UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
CÂMPUS DE ILHA SOLTEIRA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

JOVANNY BEDOYA GUAPACHA

LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS (SLAM) VISUAL USANDO SENSOR RGB-D PARA AMBIENTES INTERNOS E REPRESENTAÇÃO DE CARACTERÍSTICAS

Ilha Solteira
2017

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

JOVANNY BEDOYA GUAPACHA

LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS (SLAM) VISUAL USANDO SENSOR RGB-D PARA AMBIENTES INTERNOS E REPRESENTAÇÃO DE CARACTERÍSTICAS.

Tese de Doutorado apresentada à Faculdade de Engenharia de Ilha Solteira - UNESP, como parte dos requisitos para obtenção do título de Doutor em Engenharia Elétrica.
Área de Concentração: Automação.

Profa. Dra. Suely Cunha Amaro Mantovani
Orientadora

Ilha Solteira

2017

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

B412l

Bedoya Guapacha, Jovanny.
Localização e mapeamento simultâneos (SLAM) visual usando sensor RGB-D para ambientes internos e representação de características / Jovanny Bedoya Guapacha. -- Ilha Solteira: [s.n.], 2017
107 f. : il.

Tese (doutorado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2017

Orientador: Suely Cunha Amaro Mantovani
Inclui bibliografia

1. SLAM visual. 2. Navegação de robôs. 3. Visão computacional. 4. ROS. 5. OpenCV.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO DA TESE: LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEO-(SLAM) VISUAL USANDO SENSOR RGBD PARA AMBIENTES INTERNOS E REPRESENTAÇÃO DE CARACTERÍSTICAS

AUTOR: JOVANNY BEDOYA GUAPACHA
ORIENTADORA: SUELY CUNHA AMARO MANTOVANI

Aprovado como parte das exigências para obtenção do Título de Doutor em ENGENHARIA ELÉTRICA, área: AUTOMAÇÃO pela Comissão Examinadora:

Profa. Dra. SUELY CUNHA AMARO MANTOVANI
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. RICARDO TOKIO HIGUTI
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. AILTON AKIRA SHINODA
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. MÁRIO ANDERSON DE OLIVEIRA
Departamento de Eletro-eletrônica / Instituto Federal de Mato Grosso

Prof. Dr. MARCO APARECIDO QUEIROZ DUARTE
Departamento de Matemática / Universidade Estadual de Mato Grosso do Sul

Ilha Solteira, 04 de setembro de 2017

DEDICATÓRIA

Este trabalho é dedicado à memória de meu avô Gabriel Guapacha que está no céu, cuidando de mim, obrigado avô por estar sempre atento a mim.

Pajaro.....!

AGRADECIMENTOS

Agradeço a Deus e Madre Maria por sua infinita misericórdia para comigo e pelas bençãos recebidas.

À minha maravilhosa esposa Luz Piedad Valencia Montoya por seu amor e apoio incondicional, à minha admirável mãe Luz Mery Guapacha Villaneda por suas orações, minha irmã Adriana Bedoya Guapacha por se preocupar e ficar sempre perto de mim, às minhas sobrinhas Laura e Camila, a toda a minha família por ser meu sorriso.

À minha orientadora Profa. Dra. Suely Cunha Amaro Mantovani pela paciência, dedicação, ajuda e valiosa orientação que resultou na elaboração do presente trabalho.

Aos meus irmãos SOLO GORDOS (Alejandra, Nataly, Nikolas, Natalia) pela amizade e apoio, aos meus amigos Rodolfo Castro e Silva, Ricardo Fernando Nunes e Aref Sakr pelos momentos compartilhados no laboratório.

À minha segunda família Fazzan e Longo (Hortência, Zenaide, Ana, Marcos, Carlos, João, Alice, Joseli, Jair, Lázaro) por se preocupar por mim, pelo acolhimento, pelo imenso apoio e orações, nos momentos difíceis da minha estadia em Ilha Solteira, incluindo o grupo de oração que me deu fortaleça espiritual neste caminho.

À CAPES, pelo aporte financeiro necessário para a realização deste projeto de pesquisa.

Muito Obrigado, Deus abençoe vocês.!

RESUMO

A criação de robôs que podem operar autonomamente em ambientes controlados e não controlados tem sido, um dos principais objetivos da robótica móvel. Para que um robô possa navegar em um ambiente interno desconhecido, ele deve se localizar e ao mesmo tempo construir um mapa do ambiente que o rodeia, a este problema dá-se o nome de Localização e Mapeamento Simultâneos- SLAM. Tem-se como proposta neste trabalho para solucionar o problema do SLAM, o uso de um sensor RGB-D, com 6 graus de liberdade para perceber o ambiente, o qual é embarcado em um robô. O problema do SLAM pode ser solucionado estimando a pose - posição e orientação, e a trajetória do sensor no ambiente, de forma precisa, justificando a construção de um mapa em três dimensões (3D). Esta estimação envolve a captura consecutiva de frames do ambiente fornecidos pelo sensor RGB-D, onde são determinados os pontos mais acentuados das imagens através do uso de características visuais dadas pelo algoritmo ORB. Em seguida, a comparação entre frames consecutivos e o cálculo das transformações geométricas são realizadas, mediante o algoritmo de eliminação de correspondências atípicas, bPROSAC. Por fim, uma correção de inconsistências é efetuada para a reconstrução do mapa 3D e a estimação mais precisa da trajetória do robô, utilizando técnicas de otimização não lineares. Experimentos são realizados para mostrar a construção do mapa e o desempenho da proposta.

Palavras – chave: SLAM visual. Navegação de robôs. Visão computacional. ROS. OpenCV.

ABSTRACT

The creation of robots that can operate autonomously in controlled and uncontrolled environments has been, one of the main objectives of mobile robotics. In order for a robot to navigate in an unknown internal environment, it must locate yourself and at the same time construct a map of the surrounding environment this problem is called Simultaneous Location and Mapping - SLAM. The purpose of this work for solution to SLAM's problem is to use an RGB-D sensor with 6 degrees of freedom to perceive the environment, which is embedded onto a robot. The SLAM's problem can be solved by estimating the position and orientation, and the path of the sensor/robot in the environment, in precise form, justifying the construction of a 3D map. This estimation involves the consecutive capture of the environment's frames provided by the RGB-D sensor, where the pronounced points of the images are determined through the use of visual characteristics given by the ORB algorithm. Then, the comparison between consecutive frames and the calculation of the geometric transformations are performed using the algorithm of elimination of atypical correspondences, bPROSAC. Finally, a correction of inconsistencies is made for the reconstruction of the 3D map and the more accurate estimation of the robot trajectory, using non-linear optimization techniques. Experiments are carried out to show the construction of the map and the performance of the proposal

Keywords: SLAM visual. Robots navegation. Computer vision. ROS. OpenCV.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DoF	**D**egrees **o**f **F**reedom
BRIEF	**BinaryR**obust **I**ndependent **E**lementary **F**eatures
BFM	**B**rute **F**orce **M**atching
FAST	**F**eatures from **A**ccelerated **S**egment **T**est
FLANN	**F**ast **L**ibrary for **A**pproximate **N**earest **N**eighbors
g2o	**g**eneral **g**raph **o**ptimization
ORB	**Or**iented FAST and **R**otated **B**RIEF
PROSAC	**PRO**gressive **SA**mple **C**onsensus
RANSAC	**Ran**dom **Sa**mple **C**oncesus
SLAM	**S**imultaneus **L**ocalization **a**nd **M**apping
SIFT	**S**cala**r** **I**nvariant **F**eature **T**ransform
SURF	***S***peeded-***U***p ***R***obust ***F***eatures

LISTA DE SÍMBOLOS E VARIÁVEIS

ϵ	Relação entre inliers e o numero de correspondências
Θ	Espaço de parâmetros
α	autovalor com a magnitude maior da traça da matriz Hessiana
β	Autovalor com a magnitude menor da traça da matriz Hessiana
θ^*	Modelo de parâmetros
θ	Parâmetros do modelo
J_s	Função de custo
Δ	Limite de erro para inliers
η_o	Probabilidade de retornar uma solução ruim
τ	Transformação
$\mathcal{U}_{\mathcal{N}}$	Conjunto de Pontos
$\mathcal{M}$	Subconjunto de $\mathcal{U}_{\mathcal{N}}$
Ψ	Limiar para valores atípicos
X	Vetor de Parâmetros

Sumário

1 INTRODUÇÃO

A criação de robôs que possam operar autonomamente em ambientes do mundo real tem sido um dos principais objetivos da robótica móvel visando as mais diversas aplicações, tais como, veículos não tripulados (BUEHLER, 2009), veículos submarinos autônomos (CRUZ, 2011), rovers planetários (TUNSTEL et al, 2005), robôs domésticos (PRASSLER; KOSUGE, 2008), robôs para aplicações de alto risco para os seres humanos e um grande número de uso nas indústrias. Um robô é uma máquina capaz de reproduzir tarefas associadas aos seres humanos e na sua construção tem-se uma parte mecânica e a outra parte com dispositivos elétricos, eletrônicos, de comunicações e sistemas de informação para seu controle, fornecendo ao robô móvel (ou terrestre) certo grau de autonomia. Nos últimos anos, as aplicações envolvendo robôs móveis vêm crescendo significativamente, devido principalmente, à grande quantidade de tarefas que estes podem realizar, por sua capacidade de se locomover livremente pelo espaço de trabalho, limitados apenas por eventuais obstáculos.

As pesquisas se intensificam no desenvolvimento de robôs que realizam tarefas enquanto se deslocam autonomamente em espaços (ambientes) dinâmicos e complexos, como são os espaços exteriores (terrestres, na água e no ar) e nos espaços interiores (laboratórios, casas, escritórios etc.) (MIGLINO; LUND; NOLFI, 2010).

Um robô móvel é considerado autônomo quando apresenta comportamento inteligente devido a integração de seus sensores, responde à mudanças no ambiente, realiza múltiplas tarefas, apresenta robustez e flexibilidade, velocidade de processamento, ou seja, tem raciocínio global. Neste contexto os problemas típicos da robótica móvel são os relacionados com a navegação do robô em um ambiente geralmente desconhecido.

Um robô autônomo para navegar, deve sair de um lugar considerado a origem e alcançar uma posição considerada o destino, desviando dos obstáculos, para isso deve ter sensores que fornecem informações sobre o ambiente. Entre os tipos de sensores usados estão os sensores infravermelhos, sonares e o laser que apresenta um alto custo. Atualmente, o sistema de visão tem sido usado como principal fonte de informação sensorial, não somente pela grande quantidade de informação que fornece, mas também, pela evolução da tecnologia das câmeras, que ao longo dos anos tem reduzido seu preço.

O alcance e a resolução de um sensor apresentam limitações físicas, tais como ruídos, que influenciam no resultado das medidas, comprometendo a informação que pode ser extraída. No caso das câmeras pesam pouco e permitem medições de grande alcance e com uma imagem de boa resolução. Um sistema de visão para ser considerado robusto deve ter a capacidade de detectar objetos e fornecer uma representação adequada do mundo, de forma a realizar um planejamento inteligente de seus movimentos e uma otimização de sua trajetória (LEMAIRE et al, 2007).

Baseando-se em contínuas observações do ambiente que o cerca (robô), este processo pode ser organizado de forma hierárquica, em cinco níveis de autonomia: Mapeamento do Ambiente, Localização, SLAM e SLAM Visual, Planejamento de Caminho e Geração de Trajetória e Execução de trajetória (ALSINA et al., 2002). Os quatro primeiros tópicos são de interesse deste trabalho e por isso são comentados, os restantes podem ser encontrados em (PINHERO; ALSINA; MEDEIROS, 2003; SOUZA, 2008) para mais detalhes.

No **Mapeamento** o robô deve obter o conhecimento do entorno, ser capaz de perceber os objetos da cena e utilizar esse conhecimento para atingir os seus objetivos. Faz a coleta de dados por meio de seus sensores, conforme se movimenta pelo ambiente, realizando observações, para poder reagir de acordo com comportamentos programados ou então armazenar a informação dessas observações, para futuramente serem recuperadas. O armazenamento dessas informações pode gerar modelos computacionais com as principais características, de modo a mapear o ambiente, em outras palavras, geram um mapa, 3D, que representa o modelo que o robô possui do ambiente. (BALCH; ARKIN, 1993; THRUN, 2002).

Cada observação realizada pelo robô deve determinar a região do ambiente que esta observação está vinculada e então atualizá-la no mapa de maneira que seja fiel ao estado observado. Esse reconhecimento de regiões é feito a partir de um conjunto de características que cada variável possui e que a torna única dentro do mapa, cujo conteúdo depende do tipo de sensor e do tipo de mapa que está sendo construído.

É fundamental que o robô, em um sistema autônomo de navegação, tenha a capacidade de determinar sua **Localização e Orientação—** pose, dentro do ambiente no qual se encontra e se movimenta. A combinação da localização e orientação do robô têm como parâmetros as coordenadas cartesianas do robô e sua orientação com respeito a um referencial. A localização pode ser relativa ou global, a relativa está baseada na navegação a partir de uma posição inicial conhecida, envolvendo apenas parâmetros

internos do robô, tais como, as informações odométricas obtidas pelos sensores acoplados às suas rodas, ou seja, não depende das informações do ambiente ao seu redor. Por outro lado, na localização global onde não se conhece a posição inicial do robô, deve haver múltiplas hipóteses para determinar sua posição no ambiente, tornando-a mais complexa (BORENSTEIN; FENG, 1996; JENSFELT; KRISTESEN, 2001).

Portanto, para a navegação em um meio desconhecido, um robô móvel necessita construir um mapa do ambiente e ao mesmo tempo se autolocalizar, ou seja determinar sua pose. O processo que agrega os dois problemas é chamado **Mapeamento e Localização Simultânea**, do inglês, *Simultaneous Localization and Mapping* **(SLAM),** originalmente desenvolvido por Leonard e Durrant-White (1991) baseado em trabalhos de Smith et al (1987). Em um meio externo (*outdoor*) um *Global Position Systems* (GPS) pode resolver o problema do SLAM fornecendo as condições para a navegação de um robô. Contudo, quando se movendo em ambientes internos (*indoor*) onde os dados do GPS não estão disponíveis ou não são confiáveis o suficiente, pode ser difícil estimar a pose do robô, por isso são adotadas outras soluções (SANTANA, 2011; DURRANT; FELLOW; BAILEY,2006a; 2006b).

Por outro lado, soluções individuais não são suficientes, pois cada estimativa possui erros que se propagam entre si e se acumulam. Um mapa impreciso não será capaz de gerar uma estimativa precisa de localização do robô, portanto a resolução do problema SLAM fornece um modelo detalhado do entorno enquanto mantém sua exata localização (DISSANAYAKE et al., 2001).

As incertezas nas medidas devido às limitações técnicas ou ao ruído dos sensores vem a ser o principal problema do SLAM. Para reduzir os erros inerentes fornecendo estimativas satisfatórias são usados modelos probabilísticos. Enquanto este processo é geralmente baseado em dados fornecidos pelos sensores, tais como, scanner a laser, combinados com a odometria, a visão vem sendo utilizada nas técnicas de percepção acopladas ao SLAM, porque as metodologias aplicadas de visão computacional consistem em algoritmos eficientes de extrair e comparar características visuais, surgindo o **SLAM Visual,** ou *Visual Simultaneous Localization and Mapping* (VSLAM) (KARLSOSN et al., 2005). No processo do VSLAM enquanto a câmera se move é realizada a estimação das poses da câmera (e, portanto, a do robô), por meio do fluxo de dados de entrada (imagens e profundidade), visando construir uma representação em três dimensões do ambiente, o mais próximo do ambiente real.

Nota-se na última década, uma tendência pelo uso da visão como único sistema de percepção sensorial externa para resolver o problema do SLAM (PAZ et al., 2008; CLEMENTE; DAVISON, 2007; KLEIN; MURRAY, 2007; PINIES; TARDOS, 2008). A razão principal é atribuída à capacidade do sistema com visão, obter uma grande quantidade de informação, tais como, cor, textura, iluminação e distância dos diferentes elementos presentes na cena, fornecendo ao robô a possibilidade de integrar outras tarefas de alto nível, como a detecção e reconhecimento dos objetos, pessoas e locações. Desde a sua criação pela PrimeSense para o Microsoft Xbox 360, em 2010, a Microsoft Kinect- câmera (ou sensor RGB-D) tem sido usada para percepção sensorial de visão, principalmente pelo custo reduzido por bit de dado, rápida integração ao circuito do robô, menor consumo de potência, medição sem contato, entre outros.

As imagens fornecem uma enorme quantidade de informação, propriedades geométricas e fotométricas da cena, das fontes de luz e da câmera, seus parâmetros intrínsecos e extrínsecos. Para utilizar essas informações visuais em tarefas robóticas, é necessário o desenvolvimento de algoritmos precisos, todavia simples o suficiente para operar em tempo real e embarcado em um robô (ZHOU; MASON, 2012).

Muitas contribuições são encontradas na área do SLAM visual, mas ainda existem muitos problemas para resolver, por exemplo, acúmulo de erros devido aos parâmetros intrínsecos da câmera e das medições, que geram estimações inconsistentes da pose do robô e mapas totalmente incoerentes. Soluções possíveis para esses problemas se baseiam na aplicação ao sistema de procedimentos matemáticos robustos, entre esses, caracterização, estimação e otimização (KONOLIGE; AGRAWAL 2008; KONOLIGE, BOWMAN; CHEN, 2009; MEI et al, 2010).

Portanto, na navegação visual determina-se um caminho adequado para que um robô se locomova de forma autônoma e segura, entre um ponto de partida e um ponto de destino. A navegação visual para os robôs móveis nas últimas décadas tem contribuído para aumentar a quantidade de aplicações de veículos móveis autônomos, entre essas, vigilância, busca e salvamento, inspeção de minas, missões aéreas, mapeamento, etc.

1.1 Justificativa

Historicamente, o homem tem utilizado os recursos à sua disposição, como ferramentas (software e hardware) para auxiliar nas tarefas de forma mais eficaz, rápida e segura. Enquanto a tecnologia avança estas ferramentas oferecem vantagens capazes

de realizar trabalhos em substituição ao homem, com alta precisão, por exemplo, os robôs domésticos.

No sentido de contribuir com as pesquisas na área de navegação autônoma, neste trabalho propõe-se uma solução para o problema de localização e mapeamento simultaneos de um robô móvel e um ambiente indoor, usando um sensor Kinect baseado principalmente, nos trabalhos de Henry et al. (2012) e Endres et al. (2012).

1.2 Objetivo Geral

O objetivo deste trabalho é o desenvolvimento de um sistema que permita a construção de mapas do entorno em 3D de um ambiente interno desconhecido, usando dados de uma câmera Kinect - câmera RGB e o sensor de profundidade - D (*Deep*), com 6 graus de liberdade (DoF) embarcado em um robô. Para estimar a pose do robô de forma precisa, são realizadas técnicas baseadas em primitivas geométricas também chamadas de características visuais, onde com as informação das imagens do sensor faz-se:

- A extração de características dos *frames* (imagem e profundidade) utilizando algoritmos *open-source* adaptados ao problema em questão, em 2D;
- A determinação das correspondências de pontos de interesse entre pares de imagens visando diminuir o conjunto de pontos atípicos entre os frames;
- A aproximação das poses mediante métodos de estimação robusta para a criação do mapa;
- Um refinamento da pose por métodos iterativos e a otimização do mapa construído em 3D.

Com isso espera-se obter uma redução do ruído das medições na construção do mapa , melhores estimativas da trajetória da câmera/robô e a reconstrução mais precisa e em tempo reduzido do mapa 3D do ambiente.

1.3 Contribuições

Visando dar autonomia a um robô móvel em ambientes indoor usando a técnica do SLAM Visual e visão monocular (uma câmera) tem-se como contribuições:

- Integração de métodos matemáticos que apresentam melhor desempenho no processamento de imagens, para o acoplamento de imagens consecutivas;

- Implementação de um algoritmo baseado no PROSAC (*Progressive Sample Consensus*) para a eliminação de valores atípicos na construção do mapa 3D, agilizando a convergência do sistema na construção do mapa e auxiliando na estimação da pose do robô;

1.4 Organização do texto

Neste trabalho, além desta introdução, tem-se no capítulo 2 a revisão da literatura sobre o Mapeamento e Localização Simultâneos (SLAM), a sua evolução e o estado da arte para o SLAM usando a visão com principal sensor em robôs.

Descrevem-se no capítulo 3 são vistos a construção dos mapas e os diferentes tipos de representação dos ambientes para resolver o problema do SLAM, utilizando a visão artificial como principal sistema de percepção (RGBD).

No capítulo 4 apresentam-se os métodos encontrados na literatura mais utilizados na detecção e descrição de características em uma imagem.

Descrevem-se no capítulo 5 alguns dos principais algoritmos de estimação de parâmetros, visando encontrar as correspondências entre características mais semelhantes entre duas imagens consecutivas.

Apresenta-se no capítulo 6 a proposta deste trabalho para o mapeamento e a localização de robôs móveis em um ambiente interno, VSLAM monocular e a construção de um mapa 3D.

No capítulo 7 apresentam-se os resultados obtidos ao aplicar a metodologia proposta para sequências de imagens em um cenário de teste para o mapeamento e localização.

Finalmente no capítulo 8 apresentam-se as conclusões e sugestões de trabalhos futuros, seguido pelos apêndices.

2 REVISÃO DA LITERATURA

Para fornecer os subsídios para o entendimento deste trabalho, neste capítulo apresentam-se a evolução histórica e o estado da arte para resolver o problema do Mapeamento e Localização Simultâneos (SLAM), especialmente para o SLAM Visual ou VSLAM utilizando como principal sensor a câmera Kinect RGB-D.

2.1 Evolução Histórica

O problema do SLAM tem sido um dos assuntos mais estudados pelos pesquisadores da área. Grande ênfase foi dada em uma conferência do IEEE em San Francisco - USA, IEEE *Robotics and Automation Conference*, em 1986, quando alguns pesquisadores apresentaram métodos teóricos de estimação para os problemas de localização e mapeamento, baseadas no teorema de Bayes. Do resultado das discussões percebeu-se a necessidade de trabalhar com soluções probabilísticas para o mapeamento do ambiente e a localização de robôs (SMITH; CHEESEMAN, 1986).

Depois disso foi desenvolvida uma série de trabalhos que estabeleceram uma base estatística relacionando os "marcos" (*landmarks* ou pontos de referência) de um mapa, entre esses o de Smith e Cheeseman (1986), e o de Leonard, Durrant e Whyte (1991). As *landmarks* são estruturas do ambiente utilizadas pelo robô para se localizar durante a navegação, identificadas por meio de incertezas geométricas. No trabalho de Smith e Cheeseman é mostrado como estabelecer uma base estatística entre os marcos e o alto grau de correlação crescente entre as estimativas da localização dos diferentes marcos em um mapa (a correlação entre os marcos não eram eliminados, dificultando a convergência do algoritmo).

Outro trabalho importante na evolução do tema SLAM é o trabalho publicado por Crowley (1989) em navegação de robôs móveis usando um sonar e a técnica de estimação de estados, Filtro de Kalman (KF), onde é mostrado que para um robô em um ambiente desconhecido coletando observações de marcos, as estimativas de todos os marcos são correlacionadas (umas com as outras) devido ao erro comum na estimativa de localização do robô. Uma solução completa e consistente para o problema combinado de localização e mapeamento após cada marco observado, é que o sistema deve ser atualizado, empregando um estimador para um vetor de estados com grandes

dimensões, para a redução do erro e visando a convergência, sendo necessário então, aumentando o custo computacional, pois é armazenada a medição e a nova localização desta medição.

A estrutura do problema de SLAM, o resultado da convergência e a criação do acrônimo SLAM foram apresentados pela primeira vez, no *Internacional Symposium on Robotics Research* (1996) em um trabalho de pesquisa de robótica móvel de Durrant-Whyte et al (1996), onde a teoria fundamental sobre a convergência e os primeiros resultados foram desenvolvidos, juntamente com Csorba, Uhlmann e Durrant (1996).

Outros eventos marcaram a evolução do SLAM, por exemplo, o *Internacional Symposium on Robotics Research* em 1999 quando dedicaram a primeira sessão sobre SLAM. Foram apresentados neste evento os métodos probabilísticos de mapeamento e localização baseados em filtro de Kalman, introduzidos por Thrun (1998). Depois disso foi realizado com sucesso em Estocolmo-Suécia o 1st SLAM Summer School no ano de 2002. Neste evento, participaram 50 pesquisadores da área, com destaque para Hugh Durrant–Whyte, John Leonard, Raja Chatila, Sebastian Thrun e Wolfram Burgard. Outras escolas de verão aconteceram, em 2004, 2006 e 2009, quarta e última edição do SLAM Summer School em Sidney-Austrália, onde foi observado o crescente interesse de pesquisadores pelo tema do SLAM (SANTANA, 2011).

Posteriormente, os trabalhos sobre o problema do SLAM tinham a preocupação com a eficiência computacional e a solução de questões como associação de dados. Também buscavam a redução da complexidade dos algoritmos como, em Paz et al (2008), enquanto que estudos sobre consistência de algoritmos são apresentados por Huang e Dissanayake (2007) e Huang, Wang e Dissanayake (2008).

Diferentes grupos de pesquisa em robótica e visão por computador em todo o mundo têm desenvolvido muitas técnicas para resolver o SLAM utilizando diferentes tipos de sensores, tais como, os scanners a laser 2D (MONTEMERLO et al , 2002; GRISETTI et al, 2007), scanners 3D (NÜCHTER et al, 2007), câmeras monoculares (STÜHMER;GUMHOLD;CREMERS, 2010; KOESER; BARTCZAK; KOCH, 2007; STRASDAT; MONTIEL; DAVISON, 2010), sistemas estéreo (NISTER; NARODITSKY; BERGEN, 2004; AKBARZADEH et al, 2006; KONOLIGE; AGRAWAL, 2008; COMPORT; MALIS; RIVES, 2010), até coleções de fotos sem classificar (SNAVELY; SEITZ; SZELISKI, 2006).

Devido ao crescimento exponencial no poder de armazenamento e processamento na computação, inicia-se por volta do ano 2000, com Andrews Davison

uma solução para o SLAM utilizando câmera, com alguns resultados mostrados em Davison e Murray (2002). O maior atrativo da navegação autônoma com os dispositivos para capturar imagens é a riqueza das informações do ambiente que eles são capazes de fornecer, de modo a orientar às decisões do robô, principalmente se comparada às medidas obtidas pelos lasers e sonares. Esta informação pode ser mais bem aproveitada para construir mapas mais detalhados e manter atualizada a pose do robô. (SUJAN; MEGGIOLARO; BELO, 2005; SE; LOWE; LITTLE, 2001).

2.2 Estado da Arte para o SLAM Visual

Uma imagem é uma caracterização espacial densa do ambiente que define cada uma das suas regiões com um parâmetro de cor, que pode variar dentro de um intervalo de possibilidades, permitindo estabelecer uma grande quantidade de padrões que descrevem bem as propriedades do ambiente, a curvatura de áreas na imagem em busca de padrões como retas, curvas e cantos, comuns em ambientes internos (ASADA; BRADY, 1986)

Localização e mapeamento simultâneos a partir de câmeras chamado SLAM Visual é objeto de estudo desde o surgimento das primeiras pesquisas em visão computacional (MORAVEC, 1977). Esta classe de algoritmos funciona extraindo características como pontos em Davison (2003) ou bordas, Tomono (2010), de uma ou mais câmeras, Goncalves et al. (2005) e Davison (2003), e relacionando estas informações por um mapa. Extratores de pontos característicos como o operador de cantos de Harris e Stephens (1988) ou de Shi e Tomasi (1994) e extratores de linhas como a transformada Hough, em Trucco e Verri (1998) são geralmente utilizados para processar a imagem.

Muitas aplicações relevantes na robótica e na visão por computador devem ter a capacidade de adquirir rapidamente modelos 3D do entorno e fazer a estimação da câmera com respeito a este modelo. Um requisito importante para a autonomia de robôs móveis na realização de tarefas como a localização, planificação e navegação, especialmente para trabalhos em ambientes dinâmicos e complexos onde, para se deslocar é necessário conhecer a sua localização, é a disponibilidade de um mapa do espaço de trabalho do robô. Portanto, a localização da câmera no ambiente exige o modelo 3D desse ambiente, e construir o modelo 3D, por sua vez, requer a pose da

câmera, assim a trajetória da câmera e o modelo 3D precisam ser estimados ao mesmo tempo.

Com Olson, Matthies e Schoppers (2003) tem-se um dos primeiros trabalhos sobre a navegação visual baseados numa configuração estérea (mais de uma câmera), que são capazes de calcular facilmente e com boa precisão as posições reais em 3D dos pontos de referência de uma cena, por meio da triangulação encontrada em Hartley e Sturm (1997) que fornece uma informação importante no problema do SLAM Visual. Trabalhos mais recentes e com sistemas mais eficazes em SLAM estéreo, podem ser encontrados em Konolige e Agrawal (2008) e Konolige, Bowman e Calonder (2009).

O surgimento no mercado das câmeras Kinect que fornecem imagens coloridas e densos mapas de profundidade, permitiram uma nova abordagem para o SLAM Visual, a geração de mapas 3D. Fioraio e Konolige (2011) apresentam um sistema para estimar as características visuais e gerar uma ótima estrutura em 3D por meio do método *Bundle Adjustment* (BA) que alinham densas nuvens de pontos, ou seja, banco de dados que contém pontos no sistema de coordenadas tridimensional com registro preciso de objetos no espaço, usando sensor Kinect. (TRIGGS et al.,2000)

Outro trabalho sobre SLAM visual baseado em RGB-D foi proposto por Henry et al.(2012), que geram modelos 3D de ambientes interiores, integrando as informações da aparência e forma do ambiente dada pelo sensor. O alinhamento entre os frames é calculado pela otimização das relações de aparência e às correspondências de caracteristicas encontradas.

Em Endres et al (2012) descrevem um sistema SLAM com sensor RGB-D , sensor da Microsoft Kinect para gerar um mapa 3D, com o robô e a mão livre, sem o uso de outros sensores e da odometria . Analisam e tratam a influência de parâmetros para gerar mapas com os descritores/detectores de características, quantidade de características visuais e os métodos de avaliação como o *Environment Measurement Model* (EMM) para realizar a avaliação das transformações estimadas por correspondências de características e o algoritmo de interações.

Ao se modelar o SLAM visual com mecanismos de filtragens probabilísticas, os mapas são constituídos pela média e matrizes de covariância das posições espaciais (em três dimensões) correspondentes às características observadas, sendo estas computadas com o mesmo formalismo matemático desenvolvido em SLAM com sensores de profundidade. O filtro de Kalman ou o filtro de partículas, presentes nos sistemas MonoSLAM (DAVISON, 2003) e VSLAM (KARLSSON et al., 2005), são exemplos

da literatura das primeiras abordagens com êxito em SLAM visual. Todavia, o custo computacional envolvido na atualização das matrizes de covariância destes filtros limita o processo de reconstrução a algumas centenas de características.

No Brasil existem vários grupos trabalhando atualmente em SLAM e o VSLAM com alguns trabalhos publicados, entre eles o de Santana (2011), Peñafiel e Pereira (2014), Silveira e Rives (2009), Botelho et al. (2009) com pesquisas em visão computacional para ambientes aquáticos, gerando material de pesquisa importante como, em Romero et al (2014).

2.3 Comentários

Neste capitulo foi apresentado o problema da Localização e Mapeamento Simultâneos, que consiste em capacitar um robô móvel a construir um mapa de um ambiente e ao mesmo tempo calcular e manter atualizada a sua pose. E ainda a evolução das pesquisas e como chegou ao SLAM Visual usando a câmera Kinect RGB-D que fornece a imagem colorida e a profundidade.

A visão por computador é a base para diversas aplicações que variam desde o reconhecimento e rastreamento de objetos até a reconstrução de ambientes, enfoque do trabalho em questão cuja proposta é, a partir da informação do entorno de um ambiente interno e desconhecido, gerar um mapa em três dimensões (3D).

3 SLAM VISUAL

Neste capítulo são abordados com maiores detalhes os conceitos que envolvem a construção do mapa e os tipos de representações para o VSLAM visando dar autonomia ao robô.

3.1 SLAM VISUAL (VSLAM)

Nos últimos anos, a tendência na robótica é usar a visão artificial como único sistema de percepção sensorial para resolver o problema de construção de mapas, enquanto o sistema (robô) se localiza em relação ao mapa conhecido como SLAM Visual sendo um dos problemas desafiantes e essencial para robôs autônomos. O uso da visão proporciona ao sistema a capacidade de obter informação de distâncias, texturas, iluminação e cores dos diferentes elementos que conformam o ambiente (KARLSSON et al , 2005), permitindo que os robôs autônomos integrem tarefas de alto nível como a detecção e reconhecimento de padrões numa cena (KLEIN; MURRAY, 2007; CLEMENTE; DAVISON, 2007; LIM; LIM; KIM, 2014).

O SLAM Visual é uma técnica que constrói um mapa visual que consiste, normalmente, de um conjunto único de marcos identificados pelo robô no seu caminho. Em um ambiente desconhecido o robô automaticamente identifica marcos, e quando revisita áreas mapeadas, utiliza os marcos para estimar e melhorar a sua pose e a localização dos marcos. Essa atualização contínua na posição dos marcos e na pose do robô, incrementalmente melhora o mapa calculado.

Os principais tipos de sistemas de percepção visual utilizados pelos robôs são a visão monocular, visão estérea e visão omnidirecional. Diz-se que um sistema é de visão monocular quando utiliza uma câmera (uma imagem) para capturar as imagens do ambiente, não fornecendo diretamente a informação de profundidade, apresentado em Davison (2003). No sistema de visão estérea são utilizadas duas ou mais câmeras para capturar as imagens do ambiente e por meio da técnica de triagulação de pontos é possível obter a informação de profundidade, porém aumentando o custo computacional, um exemplo deste tipo de sistema é encontrado em Davison e Murray (2002). Um sistema de visão omnidirecional apresenta uma imagem panorâmica do

ambiente, por outro lado a imagem pode sofrer deformação quando utiliza um espelho ou o casamento das imagens, quando em sistemas multicâmeras (SANTANA, 2011).

Com a difusão dos sensores RGBD de baixo custo tem-se a abordagem para o SLAM Visual, também chamada de RGBD-SLAM, que utiliza a informação visual e da profundidade de cada pixel na imagem de somente um sensor, o que permite ter conhecimento da localizaçao dos elementos ou características presentes no ambiente, onde essas características são utilizadas para relacionar as imagens obtidas do entorno.

Uma característica é considerada de boa qualidade quando contém as seguintes propriedades:

- Ser notável - fácil de obter;
- Válida - poder ser medida de maneira exata;
- Invariante - na rotação, translação, escala e iluminação.

O processo de obtenção de características é composto de duas etapas, a detecção e a descrição. A etapa de detecção consiste no processamento da imagem para obter uma série de elementos sobressalentes. Na descrição é construido um vetor de características baseado na imagem. Estas duas etapas são importantes ao problema de associação de dados, permitindo a criação de ambientes em três dimensões.

Apesar das muitas contribuições feitas na área do SLAM Visual, muitos sistemas sofrem de erros de acúmulo na exploração do ambiente, erros de hardware (câmeras) e software (algoritmos probabilísticos de estimação), as causas principais são:

- Movimentos da câmera/robô geralmente não suaves para a detecção de características - os robôs se movimentam em superfícies não uniformes gerando uma imprecisão no cálculo da trajetória do sensor e como consequência a câmera captura as imagens com baixa textura ou embassadas (DAVISON, 2003);
- Geralmente os entornos explorados pelos robôs contêm elementos em movimento, se isto não é considerado, erros são gerados em todo o sistema (WANG; THORPE; THRUN, 2007);
- Ambiente é repetitivo visualmente, ou seja, uma área anteriormente vista é mais difícil de reconhecer, tornando complexa a tarefa do VSLAM.

Por meio da coleta de dados de seus sensores o robô gera os modelos computacionais com as principais características estruturais do ambiente e essas informações são usadas para a construção do mapa desse ambiente. Na linha de

mapeamento robótico surgem na literatura várias formas de mapear o ambiente, em Elfes (1989) são usadas diversas estruturas gemétricas e topológicas com o objetivo de obter a representação do ambiente com boa precisão. Algumas representações são comentadas a seguir, visando sua utilização no trabalho.

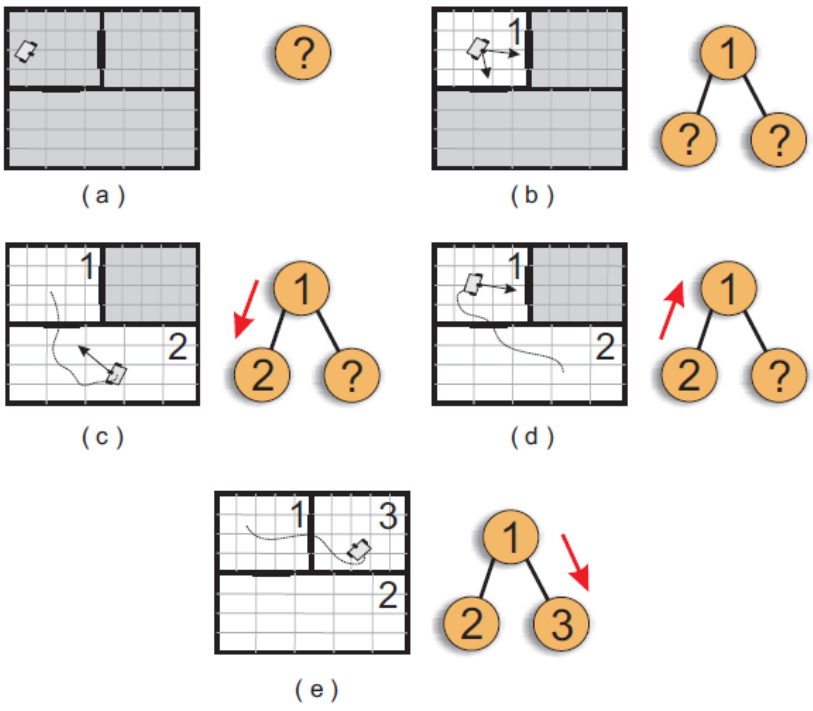
3.2 Representação dos Ambientes

Entre os diversos tipos de representação de mapas do ambiente tem-se três principais métodos, os mapas Métricos, os Topológicos (THRUN, 2002) e os OctoMaps (CHOSET;FOX, 2004; ROCHA, 2006).

3.2.1 Mapas topológicos

Os mapas topológicos são aqueles representados computacionalmente por meio de um grafo que descreve um arranjo de nós (ou vértices) conectados entre si por uma série de arestas (elos), ilustrado na Figura 1. Normalmente, os grafos descrevem os espaços livres para a execução das tarefas, os nós são as regiões com as informações sensoriais homogêneas e os elos refletem a conexão entre estas regiões.

Figura 1 – Mapa topológico



Fonte: Santana (2011)

Também podem ser chamados de mapas relacionais porque fornecem as relações entre os vários locais ou objetos presentes no ambiente. Mapas topológicos registram informações sobre determinados elementos ou locais do ambiente, chamados marcos, assim, os elementos dos mapas topológicos são os marcos e as relações. Essas relações podem ser de vários tipos, tais como, podem indicar o descolamento entres dois marcos, a existência de um caminho entre eles, os comandos de controle que movem o veículo de um marco ao outro, e assim por diante (FOX; BURGARD; THRUN, 1999). Porém a localização do robô usando mapas topológicos é abstrata, ou seja , não há como definir explicitamente a posição e a orientação do robô, pode-se afirmar somente em qual nó do grafo ou em qual região do ambiente ele se encontra.

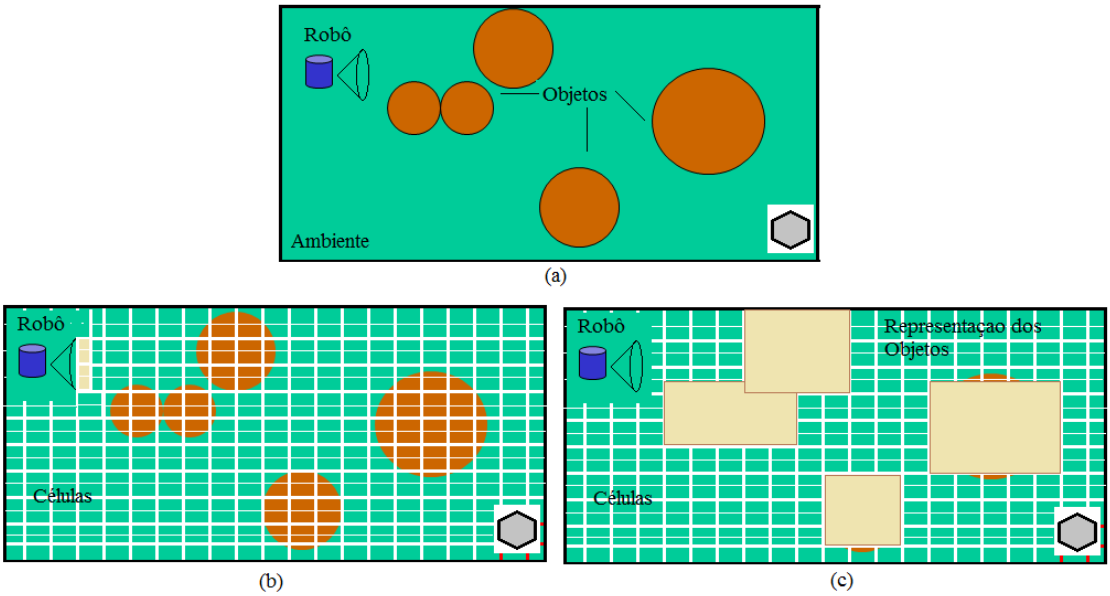
Produzem mapas compactos para processamento e armazenamento e permitem definir de maneira mais simples as tarefas de mais alto nível, como por exemplo, planejamento de rotas por algoritmos tradicionais de busca em grafos como em Shatkaye e Kaelbling (1997), porém não é recomendado para grandes ambientes e complexos que podem apresentar informações ambíguas, dificultando a construção e manutenção desse tipo de mapa. Segundo Dudek, Freedman e Hadjres (1993), a principal vantagem desta abordagem é sua natureza qualitativa e sua relação com a teoria de cognição humana. Outros trabalhos que utilizam este tipo de representação são os de Thrun (1998) e Cheong, Park e Kee Park (2008).

3.2.2 Mapas Métricos

O mapa métrico reproduz com um determinado grau de fidelidade, a geometria do ambiente no qual o robô está inserido. Para que isso seja possível, os mapas métricos dividem o espaço em células regulares, as quais representam características de interesse sobre o local. Estes mapas retratam bem o mundo real, por isso podem ser usados para identificar paredes, obstáculos e passagens. Os mapas métricos são representados por grades de ocupação e pelos mapas de características.

A representação usando grade de ocupação, proposta formalizada por Elfes (1989), representa os espaços contínuos do ambiente na forma discretizada como uma grade ou matriz multi-dimensional (2D ou 3D). Cada elemento da matriz, também chamado de célula, contém a probabilidade de que esse espaço esteja ocupado por um obstáculo, vazio ou pode ainda não ter sido explorado. Ilustra-se este tipo de representação pela Figura 2.

Figura 2- Grade de Ocupação. a) Robô no ambiente. b) Divisão do ambiente em grades c) Representação dos objetos no ambiente.

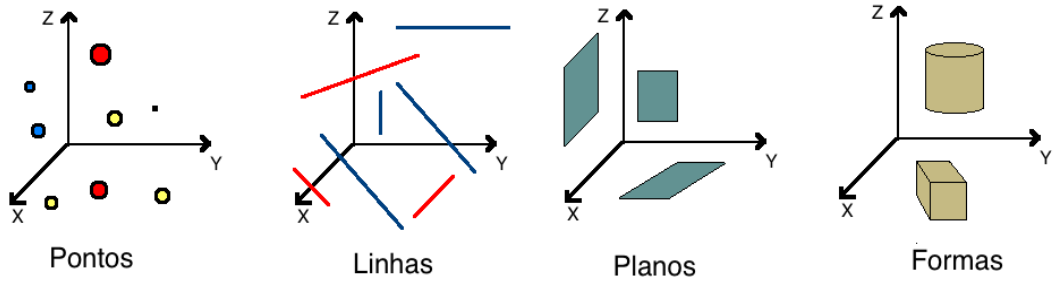


Fonte: Própria do autor

Segundo Thrun (1998), a representação dos mapas em grade de ocupação são simples de serem construídas e mantidas incluindo ambientes complexos e grandes. Dentro de um sistema de coordenadas global, a posição geométrica do robô é simplificada com a representação do mapa em grade de ocupação e conforme sua resolução permite representar melhor os detalhes do ambiente. Por outro lado, a representação em grade de ocupação necessita de um espaço de armazenamento maior e um tempo maior de processamento (SELVATICI, 2009).

No caso da representação por mapa de características são filtradas informações da câmera em busca de padrões relevantes encontrados em localizações específicas do ambiente. Esses padrões ou características (*feature*) são elementos facilmente notados pelo robô, como comentado anteriormente, distinguível entre si e devem ser reobserváveis de diferentes pontos de vista do ambiente. É comum identificar nas imagens, pontos, linhas, planos e formas para a construção dos mapas de características, como ilustrada pela Figura 3.

Figura 3 - Principais tipos de características extraídas em imagens



Fonte: Própria do autor.

Nas soluções propostas para o SLAM visual frequentemente são utilizados pontos como características do ambiente, seja para visão monocular (FU; YANG, 2009) ou visão estérea (HAN et al, 200 7). Isto é devido não somente a serem os pontos, objetos geométricos simples, mas também devido a existência de vários algoritmos usados para detectar pontos notáveis (pontos de interesse) em imagem.

Existem várias técnicas para a obtenção dos pontos de interesse. Entre essas estão o algoritmo Harris, SIFT, SURF que apresentam forte invariância à rotação, escala, iluminação e ruído de imagem, e que serão tratados no próximo capítulo.

Essa abordagem gera um menor custo computacional de manutenção, pois lida com uma menor quantidade de informações, contudo, surge o custo decorrente da busca por padrões nas informações que sejam relevantes para aquela aplicação, assim como a caracterização desses padrões de maneira a permitir a correspondência com estruturas semelhantes.

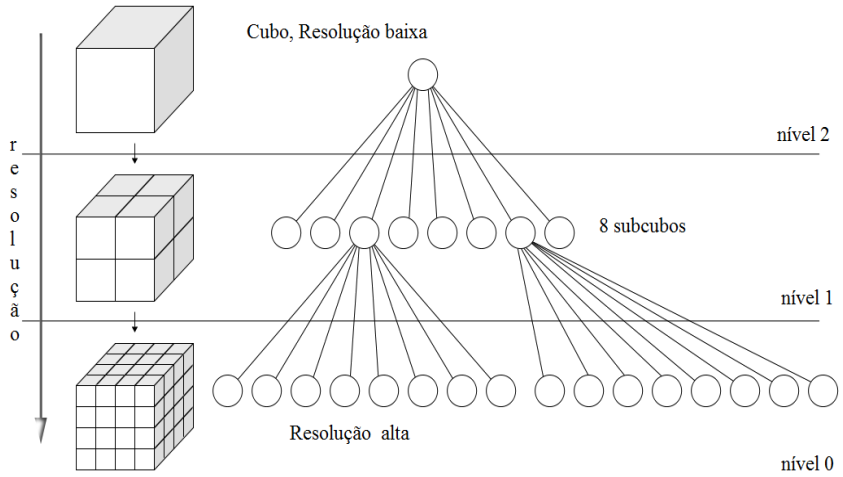
Como as variáveis em um mapa de características não possuem posições fixas no espaço como acontece em um mapa métrico, a incerteza quanto a essa estimativa pode ser incorporada diretamente no parâmetro que a define, sob a forma de uma distribuição de probabilidades similar à utilizada no tratamento do problema de localização (ROMERO; CARZOLA, 2009).

3.2.3 *OctoMap*

O *OctoMap* está baseado na representação em árvores (*Octrees*) para fornecer máxima flexibilidade considerando a área de mapeamento e a resolução, desenvolvido por Hornung et al (2013). Estima a probabilidade de ocupação para garantir a atualização e lidar com o ruído do sensor. Usa métodos de compactação para reduzir o tempo de resposta.

O *OctoMap* é uma estrutura de dados hierárquicos que utiliza uma subdivisão do espaço 3D, e cada nó na estrutra representa o espaço contido em um volume cúbico, chamado *voxel* (volume /pixel). Este volume se subdivide recursivamente em oito sub-volumes, até que o tamanho do voxel seja mínimo (valor determinado pela resolução), ilustrada na Figura 4. Devido a grande quantidade de dados esta representação de mapas tem um crescimento exponencial, mas modela o ambiente de forma a reduzir o requisito de memória (HORNUNG et al , 2013).

Figura 4 - Estrutura *OctoMap* do ambiente.



Fonte: Própria do autor

3.3 Conceitos gerais para o SLAM

O problema do SLAM pode ser visto como a busca da melhor estimação da pose do robô/câmera, reduzindo as incertezas devido ao efeito do ruído nas medidas, por isso são usadas técnicas probabilísiticas descritas por exemplo em Thrun, Burgard e Fox (2005).

A ideia baseia-se na construção do mapa o mais fiel possível, estimando a todo o momento a pose do robô atualizando os mapas anteriores. A estimação da posição (crença) é representada pela função densidade de probabilidade.

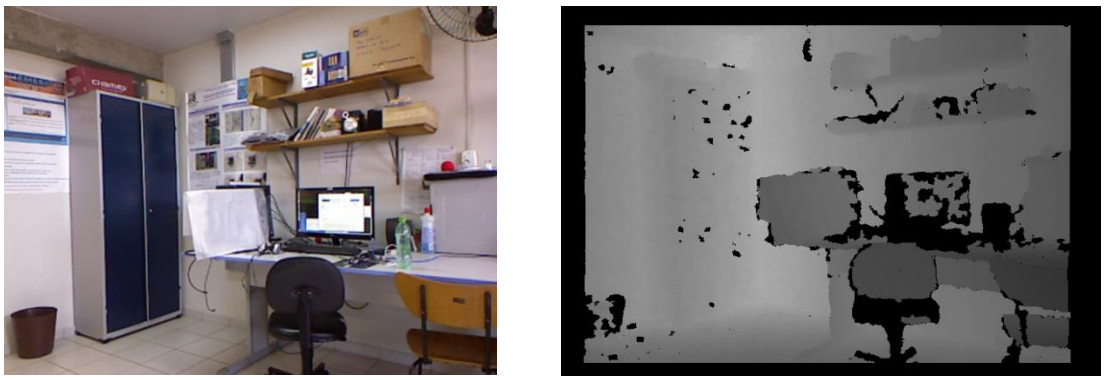
Seja x_t o estado do robô no tempo *t*, representando as variáveis de movimento, por exemplo, a pose , sendo z_t , a medida da imagem da câmera e u_t os dados de controle (denotando a mudança de estado), a crença, C sobre a variável x_t é dada pela Equação (1),

$$C\ (\ x_t\ ) = p(x_t|z_{1:t}\ , u_{1:t}) \qquad (1)$$

onde p é a distribuição de probabilidade sobre o estado x_t para o tempo t, condicionado a todas as medidas $z_{1:t}$ e a todos os sinais de controle $u_{1:t}$ (HOGMAN, 2013).

No caso do SLAM Visual com uso do sensor RGB-D considera-se a sequência de vídeos como uma conjunto de frames (dados da câmera RGB e a profundidade). Neste caso cada medida z_t é um par RGB e a profundidade para o tempo t. O número total de frames depende da duração da sequência, e da taxa efetiva de frame que é limitada pela taxa máxima do sensor Kinect (30Hz). No sensor RGD-D a diferença entre a observação e a projeção é utilizada para calcular a profundidade por triangulação usando uma linha base conhecida. Como ilustrações são apresentadas na Figura 5 os sinais da câmera e do sensor de profundidade.

Figura 5 - a) Imagem da câmera. b) Imagem de profundidade



(a) (b)

Fonte: Própria do autor.

3.3.1 Filtragem e Suavização

Encontram-se na literatura diferentes técnicas classificadas como filtragem e suavização que resolvem o problema do SLAM. Na filtragem modela-se o problema como a estimação de estados *on-line*, cujo estado do sistema consiste na posição atual do robô e do mapa. A estimativa é melhorada à medida que novos dados sejam incorporados (HOGMAN, 2013).

Extended Kalman Filtering (EKF) (THRUN; BURGARD; FOX, 2005) e Filtro de partículas (GRISSETTI et al, 2007) são as técnicas mais comuns de filtragem e consideradas métodos de SLAM *on-line*. O EKF tem custo computacional baixo devido a que armazena a hipótese anterior e a atual, diferente do filtro de partículas que

armazena várias hipóteses das variáveis de estados, que crescem com os movimentos do robô (HOGMAN, 2013).

Na técnica de suavização estima-se a trajetória completa do robô baseado no conjunto de medidas usando a minimização do erro de mínimos quadrados.

3.3.2 Grafo da pose

O problema do SLAM pode ser descrito por modelo de grafos combinando as teorias de grafos e probabilidade, gerando o grafo da pose, onde os nós (ou vértices) representam a variável de estado descrevendo as poses das câmeras e os elementos representam as restrições entre as observações ligando um par de nós. A otimização matemática tem o objetivo de encontrar a configuração de nós semelhantes e está baseada no modelo do grafo. Existem na literatura diferentes estruturas para modelos de grafos, entre esses estão o GraphSLAM (THRUN; BURGARD; FOX, 2005), TORO (GRISSETTI et al, 2007), HOG-Man (ENGELHARD et al, 2011) e g^2o (KUMMERLE et al. 2011).

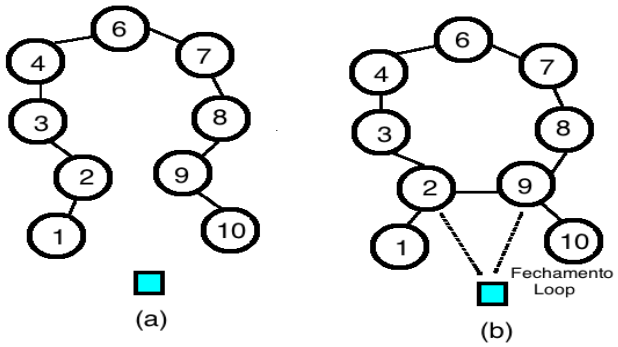
O *General Graph Optimization* - g^2o ou otimização geral de grafos desenvolvido por kümmerle et al (2011), fornece a possibilidade de redefinir, de modo preciso, a função não linear do erro e como pode ser executada. Este algoritmo é um dos mais utilizados por sua eficiência e, portanto também foi usado neste trabalho.

3.3.3 *Loop closure e a Otimização do grafo*

Quando um robô revisita uma posição este evento é denominado de detecção de *loop closure*, desenvolvida por Angeli et al (2008). Permite estimar mais precisamente o caminho real percorrido e corrigir o mapa correspondente.

Baseia-se em fazer uma suposição sobre o caminho seguido pelo robô, mantendo o histórico dos frames, de forma a verificar se o frame atual tem alguma correspondência com alguns dos anteriores. Se a observação atual contém suficientes similaridades com outra observação do passado, calcula-se uma transformação entre os frames e uma nova restrição pode ser inserida, repetindo-se várias vezes esse procedimento. Considera-se que com estas novas restrições, o erro acumulado do grafo é reduzido. Teoricamente, todas as poses podem ser associadas se esta relação existe. (HOGMAN, 2013). Mostra-se na Figura 6 um exemplo deste procedimento.

Figura 6 - Detecção de *loop closure* (a) Um exemplo. (b) A redução de um nó.



Fonte: Adaptado de Hogman (2013).

A detecção de *loop closure* fornece novas restrições que podem ser inseridas no grafo como novos elementos, juntando as poses relacionadas. A otimização do grafo consiste em encontrar uma configuração ótima de vértices que respeite todas estas restrições.

3.4 Comentários

Neste capítulo foram apresentados os conceitos gerais que envolvem o SLAM e o VSLAM. Em sistemas visuais a representação do ambiente torna-se um importante parâmetro e a sua escolha proporciona a reconstrução de mapas do ambiente confiáveis e com boa resolução, e ainda mapas otimizados com relação ao custo computacional e ao armazenamento, permitindo a navegação do robô e a sua localização no ambiente.

4 EXTRAÇÃO DE CARACTERÍSTICAS

Neste capítulo apresentam-se os subsídios necessários de algoritmos para extração de caracterísiticas, depois da captura da imagem, no processo do VSLAM. Abordam-se alguns dos métodos matemáticos encontrados na literatura que permitem detectar e caracterizar áreas ou pontos de interesse nas imagens. Muitas técnicas em visão computacional e mais especificamente em reconhecimento de objetos são baseadas na detecção/descrição de pontos de interesse do objeto ou superfície, isto é feito baseado na extração de características.

4.1 Determinação das características visuais

Para rastrear os pontos de interesse durante o movimento da câmera /ou robô, diz-se que uma característica é confiável se é invariante a localização da imagem, escala e rotação. O processo de extração de características de uma imagem se compõe de duas fases, a detecção de um ponto de interesse ou *keypoint*, o qual identifica a área de interesse e o descritor que caracteriza a região.

Tipicamente o detector identifica a região contendo uma forte variação da intensidade, tal como, um canto (*corner*) . O descritor cria um vetor de características multidimensional calculando a medida das principais orientações dos pontos ao redor, o qual vetor identifica o ponto de interesse dado. A invariância do descritor às trocas de posição e orientação permite uma maior comparação de imagens ou *matching*, de forma a associar alguns pares de pontos de interesse entre um par de frames (imagem + profundidade).

4.1.1 Detectores

As propriedades de um bom detector de características de uma imagem são a precisão na pose (posição e orientação) no interior de imagem, repetitividade, eficiência nos tempo de execução, robustez, distinção, não variação a luminosidade, rotação, escala e zoom (TUYTELAARS; MIKOLAJCZYK, 2008). No processo de detecção de características de uma imagem utilizam-se os detectores de esquinas (intersecção de

duas ou mais bordas), de blobs (região da imagem diferente do entorno), que estão baseados na função *Hessiana*.

4.1.2 Descritores

Quanto aos descritores existem trabalhos na literatura que fazem estudos comparativos em algoritmos de descrição de características centrados no problema do VSLAM (GIL et al., 2010). A avaliação se baseia em números de conincidências corretas e incorretas que se interceptam por meio de sequências de imagens, com trocas significativas na escala, ponto de vista e de iluminação.

Visando fornecer os subsídios para o trabalho em questão, alguns algoritmos de detectores/descritores são brevemente apresentados. Dentre os principais, tem-se os Harris *Corner* , SIFT, SURF , NARF, BRIEF e o ORB , dos quais alguns são detectores e/ou descritores conforme consta da Tabela 1.

Tabela 1 - Detectores e descritores de características.

Método	Detector	Descritor
HARRIS *Corner*	SIM	NÃO
SIFT	SIM	SIM
SURF	SIM	SIM
FAST	SIM	NÃO
BRIEF	NÃO	SIM
ORB	SIM	SIM

Fonte: Própria do autor.

4.1.3 Harris *Corner*

Foi um dos primeiros detectores proposto por Harris e Stephens (1988), para o tratamento das imagens, detectando não somente cantos, mas também bordas e *keypoints*. Está baseado no cálculo de uma matriz de autocorrelação das intensidades das imagens descrevendo as variações locais, e esse cálculo deve ser renovado cada vez que houver mudança de escala. A matriz de autocorrelação é mostrada na Equação (2),

$$\mu(x, \sigma_1, \sigma_D) = \begin{bmatrix} \mu_{11} & \mu_{12} \\ \mu_{21} & \mu_{22} \end{bmatrix} = \sigma_D^2 \mathrm{g}(\sigma_1) * \begin{bmatrix} L_x^2(x, \sigma_D) & L_x L_y(x, \sigma_D) \\ L_x L_y(x, \sigma_D) & L_y^2(x, \sigma_D) \end{bmatrix} \quad (2)$$

onde, σ_1é a escala, σ_D é a diferenciação da escala e L é a derivada calculada na direção do ponto *(x, y).*

Para a detecção de esquinas combinam-se o traço e o determinante da matriz, sendo esquinas, E_{esq} os máximos locais da função, como mostrado na Equação (3),

$$E_{esq} = \det(\mu(x, \sigma_1, \sigma_D) - k * trace^2(\mu(x, \sigma_1, \sigma_D))) \quad (3)$$

4.1.4 Caracterizador SIFT

Scalar Invariant Feature Transform **(SIFT)** apresentado por Lowe (2004) atualmente é muito utilizado em robótica e visão computacional. Este método detecta pontos característicos da imagem, invariantes (a escala) e distintos, que facilmente podem ser combinados para executar tarefas como detecção de objeto e reconhecimento, ou para calcular a transformação geométrica entre imagens. Consiste de uma série de operações com complexidade crescente, de modo que parte de processamento mais pesado se execute somente aos candidatos mais prováveis, cujo algoritmo é descrito a seguir:

1. A primeira etapa está baseada na pirâmide de Diferença de Gauss (DoG) para a detecção de extremos (máximos e mínimos), de modo a identificar pontos de interesse invariáveis à escala e rotação;
2. Na segunda etapa tem-se a determinação de pontos de interesse estáveis usando modelos matemáticos para determinar sua localização e escala;
3. Na terceira etapa apresenta-se a direção do gradiente da imagem que é usada para assinalar uma ou mais orientações dos pontos de interesse;
4. Na última etapa, o gradiente de imagem local é transformado para ser estável às distorções e mudanças na iluminação (HOGMAN, 2013).

Detector: a invariância a escala do detector **SIFT** é fundamental para aplicações em robótica móvel. Como a câmera se move, as áreas contendo os pontos de interesse aparecerão maiores ou menores relativamente ao fator da escala, a ideia é mostrada na Figura 7 (a) e (b). Matematicamente, uma função contínua chamada espaço de escala, *L(x, y, σ),* é definida como a convolução entre a gaussiana *G(x, y, σ)* com variância, σ e a imagem, *I(x, y)*, Equação (4),

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (4)$$

onde, *G(x,y,z)* é expressa pela Equação (5),

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \quad (5)$$

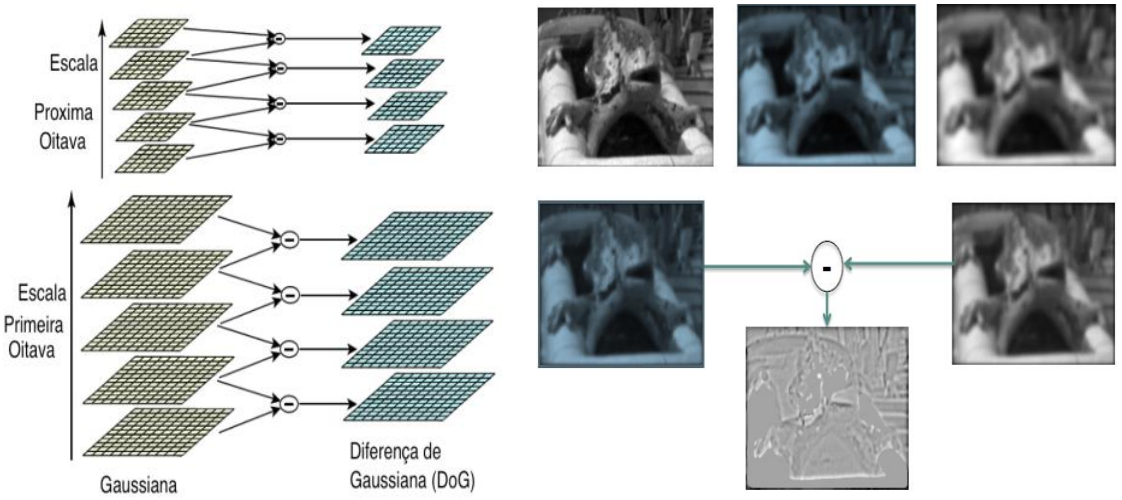
O DoG é dado pela diferença de imagens filtradas em escalas próximas, separadas por uma constante de escala *k*, definida pela Equação (6),

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma) \quad (6)$$

O resultado de efetuar a convolação entre uma imagem e o filtro DoG, *D* permite localizar pontos invariantes à escala e orientação, dado pela Equação (7),

$$D(x, y, \sigma) = \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (7)$$

Figura 7 – Detecção. a) DoG em diferentes escalas. b) Imagem com DoG.

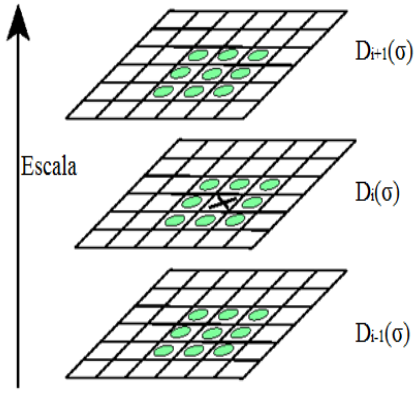


(a) (b)

Fonte: Própria do autor.

Na etapa 2, a cada ponto de interesse dos candidatos extraídos na etapa anterior, aplica-se um algoritmo para determinar sua localização e escala, de forma a detectar os extremos locais de DoG, $D(x, y, \sigma)$. O ponto (x_0,y_0) será um máximo relativo ou um mínimo relativo, se é maior ou menor aos seus 8 pontos vizinhos dentro de seu nível e aos seus 9 pontos vizinhos de cada um dos níveis inferior, Figura 8.

Figura 8 - Comparação do pixel com a vizinhança



Fonte: Adaptado de Lowe (2004)

A escolha recai somente os pontos mais estáveis (pontos extremos), cuja estabilidade é mantida, eliminando os pontos com baixa resolução e maior sensibilidade ao ruído, por meio de um limiar e a matriz Hessiana, *H*, Equação (8),

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad (8)$$

Seja H de $D(x, y, \sigma)$ avaliada em um ponto determinado por (x_0,y_0,σ_0), com seus valores próprios, $\alpha = D_{xx}$ e $\beta = D_{yy}$, se, um valor é maior e o outro é menor, tem-se um ponto em uma borda e não em um canto, portanto, é necessário definir uma condição para tirar esse ponto. A condição depende do traço, *Tr* e do determinante da matriz *H* (HARRIS; STEPHENS, 1998), como definido na Equação (9),

$$Tr(H) = D_{xx}+D_{yy} = \alpha + \beta \quad , \quad det(H) = D_{xx}D_{yy} - D_{xy}^2 = \alpha\beta \quad (9)$$

onde, se *det* < 0 , o ponto é descartado (não é considerado um extremo).

Sabendo que $\alpha = r\beta$, a equação anterior pode ser reescrita pela Equação (10),

$$\frac{[Tr(H)]^2}{det(H)} < \frac{(r+1)^2}{r} \quad (10)$$

onde , *r*=10 é um limiar proposto por Lowe (2004) para eliminar esses pontos.

Na etapa 3, para cada imagem *I(x, y)* em 2D e para uma determinada escala, calcula-se a magnitude do gradiente *m(x, y)* e sua orientação *θ(x, y)*, fazendo a diferença entre os pixels, conforme Equações (11) e (12), respectivamente,

$$.m(x,y) = \sqrt{(I(x+1,y) - I(x-1,y))^2+(I(x,y+1) - I(x,y-1))^2} \quad (11)$$

$$\theta(x,y) = \tan^{-1}\frac{I(x,y+1) - I(x,y-1)}{I(x+1,y) - I(x-1,y)} \quad (12)$$

Descritor: os descritores devem fornecer uma base para a combinação entre as imagens (*matching*). Deve ser invariante a mudança de iluminação ou ponto de vista 3D. O SIFT descritor baseia-se em um histograma de gradientes orientados locais (HoG) na vizinhança dos pontos de interesse. Nas Figuras 9 (a) e (b) mostra-se o resultado da aplicação do método SIFT para uma cena.

Figura 9 - Aplicando o SIFT. a) Imagem original. b) Imagem com *keypoints*.

(a) (b)

Fonte: Própria do autor.

4.1.5 Caracterizador SURF

Speeded-Up Robust Features **(SURF)** é um detector/descritor que pode ser utilizado para reconstrução de ambientes, desenvolvido por Bay, Tuytelaars, Gool (2008). Está baseado no SIFT, apesar da versão padrão do SURF ser mais rápida e mais robusta que o SIFT. Mediante as diferentes transformações de imagem, detecta pontos de interesse invariantes à iluminação, rotação e escala, ambos usam histogramas de gradiente local.

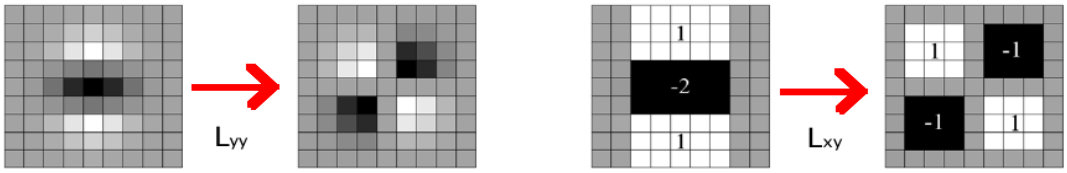
Inicialmente, utiliza-se o detector *Fast Hessian* (HAN et al., 2007), para extrair os pontos de interesse, depois encontra-se a orientação do ponto baseado em uma região circular em torno dele mesmo. Em seguida, se constroi uma região quadrada alinhada com a orientação e se extrai o descritor.

O SURF usa uma aproximação da matriz Hessiana para reduzir o tempo de processamento. Portanto, seja um ponto $X=(x, y)$ e uma imagem $I(x,y)$, a matriz Hessiana $H(X,\sigma)$ em X, com uma escala σ é definida pela Equação (13),

$$H = \begin{bmatrix} L_{xx}(X,\sigma) & L_{xy}(X,\sigma) \\ L_{xy}(X,\sigma) & L_{yy}(X,\sigma) \end{bmatrix} \quad (13)$$

onde, $L_{xx}(X,\sigma)$ é a convolução da derivada parcial de ordem dois da gaussiana $\frac{\delta^2}{\delta x^2}\, g(\sigma)$ com a imagem *I(x,y)*, no ponto *x* , na direção *xx*. De forma similar, para $L_{xy}(X,\sigma)$ e $L_{yy}(X,\sigma)$ nas direções *xy* e *yy* , respectivamente, como mostrado na Figura 10.

Figura 10 - Derivadas parciais gaussianas de segunda ordem



Fonte: Adaptado de Bay, Tuytelaars, Gool (2008)

No cálculo do determinante obtêm-se três aproximações D_{xx}, D_{xy} *e* D_{yy}, onde o determinante da Hessiana indica a localização dos pontos e sua escala a Hessiana é calculada como mostrado na Equação (14),

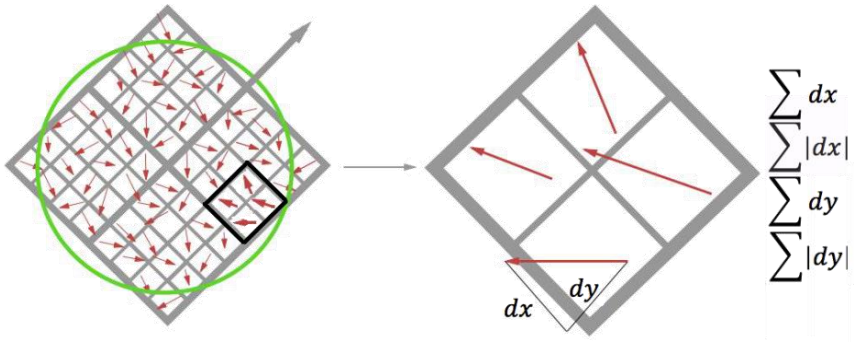
$$\det(H_a prox) = D_{xx}D_{yy} - (0.9D_{xy})^2 \quad (14)$$

Depois de conhecida a escala, se calcula a orientação do ponto de interesse. Para obter um ponto invariante à orientação, calcula-se o *Harr-Wavelet* (BAY; TUYTELAARS; GOOL, 2008) para as direções *x* e *y* em uma região circular *d*, raio 6s, onde s é a escala do ponto. Em seguida calcula-se a orientação dominante somando todos os resultados dentro de uma janela deslizante de 60 graus.

Calcula-se o descritor construindo uma região quadrada em torno do ponto com um tamanho de 20s, conforme mostrado na Figura 11. Essa região é dividida em 4 sub-regiões, calcula-se *Harr-Wavelet* novamente para *x* e *y*, obtendo *dx* e *dy*, assim cada sub-região fornece un vetor, *v*, dado na Equação (15),

$$v = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right). \quad (15)$$

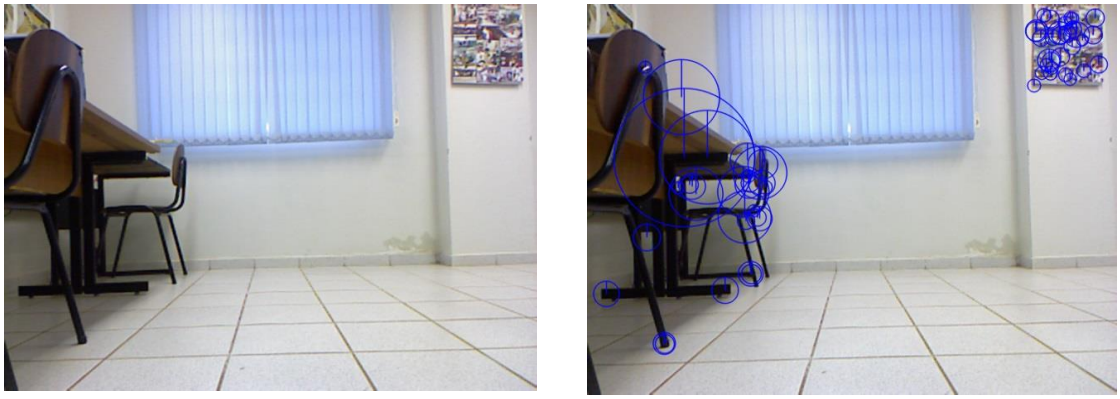
Figura 11- Orientação do ponto de interesse



Fonte: Adaptado de Bay, Tuytelaars e Gool (2008)

Nas Figuras 12 (a) e (b) mostram-se a extração de características com SURF.

Figura 12 - a) Imagem real. b) Características da imagem com SURF.



(a) (b)

Fonte: Própria do autor.

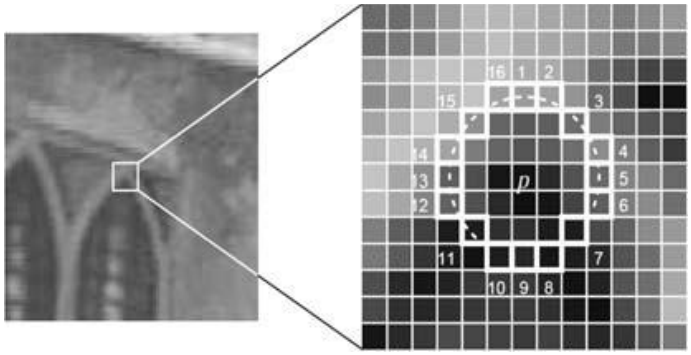
4.1.6 Caracterizador FAST

Features from Accelerated Segment Test **(FAST)** é somente um detector de características (canto), o qual pode ser usado para extrair pontos característicos para o rastreamento de objetos ou mapear um ambiente em muitas tarefas de visão computacional. Foi desenvolvido por Rosten e Drummond (2006) e tem como vantagem a eficiência computacional, por isso o nome. É mais rápido que o DoG usado no SIFT e outros, e adequado para processar vídeo em tempo real.

O FAST usa círculos de 16 pixels para classificar se um ponto candidato *p* é realmente um canto (ou esquina) e cada pixel no círculo é nomeado de um número inteiro de 1 a 16 em ordem crescente. Se um conjunto de *n* pixels contínuo no círculo é mais brilhante do que a intensidade do pixel candidato *p* (denotado por I_p) mais um limiar de valor *t*, ($Ip + t$) ou todos são mais escuros que a intensidade do pixel candidato

p menos o limiar *t*, (*Ip* – *t*), então *p* é classificado como canto ou ainda *p,* deve estar na faixa de , *Ip* – *t* < *p* < *Ip* + *t*. Este processo é ilustrado na Figura 13.

Figura 13 - Deteção de características em uma zona da imagem pelo FAST



Fonte: Rosten e Drummond (2005).

Na figura mostra-se o conjunto de imagens com a aplicação do detector FAST, onde os quadrados mais brilhantes (dentro do limiar definido) são usados na detecção. O pixel *p* é o centro da detecção e as linhas tracejadas passam através de 12 pixels contínuos que são mais brilhantes do que *p,* portanto são excluídos dos pontos considerados como pontos de interesse, a avaliação é feita em torno dos pontos cardeais, 1, 5, 9, 13. Este detector apresenta algumas desvantagens quanto, por exemplo, a avaliação não se generaliza bem para n < 12, quando múltiplas características são detectadas. Estas correções foram solucionadas por Rosten e Drummond (2006), de forma que foram propostos outros detectores baseados no FAST.

4.1.7 Caracterizador *BRIEF*

Binary Robust Independent Elementary Features **(BRIEF)** é somente um descritor de pontos característicos, sendo usado na maioria das aplicações de visão computacional para caracterização de uma imagem, por apresentar processamento rápido e binário. Não varia com a mudança de escala e rotação, a menos que utilizado juntamente com um detector capaz de dar-lhe esta propriedade.

Proposto por Calonder et al (2010), BRIEF é um descritor de palavra binária que consegue um alto grau de discriminação utilizando testes simples de diferença de intensidades no seu cálculo, considerado rápido para a construção de mapas e correspondências, avaliado normalmente pela distância de *Hamming* que é mais eficiente. Comparando com o SURF e suas variações, nota-se um desempenho similar

no reconhecimento de pontos de interesse, embora apresente o mesmo comportamento em termos de tempo de processamento.

Matematicamente, Calonder et al (2010) definiu para uma imagem, um teste binário , τ , na região (ou zona) *r*, de tamanho, *SxS* , em torno de um ponto de interesse, *p*, como sendo , Equação (16),

$$\tau(r; x, y) := \begin{cases} 1 & se\ r(x) < r(y) \\ 0 & em\ outro\ caso \end{cases} \quad (16)$$

onde *r(x)* é a intensidade do pixel em uma versão suavizada de *r* , em $x = (u, v)^T$.

A escolha exclusiva de um conjunto de pares de localização $n(x, y)$, define um conjunto de testes binários, Equação (17),

$$f_n(r) = \sum_{1 \le i \le n} 2^{i-1} \tau\,(r; x_i, y_i) \quad (17)$$

BRIEF portanto, utiliza uma região de imagem suavizada para um conjunto de pares de localização, comparando a intensidade do pixel nestes locais, cujo o resultado é 1 ou 0. Ainda segundo Calonder et al. (2010), experimentos mostram que apenas com a resolução do vetor de 256 bits (de tamanho) ou até mesmo 128 bits, muitas vezes são suficientes para obter resultados eficientes de correspondência.

4.1.8 Caracterizador ORB

Oriented FAST and Rotated BRIEF (ORB) ou FAST orientado e BRIEF rotacionado é um detector /descritor construído a partir do detector de pontos FAST e o descritor BRIEF. ORB é um descritor binário muito rápido baseado no BRIEF, invariante a rotação e resistente ao ruído gaussiano da imagem, (mais rápido que o SIFT). Ambas às técnicas (FAST e BRIEF *rotacionado*) apresentam bom desempenho e baixo custo computacional. Foi proposto por Rublee et al (2011) que implementou algumas contribuições no FAST e BRIEF:

- Adiciona um rápido e preciso componente de orientação para o *FAST*;
- Eficiência computacional orientada e análise de variância e correlação para o BRIEF;
- Método de aprendizagem para de–correlação para o BRIEF sob invariância rotacional fornecendo, melhor desempenho em aplicações de vizinhos mais próximos;

As modificações propostas, primeiramente, são feitas para o FAST:

- Para fornecer o componente de orientação emprega-se um medidor de canto, Harris, ordenando os pontos de interesse no FAST;
- FAST não produz características de multi-escala, por isso, emprega-se uma escala piramidal da imagem, para produzir um FAST (filtrado por Harris) para cada nível na pirâmide;
- A orientação do FAST é dada pela medida de orientação das esquinas, o centroide de intensidade assume que a intensidade de esquina é deslocada do centro e este vetor pode ser usado para calcular a orientação. O método define o momento da região, m_{pq} , pela Equação (18),

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{18}$$

Com estes momentos encontra-se o centroide, *C*, Equação (19),

$$C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}) \tag{19}$$

Assim é construido um vetor de centro de esquina, O, para o centroide, $\overrightarrow{OC}$. Finalmente, a orientação da região , θ é dada pela Equação (20),

$$\theta = \tan^{-1} \frac{m_{01}}{m_{10}} \tag{20}$$

Reescrevendo a equação (20) em função da imagem, *I*, tem-se a orientação da região, θ, pela Equação (21),

$$\theta = \tan^{-1} \frac{\sum_{x,y} yI(x,y)}{\sum_{x,y} xI(x,y)} \tag{21}$$

No caso do BRIEF tem-se uma modificação de acordo com a orientação dos pontos de interesse, denominado BRIEF dirigido (em inglês, *steer* BRIEF). Para qualquer conjunto de características na localização, (x_i, y_i), define-se *S* como sendo a matriz 2xn, para *n* testes binários , Equação (22),

$$S = \begin{pmatrix} x_1, \dots \dots \dots, x_n \\ y_1, \dots \dots \dots, y_n \end{pmatrix} \tag{22}$$

Utilizando a orientação da região, θ e a correspondente matriz de rotação, R_θ, tem-se uma versão dirigida, S_θ, de S , dada pela Equação (23),

$$S_\theta = R_\theta \mathrm{S} \tag{23}$$

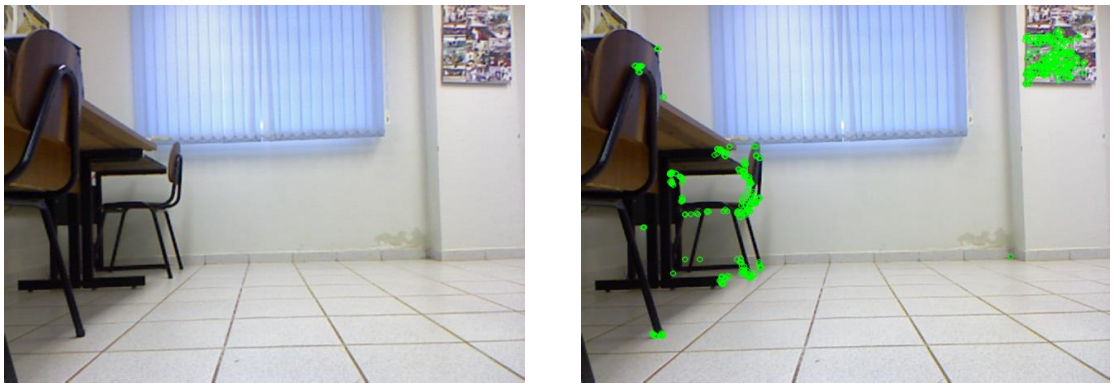
O operador BRIEF dirigido, $g_n(r, \theta)$ se converte na Equação (24),

$$g_n(r, \theta) = f_{nd}(r) | (x_i y_i) \epsilon S_\theta \tag{24}$$

onde r é a região da imagem suavizada para n testes binários.

O ORB é mais rápido que o SIFT duas ordens de grandeza de acordo com inúmeros testes realizados em aplicações do mundo real , incluindo deteção de objetos. É considerado como uma busca gananciosa para um conjunto de testes binários, que atende aos requisitos de ter a maior variação e a menor correlação. O principal interesse neste detector é o seu bom desempenho. Nas Figuras 14 (a) e (b), mostram-se a detecção e a descrição dos pontos de interesse de uma imagem em ORB.

Figura 14 - Caracterizador ORB. a) Imagem original. b) Com ORB



(a) (b)

Fonte: Própria do autor.

4.2 Comentários

Neste capítulo foram apresentados alguns dos principais detectores/descritores usados para o tratamento da imagem em visão computacional. Na detecção identifica-se uma área de interesse que contém uma variação de intensidade e luminosidade, e o descritor caracteriza a região da imagem onde está localizado o ponto de interesse, formando um vetor de características para a associação dos pontos com seus vizinhos. Dentre estes o ORB é um dos mais utilizados atualmente, pelo seu melhor desempenho no custo computacional e de armazenamento e ter formatação binária.

5 CORRESPONDÊNCIAS ENTRE PONTOS NAS IMAGENS

Depois de computados os pontos de interesse pelos detectores/descritores por meio destes, deve-se encontrar as correspondências entre imagens ou a estimação da transformação 3D. Neste capítulo apresentam-se alguns métodos de correspondência (*matching*) entre estes pontos de interesse, mais usados nas pesquisas nesta área, o RANSAC e o mais recente PROSAC, considerados robustos para eliminação de dados atípicos (correspondências incorretas entre pontos 3D). Apresentam–se também um algoritmo iterativo para refinar a estimação da transformação rígida entre correspondências, o ICP, método muito aplicado na robótica móvel.

5.1 Correspondência *Inicial* entre *keypoints*

Antes da estimação da transformação 3D tem-se a tarefa de comparação inicial e união entre pontos, conhecidas como correspondências, do inglês *matching* inicial, entre um par de frames, e está baseada em métodos de comparação ponto a ponto das características dadas pelos detectores/descritores em 2D. Um método normalmente usado é o ***Brute-Force-Matcher* (BFM).**

O BFM está baseado no método da busca do vizinho mais próximo usando a distância Euclidiana sobre os valores dos pontos de interesse. Para cada frame alvo, dois vizinhos são buscados no frame fonte, se estes vizinhos são próximos o suficiente, com respeito a um limiar fixado, o resultado é considerado válido- *inliers*, do contrário, considera-se *outliers*. Esta busca inicial é feita em 2D e para melhorar a correspondência quando a razão entre distâncias de vizinhos mais próximos é maior que o limiar dado, alguns pontos de interesse são rejeitados. Depois disso, a informação de profundidade é usada para manter as características que estão próximas o suficiente da câmera, pois a precisão da informação de profundidade não é considerada confiável acima do intervalo aproximado de 5 a 6 metros (HOGMAN, 2013).

Um exemplo de aplicação do método BFM para ambientes internos com a classificação nas correspondências de pontos *inliers e outliers* para imagens consecutivas é apresentado na Figura 15. No Apêndice B tem-se o pseudocódigo (Algoritmo 1) do estimador BFM.

Figura 15 - Método BFM - classifica correspondências (imagens consecutivas).

Fonte: Própria do autor.

Embora o algoritmo BFM realize uma boa classificação de correspondências entre os conjuntos de imagens, nota-se na figura que o ponto P_1 (na imagem origem) e o ponto Q_1 (na imagem destino), estão bem classificados, representando pontos *inliers*. Enquanto que, os pontos mostrados nos círculos vermelhos, e em P_2 e Q_2 são considerado *outliers* (mal classificados). Mesmos que os pontos tenham a mesma distância, devido à localização geométrica do ponto (translação e rotação), podem apresentar uma classificação errada, estes erros podem ser corrigidos usando métodos de estimação robusta.

5.2 Estimação da transformação 3D

Na correspondência entre pontos de diferentes frames, muitas vezes, pontos localizados em diferentes lugares das imagens, mesmo tendo as mesmas características, podem não ser diferenciados pelo algoritmo de correspondências, tornando o acoplamento entre imagens não confiável. Por isso, são utilizados algoritmos de estimação robustos ou de transformação 3D (translação e a rotação do sensor entre um frame origem e outro destino) para a eliminação de dados atípicos (*outliers*) *ou* também chamados falsos positivos - medições que tem erros tão grandes, considerados com baixa validade. Entre os algoritmos mais aplicados estão:

- RANSAC e o PROSAC - correspondências de características visuais e aproximação a pose;
- ICP – refinamento da pose.

O enfoque de aproximação visual a pose com RANSAC e refinamento com ICP, tem sido bastante eficaz na construção do mapa de interiores, por isso, muito usado por pesquisadores (HARTLEY, ZISSERMAN, 2004; FISCHLER, BOLLES; 1981).

5.3 RANSAC

Random Sample Consensus **(RANSAC)** é um dos algoritmos mais populares de estimação robusta que visa eliminar ruídos nos dados inerentes aos sensores. Em outras palavras, é um algoritmo iterativo utilizado para adaptar os parâmetros de um modelo matemático aos dados experimentais. Este método foi apresentado nos anos oitenta por Fischler e Bolles (1981) visando automatizar a análise de imagens e é usado em visão computacional quando um conjunto de dados (nuvem de pontos) contém uma alta porcentagem de valores atípicos.

É importante contar com um método que permita a eliminação de dados atípicos, uma vez que se considera que sempre haverá correspondências incorretas entre pontos 3D, gerando um problema no momento de estimar a transformação entre as nuvens de pontos e criar mapas consistentes.

Dado um conjunto de medidas, *M* e um modelo matemático que tenha $n_{m,}\, m \in \{1, \cdots, M\}_{,}$ parâmetros livres que possam ser estimados, o número de medições em *M* tem que ser maior que n_m e sejam S *e* T_p os subconjuntos de variáveis diferentes de *M*, o pseudocódigo do RANSAC (Algoritmo 2) no Apêndice B, funciona da seguinte maneira:

- Seleciona de modo aleatório um subconjunto, S das medições em *P* - com *S* tem-se a primeira estimação dos n_m parâmetros livres;
- Usa a estimação atual do modelo para selecionar um novo subconjunto de pontos T_p, considerando as medições que estão dentro do limiar de erro do modelo;
- Se $T\mathrm{p} > Limiar$, volta a estimar n_m de acordo com o novo subgrupo. Calcula e armazena um valor de coincidência entre T_p e o modelo, e escolhe um novo subconjunto S;
- Se T$p < Limiar$ (com relação às medições) escolhe um novo subconjunto S de M e inicia novamente o algoritmo;
- Se nenhum dos subconjuntos tem mais medições que o limiar, termina o algoritmo com fracasso, ou se atinge o número máximo de iterações.

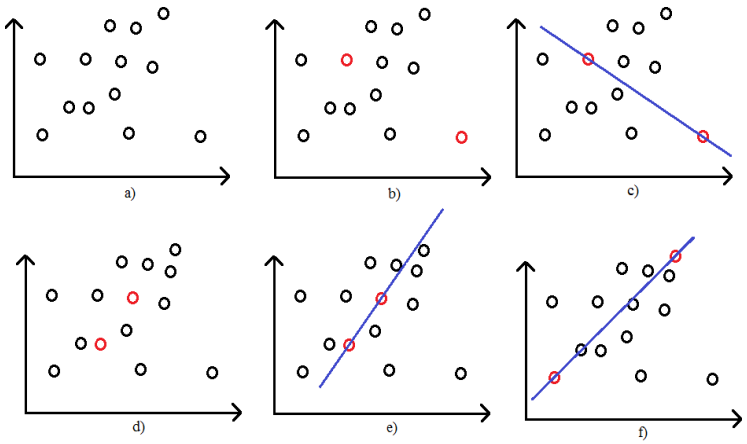
O algoritmo do RANSAC apresenta parâmetros que devem ser ajustados experimentalmente, para cada aplicação, a *Tolerância do erro* que determina quando os dados do modelo não são parte do modelo; o *Número máximo de iterações* e o *Número mínimo de medições*.

O RANSAC trata os valores atípicos de um conjunto de dados de uma forma robusta, porém não garante alguma solução e nem que a solução encontrada é ótima.

No caso da estimação da transformação rígida, o modelo que é estimado é a matriz de Homografia de 4x4, que contém o vetor de translação e a matriz de rotação que melhor explica a translação e a rotação das correspondências de pontos 3D, equivalentes às correspondências em 2D - no mínimo três pontos são necessários para estimar os parâmetros da transformação. Depois de estimar o modelo em cada iteração, são considerados inliers, os correspondentes de pontos 3D em que a distância entre os pontos de interesse, seja menor que o limiar determinado.

Para ilustrar o algoritmo RANSAC apresenta-se na Figura 16 a estimativa de um modelo de uma linha.

Figura 16 - Aplicando o RANSAC. a) Conjunto de pontos. b) Dois pontos aleatórios. c) Primeiro modelo estimado. d) Novos pontos - nova estimação. e) Novo modelo estimado. f) Melhor modelo estimado.



Fonte: Própria do autor

5.4 ICP

***Iterative Closest Point* (ICP)** é um dos métodos mais intuitivo e foi introduzido por Chen (1992 ou 1991). Faz o refinamento da pose aplicando uma transformação às

nuvens de pontos captadas em tempos diferentes, para que esteja o mais próximo quanto possível das outras.

Sejam duas nuvens de pontos, onde um conjunto é chamado *pontos do modelo*, $Q = q_1, q_2, q_3, \dots\dots, q_N$ e o outro conjunto é denominado *pontos de dados*, $P = p_1, p_2, p_3, \dots\dots, p_N$, aplica-se uma transformação a *P* com o propósito que estes tenham um melhor afastamento com relação a *Q*. Matematicamente, o ajuste ou métrica de erro pode ser dado pela função objetivo *E*, Equação (25),

$$E = \sum_{i=1}^{N} \|\tau(p_i) - q_i\|^2 \tag{25}$$

Esta equação é a soma dos erros quadráticos entre as distâncias ao quadrado de p_i , aos seus vizinhos mais próximos, q_i, depois de aplicada a transformação, τ , cuja transformação é feita de acordo com o problema (BELS; MACKAY, 1992).

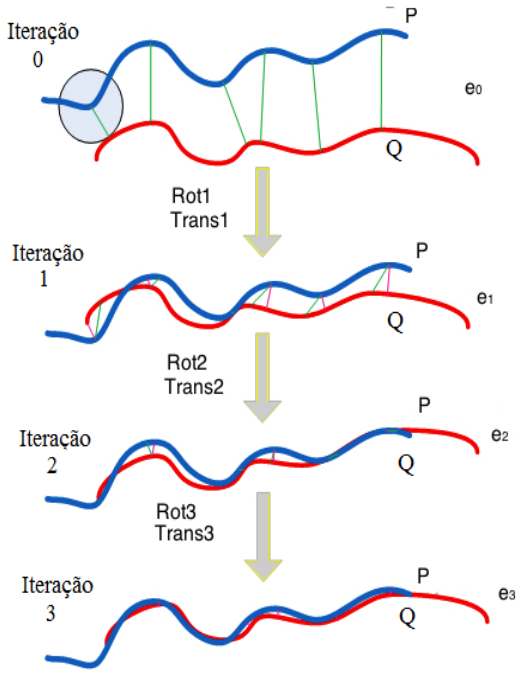
No espaço 3D, onde a transformação tem 6 DoF, a função objetivo, *E* se converte em uma função de 6 variáveis. Dada a matriz de rotação, *R* e o vetor de translação, *T*, *E* é reescrito pela Equação (26),

$$E = \sum_{i=1}^{N} \left\| \left(R(p_i) + \vec{T}\right) - q_i \right\|^2 \tag{26}$$

Como não se conhece qual ponto, q_i será o vizinho mas próximo a p_i , depois da transformação, esta informação é aproximada pela busca do vizinho mais próximo. Os métodos de minimização não lineares apresentam alguma transformação para determinar a redução de *E*. O pseudocódigo padrão do ICP, Algoritmo 3, é apresentado no Apêndice B. Na Figura 17 mostra-se um exemplo do refinamento com o ICP que é resumido a seguir:

- Determinação das correspondências entre as nuvens de pontos;
- Minimização das distâncias entre as correspondências;
- Transformação estimada dos *pontos de dados* para aproximar aos *pontos* do modelo.

Figura 17 - Exemplo do refinamento pelo ICP em 4 iterações



Fonte: Adaptado de Zang (2006)

5.5 PROSAC

***PROgressive SAmple Consensu* (PROSAC)** é uma abordagem mais recente para encontrar as correspondências entre duas ou mais imagens, proposto pelo Chum e Matas (2005). Este algoritmo usa uma função de similaridade para estabelecer tentativas de correspondências ordenando-as de forma linear. As amostras do PROSAC são organizadas de forma progressiva, à medida que as correspondências são classificadas, com isso reduz o custo computacional (mais do que 100 vezes o do RANSAC).

O objetivo de PROSAC é encontrar os *inliers* no conjunto de todas as tentativas de correspondências, no menor tempo possível, para garantir com determinada probabilidade, que todos os *inliers* do conjunto sejam encontrados.

A estrutura do algoritmo PROSAC é similar ao RANSAC embora este trate todas as correspondências igualmente e de forma aleatória. Primeiro, são geradas hipóteses por amostragem aleatória. Diferente do RANSAC, as amostras são extraídas de subconjuntos dos dados com alta qualidade. O tamanho do conjunto de geração de hipóteses cresce gradualmente, e as amostras prováveis a serem *inliers* são examinadas mais rápidas. Então o PROSAC é projetado para extrair as mesmas amostras que o RANSAC, mas de forma ordenada. Aspectos importantes do algoritmo é a escolha do

tamanho do conjunto na geração das hipóteses e o critério de parada do processo de amostragem.

Este algoritmo, por ser relativamente novo tem sido pouco utilizado para correspondência de imagens. Na literatura encontram-se somente os trabalhos aplicados de Chum e Matas (2005) e Jeon, Jeong e Lee (2015), cujos resultados são promissores em relação a qualidade do resultado de correspondências e o menor tempo de execução na construção do mapa. O pseudocódigo do PROSAC, algoritmo 4, é apresentado no Apêndice B.

5.6 Comentários

Apresentaram-se neste capítulo os métodos que são usados na busca por características para identificar correspondências de candidatos para associá-los em pares entre duas fontes, tais como o RANSAC, PROSAC e o ICP. Estes métodos fornecem boas respostas frente às possíveis variações de luminosidade, rotação, translação e escala dos pontos característicos que definem as imagens. O interesse deste trabalho recai sobre o algoritmo de correspondência PROSAC visando a eliminação dos dados atípicos e a construção do mapa 3D mais rapidamente.

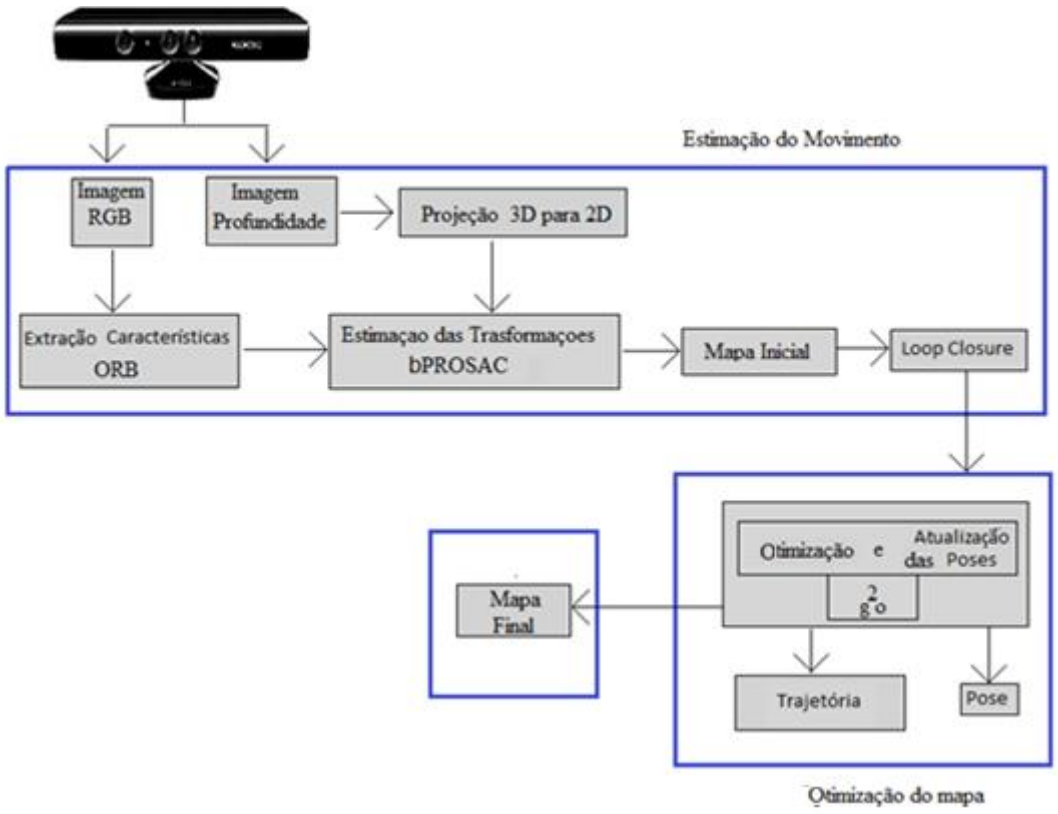
6 SISTEMA PROPOSTO

Depois ter sido explanado até o momento, cada uma das partes do tratamento da imagem para o mapeamento, neste capítulo apresenta-se a proposta para a solução do problema do VSLAM monocular com a construção de um mapa 3D de um ambiente indoor. A proposta está baseada na resposta da estimação de parâmetros de correspondências entre frames consecutivos usando o PROSAC.

6.1 Proposta do VSLAM em Diagrama de blocos.

A localização e geração de um mapa tridimensional de um robô, como parte do planejamento para navegar de forma autónoma em ambientes interiores, pode ser solucionada com o tratamento das características das imagens fornecidas por sensores visuais, neste caso a câmera RGB-D. Por métodos matemáticos probabilísticos realiza-se o algoritmo do VSLAM — diagrama de blocos, Figura 18.

Figura 18 - Diagrama de blocos do algoritmo VSLAM



Fonte: Própria do autor

A técnica consiste das etapas de estimação do movimento, otimização do mapa e a representação do mapa final, compostos pelas seguintes sub-tarefas:

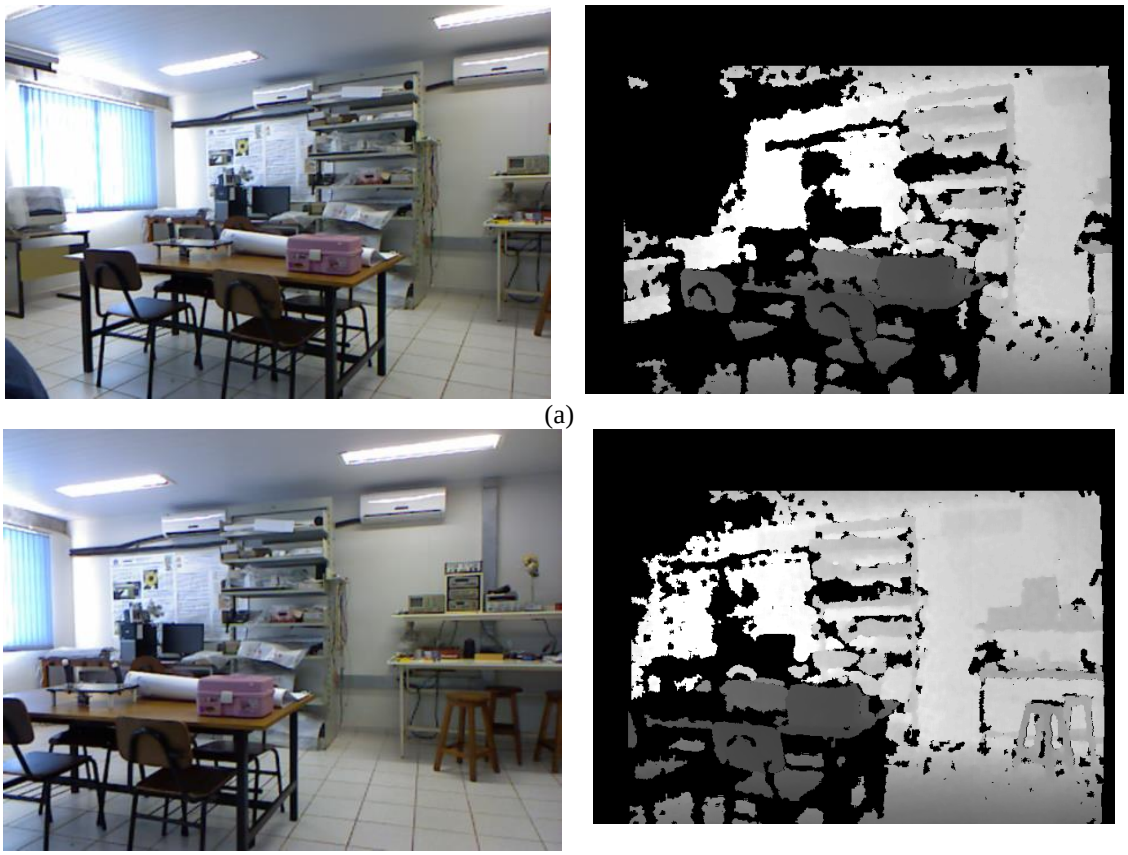
- Captura e armazenamento das Imagens RGB e profundidade (*Depth*) do ambiente por meio do sensor Kinect;
- Cálculo e detecção dos *keypoints usando* ORB - extração de características relevantes;
- Acha as correspondências iniciais usando BFM;
- Transformação dos pontos 3D para 2D;
- Estimação das transformações (matriz de Homografia) pelo PROSAC;
- Eliminação de frames já conhecidos usando o *loop closure*;
- Otimização do mapa construído e a determinação das poses;
- Mapa final.

Inicia-se a descrição dos algoritmos e o seu desenvolvimento matemático necessário para o entendimento teórico da proposta.

6.2 Captura da Imagem

A Microsoft câmera Kinect fornece a imagem colorida e a profundidade com taxa de amostragem de 30fps. Nesta câmera as informações das imagens são capturadas com *strings* de 8 bits em cada canal, R,G e B (24bits no total) com uma resolução de 640x480 pixels, e as informações fornecidas pelo sensor de profundidade compõe uma matriz de resolução de 11 bits em cada pixel. Nas Figuras 19 (a) e (b) são apresentadas 2 imagens consecutivas e a sua profundidade.

Figura 19 - Sequência de imagem RGB-D de entrada. a) Imagem 1 e a sua profundidade. b) Imagem 2 e a sua profundidade.



(a)

(b)

Fonte: Própria do autor.

6.3 Detecção/descrição dos pontos característicos das imagens

Nesta etapa faz-se a detecção dos frames (imagens e profundidade) para encontrar os *keypoints* e a descrição para a orientação desses pontos. Os dois procedimentos são utilizados para encontrar os pontos característicos nas imagens em 2D . O detector usado é o ORB (oFast) e o descritor o ORB (rBrief), conforme teoria visto anteriormente. O ORB tem um melhor desempenho, tanto à imunidade ao ruído gaussiano, quanto a velocidade de processamento.

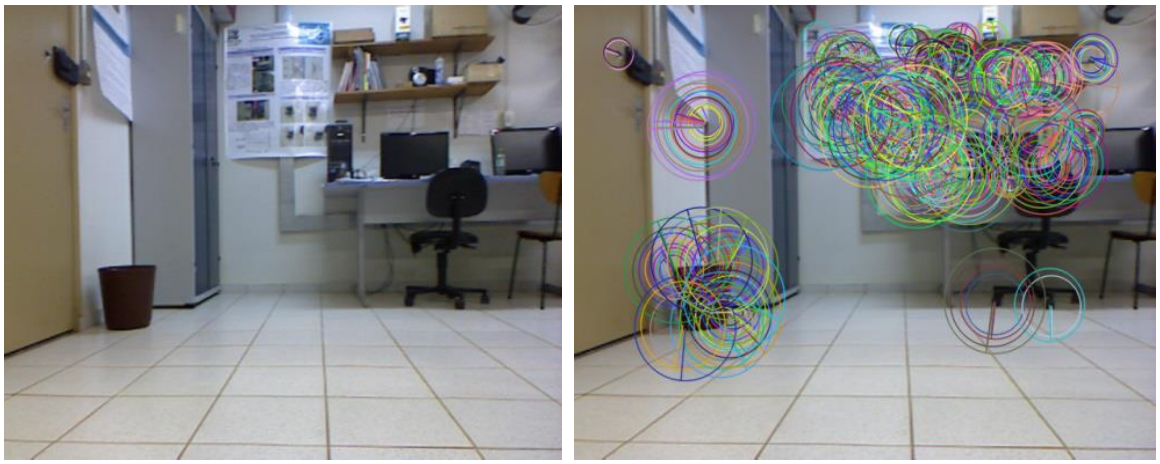
O algoritmo do ORB deve operar com restrições nos parâmetros da distância entre os pixels vizinhos e o limiar de pixel baixo, reduzindo a quantidade de memória e com respostas mais rápidas. O algoritmo que resolve o ORB é baseado nas equações (19) a (24), vistas anteriormente com suas etapas apresentadas a seguir:

- Identifica os *Keypoints* por meio do algoritmo FAST;
- Identifica os melhores pontos por meio do algoritmo de identificação de cantos Harris (pontos com direções diferentes);

- Identifica os keypoints em diferentes escalas usando a pirâmide gaussiana;
- Identifica a orientação em relação ao centróide com BRIEF (fornece informação binária entre pares de pixels);
- Reafirma a orientação identificada pelo FAST;
- Rotaciona a matriz de descritores conforme o ângulo da orientação da direção dominante do ponto.

O resultado da aplicação do algoritmo ORB à sequência de imagens do ambiente do robô é mostrado nas Figuras 20 (a)imagem 3 e em (b) a Imagem 4 com os *keypoints* (círculos) nas diferentes escalas (raios diferentes) e a sua orientação (identificada pela linhas no interior de cada círculo). Na Tabela 2 apresentam-se alguns parâmetros de aplicação do ORB.

Figura 20 - Imagens consecutivas. a) Imagem 3. b) Imagem 4.



(a) (b)

Fonte: Própria do autor.

Tabela 2 - Parâmetros do ORB.

Quantidade de pontos característicos	485
Classificação	108 correspondências
Tempo de execução	706 microssegundos

6.4 *Matching* Inicial

Nesta etapa combinam-se os pares de imagens obtidas da extração de características visando classificar os pontos que tenham distâncias similares e rejeitar aqueles que tenham distâncias diferentes dos pontos de referência (origem) usando o

BFM. O método BFM apresenta invariância à rotação, escala e pode encontrar padrões frequentes em séries temporais com grande eficiência.

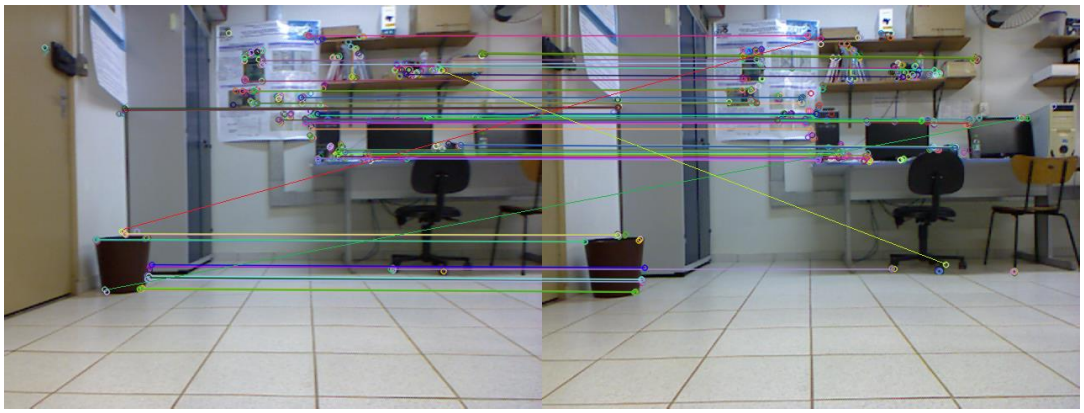
Os resultados da aplicação do algoritmo BFM em duas amostras do conjunto de imagens, Figuras 21 (a) imagem destino e em (b) imagem origem está na Figura 21 (c). Demonstram-se com este resultado as correspondências mal classificadas (círculo em vermelho) devido a dois fatores: os pontos na imagem origem apresentam valores semelhantes, mas em regiões diferentes na imagem destino ou a erros ocasionados pelas grandes distâncias que não foram descartadas pelo BFM. Os erros de falsa classificação podem ser corrigidos pela variação do limiar para classificar o *keypoints*, mostrado na Figura 21 (d), usando limiar de até 4 vezes(4x) o valor da distância, critério determinado empiricamente.

Figura 21 - a) Imagem destino. b) Imagem origem. c) Correspondências BFM. d) Correspondências aplicando o limiar de 4x.

(a)

(b)

(c)



(d)

Fonte: Própria do autor.

Na Figura 21(d) houve uma redução dos pontos de coincidência (imagem mais limpa), porém embora as correspondências entre os pontos estejam corretas, podem apresentar algum erro aleatório de distribuição gaussiana devido aos parâmetros intrínsecos da câmera ou falsos positivos (erros de similaridade na medição das distâncias). Nestes casos é necessário separar os pontos que se ajustam ao modelo da transformação (*inliers*) dos incorretos (*outliers*) usando métodos de estimação como o PROSAC.

6.5 A matriz de Homografia

A matriz de Homografia relaciona os pontos de uma superfície plana projetado sobre duas imagens, usada pelo PROSAC para reduzir o número de falsos positivos. É obtida sobre o conjunto de características X em uma imagem I_k, podendo-se detectar o descolamento de um frame I_k para outro I_{k+1}, utilizando a deformação das características, as quais podem ser modeladas como um deslocamento e uma transformação *affine*, resultando a matriz $X´$que contém as novas posições das características da imagem. Portanto, a transformação homográfica entre os pontos $\{X \leftrightarrow X´\}$ com $X = (x, y, w)$ e $X´ = (x´, y´, w´)$ é definido pela Equaçao (26),

$$\boldsymbol{X´ = HX} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix} \tag{26}$$

sendo, $\boldsymbol{H} = H(x)$ a matriz de homografia.

Visando encontrar os elementos, h_{ij}, da matriz H, manipula-se algebricamente a equaçao anterior, obtendo-se a Equação (27),

$$\boldsymbol{HX} = \begin{pmatrix} h_a\boldsymbol{X} \\ h_b\boldsymbol{X} \\ h_c\boldsymbol{X} \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} \tag{27}$$

onde , h_a, h_b, h_c são as linhas da matriz H.

Como $X\ e\ X'$ sao vetores homogêneos (têm a mesma direção), o produto vetorial entre $X' e$ HX é dado pela Equação (28),

$$\boldsymbol{X}' \times \boldsymbol{HX} = \begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} \times \begin{pmatrix} h_a\boldsymbol{X} \\ h_b\boldsymbol{X} \\ h_c\boldsymbol{X} \end{pmatrix} = \begin{pmatrix} y' h_c\boldsymbol{X} - w' h_b\boldsymbol{X} \\ w' h_a\boldsymbol{X} - x' h_c\boldsymbol{X} \\ x' h_b\boldsymbol{X} - y' h_a\boldsymbol{X} \end{pmatrix} = 0 \tag{28}$$

Substituindo na equação anterior os vetores h_a, h_b, h_c pelos seus respectivos elementos, tem-se a Expressão (29),

$$\begin{pmatrix} y'[h_{31}h_{32}h_{33}]\boldsymbol{X} - w'[h_{21}h_{22}h_{23}]\boldsymbol{X} \\ w'[h_{11}h_{12}h_{13}]\boldsymbol{X} - x'[h_{31}h_{32}h_{33}]\boldsymbol{X} \\ x'[h_{21}h_{22}h_{23}]\boldsymbol{X} - y'[h_{11}h_{12}h_{13}]\boldsymbol{X} \end{pmatrix} = 0 \tag{29}$$

Manipulando algebricamente a expressão anterior obtém-se a Expressão (30),

$$\begin{bmatrix} 0+0+0-w' h_{21}x - w' h_{22}y - w' h_{23}w + y' h_{31}x + y' h_{32}y + y' h_{33}w \\ w' h_{11}x + w' h_{12}y + w' h_{13}w + 0 + 0 + 0 - x' h_{31}x + x' h_{32}\, y + x' h_{33}w \\ -y' h_{11}x - y' h_{12}y - y' h_{13}w + x' h_{21}x + x' h_{22}y + x' h_{23}w + 0 + 0 + 0 \end{bmatrix} = 0 \tag{30}$$

Esta expressão em termos de matrizes é dada por (31),

$$\begin{bmatrix} 0^T & -w'\boldsymbol{X}^T & y'\boldsymbol{X}^T \\ w'\boldsymbol{X}^T & 0^T & -x'\boldsymbol{X}^T \\ -y'\boldsymbol{X}^T & x'\boldsymbol{X}^T & 0^T \end{bmatrix} \begin{bmatrix} h_a{}^T \\ h_b{}^T \\ h_c{}^T \end{bmatrix} = 0 \tag{31}$$

Como a terceira linha da matriz A_i é uma combinação linear das outras duas, A_i pode ser reescrita como, equação (32),

$$\begin{bmatrix} 0^T & -w'\boldsymbol{X}^T & y'\boldsymbol{X}^T \\ w'\boldsymbol{X}^T & 0^T & -x'\boldsymbol{X}^T \end{bmatrix} = A_i \tag{32}$$

Desta forma, pode-se reescrever a equação (32) em função de A_i, equação (33),

$$\begin{bmatrix} 0 & 0 & 0 & -w'x & -w'y & -w'w & y'x & y'y & y'w \\ w'x & w'y & w'w & 0 & 0 & 0 & -x'x & -x'y & -x'w \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0 \quad (33)$$

Para calcular a matriz de homografia planar são suficientes 4 pares de pontos de correspondências entre as imagens, dado pela equação (34),

$$\begin{bmatrix} 0 & 0 & 0 & -w_1'x_1 & -w_1'y_1 & -w_1'w_1 & y_1'x_1 & y_1'y_1 & y_1'w_1 \\ w_1'x_1 & w_1'y_1 & w_1'w_1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1'w_1 \\ 0 & 0 & 0 & -w_2'x_2 & -w_2'y_2 & -w_2'w_2 & y_2'x_2 & y_2'y_2 & y_2'w_2 \\ w_2'x_2 & w_2'y_2 & w_2'w_2 & 0 & 0 & 0 & -x_2'x_2 & -x_2'y_2 & -x_2'w_2 \\ 0 & 0 & 0 & -w_3'x_3 & -w_3'y_3 & -w_3'w_3 & y_3'x_3 & y_3'y_3 & y_3'w_3 \\ w_3'x_3 & w_3'y_3 & w_3'w_3 & 0 & 0 & 0 & -x_3'x_3 & -x_3'y_3 & -x_3'w_3 \\ 0 & 0 & 0 & -w_4'x_4 & -w_4'y_4 & -w_4'w_4 & y_4'x_4 & y_4'y_4 & y_4'w_4 \\ w_4'x_4 & w_4'y_4 & 4w_4 & 0 & 0 & 0 & -x_4'x_4 & -x_4'y_4 & -x_4'w_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} \quad (34)$$

$$= 0$$

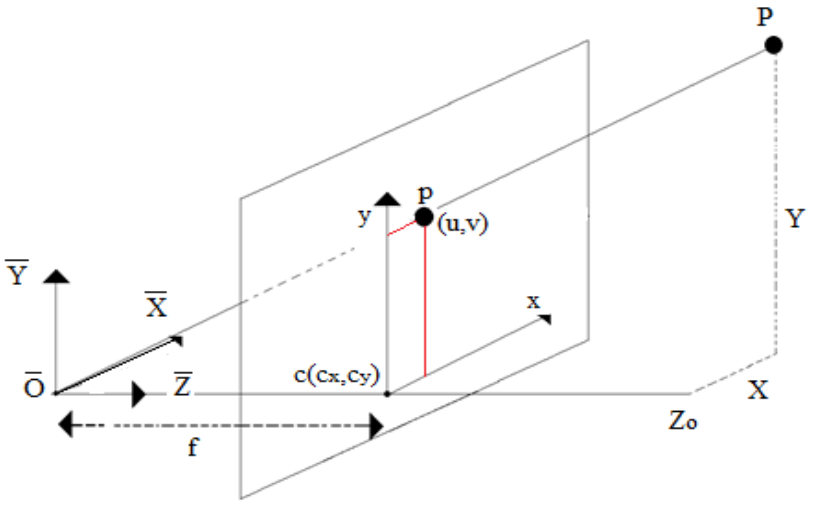
6.6 Conversão de pontos no espaço para o plano

Para implementar o algoritmo PROSAC como base necessária para a construção do mapa, o procedimento inicial é a conversão dos dados do sensor Kinect para obter uma representação no espaço. A conversão dos pontos é baseada na transformação entre planos $\mathbb{P}^3$ para $\mathbb{P}^2$ da geometria projetiva das imagens (HARTLEY; ZISSERMAN, 2004).

Seja um ponto no espaço com coordenadas P [*X, Y, Z*] mostrado na Figura 22, que pode ser representado no plano em termos de um pixel *p [u, v, 1]*, ou em termos da profundidade, *d* do pixel, *p [u, v, d]*, três transformações são necessárias:

1. Representar essas coordenadas no espaço em coordenadas da câmera;
2. Transformar o sistema de coordenadas da câmera para o sistema de coordenadas da imagem;
3. Transformar o sistema de coordenadas da imagem para o sistema de coordenadas do pixel.

Figura 22 - Projeção 3D para 2D (representação de *P* como pixel)



Fonte: Própria do autor.

A representação das coordenadas no espaço em coordenadas da câmera é realizada por meio da matriz definida pela equação (35),

$$\begin{bmatrix} \bar{X} \\ \bar{Y} \\ \bar{Z} \\ \bar{1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{35}$$

sendo *R*, a matriz de rotação do ponto da imagem e *T*, vector de translação do ponto, parâmetros extrínsecos da câmera.

A transformação do sistema de coordenadas da câmera para o sistema de coordenadas da imagem é realizada através de uma projeção perspectiva definida pela equação (36),

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \bar{X} \\ \bar{Y} \\ \bar{Z} \\ \bar{1} \end{bmatrix} \frac{1}{z} \tag{36}$$

onde f é a distância focal.

Na terceira e última transformação utiliza-se a transformação *affine* que leva do sistema de coordenadas da imagem para o sistema de coordenadas do pixel, modelada pela equação (37),

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & c_x \\ 0 & s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{37}$$

sendo, s_x, s_y número de pixels por unidade de comprimento , c_x, c_y a posição, em pixels da projeção ortogonal, *c*, da origem sobre o plano de projeção.

Portanto, a transformação que leva um ponto *P* do sistema de coordenadas do mundo à sua projeção em sistema de coordenadas do pixel *p,* é obtida fazendo a composição das três transformações, equação (38),

$$[p] = \begin{bmatrix} s_x & 0 & c_x \\ 0 & s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \\ 0 & 1 \end{bmatrix} [P] \tag{38}$$

onde, a matriz $\begin{bmatrix} fs_x & 0 & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$ é chamada matriz de calibração.

Geralmente na maior parte dos casos é conveniente expressar a matriz de calibração por parâmetros fornecidos pelos construtores das câmeras (constantes), equação (39),

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 518.0 & 0 & 325.5 \\ 0 & 519 & 253.5 \\ 0 & 0 & 1 \end{bmatrix} \tag{39}$$

Então, a transformação para a expressão do pixel fixo em termos das coordenadas do plano pode ser resumida nas equações (40) e (41),

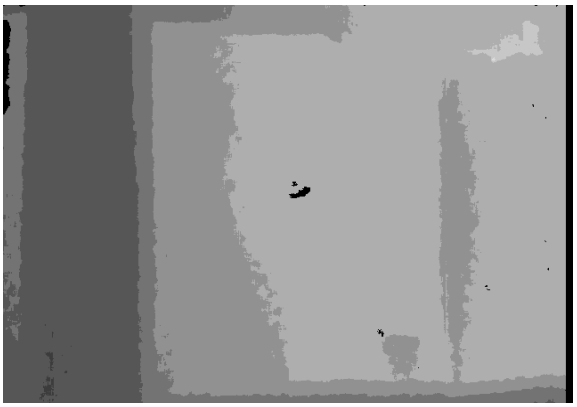
$$u = \frac{x.f_x}{z} + c_x \tag{40}$$

$$v = \frac{y.f_y}{z} + c_y \tag{41}$$

Conhecida a posição do ponto *P* em termos das coordenadas do pixel p[*u,v,1*], e e a profundidade *d*, pode-se reescrever *p* como *p*[*u,v,d*] . Para ilustrar esta conversão de pontos no espaço a pontos em coordenadas de pixels apresenta-se na Figura 23 (a) a imagem original do ambiente, em (b) a imagem da profundidade, em (c) a imagem convertida e em (d) o perfil da imagem convertida.

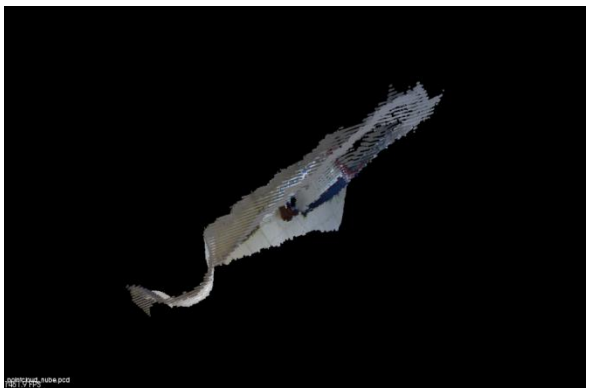
Figura 23 - a) Imagem original. b) Imagem da profundidade. c) Resultado da conversão. d) Perfil do resultado.

(a)



(b)

(c)



(d)

Fonte: Própria do autor.

6.7 Estimação baseada no PROSAC

O algoritmo de estimação proposto para determinar as melhores correspondências entre frames 3D é denominado de bPROSAC, pois é baseado no algoritmo PROSAC original que visa o acoplamento pela homografia. Este algoritmo é parte da solução do SLAM e permite a construção do mapa do ambiente de forma mais rápida, devido a sua maneira de estimar os *inliers* de maior credibilidade que estão presentes entre dois frames ou sequência de frames. As diferenças do bPROSAC para o algoritmo original PROSAC estão em:

- Realiza todo o tratamento dos dados (estimação e transformação) em 3 dimensões. Utiliza a conversão de pontos 3D a 2D para encontrar a matriz H;
- Constroi o mapa em 3D usando as carcterísticas principais dos frames. Apresenta-se a seguir o pseudocódigo para o bPROSAC.

Algoritmo - bPROSAC proposto

Entradas: N= correspondências, *m*= tamanho amostras, *n*=tamanho inicial de u_n, I_{nmin}=inliers mínimos, t = iterar,I_{Nbest}= melhor número de amostras, I_N=número total de inliers para a mostra, *isInlier* = vetor de correspondências.

1: Calcular T_N usando RANSAC;

2: Calcular I_{Nmin};

3: Calcular T_n a média das amostras de $\{\mathcal{M}_i\}_{i=1}^{T_N}$;

4: Definir $T_n^{'} = 1$ e $k_{nstart} = T_N$;

5: **Critério de parada**

6: **while**((((I_{Nbest}<I_{nmin}) || $t \leq k_{nstart}$) &&$t < T_N$&&$t <$ limiar de extrações)

7: *__Escolher o conjunto para gerar a hipótese__

8: t := t+1;

9: função de crescimento $g(t) = \min\{n: T_n^{'} \geq t\}$

10: **if** (($t > T_n^{'}$) && ($n < n_{star}$))

11: Calcular T_{n+1};

12: n=n+1;

13: Atualizar $T_n^{'}$;

14: $T_n = T_{n+1}$;

15: **endif**

16:*Extração do conjunto semi- randômico, $\mathcal{M}_t$ de tamanho ***m***

17: **if** ($T_n^{'} < t$)

18: As amostras contem *m-1* pontos escolhidos aleatoriamente de $\mathcal{U}_{n-1}$ e $\mathcal{U}_n$;

19: **else**

20: Seleção aleatória de *m* pontos de $\mathcal{U}_n$;

21: **endif**

22:*__Estimação do Modelo de parâmetros__

23: Cálculo dos parâmetros $P_t$do modelo a estimar, das amostras do subconjunto $\mathcal{M}_t$

24:*__Modelo de Verificação__

25:I_N Pontos consistentes do modelo (inliers), com os parâmetros P_t, "A hipótese é verificada contra todos os dados"

26: **if** ($I_N > I_{Nbest}$)

27: $I_{Nbest} = I_N$;

28: armazenar o melhor modelo

29: $\varepsilon_n = I_n\ /n$,$n_{best} = N$, $I_{nbest} = I_N$;$\varepsilon_{nbest} = I_{nbest}\ /n_{best}$;

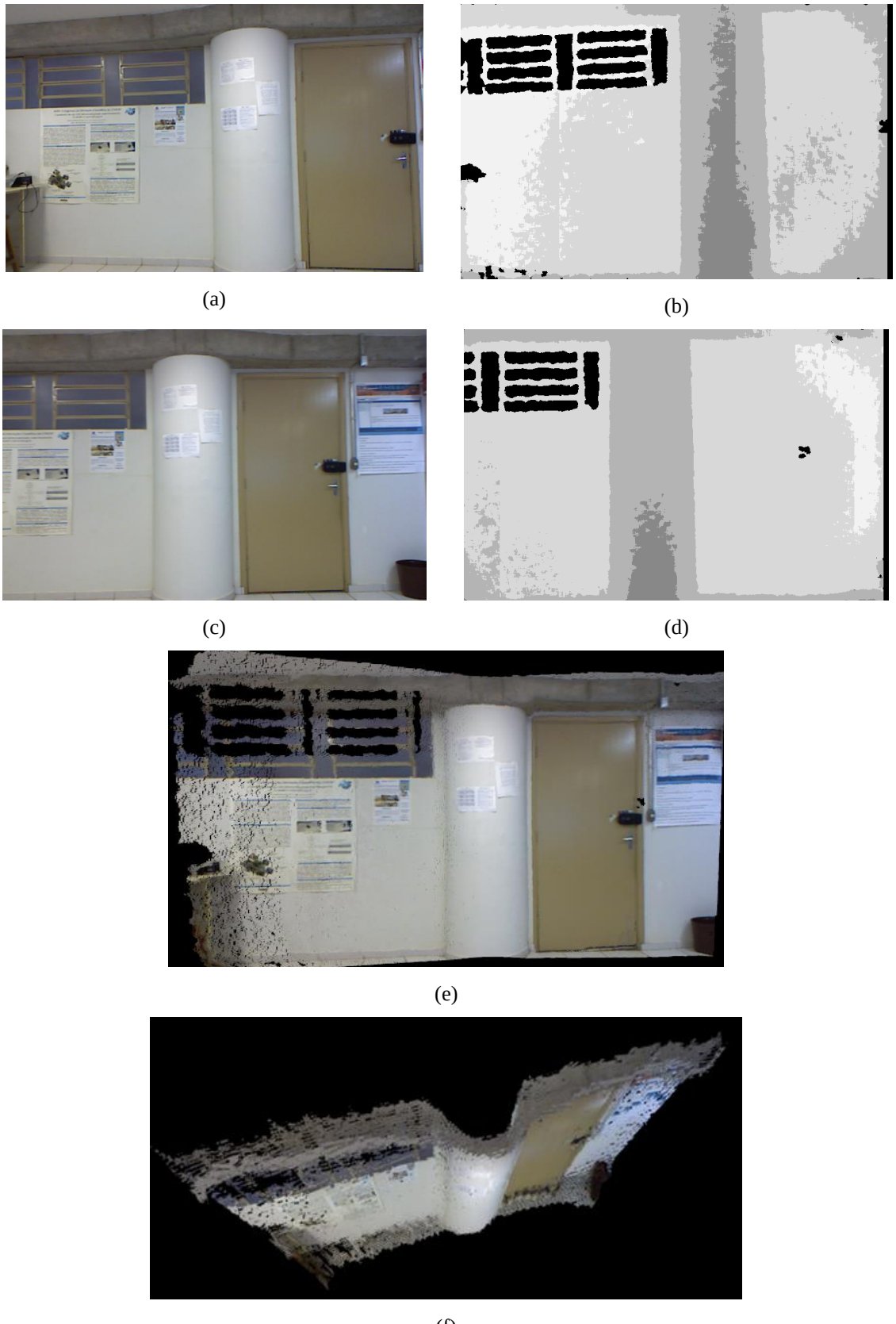
30: **for** ($n_{test} = N, I_{ntest} = I_N; n_{test} > m; n_{test} - -$)

31: Não Aleatoriedade;

32 Máximalidade com $\eta = 0.01$;
33: Calcular k_{nstar};
34: **if** $((I_{ntest} * n_{best} > I_{nbest} * n_{test})$ &&$(I_{ntest} > \varepsilon_{nbest} * n_{test} + sqrt(\varepsilon_{nbest} * n_{test} * (1 - \varepsilon_{nbest}) * 2.706)))$
35: calcular I_{nmin};
36: **if** $(I_{ntest} < I_{nmin})$
37: break;
38: **end if**
39: $n_{best} = n_{test}$;
40: $I_{nbest} = I_{ntest}$;
41: $\varepsilon_{nbest} = I_{nbest} / n_{best}$;
42: **end if**
43: $I_{ntest} = isInlier\ [n_{test} - 1]$;
44: **end for**
45: **if**$(I_{nbest} * n_{star} > I_{nstar} * n_{best})$
46: calcular k_{nstar}(RANSAC);
47: **end if**
48: **end if**
49: **end while**

Faz-se uma avaliação do algoritmo bPROSAC utilizando como entradas de dados, dois pares de frames consecutivos, mostradas nas Figuras 24 (a) a (d). O resultado da aplicação deste algoritmo é apresentado nas Figuras 24 (e) e (f).

Figura 24- Aplicando bPROSAC em 3D. a) e c) Imagens. b) e d) Profundidade. e) Resultado do acoplamento 3D. f) Perfil da imagem.



(a) (b) (c) (d) (e) (f)

Fonte: Própria do autor.

No resultado da figura 24(e) pode-se notar que o acoplamento entre os dois frames consecutivos da figura 24 (a) a (d) não apresenta inconsistências, eliminadas pelo bPROSAC.

6.8 Estimação das poses para a localização

Como visto anteriormente, no VSLAM o robô se localiza pelas poses enquanto, simultaneamente constrói o mapa do ambiente. A localização do robô tem como referência a primeira pose. Uma vez construído o mapa do ambiente e armazenado no processador, é possível realizar a tarefa de localização do robô neste ambiente. No instante que o robô realiza a captura do frame sua localização pode ser determinada por meio da comparação das transformações das características de todos os frames armazenados que fazem parte do mapa.

A estimação das poses de um conjunto de frames é feita assumindo uma pose inicial P_0 e considerando uma sequência finita de frames $(0,N)$, com $\boldsymbol{P_k}$ a pose de índice $k \in (0,N)$ que pode ser representada por uma matriz composta por uma submatriz de rotação $\boldsymbol{R}$ e um vetor de translação $\boldsymbol{t}$. O resultado final é uma matriz de coordenadas homogêneas com 6 DoF, como determinado na equação (42),

$$\boldsymbol{P_k} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{42}$$

Para $i > 0$, existe uma matriz de transformação T_{i-1}^{i} que acopla a posição P_{i-1} para a posição P_i, onde, T é representada como uma matriz (4x4), similar a $\boldsymbol{P_k}$. Se P_0 determina a posição inicial pode-se calcular uma posição P_i qualquer, combinando todas as transformações, conforme a equação (43),

$$\boldsymbol{P_i} = \prod_{i=k}^{1} T_{i-1}^{i} \boldsymbol{P_0} \tag{43}$$

Geralmente a posição inicial está definida pela orientação inicial da câmera para cada eixo, armazenada em uma matriz de rotação inicial, R_0 e o vetor de translação inicial t_0 , portanto P_0 é determinado pela equação (44) ,

$$P_0 = \begin{bmatrix} R_{0_{11}} & R_{0_{12}} & R_{0_{13}} & t_{0x} \\ R_{0_{21}} & R_{0_{22}} & R_{0_{23}} & t_{0y} \\ R_{0_{31}} & R_{0_{32}} & R_{0_{33}} & t_{0z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (44)$$

No instante inicial o robô está parado e por isso não tem translação, **t**=[0,0,0], e nem rotação (também é nula), então, desta forma, a posição inicial P_0 é determinada pela expressão (45),

$$P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (45)$$

6.9 Comentários

Neste capítulo foi apresentada a proposta do algoritmo implementado, bPROSAC para o VSLAM usando uma câmera Kinect, limitado a interiores de ambientes. Observa-se que a escolha dos detectores/descritores, o algoritmo de correspondências e transformações de pontos 3D para o espaço 2D e estimadores foram escolhidos visando obter uma redução no tempo de processamneto e de armazenamento com o objetivo de sua aplicação *on-line*.

7 RESULTADOS EXPERIMENTAIS

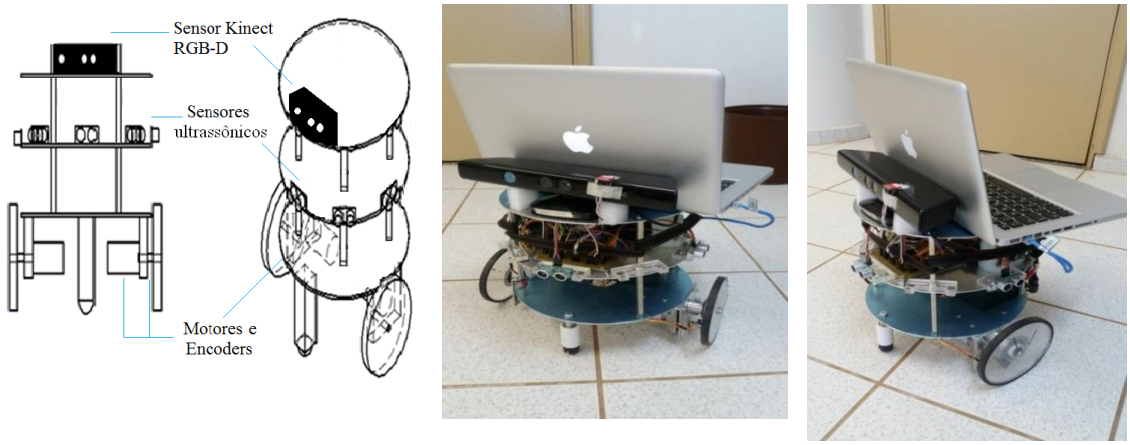
Após o estudo das principais etapas usadas para resolver o problema do SLAM Visual de um ambiente indoor, e usando sensor RGB-D para computar as poses da câmera foram realizados alguns experimentos. Descreve-se o robô construído no laboratório e usado nos experimentos e o programa desenvolvido com o objetivo de computar as poses por meio da câmera Kinect e produzir os mapas 3D, construídos para dois conjuntos de dados de tamanhos diferentes, feitos para o mesmo ambiente. Para validação do algoritmo bPROSAC baseado em ORB utiliza-se um conjunto de provas gerados pelo uso do mesmo sensor, das Universidades de *Poznan* (PUTSLAM), *Newcastle* (RGB-D *DataSet*) e a Universidade Técnica de Munique (Computer Vision Group).

7.1 Robô móvel

O robô utilizado nos experimentos foi construído no Laboratório de Sistemas Digitais (LSD) – DEE-FEIS-UNESP. É constituído de dois encoders óticos acoplados aos eixos (um em cada roda), dois motores DC e uma roda livre; e ainda 3 sensores ultrassônicos, bússola, placa de acionamento fontes (CC/CC) de bateria 12V, 7 A/h ; um sensor Kinect RGB-D dedicado para o sistema do SLAM Visual e uma *single board*- Beagle Bone Black-BBB para o acionamento e controle.

Um *notebook* Apple MacBook Pro 13, com processador Intel core i5, CPU de 2.5 Ghz, memória RAM, 16 GB, e sistema operacional virtual Linux Ubuntu 14.04 LTS, faz a captura do conjunto de imagens e armazenamento, e todo o sistema de conexão com periféricos (teclado e mouse) e a internet. A câmera kinect é posicionada na frente no robô, a 32 cm da base, conforme mostrado na Figura 25.

Figura 25 - Protótipo do Robô móvel.



Fonte: Própria do autor

O protótipo foi dimensionado visando o desenvolvimento de algoritmos para a navegação de robôs móveis e para o SLAM, suas especificações são resumidas na Tabela 3. Outros detalhes do hardware/software encontram-se no Apêndice A.

Tabela 3 - Dimensões do robô.

Peso	4 Kg
Altura	30 cm
Diâmetro	26 cm
Velocidade dos motores	8,5 rpm
Alimentação dos motores	12 V

Fonte: Própria do autor.

7.2 Parâmetros do sistema

Alguns parâmetros usados no sistema do SLAM Visual podem ser ajustados para melhorar o desempenho, entre os mais importantes têm-se os relacionados na Tabela 4. Nos experimentos realizados esses parâmetros são fixados, de forma a mostrar a robustez para os diferentes conjuntos de dados.

Tabela 4 - Parâmetros do bPROSAC para o SLAM Visual.

Máximo número de iterações	1000
Mínima quantidade de inliers para correspondência	5
Mínima quantidade de amostras na **matriz de Homografia**	7

Fonte: Própria do autor.

7.3 Algoritmo completo para o sistema VSLAM

O programa deve executar as seguintes tarefas:

1. Adquirir um conjunto RGB-D de dados da câmera Kinect;

2. Executar a extração de características do ambiente e *matching* com ORB;

3. Calcular a estimação das transformações relativas com bPROSAC;

4. Detectar os pontos repetidos com *loop closure* e inserir os elementos correspondentes no grafo;

5. Otimizar o grafo com g^2o e atualizar as poses da câmera;

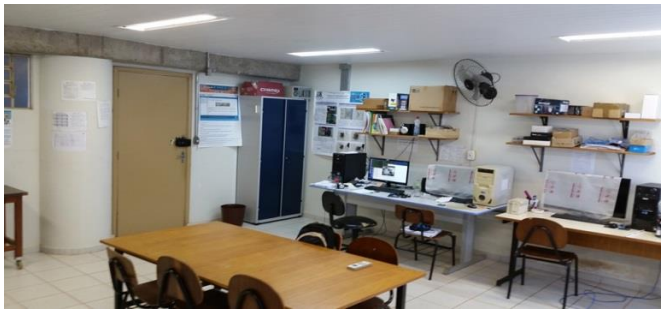
6. Reconstruir o mapa por meio do arquivo de nuvem de pontos.

7.4 Construção do Mapa

O cenário escolhido para o experimento é o laboratório de pesquisa de Sistemas Digitais do Departamento de Engenharia Elétrica - FEIS-UNESP. Neste cenário são capturados dois conjuntos de imagens diferentes, gerando dois experimentos que foram usados para a construção dos mapas do interior do laboratório:

- **Cenário1**- no primeiro experimento utiliza-se a plataforma (mesa) a uma altura de 75 cm do piso, onde o sensor/robô é posicionado no meio da mesa e gira em torno do seu eixo, baseado nas Figuras 26 (a) e (b);
- **Cenário 2** - utiliza-se o robô ao nível do piso, Figura 26 (c).

Figura 26 - Cenário 1 em a) e b). c) Cenário 2.



(a)

(b)

(c)

Fonte: Própria do autor.

Para os dois experimentos o processamento da imagem é realizado no PC com processador Intel core^{TM} i5, CPU de 3.0 Ghz, RAM de 16 GB, Sistema operacional Linux Ubuntu 14.04 LTS. No *laptop* os frames são somente capturados e armazenados por um algoritmo na linguagem C++ utilizando a plataforma de desenvolvimento ROS (*Robot Operating System*).

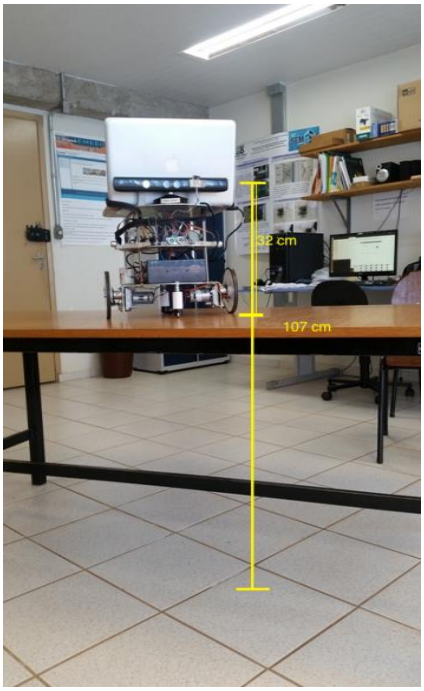
O armazenamento dos frames (o par, RGB e profundidade, salvos ao mesmo tempo) ocupam mais de 1 GB de memória devido a sua natureza, e a capacidade de processamento na máquina virtual que é limitada em termos de hardware e software pelo compartilhamento dos recursos do sistema. Os dados dos encoders são armazenados na *single board* utilizados depois na validação da trajetória.

No algoritmo implementado a captura dos dados do entorno é realizada pelo robô/sensor a uma velocidade constante de 5 cm/s, para que não haja perda de sincronização pela câmera.

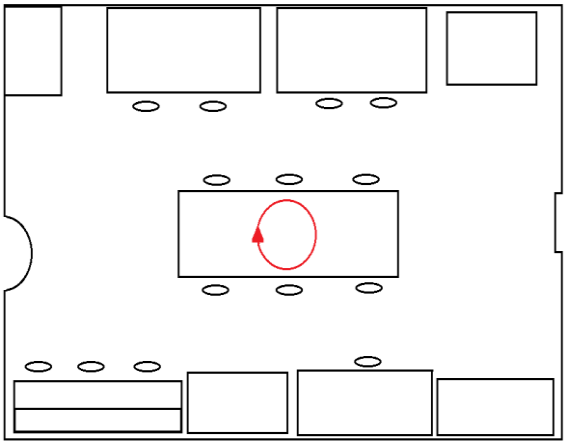
7.4.1 Primeiro experimento

No primeiro experimento utiliza-se o **cenário 1**, Figuras 27 (a), cujo robô/sensor é posicionado em cima da mesa e gira sob seu próprio eixo fazendo uma varredura em forma circular, para capturar a sequência de imagens. Na Figura 27 (b) tem-se a planta baixa do cenário.

Figura 27 - a) Cenário 1. b) Planta baixa do cenário.



(a)

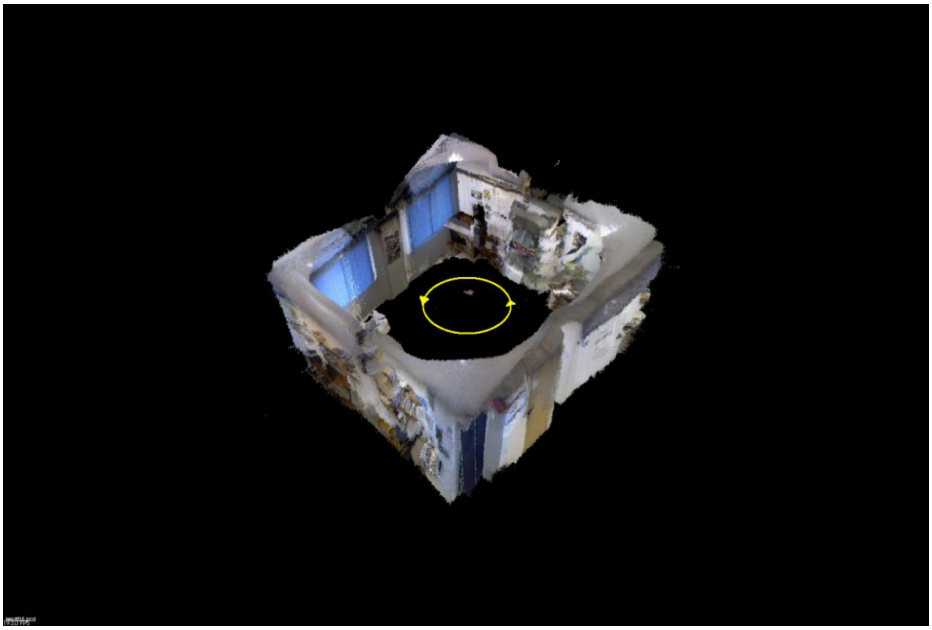


(b)

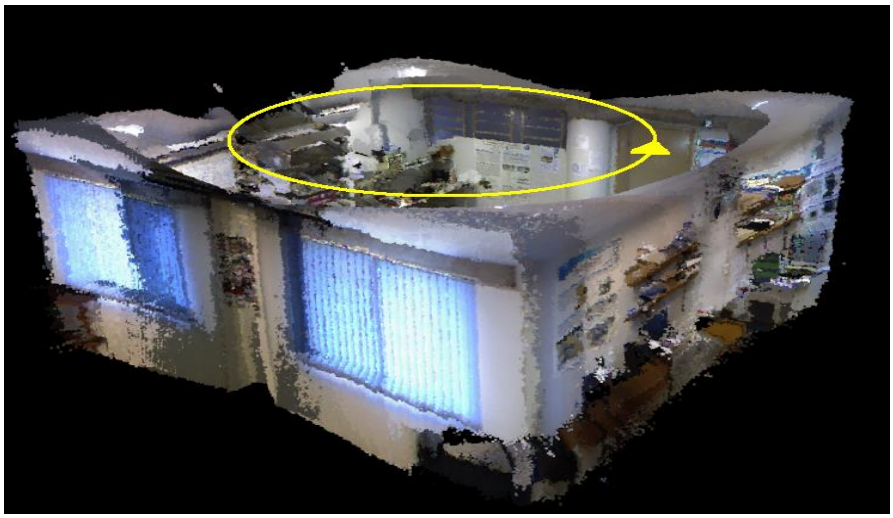
Fonte: Própria do autor.

Aplicando o algoritmo para o SLAM visual proposto obteve-se o mapa 3D colorido, em perspectivas frontal e lateral , Figuras 28 (a) e (b) ; o perfil superior e duas vistas do interior do mapa, apresentados nas Figuras 28 (c) a (e).

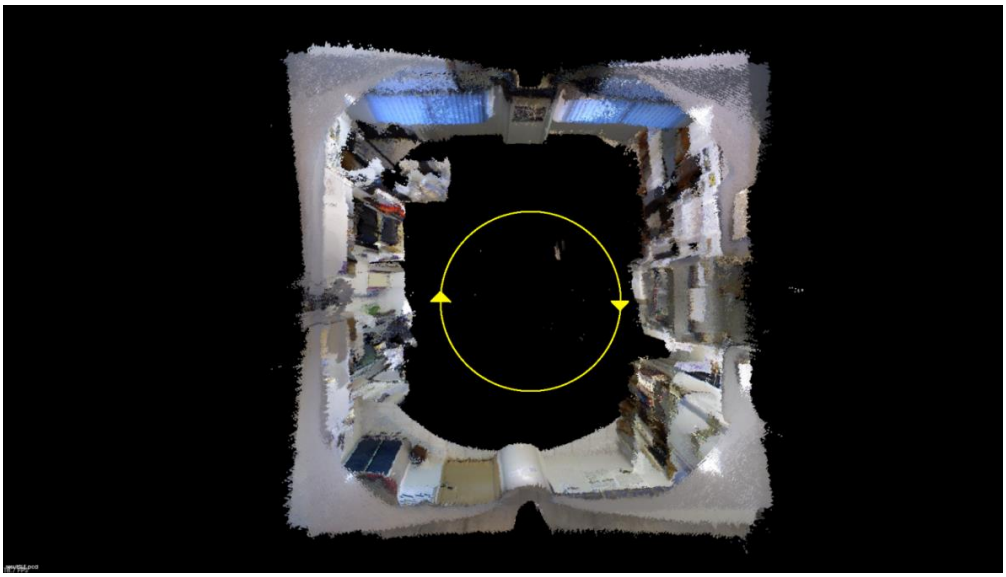
Figura 28 - Mapas. a) Perspectiva frontal. b) Lateral. c) Perfil superior. d) e e) interior.



(a)



(b)



(c)

(d) (e)

Fonte: Própria do autor.

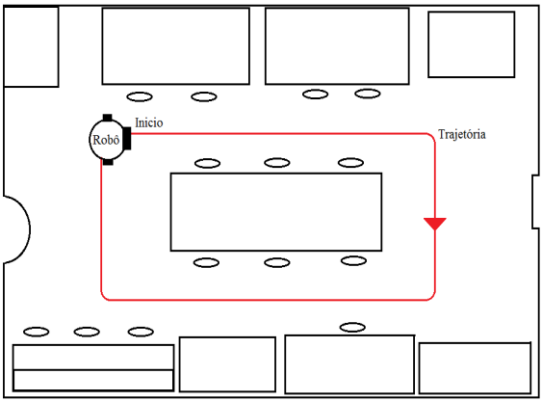
Nos resultados dos mapas 3D para o cenário 1 observa-se a visão completa do ambiente capturada pelo sensor Kinect, demonstrando que o desempenho do algoritmo do SLAM Visual proposto foi satisfatório, mesmo com o tamanho reduzido do robô, com a plataforma consegue capturar melhor os dados do ambiente. Nas duas vistas do interior do mapa, figuras (d) e (e) observam-se que os diferentes níveis de iluminação não afetaram de forma que prejudicasse a qualidade da cena, demostrando que o detector/descritor ORB e o estimador bPROSAC são invariantes à mudança desse parâmetro.

7.4.2 Segundo experimento

Para o **cenário 2** , o robô ao nível do piso, Figura 29 (a), foi usada a planta baixa (simplificada) do ambiente e a trajetória - percurso fechado, Figura 29 (b).

Figura 29 - a) Cenário 2. b) Planta baixa do cenário.

(a)

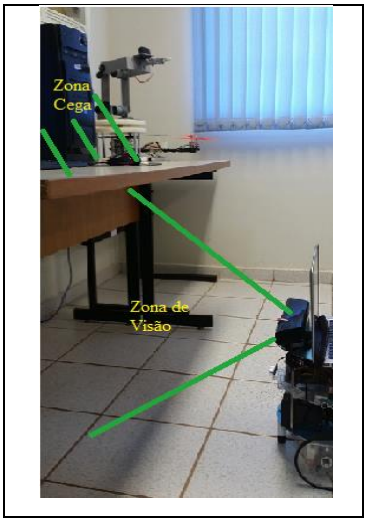


(b)

Fonte: Própria do autor.

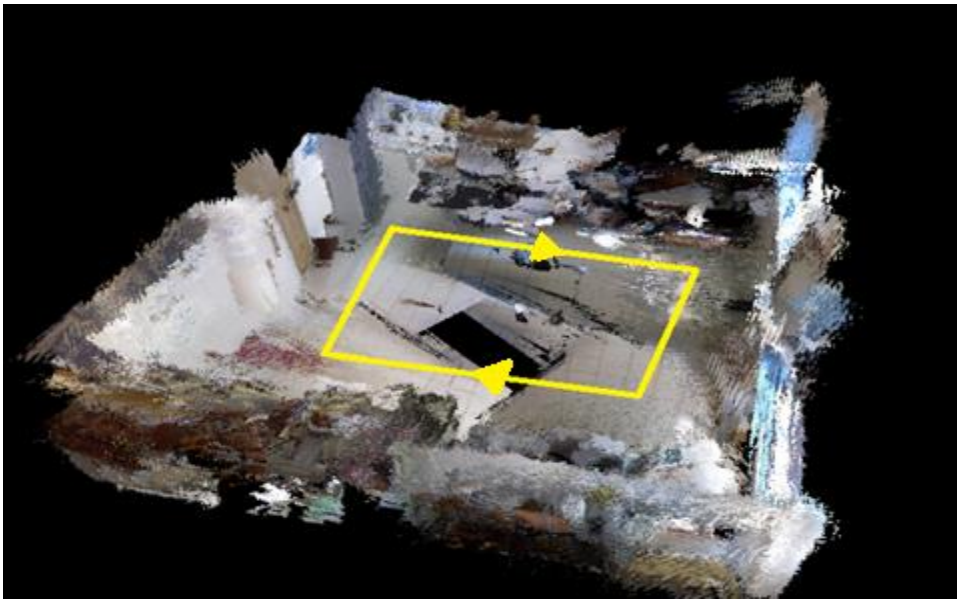
Neste experimento o robô/sensor está localizado a uma altura aproximada de 1/6 do pé direito do ambiente, por isso, a construção do mapa fica prejudicada pela pouca visão do robô- 42 graus, Figura 30. O mapa construído não traduz o mapa completo do ambiente, por que aparecem zonas cegas (onde o sensor não pode “ver”). Os resultados são apresentados nas Figuras 31 (a) e (b) e na Figura 32 (a) a (d).

Figura 30 - Faixa de visão do robô

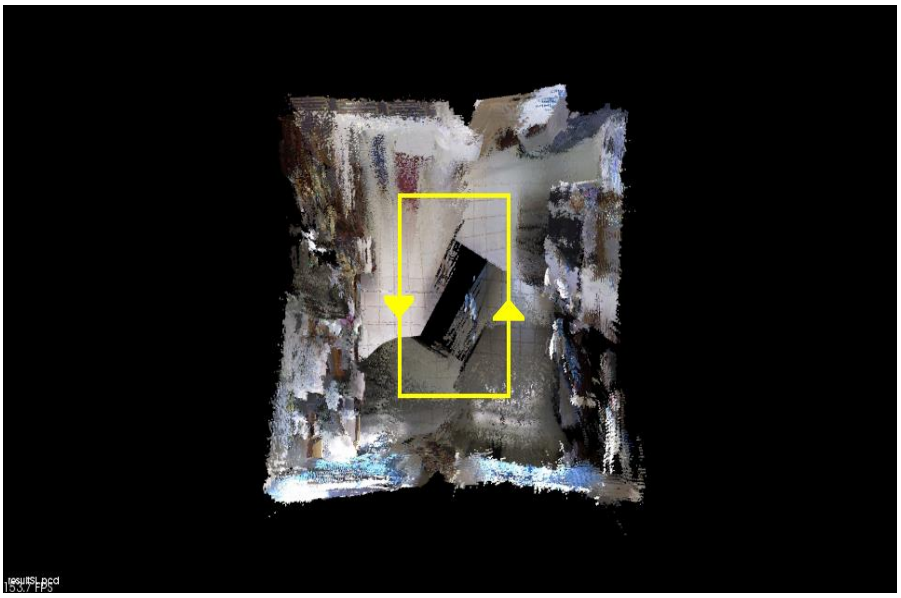


Fonte: Própria do Autor.

Figura 31- Mapas. a) Perspectiva frontal. b) Perfil superior.



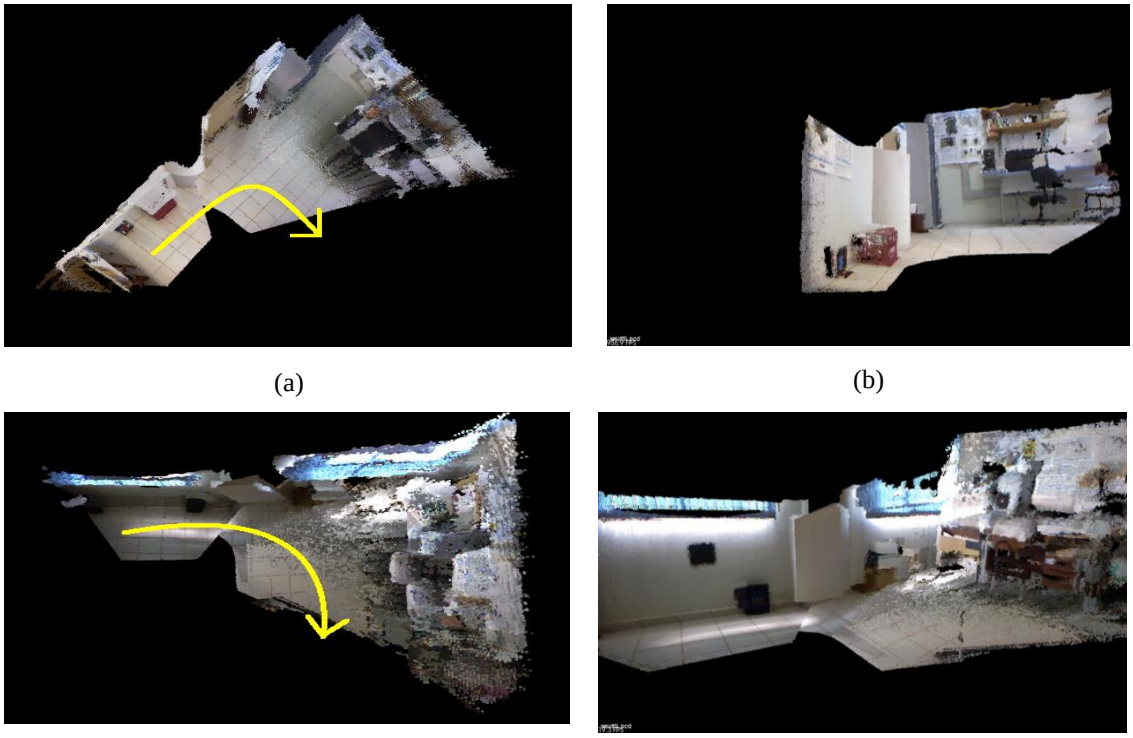
(a)



(b)

Fonte: Própria do autor.

Figura 32 - Mapa. a) Fração do perfil superior-boa iluminação. b) Interior - perfil superior.c) Fração do perfil superior-iluminação média. d) Interior - perfil superior.



(a) (b)

(c) (d)

Fonte: Própria do autor.

São observadas nessas figuras a construção do mapa 3D do ambiente, capturado ao nível do sensor, o desempenho do algoritmo proposto e a sua invariância à iluminação, embora sua perspectiva superior esteja confusa.

Na Tabela 5 apresentam-se alguns dos parâmetros de avaliação de execução, tais como, números de frames, tempo por frame, tempo total que engloba todo o processamento do algoritmo e o uso em porcentagem da CPU, obtidos na construção dos mapas usando o algoritmo proposto.

Tabela 5 - Parâmetros de avaliação de execução do algoritmo proposto.

Movimento	**Número de Frames**	**Tempo por Frame (ms)**	**Tempo total (s)**	**Uso da CPU (%)**
Experimento 1	780	278	217	80
Experimento 2	2058	278	572	80

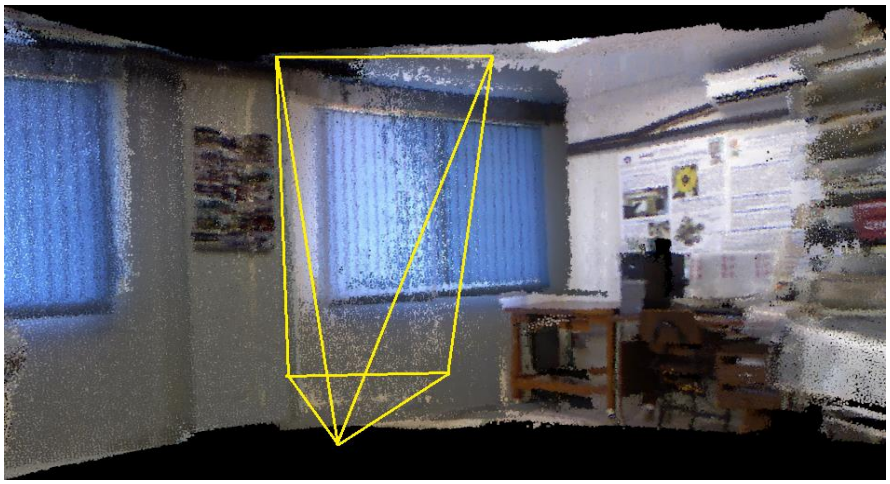
Fonte: Própria do autor.

Os resultados obtidos no processamento dos frames apresentam um tempo de execução por frame menor que o tempo apresentado em Endres (2012) que foi de 350 (ms) que utilizou SURF e RANSAC e o apresentado por Henry et al (2014), 730 (ms), que utilizou SIFT e RANSAC. Outros bancos de dados podem ser testados para a validação do algoritmo.

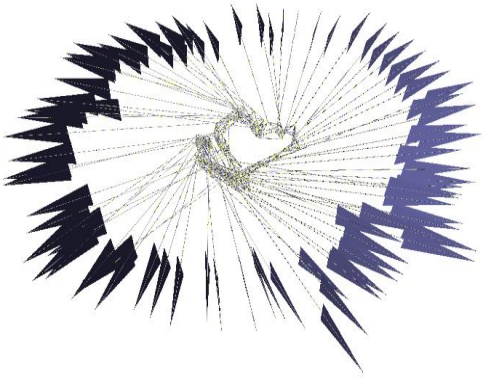
7.5 Localização

A localização do robô como visto no diagrama de blocos do algoritmo proposto para o SLAM Visual, é obtida pelas poses (orientação e posição). No primeiro experimento, usando o cenário 1 tem-se uma das poses do robô no ambiente, Figura 33 (a) e em (b) a sequência das poses da câmera mostradas pelo algoritmo g^2o. Com estas informações obtém-se a trajetória feita pelo robô que é comparada com a trajetória desejada, calculada por meio dos dados dos encoders, Figura 33 (c).

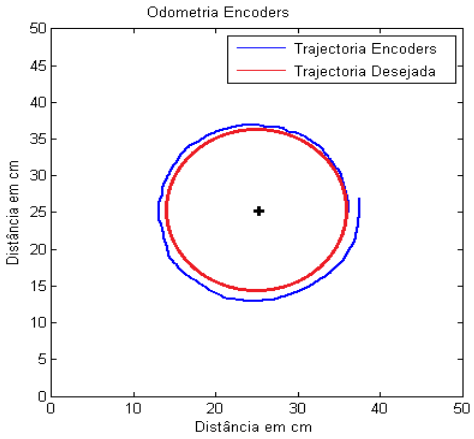
Figura 33 - Localização. a) Uma pose da câmera. b) Sequência de poses. c) Trajetória desejada e estimada.



(a)



(b)

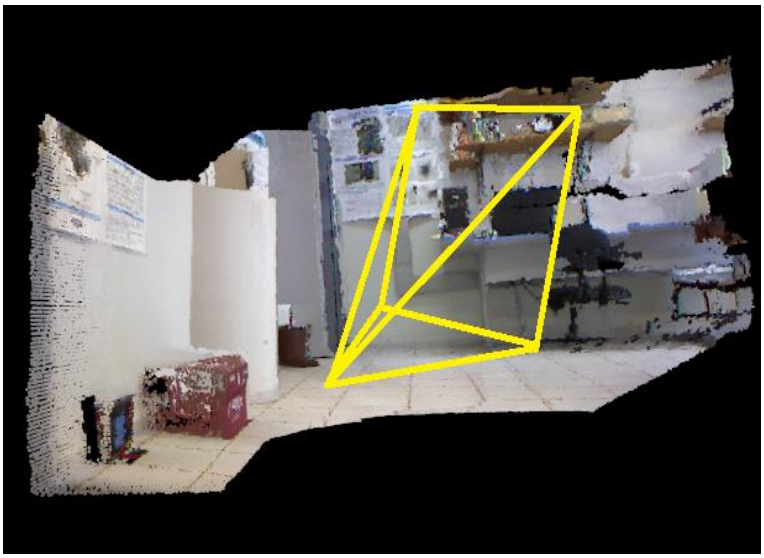


(c)

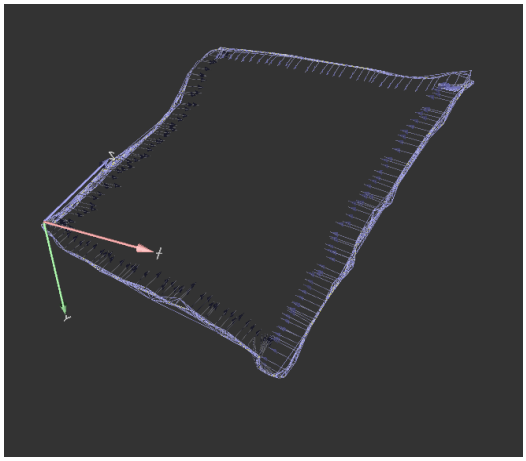
Fonte: Própria do autor.

Para o experimento 2, no mesmo cenário, uma das poses do robô no ambiente é dada pela Figura 34 (a) e em (b) tem-se a sequência das poses da câmera mostradas pelo algoritmo g^{2}o. Com estas informações obtém-se a trajetória feita pelo robô que é novamente comparada com a trajetória calculada por meio dos dados dos encoders, Figura 34 (c).

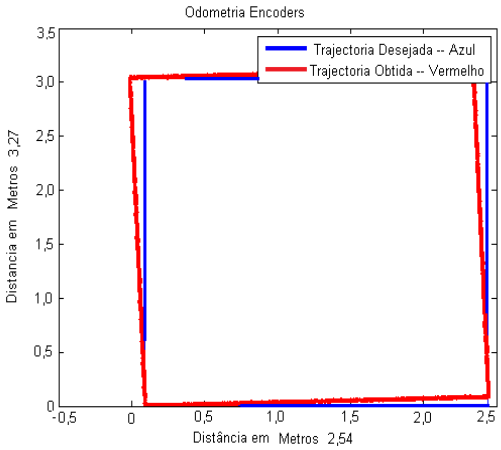
Figura 34 - Localização. a) Uma pose da câmera. b) Sequência de poses. c) Trajetória desejada e estimada.



(a)



(b)



(c)

Fonte: Própria do autor.

7.6 Testes para validação

Visando sua validação, o algoritmo bPROSAC foi aplicado a três *datasets* obtidos das seguintes universidades:

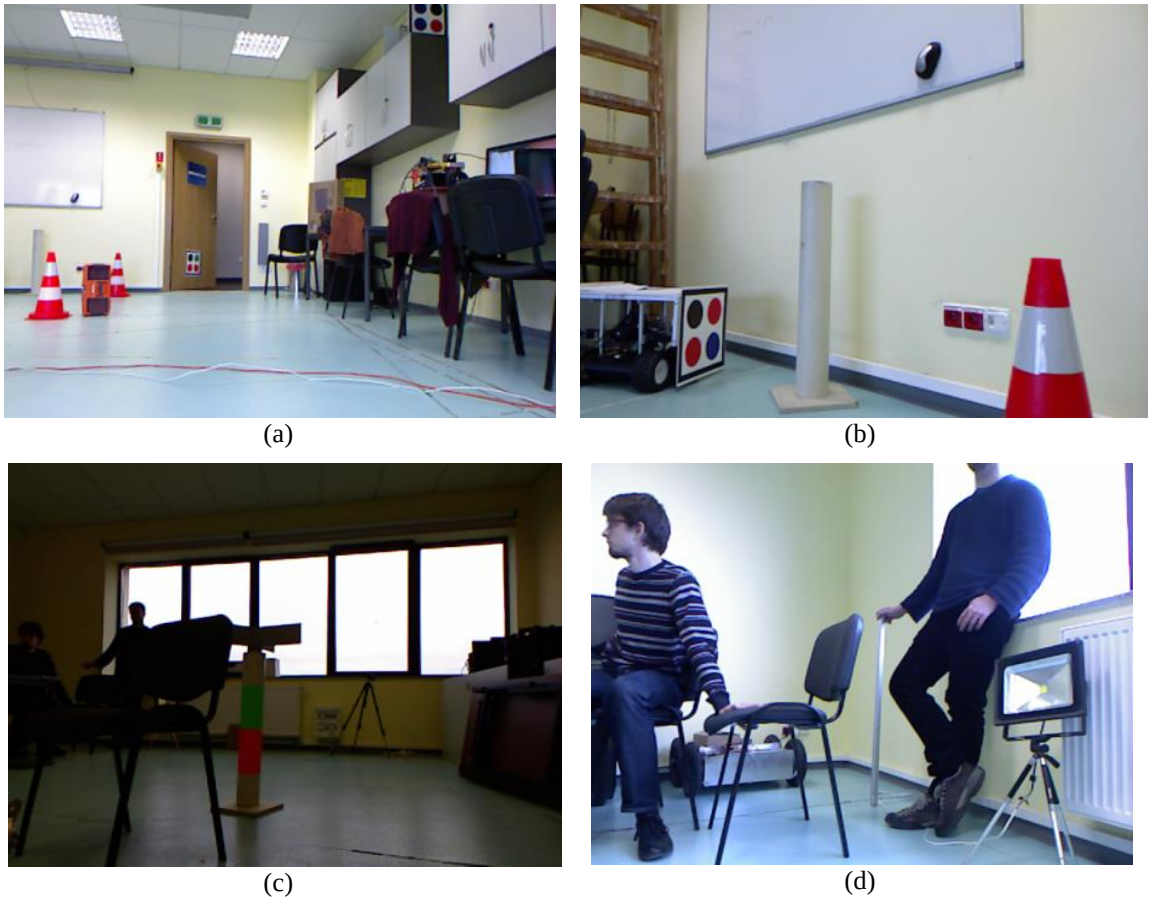
a) Universidade de tecnologia de *Puzman* (*PUTSLAM TEAM)* (POZNAN UNIVERSITY OF TECHNOLOGY, 2017);

b) Universidade de *Newcastle & Microsoft Research* (NEWCASTLE & MICROSOFT, 2017);

c) Universidade de Munique (TECHNICAL UNIVERSITY OF MUNICH, 2017).

Apresentam-se nas Figuras 35 (a) a (d), os cenários, conjunto de dados da Universidade de *Puzman* para os testes para validação da construção do mapa usando o algoritmo bPROSAC. Os resultados para esta avaliação, correspondentes a estes cenários, são mostrados nas Figuras 36 (a) a (d).

Figura 35 – Imagens do *dataset* do laboratório da Universidade de *Puzman*

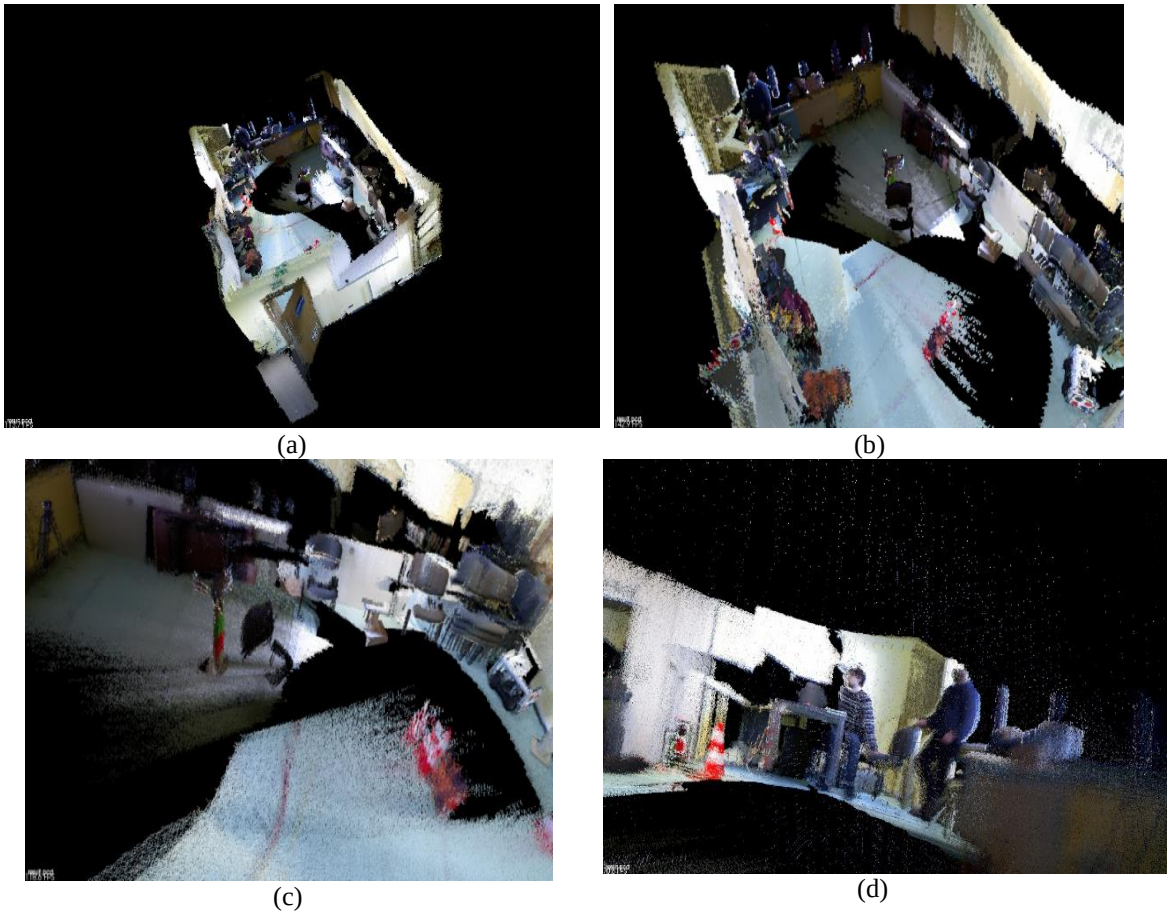


(a) (b)

(c) (d)

Fonte: Poznan University of Technology (2017)

Figura 36 –Mapas gerados com os *datasets* de *Puzman*



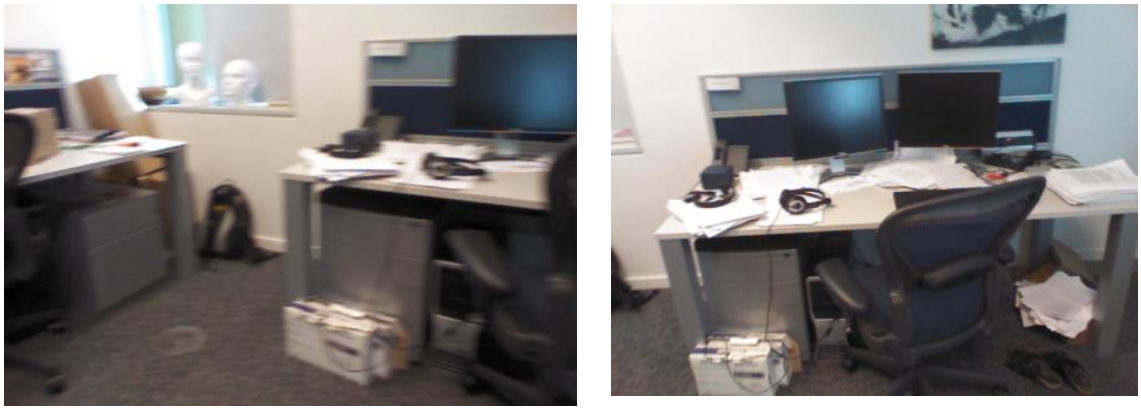
(a) (b)

(c) (d)

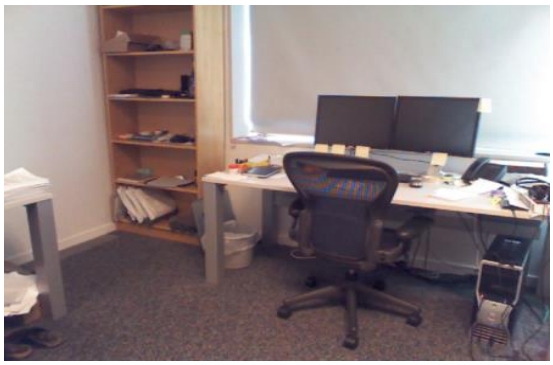
Fonte: Própria do autor

Na Figura 37 (a) a (c) apresenta-se os cenários do *dataset* da Universidade de Tecnologia de *Newcastle & Microsoft Research*, cuja respostas são mostradas nas Figuras 38 (a) a (c) .

Figura 37 – Imagens do conjunto de dados do laboratório de *Newcastle*



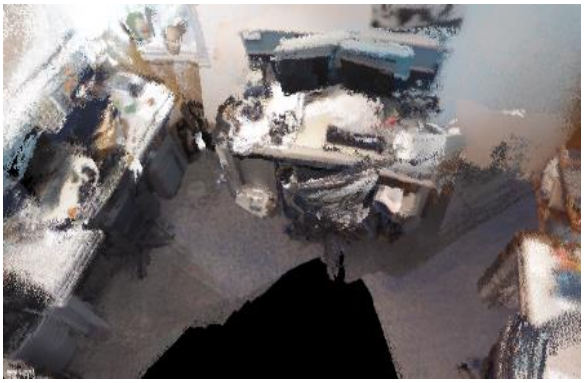
(a) (b)



(c)

Fonte: Newcastle & Microsoft (2017).

Figura 38 – Mapas gerados com os *datasets* de *Newcastle & Microsoft Research*



(a)

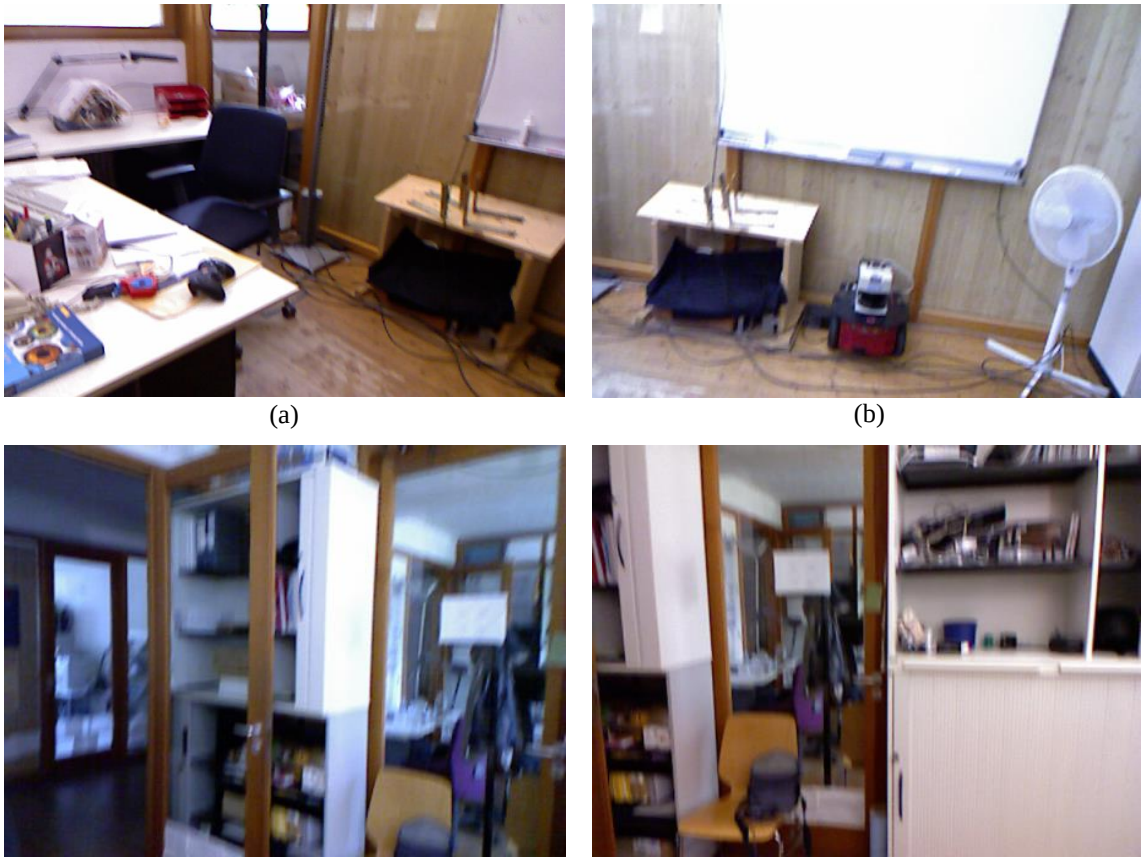
(b)

(c)

Fonte: Própria do autor.

Finalmente, para o banco de dados da Universidade de Tecnologia de Munique *dataset* das Figuras 39 (a) a (d), e a resposta deste cenário na construção do mapa usando os algoritmos bPROSAC , apresentado nas Figuras 40 (a) a (d).

Figura 39 – Imagens do conjunto de dados da Universidade de Munique.

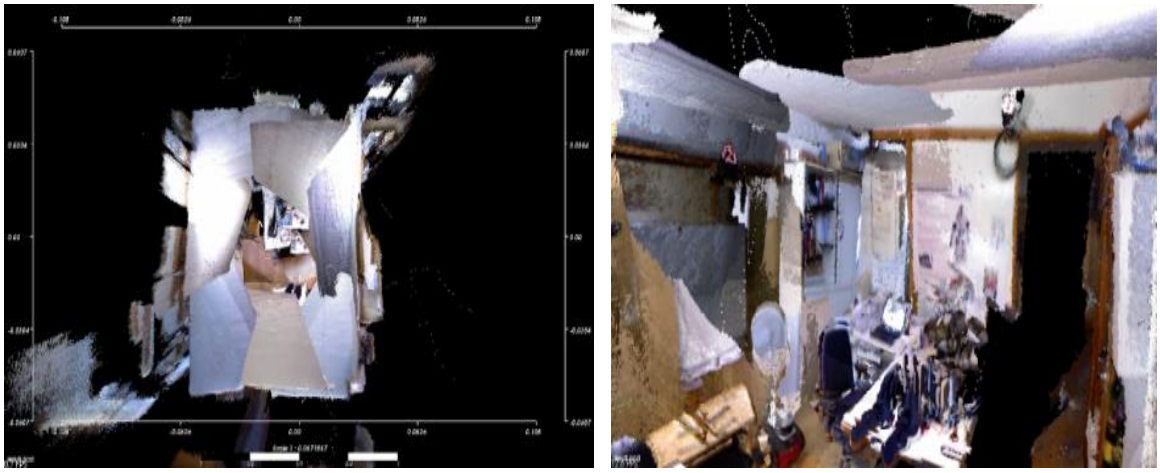


(a) (b)

(c) (d)

Fonte: Technical University of Munich (2017).

Figura 40 - Mapas gerados com os *datasets* de *Munique*



(a) (b)

(c) (d)

Fonte: Própria do autor.

7.7 Comentários

Dos resultados obtidos para a proposta do VSLAM onde é usado ORB como extrator de características e o estimador bPROSAC usando o sensor RGB-D, nota-se um desempenho que deve ser mais bem avaliado para a construção do mapa do ambiente usando o algoritmo proposto, embora haja uma redução nos tempos de processamento do algoritmo. Estudos comparativos variando luminosidade e extratores, por exemplo, poderiam fornecer maior confiabilidade a proposta.

8 CONCLUSÕES E SUGESTÕES DE TRABALHOS FUTUROS

Neste trabalho foi realizada uma proposta para o algoritmo VSLAM (monocular) que permite a construção de mapas 3D internos, a partir dos dados capturados por meio de um sensor Kinect, embarcado em um robô construído em laboratório, que fornece a informação da câmera e do sensor de profundidade.

O problema do SLAM tem sido amplamente pesquisado, principalmente devido ao surgimento da câmera RGB-D que tem como vantagens, as ricas informações de imagens e o preço acessível. Além disso, a comunidade de pesquisadores têm disponibilizado informações, software de código aberto (*open-source*) e banco de dados de imagens que podem ser usados para validação de propostas de novos algoritmos, visando a automação dos sistemas robóticos.

Uma correta escolha do sensor na solução do VSLAM garante uma alta probabilidade de sucesso, pois, podem surgir inconsistências nos mapas construídos, devido a objetos, fora da faixa de medição do sensor, na cena.

Na solução do VSLAM a captura dos frames do ambiente pela câmera/robô deve ser a uma velocidade baixa e constante para que o sensor não forneça imagens embaçadas do ambiente, gerando mapas inconsistentes. Mapas inconsistentes podem ser gerados também quando as informações estão fora ou no limite da visão do sensor, fazendo o robô perder a sua trajetória.

Na proposta de solução do VSLAM foi utilizada uma plataforma de software livre, que permitiu a integração dos diferentes programas, na busca da melhor solução para a construção de um mapa consistente do ambiente.

Dos resultados obtidos pela construção de dois mapas do mesmo ambiente, utilizando as mesas ferramentas, de classificação, estimação e otimização, observa-se que um dos mapas construídos permite visualizar com detalhes os elementos que pertencem ao ambiente, isto é devido a localização do robô/câmera no ambiente. A posição da câmera deve estar a uma altura com ângulo de visão de 43^0 vertical por 57^0 horizontal, com faixa de inclinação vertical de $\pm\ 27^0$, de forma a proporcionar uma visão completa do ambiente.

Os testes adicionais com os bancos de dados de outras universidades, ainda não são conclusivos.

8.1 Sugestões de trabalhos futuros.

Como sugestões visando aperfeiçoar o trabalho têm-se:

- Em termos de hardware, para uma melhor descrição do ambiente, pode-se acrescentar sensores externoceptivos ao ambiente, gerando uma redução dos erros devido aos rápidos movimentos do sensor embarcado no robô e ao tamanho da cena;
- No VSLAM monocular a escala absoluta do mapa não pode ser determinada, assim é necessário uma etapa adicional de normalização durante a otimização. No caso do SLAM estéreo não há esta limitação uma vez que a profundidade pode ser calculada da disparidade entre imagens. Uma solução para este problema seria o uso de uma câmera de teto – VSLAM estéreo.

REFERÊNCIAS

ALSINA, P.; GARCIA, L.; MEDEIROS, A. D.; PINHEIRO, D. F.; VIEIRA, C. Navegação e controle de robôs móveis. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 2002, [S. l.]. **Mini curso...** [S. l.: s. n.], 2002.

AKBARZADEH, A.; FRAHM, J.M.; MORDOHAI, P.; CLIPP, B.; ENGELS, C.; GALLUP, D. Towards urban 3D reconstruction from vídeo. In: INTERNATIONAL SYMPOSIUM ON 3D DATA PROCESSING, VISUALIZATION AND TRANSMISSION (3DPVT), 3., 2006. **Proceedings**… [S. l.: s. n.], 2006. p. 1– 8,

ANGELI, A.; DONCIEUX, S.; MEYER, J. A.; FILLIAT, D. Real-time tisual loop-closure detection. IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION PASADENA, 2008, [S. l.]. **Proceedings…** [S. l.: s. n.], 2008. p. 19-23.

ASADA, H.; BRADY, M. The curvature primal sketch. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Piscataway, v. 8, n. 2, p. 2-14, 1986.

AUDRAS, C.; COMPORT, A. I.; MEILLAND, M.; RIVES, P. Real-time dense RGB-D localisation and mapping. In: CONFERENCE ON ROBOTICS AND AUTOMATION, 2011, [S. l.]. **Porceedings…** [S. l.]: Monash University, 2011.

BALCH, T.; ARKIN, R. Avoiding the past: a simple but effective strategy for reactive navigation. IN: IEEE, INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1993, Atlanta. **Proceedings…** [S. l.: s. n.], 1993.

BAY, H.; TUYTELAARS, T.; GOOL, L,C. SURF: speeded up robust features. Computer Vision and Image Understanding (CVIU), v. 110, n. 3, p. 346-359, 2008.

BEAGLEBOARD. **BeagleBone Black Rev-C**. [S. l.], 2017. Disponível em: <https://beagleboard.org/black>. Acesso em: 10 jan. 2017.

BELS, P.; MACKAY, N. A method for registration of 3-D shapes. In: **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Piscataway, v. 14, n. 02, p. 239-256, 1992.

BORENSTEIN, J.; L. FENG. Measurement and correction of systematic odometry errors in mobile robots. **IEEE Transactions on Robotics and Automation**, Piscataway, v. 12, n. 6, p. 869-880, 1996.

BOTELHO, S. S. C., DREWS, P.; OLIVEIRA, G.; FIGUEIREDO, M. Visual odometry and mapping for Underwater Autonomous Vehicles. In: SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE (SBAI), 2009, [S. l.]. **Proceedings...** [S. l.: s. n.], 2009.

BUEHLER, M.; IAGNEMMA, K.; SINGH, S. (Ed.). **The DARPA urban challenge: autonomous vehicles in city traffic**. Berlin: Springer, 2009.

CALONDER, M.; VINCENT, L.; STRECHA, C.; FUA, P. BRIEF: binary robust independent elementary features. In: EUROPEAN CONFERENCE ON COMPUTER VISION (ECCV), 2010, Berlin. **Proceedings…** [S. l.: s. n.], 2010. p. 778–792.

CHEN, Y.; MEDIONI, G. Object modeling by registration of multiple range images. **Image and Vision Computing**, Amsterdam, v. 10, n. 3, p. 145–155, 1992.

CHEONG, H.;PARK, S.; KEE PARK, S. Topological map building and exploration based on concave nodes. In: INTERNATIONAL CONFERENCE ON CONTROL, AUTOMATION AND SYSTEMS, 2008, Seoul. **Proceedings…** [S. l.: s. n.], 2008.

CHOSET, H.; FOX, D. The world of mapping. In: TEC WORKSHOP ON REVIEW OF UNITED STATES RESEARCH IN ROBOTICS, NATIONAL SCIENCE FOUNDATION (NSF), Arlington, Virginia, USA. 2004. **Proceedings…** Arlington: [s. n.], 2004.

CHUM, O.; MATAS, J. **Matching with PROSAC:** progressive sample consensus. [S. l.], CVPR 2005

CLEMENTE, R. I.; DAVISON, A. Mapping large loops with a single hand-held camera. ROBOTICS: SCIENCE AND SYSTEM, 2007, [S. l.]. **Proceedings…** [S. l.: s. n.], 2007.

COMPORT, A.; MALIS, E.; RIVES, P. Real-time quadrifocal visual odometry. **Intl. Journal of Robotics Research (IJRR),** London, v. 29, p. 245–266, 2010.

CROWLEY, J. L. World modeling and position estimation for a mobile robot using ultrasonic ranging. In: (ICRA89) IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1989, Scottsdale. **Proceedings…** [S. l.: s. n.], 1989.

CRUZ, N. A. **Autonomous underwater vehicles**. Rijeka: InTech Janeza Trdine, 2011.

CSORBA, M. UHLMANN, J. K.; DURRANT H. F. **A new approach to simultaneous localisation and map building.** Orlando: SPIE Aerosense, 1996.

DAVIDSON, A.; MURRAY, W. Simultaneous localisation and map-building using active vision. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Piscataway, v. 24, n. 7, p. 865-880, 2002.

DAVISON, A. J. Real-time simultaneous localisation and mapping with a single camera. **IEEE International Conference on Computer Vision (ICCV),** Piscataway, v. 2, p. 1403-1410, 2003.

DISSANAYAKE, G.; NEWMAN, P. M.; DURRANT, H. F.; CLARK, S.; CSORBA, M.A solution to the simultaneous localization and map building (SLAM) problem. **IEEE Trans. Robot. Autom.,** Piscataway, v. 17, n. 3, p. 229‑241, 2001.

DUDEK, G.; FREEDMAN, P.; HADJRES, S. Using Local Information in a Non-Local Way for Mapping Graph-Like Worlds. INTERNATIONAL JOINT CONFERENCE OF ARTIFICIAL INTELLIGENCE, 1993, Chambery. **Proceedings…** [S. l.: s. n.], 1993. p. 1639-1647.

DURRANT, H.; FELLOW, W.; BAILEY, T.Simultaneous localization and mapping (slam): Part I". **Robotics and Automation Magazine, IEEE,** Piscataway, v. 13, n. 2, p. 99–110, 2006a.

DURRANT, H.; FELLOW, W.; BAILEY, T. Simultaneous localization and mapping (slam): Part II". **Robotics and Automation Magazine, IEEE**, Piscataway, 13, n. 3:108–117, 2006b.

DURRANT, H.; RYE, D.; NEBOT, E. Localisation of automatic guided vehicles. In: ROBOTICS RESEARCH: INTERNATIONAL SYMPOSIUM (ISRR'95), 7., 1996, New York. **Proceedings…** New York: [s. n.], 1996. p. 613–625.

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

ENDRES, F.; HESS, J.; ENGELHARD, N.; STURM, J.; CREMERS, D.; BURGARD, W. An evaluation of the RGB-D SLAM system. In: IEEE INTL. CONF. ON ROBOTICS AND AUTOMATION (ICRA), 2012, [S. l.]. **Proceedings…** Piscataway: IEEE, 2012. p. 1691-1696.

ENGELHARD, N.; ENDRES, F.; HESS, J.; STURM, J.; BURGARD, W. Real-time 3D visual SLAM with a hand-held RGB-D câmera. In: PROC. OF THE RGB-D WORKSHOP ON 3D PERCEPTION IN ROBOTICS AT THE EUROPEAN ROBOTICS FORUM, 2011, Sweden. **Proceedings…** [S. l.: s. n.], 2011.

FIORAIO, N.; KONOLIGE, K. Realtime visual and point cloud slam. In: RGB-D WORKSHOP ON ADVANCED REASONING WITH DEPTH CAMERAS AT ROBOTICS. SCIENCE AND SYSTEMS CONF. (RSS), 2011, [S. l.]. **Proceedings…** [S. l.: s. n.], 2011. p. 1-3.

FISCHLER, M. A.; BOLLES, R. C. **Random sample consensus:** a paradigm for model fitting with apphcatlons to image analysis and automated cartography. Menlo Park: Artificial Intelligence Center, 1981.

FOX, D.; BURGARD, W.; THRUN, S. Probabilistic methods for mobile robot mapping. In: IJCAI WORKSHOP ON ADAPTIVE SPATIAL REPRESENTATIONS OF DYNAMIC ENVIRONMENTS, 1999, [S. l.]. **Proceedings…** [s. n.], 1999.

FU, S.; YANG, G. Uncalibrated monocular based simultaneous localization and mapping forindoor autonomous mobile robot navigation. In: CONFRERENCE ON NETWORKING, SENSING AND CONTROL, 2009, Okayama. **Proceedings…** [S. l.: s. n.], 2009.

GIL, A.; MOZOS, O.; BALLESTA, M.; REINOSO, O. A comparative evaluation of interest point detectors and local descriptors for visual SLAM. **Machine Vision and Applications**, Heidelberg, v. 21, p. 905–920, 2010.

GONCALVES, L.; DI BERNARDO, E.; BENSON, D.; SVEDMAN, M.; OSTROWSK, J.; KARLSSON, N.; PIRJANIAN, P. A visual front-end for simultaneous localization and mapping. IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2005, [S. l.]. **Proceedings…** Piscataway: IEEE, 2005. p. 44-49.

GRISETTI. G.; GRZONKA.S.; STACHNISS.C.; PFAFF.P.; BURGARD.W. Efficient estimation of accurate maximum likelihood maps in 3D. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2007, San Diego. **Proceedings…** Piscataway: IEEE, 2007.

HAN, C.; XIANG, Z.; LIU, J.; WU, E. Stereo vision based SLAM in outdoor environments. In: INTERNATIONAL CONFERENCE ON ROBOTICS AND BIOMIMETICS, 2007, Sanya. **Proceedings…** [S. l.: s. n.], 2007.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: Vision Conference, 4., 1988, [S. l.]. **Proceedings…** [S. l.: s. n.], 1988. p. 147–151.

HARTLEY, R. I.; ZISSERMAN, A. **Multiple view geometry in computer vision.** 2. ed. Cambridge: Cambridge University, 2004.

HARTLEY, R.;STURM, P. Triangulation. **Comput Vis Image Underst**, London, v. 68, p. 146-157, 1997.

HENRY, P.; KRAININ, M.; HERBST, E.; REN, X.; FOX, D. RGB-D mapping: using Kinect-style depth cameras for dense 3D modeling of indoor environments. **The International Journal of Robotics Research**, London, v. 31, n. 5, 2012.

HENRY, P.; KRAININ, M.; HERBST, E.; REN, X.; FOX, D. **RGB-D mapping:** using depth cameras for dense 3d modeling of indoor environments. Berlin: Springer, 2014.

HOGMAN, V.; et al. **Build a 3D map from RGB-D sensors**. [S. l.]: Royal Institute of Tecnology, KTH, School of Computer Science and Communication (CSC), 2013.

HORNUNG, A.; WURM, K. M.; BENNEWITZ, M.; STACHNISS, C.; BURGARD, W. OctoMap: an efficient probabilistic 3d mapping framework based on octrees. **Autonomous Robots**, New York, v. 34, n. 3, p. 189-206, 2013.

HUANG, S.; DISSANAYAKE, G. Convergence and consistency analysis for extended kalman filter based SLAM. **IEEE Transactions on Robotics**, Piscataway, v. 23, n. 5, 2007.

HUANG, S.; WANG, Z.; DISSANAYAKE, G. Sparse local submap joining filter for building large-scale maps. **IEEE Transactions on Robotics**, Piscataway, v. 24, n. 5, 2008.

JENSFELT, P.; KRISTENSEN, S. Active global localization for a mobile robot using multiple hypothesis tracking. **IEEE Transactions On Robotics And Automation**, Piscataway, v. 17, n. 5, 2001.

JEON, H.-K; JEONG, J-M; LEE,K-Y. **Implementation of the real:** time panoramic image stitching using ORB and PROSAC. Piscataway: ISOCC IEEE, 2015.

KARLSSON, N.; DI BERNARDO, E.; OSTROWSK, J.; GONCALVES, L PIRJANIAN, P.; MUNICH, M. E. The vSLAM algorithm for robust localization and mapping. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), IEEE, 2005, Piscataway. **Proceedings…** Piscataway: IEEE, 2005. p. 24–29.

KLEIN, G.; MURRAY, D. Parallel tracking and mapping for small ar workspaces. In: IEEE AND ACM INTERNATIONAL SYMPOSIUM ON MIXED AND AUGMENTED REALITY, 6., 2007, [S. l.]. **Proceedings…** Piscataway: IEEE, 2007.

KOESER, K.; BARTCZAK, B.; KOCH, R. An analysis-by-synthesis camera tracking approach based on free-form surfaces. In: GERMAN CONF. ON PATTERN RECOGNITION (DAGM), 2007, [S. l.]. **Proceedings…** [S. l.: s. n.], 2007.

KONOLIGE.K.; AGRAWAL. M. FrameSLAM: from bundle adjustment to real-time visual mapping. **IEEE Transactions on Robotics,** Piscataway, v. 25, n. 5, p. 1066–1077, 2008.

KONOLIGE, K.; BOWMAN, CHEN, J. D.; CALONDER, M. **View-based maps**. Proceedings of robotics: Science and systems, 2009.

KÜMMERLE, R., GRISETTI, G.; STRASDAT, H.; KONOLIGE, K.; BURGARD, W. G2O: A general framework for graph optimization. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), IEEE, 2011, Piscataway. **Proceedings…** Piscataway: IEEE, 2011. p. 3607-3613.

LEMAIRE, T.; BERGER, C.; JUNG, I.; LACROIX, S. Vision-based slam: stereo and monocular approaches. **International Journal of Computer Vision**, New York, v. 74, p. 343-364, 2007.

LEONARD, J. J.; DURRANT-WHYTE, H. Simultaneous map building and localization for an autonomous mobile robot. In: IEEE/RSJ INTERNATIONAL WORKSHOP ON INTELLIGENT ROBOTS AND SYSTEMS IROS '91, [S. l.], 1991. **Proceedings…** [S. l.: s. n.], 1991.

LIM, H.; LIM, J.; KIM, J. Real-time 6-DOF monocular visual SLAM in a large-scale environment. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS & AUTOMATION (ICRA), 2014, Hong Kong. **Proceedings…** [S. l.: s. n.], 2014.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. **International Journal of Computer Vision**, New York, v. 60, n. 2, p. 91-110, 2004.

MEI, C.; FLINT, A.; REID, I..; MURRAY, D. Growing semantically meaningful models for visual SLAM. IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2010, [S. l.]. **Proceedings…** [S. l.: s. n.], 2010. p. 467-474.

MIGLINO, O.; LUND, H. H.; NOLFI, S. Evolving mobile robots in simulated and real environments. **The MIT Press Journals**, Cambridge, v. 2, n. 4, p. 417-434, 2010.

MONTEMERLO, M.; THRUN, S.; KOLLER, D.; WEGBREIT, B. FastSLAM:a factored solution to the simultaneous localization and mappingproblem. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI), 2002, [S. l.]. **Proceedings…** [S. l.: s. n.], 2002.

MORAVEC, H. P. Towards automatic visual obstacle avoidance. INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI), 1997, [S. l.]. **Proceedings…** [S. l.: s. n.], 1997.

NEWCASTLE & MICROSOFT. **RGB-D Dataset 7-Scenes**. [S. l.], 2017. Disponível em: <https://www.microsoft.com/en-us/research/project/rgb-d-dataset-7-scenes>. Acesso em: 1 abr. 2017.

NISTER, D.; NARODITSKY, O.; BERGEN, J. Visual Odometry. COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR'04), 2004, Washington. **Proceedings…** Washington: [s. n.], 2004.

NÜCHTER, A.; LINGEMANN, K.; HERTZBERG, J.; SURMANN, H. 6D SLAM– 3D mapping outdoor environments: research articles. **J. Field Robotic**., Hoboken, v. 24, p. 699–722, 2007.

OLSON, C.; MATTHIES, L. H; SCHOPPERS, M. Rover navigation using stereo ego-motion. **Robot Autonom Syst**, Amsterdam, v. 43, p. 215-229, 2003.

PAZ, L.; PINIES, P.; TARDOS, J.; NEIRA,J. Large-scale 6dof slam with stereo-in-hand. **IEEE Transactions on Robotics**, Piscataway, v. 24, p. 946-957, n. 5, 2008.

PEÑAFIEL, D. S. A.; PEREIRA, G. A. S. Geração de mapas para localização e navegação de um manipulador móvel usando múltiplos sensores. In: Congresso Brasileiro de Automática, 20., [S. l.]. **Anais...** [S. l.: s. n.], 2013. p. 1- 6.

PINIES. P.; TARDOS, J. Large scale slam building conditionally independent local maps: application to monocular vision. **IEEE Transaction on Robotics**, Piscataway, v. 24, p. 1094-1106, 2008.

PINHEIRO, D.; ALSINA, P.; MEDEIROS, A. D. Um método de geração de trajetória para robôs não-holonômicos com acionamento diferencial. In: SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE, 6., 2003, [S. l.]. **Proceedings...** [S. l.: s. n.], 2003.

POZNAN UNIVERSITY OF TECHNOLOGY**. PUTKK:** PUT Kinect 1 & Kinect 2 data set. Disponível em: <http://lrm.put.poznan.pl/putkk/>. Acesso em: 1 abr. 2017.

PRASSLER, E.; KOSUGE, K. **Domestic robotics**. In: SPRINGER HANDBOOK OF ROBOTICS, doi: 10.1007/978-3-540-30301-5_55, p. 1253-1281, 2008.

ROCHA. R. P. **Building volumetric maps with cooperative mobile robots and useful information sharing:** a distributed control approach based on enthropy. 2006. Tese (Doutorado) - Universidade Federal do Rio Grande do Norte, Natal, 2006.

ROMERO, A. M.; CARZOLA, M. Comparativa de detectores de características visuales y suaplicación al SLAM. In: WORKSHOP DE AGENTES FISICOS, 2009, Caceres. **Proceedings...** [S. l.: s. n.], 2009.

ROMERO, R. **Robótica móvel.** Rio de Janeiro: LTC, 2014.

ROSTEN, E.; DRUMMOND, T. Machine learning for high speed corner detection. In: EUROPEAN CONFERENCE ON COMPUTER VISION, 9., [S. l.], 2006. **Proceedings…** [S. l.: s. n.], 2006. p. 430-443.

RUBLEE, E.; RABAUD, V.; KONOLIGE, K.; BRADSKI, G. ORB: an efficient alternative to SIFT or SURF. IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2011, [S. l.]. **Proceedings…** [S. l.: s. n.], 2011. p. 1-8.

SANTANA, A. M.; AIRES, K. R. T.; VERAS, R. M. S.;MEDEIROS, A. A. **An approach for 2D visual occupancy grid map using monocular vision**. [S. l.]: Elsevier, 2011.

SANTANA, A. M. **Localização e mapeamento simultâneos de ambientes planos usando visão monocular e representação híbrida do ambiente.** 2011. Tese (Doutorado) - Universidade Federal do Rio Grande do Norte, Natal, 2011.

SE, S., LOWE, D.; LITTLE, J. Vision-based mobile robot localization and mapping using scale-invariant features. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), IEEE, 2001, [S. l.]. **Proceedings…** [S. l.: s. n.], 2001. p. 2051–2058.

SELVATICI, A. H. P. **Construção de mapas de objetos para navegação de robôs**. 2009. Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo, São Paulo, 2009.

SENSOR-WORKSHOP. **Sensor kinect.** [S. l.], 2016. Disponível em: <https://itp.nyu.edu/archive /physcomp-spring2014/sensors/Reports/Kinect.html>. Acesso em: 10 fev. 2016.

SHATKAY, H.; KAELBLING, L. P. Learning topological maps with weak local odometric information. In: INTERNATIONAL JOINT CONFERENCE OF ARTIFICIAL INTELLIGENCE, 1997, [S. l.]. **Proceedings...** [S. l.: s. n.], 1997. p. 920-929.

SILVEIRA, G.; RIVES, P. Registro direto de imagens para SLAM visual. In: SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE (SBAI), 2009, [S. l.]. **Proceedings...** [S. l.], 2009.

SOUZA, A. A.; SANTANA, A. BRITTO, R .S.; GONÇALVES, L. M.; MEDEIROS, A. A. Representation of odometry errors on occupancy grids. In: INTERNATIONAL CONFERENCE ON INFORMATICS IN CONTROL, AUTOMATION AND ROBOTICS (ICINCO'08), 2008, Madeira. **Proceedings…** [S. l.: s. n], 2008.

STRASDAT, H.; MONTIEL, J.; DAVISON, A. Scale drift-aware large scale monocular SLAM. In: PROC. OF ROBOTICS: SCIENCE AND SYSTEMS (RSS), 2010, [S. l.]. **Proceedings…** [S. l.: s. n.], 2010. p. 1-8.

SHI, J.; TOMASI, C. Good features to track. **IEEE Conference on omputer Vision and Pattern Recognition** (CVPR), IEEE, p. 593–600, 1994

SMITH, R. C.; CHEESEMAN, P. On the representation and estimation of spatial uncertainty. **The International Journal of Robotics Research**, London, v. 5, n. 4, p. 56-68, 1986.

SNAVELY, N.; SEITZ, S.; SZELISKI, R. Photo tourism: exploring photo collections in 3D. **ACM Transactions on Graphics,** New York, p. 1-12, 2006.

STUEHMER, J.; GUMHOLD, S.; CREMERS, D. **Real-time dense geometry from a handheld camera**. Berlin; Heidelberg: Springer-Verlag, 2010. p. 11–20.

SUJAN, V.; MEGGIOLARO, M.; BELO, F. Mobile robot localization and mapping using lowcost vision sensors. **Springer Tracts in Advanced Robotics**, Heidelberg, p. 259-266, 2005.

TECHNICAL UNIVERSITY OF MUNICH. **Computer vision group**. [S. l.], 2017. Disponível em: <https://vision.in.tum.de/data/datasets/rgbd-dataset/download>. Acesso em: 1 abr. 2017.

THRUN, S. Robotic Mapping: a survey. **Exploring artificial intelligence in the new millenium**. [S. l.]: Morgan Kaufmann, 2002a.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic robotics.** Cambridge MA: MIT, 2005.

THRUN, S. Learning metric-topological maps for indoor mobile robot navigation. **Artificial Intelligence,** Amsterdam, v. 99, p. 21-71, 1998.

TOMONO, M. 3D localization based on visual odometry and landmark recognition using image edge points. IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2010, [S. l.]. **Proceedings…** [S. l.: s. n.], 2010. p. 5953–5959.

TRIGGS, B.; MCLAUCHLAN, P.; HARTLEY, R.; FITZGIBBON, A. **Bundle Adjustment A Modern Synthesis**. Vision Algorithms: Theory & Practice, B. Triggs, A. Zisserman& R. Szeliski (Eds.), Springer-Verlag LNCS 1883, 2000.

TRUCCO, E.; VERRI, A. **Introductory techniques for 3D computer Vision.** Upper Saddle River: Prentice Hall PTR, 1998.

TUNSTEL, E.; MAIMONE, M.; TREBI-OLLENNU, A.; YEN, J. Mars Exploration Rover mobility and robotic arm operational performance. IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 2005, Waikoloa. **Proceedings…** [S. l.], 2005. p. 1-8.

TUYTELAARS, T.; MIKOLAJCZYK, K. Local Invariant feature detectors: a survey. **Foundations and Trends,** New York, v. 3, n. 3, p. p 177-280, 2008.

WANG, C.; THORPE, CH.; THRUN, S. Simultaneous localization, mapping and moving object tracking**. J Robot Res**, London, v. 26, p. 889-916, 2007.

ZHOU, X.; MASON, G. **A study of microsoft Kinect calibration.** Mason: Dept. of Computer Science George Mason University, June 2, 2012

APÊNDICE A – Software e Hardware

Neste apêndice apresentam-se os programas *open-source* utilizados para o desenvolvimento do trabalho e as especificações da câmera e BBB usados no robô para os testes deste projeto.

A.1 SOFTWARE

Para o processamento das imagens e desenvolvimento do trabalho três programas são usados, o OPENCV, PCL e ROS para Linux.

Na computação visual, o tratamento das imagens obtidas por câmeras é usado frequentemente, o software livre *OpenCV* (*Open-Source Computer Vision*), simples e amigável, uma biblioteca de código aberto oferece muitas funcionalidades escritas em *C++* e *Python* para o processamento de imagens.

O *OpenCV* é licenciado pela *Berkeley Software Distribution* (BSD) biblioteca desenvolvida pela organização *Intel*, apresentando estrutura modular, permitindo uma análise estrutural, análises de movimento, reconhecimento de modelos, etc.

O *Point Cloud Library* **(**PCL**)** também é um código livre licenciado pela BSD para processamento de imagens 2D e 3D, considerada uma biblioteca que contém numerosos algoritmos de filtro, reconstrução de superfícies, segmentação em árvores, divididos em uma série de módulos que podem ser compilados de forma separada. PCL é multiplataforma compilado no sistema Linux, Windows e outros.

Foi usada também uma plataforma de desenvolvimento open-source, o *Robot Operating System* (ROS) que é outro recurso importante para o desenvolvimento de software livre para robôs, fornecendo bibliotecas para aplicações complexas, tais como, abstração do hardware e controle de dispositivos de baixo nível (programação, depuração, execução e visualização). Facilita o desenvolvimento de software com robôs, compartilha mensagens (informação) entre módulos ou nós, sem submeter-se a alguma linguagem de programação, ou seja, tem independência de linguagem e plataforma, é composto por pacotes que podem ser gerenciados.

Basicamente o ROS é um sistema cliente/servidor com programas que se comunicam interativamente por meio de tópicos e serviços. Neste trabalho usa-se como plataforma de suporte, o Ubuntu-Linux, para todos os programas.

A.2 HARDWARE

Para o hardware descrevem-se as principais características do sensor Kinect, a plataforma de desenvolvimento para o sistema de controle e a placa de acionamento, usada no protótipo.

A.2.1 Sensor Kinect

O sensor Kinect, Figura 41, é um dispositivo desenvolvido pela Primesense inicialmente destinado ao Microsoft Xbox 360, (lançado em novembro de 2010). É composto por uma câmera RGB, e sensores de profundidade 3D, um conjunto de microfones e um motor para dar inclinação. Neste trabalho somente a câmera e os sensores são utilizados. Têm-se como principais características:

- Uma câmera RGB onde cada cor é decodificada em strings de 8 bits, somando um total de 24 bits para as três cores. Resolução VGA de 640X480 pixels, a 30 frame/sec (quadros por segundo);
- Um sensor de profundidade 3D composto de um emissor de luz infravermelha (IR) capturado por um sensor de imagem CMOS monocromático e decodificado para produzir a profundidade (distância), com dados de saída de 11 bits. A resolução do sensor de profundidade é de 320x240, com taxa de 30Hz; ângulo de visão de 43^0 vertical por 57^0 horizontal, com faixa de inclinação vertical de $\pm 27^0$;
- Formato do áudio: quatro microfones, 16bits, taxa de amostragem de 16kHz , 24 bits PCM (*Mono Pulse Code Modulation*).

Figura 41 - Sensor Kinect

Fonte: Sensor-Workshop (2016)

A.2.2 Plataforma de desenvolvimento

Uma *Single Board Computer*, *Beagle Bone Black* Rev-C (BBB), Figura 42, foi usada para o desenvolvimento do programa de controle do robô. Pertence a família Beagle Board, é baseado no processador XAM3359AZCZ100 Cortex A8 ARM da classe Sitara de Texas Instruments, CPU de 2 núcleos que executa instruções a 1GHz, 512MB de RAM DDR3 e um processador gráfico para 3D. Na Tabela 6 se apresentam especificações mais detalhadas desta BBB.

Figura 42 – *Beagle Bone Black* Rev-C

Fonte: *BeagleBoard* (2017).

Tabela 6 - Especificações detalhadas da BBB.

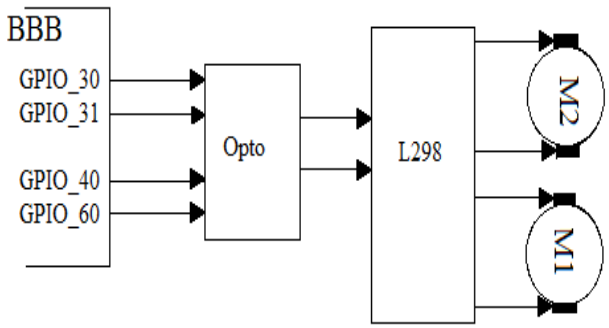
Processador	1Gz Sitara AM3359AZCZ100	
Memória SDRAM	512 MB DDR3L 400 Mhz	
Flash	4GB, 8bit Embedded MMC	
Alimentação	Mini USB or Jack DC	Alimentação: 210-460mA@5V
Indicadores	Power, Ethernet, User Leds	
Porto Serial	UART0 - 3.3V TTL.	
Ehternet	10/100, RJ45	
SD/Conector MMC	MicroSD, 3.3V	
Saída de Vídeo	16b HDMI, 1280×1024 (MAX) 1024×768,1280×720,1440x900 w/Suporte EDID	
Áudio	HDMI, Stereo	
Conectores de Expansão	Power 5V, 3.3V VDD_ADC (1.8V) 3.3V I/OMcASP0, SPI1, I2C, GPIO (69), LCD, GPMC, MMC1, MMC2, 7 AIN(1.8 MAX) , 4 Timers, 4 Serial Port, CAN0, EHRPWM(0.2) , Interrupções XDMA.	

Fonte: Beagle Board (2017).

A.2.3 Placa de acionamento

Uma placa de acionamento foi projetada com drivers para os motores DC, um LM298; têm-se as ligações dos encoders e os optoacopladores para a conexão dos sensores. Os pinos, *General Purpose Input/output* (GPIO) da BBB fazem a comunicação com a placa de acionamento, conforme mostrado na Figura 43.

Figura 43 - Comunicação entre placa de acionamento e a BBB



Fonte: Própria do autor.

Apêndice B - Pseudocódigos

B.1 Algoritmo – BFM

Algoritmo 1 BFM

Entrada: Descritor do ponto de interesse.

Saída: *Ponto mais perto*.

1: T ← T[0... n-1]

2: P ← P[0... n-1]

3: **para** *i*← 0 a *n-m* **fazer**

4: *j*← 0.

5: **enquanto** *j*< *m* **e** *P[j]=T[i+j]***fazer**

6: *j*←*j +1*

7: **fim enquanto**

8: **se** *j = m* retornar *i*

9: **fim para**

B.2 Algoritmo – RANSAC

Quando se deseja ter uma correspondência entre pontos de dois frames consecutivos e aparecem erros no processo, estes devem ser rejeitados, um dos métodos mais popular usado é o RANSAC que é um estimador robusto.

Algoritmo 2 RANSAC

Entrada: Conjunto de dados de pontos

Saída: Melhores Correspondências, Melhor Modelo

1: Definir o número de iterações N.
2: **para** i = 1 até N fazer
3: Amostras ← selecionar *k* pontos aleatoriamente.
4: Calcular *modelo atual* das Amostras.
5: *Inliers* ← 0.
6: **para** Todos os pontos **fazer**
7: Avaliar *modelo atual* para o ponto e calcular seu *erro*
8: **sim** erro < *Limiar* **então**
9: *Inliers* ← *Inliers* + ponto
10: **fim sim**
11: **fim para**
12: Contar o número de *Inliers* e calcular seu erro médio.
13: **sim** *modelo atual* e valido então
14: Recalcular *modelo atual* das Inliers
15: **sim** *modelo atual* é melhor que Melhor Modelo então
16: *Melhor Modelo* ← *modelo atual*
17: *Melhores Inliers* ← *Inliers*
18: **fim sim**
19: **fim sim**
20: **fim para**
21: **retornar** *Melhor Modelo, Melhores Inliers*.

B.3 Algoritmo – ICP

O algoritmo considera duas nuvens de pontos $\boldsymbol{Q} = q_i$ e $\boldsymbol{P} = p_i$ e uma transformação inicial, τ_0 aproxima a nuvem *Q* até *P*, retornando uma transformação que melhor alinha *P* com *Q* ao resolver o problema de minimização. Obtém, portanto, um mínimo local que pode não convergir se as nuvens não estão muito próximas. Mostra-se no Algoritmo 3 o pseudocódigo padrão do ICP.

Algoritmo 3 *Iterate Closest Point* - ICP

Entrada: Nuvens de pontos $Q = q_1, q_2, q_3, \dots\dots, q_N$, $P = p_1, p_2, p_3, \dots\dots, p_N$

1: Estimativa inicial : Transformação τ_0

Saída: *Transformação correta* τ.

2: $\tau = \tau_0$.

3: **enquanto** ainda não converge **fazer**

4: **para** i = 1 até N **fazer**

5: d_i= Encontrar o ponto mais próximo $D(\tau * s_i)$

6: **se** $\|d_i - \tau * s_i\| < e_{max}$ **então**

7: $w_i = 1$

8: **se não**

9: $w_i = 0$

10: **fim se**

11: **fim para**

12 $\tau = \text{argmin}_\tau\{\sum_i w_i \|\tau * s_i - d_i\|^2\}$

13: **fim enquanto**

B.4 Algoritmo – PROSAC

Devido a eficiência na eliminação de dados atípicos e na redução de tempo de processamento por ordenamento linear de um conjunto de correspondências. Este algoritmo é implementado para a criação de mapas 3D.

Algoritmo 4 PROSAC

Entrada: t:=0, n:=m, n* :=N

Repetir até a solução, até que a probabilidade seja menor a um limiar determinado.

1: Escolher o conjunto para **gerar a hipótese**

2: t := t+1

3: **se** $(t = T_n^{'})$& (n < n*) **então** n:=n+1

4: **semi-amostragem aleatório** $\mathcal{M}_t$ **de tamanho** *m*

5: **se** $(T_n^{'} < t)$**então**

6: As amostras contem *m-1* pontos escolhidos de aleatoriamente $\mathcal{U}_{\mathrm{n}-1}$ e $\mathrm{u_n}$

7: **senão**

8: Seleção de *m* pontos de $\mathcal{U}_{\mathrm{n}}$ aleatoriamente

9: **Modelo de estimação de parâmetros**

10: Cálculo dos parâmetros do modelo P_t da amostra $\mathcal{M}_t$

11: **Modelo de Verificação**

12: Encontrar pontos de dados consistentes do modelo de parâmetros P_t

13: Selecionar o tamanho do Critério de parada.
