Sistema FIEB | SENAI CIMATEC

PELO FUTURO DA INOVAÇÃO

PROGRAMA DE MESTRADO EM MODELAGEM COMPUTACIONAL E TECNOLOGIA INDUSTRIAL

Mestrado em Modelagem Computacional e Tecnologia Industrial

Dissertação de Mestrado

Um ROV de baixo custo como instrumento para pesquisas de algoritmos e métodos aplicáveis à robótica subaquática

Apresentada por: Jótelly Barros Oliveira

Orientador: Roberto Luiz Souza Monteiro

Salvador, 2021

Jótelly Barros Oliveira

Um ROV de baixo custo como instrumento para pesquisas de algoritmos e métodos aplicáveis à robótica subaquática

Dissertação de Mestrado apresentado ao Programa de Mestrado em Modelagem Computacional e Tecnologia Industrial, Curso de Mestrado em Modelagem Computacional e Tecnologia Industrial do , como requisito parcial para a obtenção do título de **Mestre em Modelagem Computacional e Tecnologia Industrial**.

Área de conhecimento: Interdisciplinar

Orientador: Roberto Luiz Souza Monteiro

Coorientadora: Ingrid Winkler

Salvador
2021

Dedico este trabalho a todos aqueles que tiveram coragem de ousar para além do impossível e fizeram da humanidade um lugar melhor para se viver.

Ficha catalográfica elaborada pela Biblioteca do Centro Universitário SENAI CIMATEC

O48r Oliveira, Jótelly Barros

Um ROV de baixo custo como instrumento para pesquisas de algoritmos e métodos aplicáveis a robótica subaquática / Jótelly Barros Oliveira. – Salvador, 2021.

60 f. : il. color.

Orientador: Prof. Dr. Roberto Luiz Souza Monteiro.
Coorientadora: Prof.ª Dr.ª Ingrid Winkler.

Dissertação (Mestrado em Modelagem Computacional e Tecnologia Industrial) – Programa de Pós-Graduação, Centro Universitário SENAI CIMATEC, Salvador, 2021.
Inclui referências.

1. Visual odometry. 2. Egomotion. 3. Robot vision. 4. SLAM. 5. Underwater. I. Centro Universitário SENAI CIMATEC. II. Monteiro, Roberto Luiz Souza. III. Winkler, Ingrid. IV. Título.

CDD 620.00113

NDI - 02

Sistema FIEB | SENAI CIMATEC
PELO FUTURO DA INOVAÇÃO

CENTRO UNIVERSITÁRIO SENAI CIMATEC

Mestrado Acadêmico em Modelagem Computacional e Tecnologia Industrial

A Banca Examinadora, constituída pelos professores abaixo listados, aprova a Defesa de Mestrado, intitulada **"Um ROV de Baixo Custo como Instrumento para Pesquisas de Algoritmos e Métodos aplicáveis à Robótica Subaquática**" apresentada no dia 29 de julho de 2021, como parte dos requisitos necessários para a obtenção do Título de Mestre em Modelagem Computacional e Tecnologia Industrial.

DocuSigned by:
ROBERTO LUIZ SOUZA MONTEIRO
AF95B2A4FD394A6...

Orientador: **Prof. Dr. Roberto Luiz Souza Monteiro**
SENAI CIMATEC

DocuSigned by:
EB83F32E864047D...

Coorientadora: **Prof.ª Dr.ª Ingrid Winkler**
SENAI CIMATEC

DocuSigned by:
ADA429E9E2B549F...

Membro Interno: **Prof. Dr. Marcelo Albano Moret Simões Gonçalves**
SENAI CIMATEC

DocuSigned by:
D3928C524BD546D...

Membro Externo: **Prof. Dr. Eduardo Manuel de Freitas Jorge**
UNEB

Agradecimentos

À minha **Família**, minha mãe **Rita Marques**, meu irmão **Rodrigo Barros**, sinônimo de amor e união. Obrigado por acreditarem no meu sonho e sempre me motivar a seguir em frente. É muito bom saber que posso contar com vocês em todos os momentos. Amo vocês!

À minha doce e amável filha **Kyara Barros** a qual me deu forças para continuar na batalha, te amo muito.

A minha cadela **Mel** que sempre esta ao meu lado.

Ao meu orientador, Prof. Dr. **Roberto Luiz Souza Monteiro** , pela oportunidade de realizar este trabalho. Obrigado pela confiança e por me atender com paciência todas as vezes que bati em sua porta. Agradeço por todos os ensinamentos compartilhados de forma admirável, e por me guiar nos primeiros passos da graduação. Muito obrigado por tudo!

A minha coorientador(a), A Prof^{a}. **Ingrid Winkler** , por toda a ajuda durante a realização deste trabalho. Sua contribuição foi essencial para a concretização de todas as pesquisas desenvolvidas neste Programa de mestrado. Muito obrigado.

Aos amigos do Senai CIMATEC, **Renata Barreto**, **Lucas Marins**, **Rafael Queiroz**, **Eduardo de Brito**, **Branilson Luiz**, **Ivan Tarifa**, **Mateus Amarantes**, **Erick Cerqueira**, **Mateus Nascimento**, e muitos outros amigos que tenho eterna gratidão, nós passamos mais tempo juntos do que em nossa casa, com a nossa família. Conviver com vocês ao longo desses dois anos foi sensacional. Muito obrigado por toda forma de ajuda, pela companhia durante um café, pelas inúmeras conversas e risadas. Vocês são muito especiais e tornam o trabalho muito mais agradável.

Resumo

No mercado atual modelos de Veículo submarino operado remotamente (ROVs) com funcionalidades de transmitir imagens em tempo real com baixa latência possuem um alto custo financeiro, iniciando na faixa dos R$ 4 mil reais, isso para os modelos básicos, que também são chamados de modelo de entrada a exemplo o (*OpenROV Trident Underwater Drone*) ou (*CHASING Dory Underwater Drone*). O valor elevado atribuído aos ROVs para sua construção, dificulta as pesquisas envolvendo visão computacional para navegação e coletas de dados em ambiente subaquático entre estudantes, professores e instituições de ensino, sobretudo escolas e universidades públicas. Esta pesquisa demonstra um equipamento de baixo custo para estudos envolvendo visão computacional, navegação e coletas de dados em ambiente subaquático, além de ter realizado experimentos utilizando o ROV envolvendo os algoritmos de odometria visual, sendo eles: LSD-SLAM: *Large-Scale Direct Monocular SLAM*, ORB_Slam, e SVO (*Semi-Direct Monocular Visual Odometry*), mantendo as informações acuradas das condições climáticas durante os expe rimentos. Ao longo deste estudo, foi possível concluir que o algoritmo que obteve um melhor resultado em termos de acurácia e precisão em um ambiente simulado controlado foi o SVO (*Semi-Direct Monocular Visual Odometry*) que conseguiu alcançar os melhores resultados em 4 quatro testes de um total de 6 seis por sessão demonstrando a sua superioridade em relação ao LSD_Slam e o ORB_Slam, este que por sua vez na média geral conseguiu o melhor resultado no desvio padrão médio cerca de 0.09, indicando que dentre os outros possui menos variações entre um teste e outro.

Palavras-chave: Visual Odometry, Egomotion, Robot Vision, SLAM, Underwater.

Abstract

In the current market, Remotely Operated Submarine Vehicles (ROVs) with features of transmitting images in real time with low latency have a high financial cost, starting in the range of BRL 4,000 reais, this for the basic models, which are also called template input such as (*OpenROV Trident Underwater Drone*) or (*CHASING Dory Underwater Drone*). The high value attributed to ROVs for their construction makes research involving computer vision for navigation and data collection in an underwater environment difficult among students, professors and educational institutions, especially schools and public universities. This research demonstrates a low-cost equipment for studies involving computer vision, navigation and data collection in an underwater environment, in addition to having carried out experiments using the ROV involving visual odometry algorithms, namely: LSD-SLAM: *Large- Scale Direct Monocular SLAM*, ORB_Slam, and SVO (*Semi-Direct Monocular Visual Odometry*), keeping accurate weather information during experiments. Throughout this study, it was possible to conclude that the algorithm that obtained the best result in terms of accuracy and precision in a simulated controlled environment was the SVO (*Semi-Direct Monocular Visual Odometry*) which managed to achieve the best results in 4 four tests of a total of 6 sixes per session demonstrating its superiority in relation to LSD_Slam and ORB_Slam, which in turn in the overall average achieved the best result in the average standard deviation of about 0.09, indicating that among the others have less variation between one test and another.

Keywords: Visual Odometry, Egomotion, Robot Vision, SLAM, Underwater.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo Um

Introdução

A possibilidade da criação de um mapa e sua trajetória construída por imagens contribui para o aprimoramento da visão computacional, concomitantemente, surge novas dificuldades para o seu bom funcionamento.

Neste estudo desenvolvemos um modelo de *Remotely Operated Vehicle* (ROV), utilizando-o como instrumento de pesquisa, qual apelidamos de LowcostROV, nome esse que faz referência ao seu baixo custo de projeto em relação a modelos do mercado atual, onde foi possível abordar os principais algoritmos de odometria visual da atualidade em pesquisas científicas, testando e validando os resultados em um ambiente simulado controlado com cada algoritmo implementado. É importante perceber que o assunto em questão está em constante evolução sendo aperfeiçoado cada dia mais.

1.1 Justificativa

Sabe-se que atualmente os (ROV) estão sendo bastante utilizados para a limpeza de navios em ambientes de difícil acesso, gerando assim um custo alto em sua fabricação. Compreendemos que esses altos custos se devem a qualidade do material do ROV, que precisa ser de excelência para lidar com grandes interferências ópticas devido ao movimento natural da água, redução de transparência (turbidez), variações na intensidade de luz com profundidade, distribuição não homogênea de pressão, além dos movimentos dos peixes e riscos de desconexão do sinal de localização do dispositivo, com possível perda do mesmo em situações extremas (SÀNCHEZ; VARGAS; COUCEIRO, 2018).

Sendo assim, a importância deste estudo consistiu em desenvolver um ROV de baixo custo para experimentos envolvendo robótica subaquática, além de viabilizar pesquisas em ambiente subaquático para estudantes e instituições de ensino com pouco capital disponível.

1.2 Definição do Problema

No mercado atual modelos de ROVs com funcionalidades de transmitir imagens em tempo real com baixa latência, controlados remotamente com estabilização de movimento, pos-

suem um valor muito alto, ultrapassando facilmente os R$ 4 mil reais, isso para os modelos básicos, que também são chamados de modelo de entrada. É possível encontrar no mercado ROVs como o (OpenROV Trident Underwater Drone) ou (CHASING Dory Underwater Drone), mesmo custando menos que os Rovs mais robustos ambos ainda possuem o preço um pouco elevado. O alto valor atribuído aos ROVs dificulta muito sua construção e por consequência o estudo/pesquisa entre estudantes e professores.

Devido aos custos elevados para aquisição de ROVs para pesquisas envolvendo visão computacional, navegação e coletas de dados em ambiente subaquático, vem se tornando cada vez mais difícil a realização dessas pesquisas por estudantes e instituições de ensino, sobretudo escolas e universidades públicas.

Esta pesquisa teve como propósito oferecer uma alternativa de baixo custo e alta eficiência ao desenvolvimento de ROVs.

1.3 Questão Norteadora

Como viabilizar pesquisas em ambiente subaquático para estudantes e instituições de ensino com pouco capital disponível, sobretudo quando esses estudos envolvem aquisição de dados precisos, tais como localização, imagem e leitura de variáveis climáticas. Mantendo ainda a segurança para o observador e equipamentos utilizados?

1.4 Objetivo geral

O objetivo principal desta pesquisa foi implementar um equipamento de baixo custo para estudos envolvendo visão computacional, navegação e coletas de dados em ambiente subaquático.

1.5 Objetivos específicos

Os objetivos específicos foram:

- Desenvolver um ROV de baixo custo para experimentos envolvendo robótica subaquática;
- Realizar experimentos envolvendo odometria visual utilizando o ROV;

- Demonstrar a acurácia dos sensores do ROV nos experimentos;
- Demonstrar a manobrabilidade do ROV em um ambiente controlado;
- Definir um padrão de marcadores fiduciais para calibração e realização dos experimentos;
- Manter informações acuradas das condições climáticas durante os experimentos;

1.6 Metodologia

A metodologia utilizada neste estudo foi realizar experimentos envolvendo odometria visual, utilizando os algoritmos Lsd-Slam, ORB-Slam, e *SVO (Semi-Direct Monocular Visual Odometry)*. O processo metodológico abordado nesta pesquisa possui cinco etapas e treze processos, ilustrados na Figura 1.1.

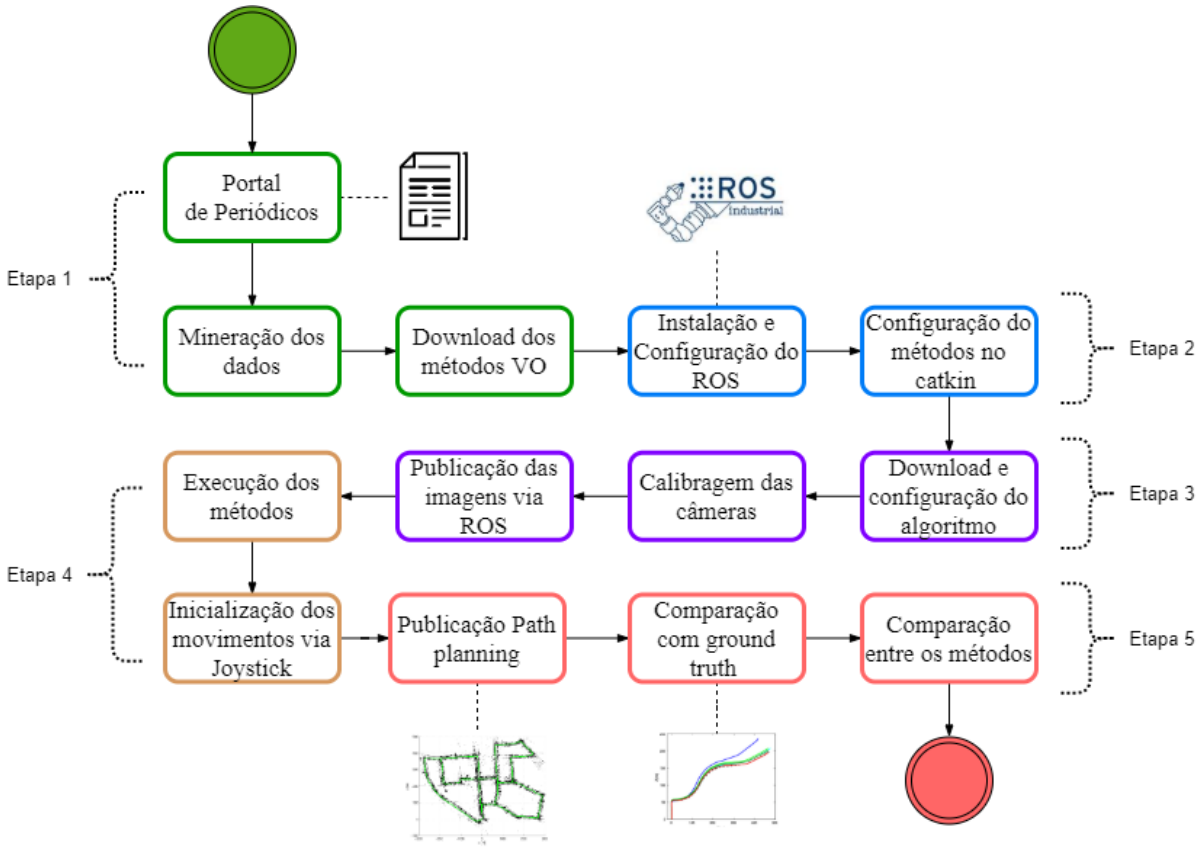


Figura 1.1: Fluxograma da metodologia para comparação dos resultados dos algoritmos utilizados. Fonte: Autor (2022)

A Etapa 1 (cor verde da Figura 1.1) abrange a elaboração do banco de dados abordando a odometria visual, seguindo os procedimentos em (OLIVEIRA et al., 2019) sendo possível extrair informações como os algoritmos mais bem avaliados ou citados, em seguida foi

feito o download e configuração dos algoritmos. Na Etapa 2 (cor azul da Figura 1.1) foi necessário efetuar a configuração de um espaço de trabalho, *workspace* ROS Catkin, o qual possibilitou editar e configurar vários algoritmos no mesmo lugar. A Etapa 3 (cor marrom da Figura 1.1) consistiu na calibração das câmeras para cada método proposto publicando imagens provenientes da câmera estéreo utilizada no modelo LowcostROV, sendo necessário para a resolução de problemas como: falta de foco ou excesso de distorção física que impossibilitaram a correta execução do método de odometria visual. Na Etapa 4 (cor roxa da Figura 1.1) o LowcostROV, nome do rov utilizado devido ao seu baixo custo, foi possível receber instruções provenientes de um controle (*joystick*), gerando informações do planejamento de movimento (*Path planning*) possibilitando a manipulação dos motores e garantindo o mapeamento baseado em sua real trajetória (*ground truth*). Por último, a Etapa 5 (cor vermelho da Figura 1.1), consistiu em analisar a trajetória gerada a partir dos algoritmos e comparação entre eles.

A conexão via ROS foi responsável por gerenciar todos os comandos, publicando e subscrevendo nos tópicos de seus respectivos algoritmos de localização e mapeamento simultâneos (SLAM). A trajetória criada foi exibida por meio do ambiente de simulação Rviz, mostrado na Figura 1.2, ao qual possibilitou efetuar ajustes e calibrações nos algoritmos utilizados.

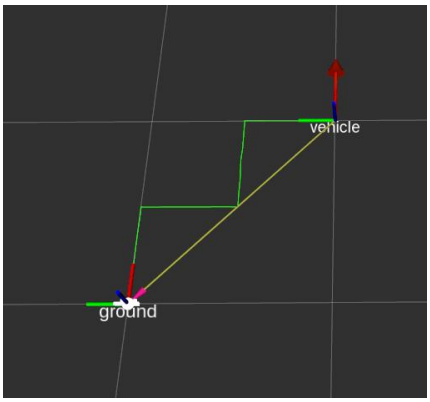


Figura 1.2: *Path planning* em relação ao *ground truth* do Rviz. Fonte: Autor (2022)

1.7 Organização do documento

- **Capítulo 1 - Introdução**: Contextualiza o âmbito, no qual a pesquisa proposta está inserida. Apresenta, portanto, a definição do problema, objetivos e justificativas da pesquisa e como este projeto de dissertação de mestrado está estruturado.;
- **Capítulo 2 - Revisão da literatura**: Traz uma revisão geral dos sistemas de odometria baseados em visão computacional.;

- **Capítulo 3 - Modelo proposto LowcostROV**: Apresenta o desenvolvimento do modelo LowcostROV proposto;
- **Capítulo 5 - Resultados e Discussão**: Compara os resultados obtidos com os algoritmos de odometria analisados.;
- **Capítulo 6 - Considerações Finais**: Apresenta a conclusão, contribuições e algumas sugestões de atividades de pesquisa a serem desenvolvidas no futuro.;

Capítulo Dois

Revisão da literatura

Esta seção discorre sobre a revisão bibliográfica dos estudos comparativos dos sistemas de odometria baseados em visão computacional aplicáveis a veículos autônomos subaquáticos.

2.1 Sistemas operacionais para robôs

De acordo com (ROS.ORG, 2018), O ROS é um conjunto de bibliotecas e ferramentas de software, responsáveis pelo gerenciamento de nós e pela comunicação entre si. Ele fornece os serviços que você esperaria de um sistema operacional, incluindo abstração de hardware, controle de dispositivo de baixo nível, implementação da funcionalidade comumente usada, passagem de mensagens entre processos e gerenciamento de pacotes.

2.2 Ferramentas de simulação

2.2.1 RViz e Gazebo (Ambiente de Simulação)

A ferramenta de visualização 3D para ROS tem o nome RViz, permitindo visualizar os dados do gazebo, possuindo plugins para vários tipos de tópicos disponíveis.

Segundo (GAZEBO, 2013), Gazebo é o simulador de física do mundo real, oferecendo a capacidade de simular com precisão e eficiência populações de robôs em ambientes internos e externos complexos.

Estrutura conceitual de arquivos:

- Xacro: é uma linguagem de macro XML, uma macro é uma sequência de entrada automatizada que imita pressionamentos de tecla ou ações do mouse;
- Yaml: Não é uma linguagem de marcação, é um padrão de serialização de dados amigável para humanos para todas as linguagens de programação;
- URDF: O *Universal Robotic Description Format*, é um formato de arquivo XML usado no ROS para descrever todos os elementos de um robô;

- SDF: O *Simulation Description Format*, ele próprio descrito usando XML, que facilita uma ferramenta de atualização simples para migrar versões antigas para novas versões, foi criado para uso no Gazebo para resolver as deficiências do URDF;

2.2.2 Parâmetros de posição e rotação no ambiente de simulação

A posição é: de cima para baixo, de um lado para o outro, de frente para trás (XYZ):

Enquanto a orientação dos eixos mundiais é fixa, seu ponto de vista no mundo não é, portanto esses eixos parecem mudar em relação ao seu ponto de vista. A posição de um objeto (valor de conversão) em qualquer eixo especificado é expressa como positiva ou negativa, com base em qual lado da origem está nesse eixo. Os eixos X, Y e Z do mundo são rotulados com Vermelho, Verde e Azul e corresponderão às cores R, G e B nos ícones das ferramentas. Na Figura 2.1 é mostrado a representação dos eixos.

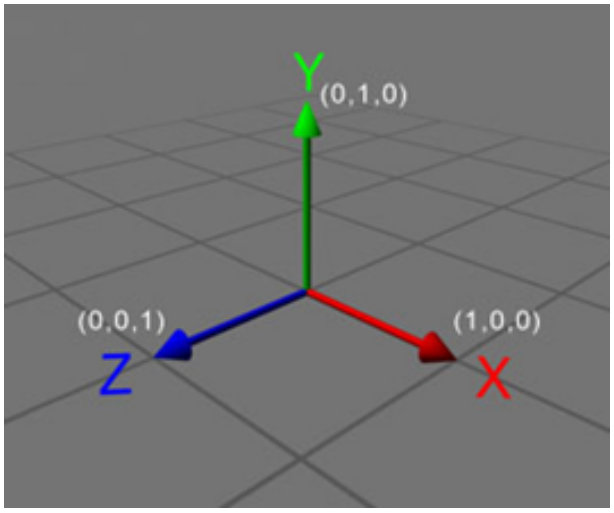


Figura 2.1: Representação dos eixos, X = Vermelho, Y = Verde e Z = Azuis. Fonte: Autor

Representação do espaço 3D virtual (XYZ):

- A Origem = O centro do palco;
- X positivo = No estágio esquerdo;
- X negativo = Na direita do palco;
- Y positivo = Acima do palco;
- Y negativo = Abaixo do estágio;
- Z positivo = Em direção ao palco / público;
- Z negativo = Rumo à fase inicial;

Rotação: Roll, Pitch, Yaw (RPY):

Os parâmetros são baseados no eixo/eixos nos quais deseja girar. Para virar a cabeça de um personagem levemente para baixo, trazendo o queixo em direção ao peito (supondo que o personagem tenha sido criado alinhado com os eixos do mundo), você alteraria o valor X de cima para baixo da cabeça, porque se você desenhar uma linha de orelha a orelha, isso seria o eixo em torno do qual você deseja girar que corresponde ao eixo X. essa regra é válida para todos os eixos.

Representação do espaço 3D virtual (RPY):

- Roll move para cima ou para baixo (eixos X);
- Pitch move para a esquerda ou para a direita (eixos Y);
- Yaw move para frente ou para trás (eixos Z);

2.3 Algoritmos de odometria visual

Egomotion ou Odometria visual, no inglês, *visual odometry* (VO), é abordada na área da visão computacional como sendo o processo para estimar a posição atual (pose) de um agente (por exemplo, veículo, humano e robô) usando apenas a entrada de uma ou várias câmeras anexadas a ele. Domínios de aplicação incluem robótica, computação vestível, realidade aumentada e automotiva (FRAUNDORFER; SCARAMUZZA, 2011). O VO é um caso particular de uma técnica conhecida como Estrutura do Movimento (SFM) que aborda o problema da reconstrução 3D da estrutura do ambiente e das poses da câmera a partir de conjuntos de imagens sequencialmente ordenados ou não-ordenados (YOUSIF; BAB-HADIASHAR; HOSEINNEZHAD, 2015).

De acordo com (FRAUNDORFER; SCARAMUZZA, 2011), o termo Visual Odometry (VO) foi criado em 2004 por Nister em seu documento de referência (NISTÉR; NARODITSKY; BERGEN, 2004). O termo foi escolhido por sua semelhança com a odometria das rodas, que estima incrementalmente o movimento de um veículo, integrando o número de voltas de suas rodas ao longo do tempo. Da mesma forma, o VO opera estimando incrementalmente a posição do veículo através das mudanças que o movimento induz nas imagens de seus sensores a bordo. A vantagem do VO em relação à odometria da roda é que o VO não é afetado pelo deslizamento da roda em terrenos irregulares ou outras condições adversas.

Foi demonstrado que, em comparação com a odometria da roda, o VO fornece estimativas de trajetória mais precisas, com erros de posição relativa variando de 0,1 a 2% (WIRTH;

CARRASCO; CODINA, 2013; FRAUNDORFER; SCARAMUZZA, 2011) - (NAWAF et al., 2017). Essa capacidade faz do VO um complemento interessante à odometria das rodas e, adicionalmente, a outros sistemas de navegação, como as Unidades de Medição Inercial (IMUs). A técnica de VO está se tornando popular nos AUVs para navegação, manutenção de estações e fornecimento de informações de feedback para manipulação. A saída da odometria visual é frequentemente combinada com as IMUs para fornecer uma alternativa mais barata aos DVLs e sistemas de navegação inercial (INS) (BELLAVIA; FANFANI; COLOMBO, 2017; WIRTH; CARRASCO; CODINA, 2013) e (YOUSIF; BAB-HADIASHAR; HOSEINNEZHAD, 2015).

Vale lembrar que as limitações da visão subaquática são amplamente conhecidas e seu desempenho depende de muitos fatores, como visibilidade, iluminação e distorção resultantes de diferentes índices de refração. De acordo com (FRAUNDORFER; SCARAMUZZA, 2011), para que o VO funcione efetivamente, deve haver iluminação suficiente no ambiente e uma cena estática com textura suficiente para permitir a extração de movimentos aparentes. Além disso, os quadros consecutivos devem ser capturados, garantindo que eles tenham sobreposição de cena suficiente.

Nos últimos anos, várias técnicas de VO foram propostas, que podem ser divididas em algoritmo para câmera monocular e estéreo (YOUSIF; BAB-HADIASHAR; HOSEINNEZHAD, 2015). Esses algoritmos ou métodos são posteriormente divididos em correspondência de recursos (recursos correspondentes em vários quadros), rastreamento de recursos (recursos correspondentes em quadros adjacentes) e técnicas de fluxo óptico (com base na intensidade de todos os pixels ou regiões específicas nas imagens sequenciais). De acordo com (WIRTH; CARRASCO; CODINA, 2013), o pipeline básico do algoritmo de odometria visual consiste nas seguintes etapas (independentemente do tipo de câmera): primeiro, os pontos-chave são identificados em cada quadro da câmera e os descritores de recursos para esses pontos são extraídos.

Em seguida, a profundidade de cada ponto de referência é estimada usando uma câmera estéreo, estrutura de movimento ou uma câmera de profundidade separada. Posteriormente, os recursos são correspondidos nos prazos e a transformação de corpo rígido que melhor alinha os recursos entre os quadros é estimada. O resultado desse processo é uma estimativa do movimento da câmera entre os quadros e, portanto, é necessário integrar esses dados ao longo do tempo para obter a posição e orientação absolutas do veículo. Normalmente, esse resultado final é refinado com uma otimização offline, ou seja, ajuste de pacote, cujo tempo de computação cresce com o número de imagens (FRAUNDORFER; SCARAMUZZA, 2011; NAWAF et al., 2017). A Figura 2.2 apresenta uma sequência típica para algoritmos VO de acordo com (YOUSIF; BAB-HADIASHAR; HOSEINNEZHAD, 2015).

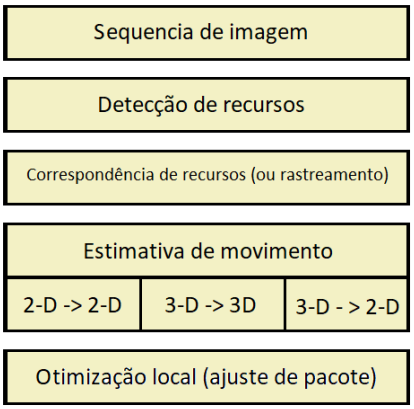


Figura 2.2: Um diagrama de blocos mostrando os principais elementos de um componente de odometria visual. Fonte: (YOUSIF; BAB-HADIASHAR; HOSEINNEZHAD, 2015).

2.3.1 Odometria visual monocular

De acordo com (YOUSIF; BAB-HADIASHAR; HOSEINNEZHAD, 2015), no VO monocular, os pontos característicos da imagem precisam ser observados em pelo menos três quadros diferentes (observe as características no primeiro quadro, re-observe e triangule em pontos 3D no segundo quadro e calcule a transformação no terceiro quadro), Os recursos de interesse geralmente são: cantos (corners), bordas (edges) e bolhas (blobs). Ao contrário dos sistemas de visão estéreo em que a transformação (rotação e translação) entre os dois primeiros quadros da câmera pode ser obtida, a transformação entre os dois primeiros quadros consecutivos na visão monocular não é totalmente conhecida e geralmente possui um valor predefinido. Em (WIRTH; CARRASCO; CODINA, 2013), esses problemas também são considerados e descritos. Segundo o autor, sistemas que usam apenas uma câmera precisam de movimento de translação para estimativa de movimento em 3D e todas as medições precisam ser dimensionadas por um fator desconhecido para estar em uma escala métrica.

A Figura 2.3 apresenta as posições relativas entre as câmeras que visualizam o mesmo ponto 3D, sendo calculados e combinados os pontos correspondentes na imagem 2D. Se a localização 3D dos pontos for conhecida, pode ser usado um método 3D para 3D ou 3D para 2D. As poses globais são calculadas concatenando as transformações relativas em relação a um quadro de referência (pode ser definido como o quadro inicial) (YOUSIF; BAB-HADIASHAR; HOSEINNEZHAD, 2015)

As abordagens de VO foram apresentadas também para o domínio subaquático (WIRTH; CARRASCO; CODINA, 2013). Usando pipelines que incluem rastreamento de recursos e estimativa de movimento, esses sistemas são capazes de calcular transformações planares (HUANG et al., 2017) ou seis graus de liberdade (DoF) (WIRTH; CARRASCO; CODINA, 2013; CORKE et al., 2007) incrementais da posição da câmera. Medidas inerciais também

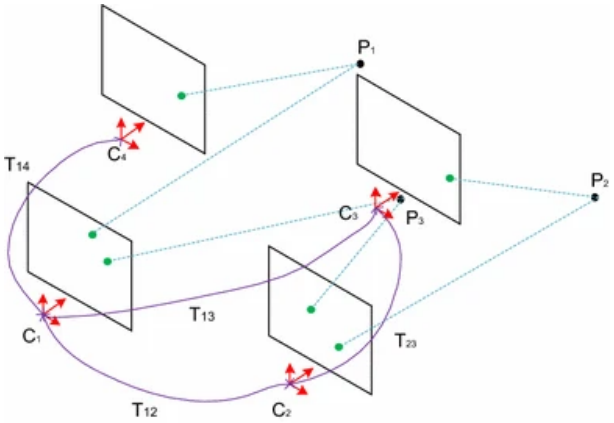


Figura 2.3: Um exemplo de sistema VO monocular. Fonte: (YOUSIF; BAB-HADIASHAR; HOSEINNEZHAD, 2015).

podem ser incluídas para melhorar o desempenho (CREUZE, 2017).

De acordo com (BELLAVIA; FANFANI; COLOMBO, 2017), os ambientes subaquáticos são tipicamente caracterizados por imagens não estruturadas, ruidosas e altamente texturizadas, com padrões repetitivos e más condições de iluminação local (efeitos de vinheta e outros artefatos). Consequentemente, os pontos-chave da imagem são difíceis de rastrear e corresponder corretamente debaixo d'água, mesmo se forem empregados detectores/descritores de recursos estáveis e robustos, como a transformação de recurso invariante em escala (SIFT) (LOWE, 2004), os recursos robustos acelerados (SURF) (BAY; TUYTELAARS; GOOL, 2006) ou descritores de recursos com base nos momentos de Zernike (EUSTICE; PIZARRO; SINGH, 2008; KIM; EUSTICE, 2009).

2.3.2 Odometria visual estéreo

Usando o mesmo conceito do sistema visual humano, o sistema de visão binocular ou estereoscópica emprega um conjunto de câmeras binoculares ou câmera estéreo, sendo necessário na utilização em algoritmos de odometria visual estéreo, como mostrado na Figura 2.4. Segundo os autores em (STIVANELLO et al., 2008), um sistema estéreo pode extrair informações da cena observada recuperando dados de profundidade de um ponto no espaço a partir da distância relativa entre dois pontos. Essa diferença nas coordenadas dos pontos correspondentes às projeções é chamada disparidade.

A maioria das soluções subaquáticas propostas prefere utilizar câmera estéreo em vez de monoculares para melhorar a robustez e a precisão da saída, evitando problemas como os citados antes (BELLAVIA; FANFANI; COLOMBO, 2017; KIM; EUSTICE, 2013).

A geometria necessária para fornecer a terceira dimensão é baseada no conceito de geo-

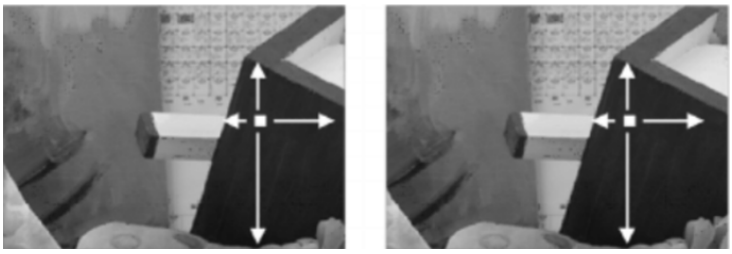


Figura 2.4: Imagem de duas lentes simultâneas. Fonte: (STIVANELLO et al., 2008).

metria epipolar, onde duas câmeras são apontadas para a mesma cena em duas posições distintas, resultando em várias conexões geométricas entre elas e, portanto, fornecendo os pontos 3D, epipolar. A geometria é apresentada com mais detalhes em (BAPPY; RAHMAN, 2012).

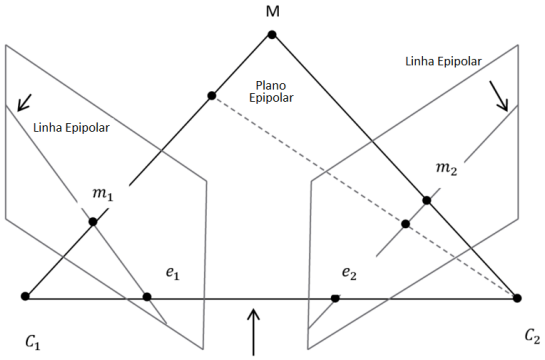


Figura 2.5: Conceito de geometria epipolar. Fonte: (BAPPY; RAHMAN, 2012).

Na odometria visual estéreo, o movimento é estimado pela observação de informações da imagem oriunda de uma câmera estéreo, em dois quadros sucessivos (nas imagens direita e esquerda). Na Figura 2.6, são mostrados quadros rastreados para formar uma nuvem de pontos 3D esparsos do ambiente e podem ser usados como base para a localização.

Figura 2.6: Os quadros rastreados formam uma nuvem de pontos 3D. Fonte: (STIVANELLO et al., 2008).

De acordo com os autores em (STIVANELLO et al., 2008), uma das principais vantagens de imagens oriundas de uma câmera estéreo, é que possuem uma quantidade enorme de informações significativas, mesmo sem a necessidade de mover a câmera, além disso,

fornecem alta precisão da localização, provando ser economicamente viável em comparação com outros sensores de proximidade. Em (ZHOU et al., 2017), o VO estéreo pode estimar os 6-Seis graus de liberdade de movimento, sendo eles: movimento linear, retilinidade horizontal, retilinidade vertical, pitch, rotação no plano horizontal e rotação ao redor do eixo de percurso, além do sensor do movimento do *egomotion*, o que não diz respeito ao tipo de ambiente em que o sistema trabalha. Atualmente, existem dois tipos de métodos de VO estéreo, 3D-3D e 3D-2D.

De acordo com (YOUSIF; BAB-HADIASHAR; HOSEINNEZHAD, 2015), essa estimativa de movimento 3D-3D é realizada triangulando pontos característicos da imagem 3D observados em uma sequência de imagens. A transformação entre os quadros oriundos da câmera é então estimada minimizando a distância euclidiana 3D entre os pontos 3D das imagens correspondentes.

Os autores também discutiram sobre o método de estimativa de movimento 3D-2D semelhante à abordagem anterior, mas aqui o erro de reprojeção 2D é minimizado para encontrar a transformação necessária. Novamente, o número mínimo de pontos requeridos varia com base no número de restrições no sistema. Embora esses algoritmos de VO estéreo tenham se mostrado promissores, a imagem observada pela câmera em determinadas condições pode impactar diretamente o resultado de trajetória gerados pelos algoritmos de odometria visual, existem alguns problemas em relação à captura de imagens, gerando desvio de trajetória e valores discrepantes, como condições de iluminação, textura ao redor, presença de água, neve, desfoque de movimento, presença de sombras, semelhança visual, configuração degenerada e oclusões.

Com o objetivo de avaliar o melhor desempenho entre os algoritmos de VO estéreo e VO monocular, foi realizado um experimento de forma comparativa entre algoritmos de VO monocular e VO estéreo comparando-os com os dados de outros sensores como (INS) e (GPS), em (CHEN; CHIANG, 2015), onde foi comprovado que a utilização de algoritmos VO estéreo promove um desempenho mais estável. A figura 2.8, a trajetória no movimento de curvatura pode ser facilmente distinguida com o caminho de referência no modo monocular. O valor da distância percorrida é de 6,3%, o principal motivo é que o resultado da comparação entre os principais pontos característicos da imagem monocular não é robusto o suficiente para fornecer uma boa estimativa de movimento quando o veículo está mudando de direção.

Portanto, o sistema VO estéreo tem desempenho mais estável que o sistema monocular, e a superioridade pode ser verificada através do experimento de torneamento dinâmico que é a combinação dos movimentos de rotação e movimento de avanço em (CHEN; CHIANG, 2015).

Na figura 2.7, é mostrada a comparação de desempenho entre o sistema de integração INS (Sistema de Navegação Inercial)/GNSS (Sistema Global de Navegação por Satélite), que é um dos mais usados na área de navegação, e a abordagem VO (Odometria Visual), respectivamente para sistemas monocular e estéreo. A taxa de amostragem mais alta tem melhor desempenho tanto na trajetória pura quanto na TD (Distância percorrida), tanto E (leste) (m) quanto N (norte) (m), quando o veículo passa por algumas áreas onde a qualidade do sinal GPS recebido é instável, o INS/GPS MEMS (Micro Sistemas Eletromecânicos) com o esquema fracamente acoplado gerará facilmente piores soluções de navegação (CHEN; CHIANG, 2015).

Modo	Monocular	Estéreo
Distância Real	919.66m	
Distância estimada	854.61 m	912.09 m
Erro de ponto final	57.94 m	10.12 m
DT%	6.3 %	1.1 %

Figura 2.7: A análise numérica de dois sistemas de odometria visual em que a comparação é curva é mostrada na Figura 2.8 e na Figura 2.9. Fonte: (CHEN; CHIANG, 2015).

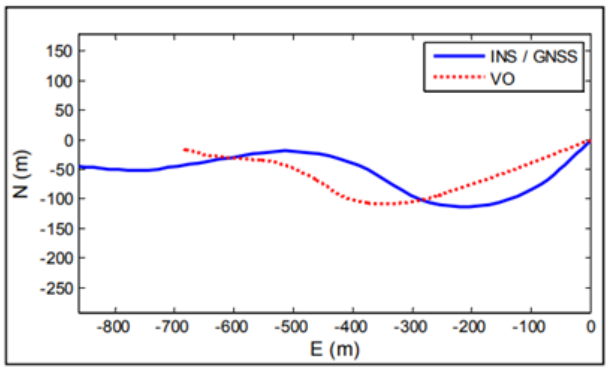


Figura 2.8: Demonstração de VO monocular de desempenhos com mais distorção. Fonte: (CHEN; CHIANG, 2015).

2.3.3 Visual SLAM

De acordo com (YOUSIF; BAB-HADIASHAR; HOSEINNEZHAD, 2015; FRAUNDORFER; SCARAMUZZA, 2011) Localização e mapeamento simultâneos (SLAM) é uma maneira de um robô se localizar em um ambiente desconhecido, enquanto constrói incrementalmente um mapa de seus arredores. O SLAM foi extensivamente estudado nas últimas décadas, resultando em muitas soluções diferentes usando sensores diferentes, in-

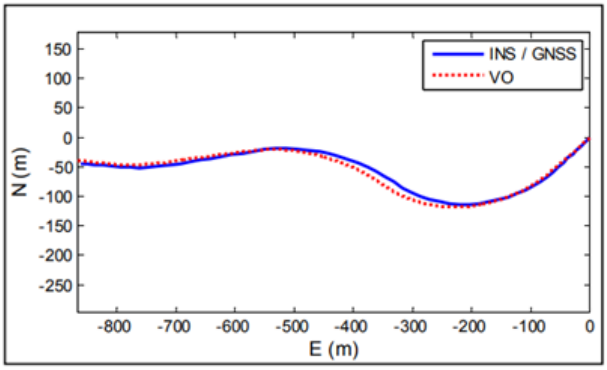


Figura 2.9: VO estéreo, demonstrativa de desempenho mais estável. Fonte: (CHEN; CHIANG, 2015).

cluindo sensores de sonar, sensores de infravermelho e *scanners* LASER. Recentemente, houve um interesse crescente no SLAM baseado em visual, também conhecido como V-SLAM, devido às informações visuais ricas disponíveis nos sensores de vídeo passivos de baixo custo em comparação aos scanners LASER.

De acordo com (FRAUNDORFER; SCARAMUZZA, 2011; NISTÉR; NARODITSKY; BERGEN, 2004), o objetivo do V-SLAM é obter uma estimativa global e consistente do caminho do robô. Isso implica em manter um controle de um mapa do ambiente (mesmo que o mapa não seja essencial) porque é necessário perceber quando o robô retorna a uma área visitada anteriormente, esse processo é conhecido como detecção de fechamento de loop. Quando um fechamento de loop é detectado, essas informações são usadas para reduzir a deriva no mapa e no caminho da câmera. De acordo com (WIRTH; CARRASCO; CODINA, 2013), os algoritmos para odometria visual - em oposição aos algoritmos SLAM completos - concentram-se em estimativas rápidas de movimento quadro a quadro, sem manter um grande histórico de fechamento de loop.

A ênfase está em medições precisas em altas frequências. Entender quando ocorre um fechamento de loop e integrar eficientemente essa nova restrição no mapa atual são dois dos principais problemas do V-SLAM. Uma das principais desvantagens da implementação das técnicas do V-SLAM depende do tempo de computação e utiliza recursos que, por sua vez, crescem significativamente para grandes ambientes (NAWAF et al., 2017). Em (STRASDAT; MONTIEL; DAVISON, 2012) é apresentada uma pesquisa extensa considerando outros aspectos relacionados ao V-SLAM e aplicações similares.

2.4 Metodologia de Slam

Nesta seção foram abordados os algoritmos de código aberto da odometria visual. Os principais critérios de avaliação que serão considerados são a compatibilidade do sistema

operacional do robô (ROS), licenças, sensores e o idioma que será usado no projeto (C++). A distribuição ROS considerada para ser usada é o Kinetic Kame. De acordo com (ALEXANDRESCU, 2001), a linguagem de programação C++ é amplamente utilizada para as mais diversas aplicações, possui recursos de programação imperativos, orientados a objetos e genéricos, além de fornecer recursos para manipulação de memória de baixo nível, e de introduzir a programação orientada a objetos (POO), sendo totalmente compatível com o ROS. Devido a esses recursos, os algoritmos que fornecem implementação de C++ são preferidos.

2.4.1 Lsd-Slam

De acordo com (ENGEL; SCHÖPS; CREMERS, 2014), O método SLAM monocular direto em larga escala (LSD-SLAM) rastreia localmente o movimento da câmera além de permitir construir mapas consistentes e em larga escala do ambiente, utilizando estimativas baseada em filtragem de mapas de profundidade semi-densos. Na Figura 2.10, é mostrada um alinhamento direto da imagem e mapas probabilísticos de profundidade semi-densos em vez de pontos-chave.

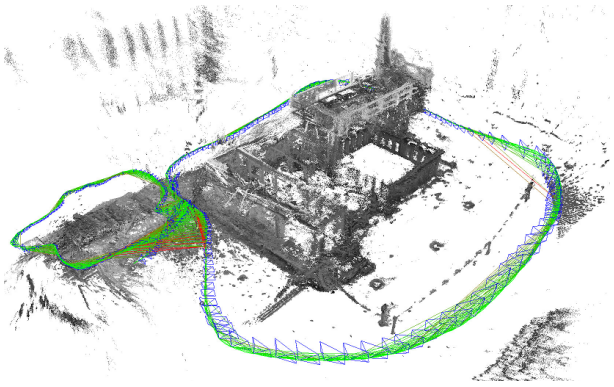


Figura 2.10: Mapa global consistente, usando alinhamento direto de imagem. Fonte: (ENGEL; SCHÖPS; CREMERS, 2014)

2.4.2 ORB-Slam

O módulo “Orb-Slam” proposto em (MUR-ARTAL; TARDÓS, 2017), é um sistema SLAM completo para câmeras monoculares, estéreo e RGB-D, incluindo recursos de reutilização de mapas, fechamento de loop e relocalização. de acordo com o autor, o sistema trabalha em tempo real em CPUs padrão em uma ampla variedade de ambientes, desde pequenas sequências manuais internas a drones voando em ambientes industriais e carros dirigindo pela cidade, o sistema inclui um modo de localização leve que utiliza recursos visuais trilhas de odometria para regiões não mapeadas e correspondências para mapear pontos

que permitem a localização de desvio zero. Na Figura 2.11, é mostrada uma trajetória e reconstrução esparsa de um ambiente urbano com vários fechamentos de loop.

Figura 2.11: Trajetória e reconstrução esparsa de ambiente urbano. Fonte: (MUR-ARTAL; TARDÓS, 2017)

2.4.3 SVO Fast Semi-Direct Monocular Visual Odometry

Em (FORSTER; PIZZOLI; SCARAMUZZA, 2014), O SVO (Semi-direct Visual Odometry), é um algoritmo de odometria visual monocular semidireto que elimina a necessidade de extração de recursos custosos e técnicas robustas de correspondência para estimativa de movimento. o algoritmo opera diretamente em intensidades de pixel, o que resulta em precisão de subpixel em altas taxas de quadros. Um método de mapeamento probabilístico que modela explicitamente as medidas que fogem do padrão (*outliers*) é usado para estimar pontos 3D, o que resulta em menos *outliers* e pontos mais confiáveis. O algoritmo foi aplicado à estimativa de estado de micro-veículos aéreos em ambientes sem suporte a GPS e é executado a 55 quadros por segundo no computador embarcado com mais de 300 quadros por segundo em um laptop comum. Na Figura 2.12 é mostrada uma estrutura adquirida durante os testes.

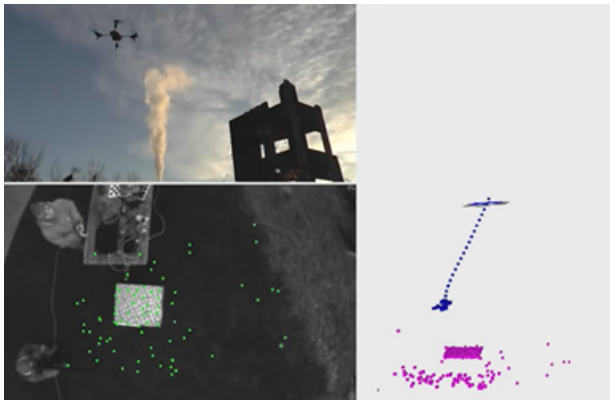


Figura 2.12: Uma estimativa 3D de pontos no modo de vôo. Fonte: Autor.

Neste capítulo abordamos uma revisão geral dos sistemas de odometria baseados em visão computacional compostos por visual monocular, visual estéreo e Slam, sendo possível

observar a utilização das ferramentas de simulação RViz e Gazebo, além da metodologia de Slam utilizada em cada um dos algoritmos sendo eles Lsd-Slam, ORB-Slam e SVO Fast Semi-Direct.

Capítulo Três

Projeto e desenvolvimento do ROV

Neste capítulo apresentaremos os detalhes do projeto e desenvolvimento do ROV utilizado para testar os algoritmos de odometria visual estudados nesta pesquisa de mestrado.

3.1 Softwares

Os softwares utilizados neste projeto foram escolhidos seguindo critérios como, estabilidade, qualidade de arquitetura, compatibilidade com o sistema operacional Linux o qual foi possível a utilização do sistema para robô (ROS).

3.1.1 Ubuntu Linux 16.04

De acordo com (UBUNTU, 2018), O Ubuntu é um sistema operacional Linux completo, disponível gratuitamente (Open-Source) com suporte comunitário e profissional. O Projeto Ubuntu é patrocinado pela (CANONICAL, 2018). A Canonical não cobrará taxas de licença pelo Ubuntu, agora ou em qualquer estágio no futuro. O modelo de negócios da Canonical é fornecer suporte técnico e serviços profissionais relacionados ao Ubuntu. Sendo este utilizado para a execução dos algoritmos de odometria visual, que tem como requisito um sistema operacional em linux.

3.1.2 ROS (Sistema operacional de robôs)

De acordo com (ROS.ORG, 2018), O ROS é um sistema meta-operacional de código aberto. Fornece os serviços como abstração de hardware, controle de dispositivo de baixo nível, comumente usado funcionalidade, passagem de mensagens entre processos e pacote gestão. O ROS foi escolhido por possuir a capacidade de conectar o LowcostROV ao Laptop (topside), por meio de mensagens, as informações trocadas estabelece um controle mais eficiente possuindo menor latência entre a comunicação.

3.1.3 Gephi (Pacote de software de análise e visualização de rede)

Segundo (GEPHI.ORG, 2017), Gephi é o principal software de visualização e exploração para todos os tipos de gráficos e redes. O Gephi é de código aberto e gratuito, sendo utilizado em vários projetos na área da pesquisa na academia. O Gephi foi utilizado para visualização das informações obtidas por meio da pesquisa efetuada para obter os algoritmos de odometria visual, retornando de sua análise os algoritmos, Lsd-Slam, ORB-Slam, e *SVO (Semi-Direct Monocular Visual Odometry).*

3.1.4 Draw.io (Aplicação de diagramação gráfica online)

Uma aplicação de diagramação totalmente gratuita, possuindo versões: Pro e Desktop, disponível no Google Drive, sem registro e sem limitações. O Draw.io foi utilizado para a construção dos diagramas e fluxogramas da pesquisa.

3.1.5 Solidworks (Software CAD 3D (Design Assistido por Computador))

De acordo com (SOLIDWORKS, 2019), O SolidWorks surgiu em 1995, criado por uma companhia de nome homônimo, nos Estados Unidos. Pelas suas características e possibilidades de uso, o programa gerou interesse na companhia francesa *Dassault Systémes.* Esta trabalhava com projetos em 3D desde o ano de 1981 e integrou o software à sua base de produtos em 1997. O SolidWorks foi a ferramenta utilizada para a construção do modelo das peças em 3D que sustentam os motores do LowcostROV, como é mostrado nas figuras Figura 3.1, Figura 3.2, Figura 3.3 e Figura 3.4

3.1.6 Ultimaker Cura (Software fatiador e servidor para impressora 3D)

Segundo (ULTIMAKER, 2016), O Ultimaker Cura é o software para impressoras 3D mais avançado do mundo, o seu principal recurso é o processo de fatiamento de impressões 3D. E foi utilizado para o fatiamento e impressão 3D dos modelos de suporte a motores desenvolvidos com o SolidWorks.

3.1.7 Desenho Técnico

Apresentação da Imagem em 3d do protótipo LowcostROV desenvolvido e renderizado no SolidWorks, como é mostrado nas figuras Figura 3.1, Figura 3.2, Figura 3.3 e Figura 3.4, vista geral, vista frontal, vista inferior e superior respectivamente.

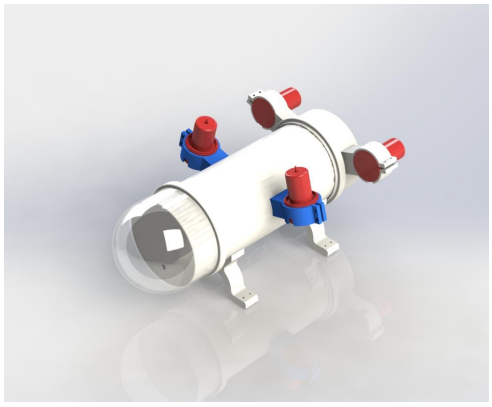


Figura 3.1: Vista Total. Fonte: Autor

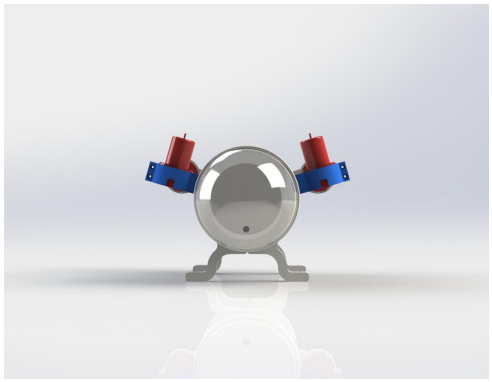


Figura 3.2: Vista Frontal. Fonte: Autor

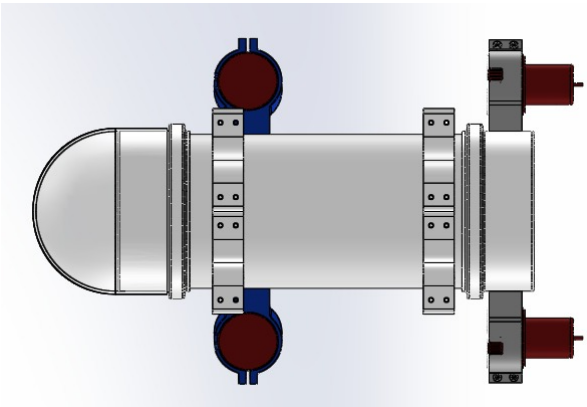


Figura 3.3: Vista Inferior. Fonte: Autor

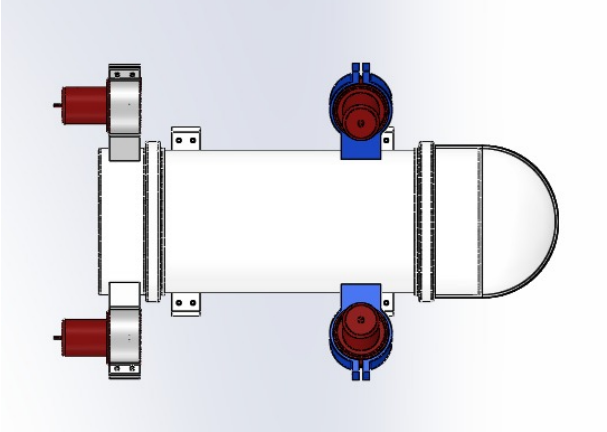
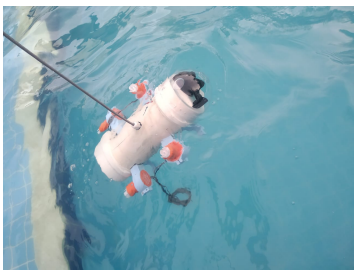


Figura 3.4: Vista Superior. Fonte: Autor

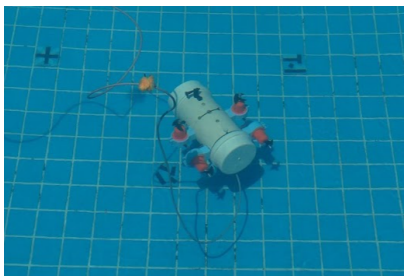
3.2 *Estrutura (Chassi)*

A estrutura do LowcostROV foi desenvolvida pensando na maior eficiência e flexibilidade, podendo utilizar diversos recursos como sensores e até mesmo braços robóticos, com mínimo de recursos necessários.



(a) Estrutura do LowcostROVcom cápsula frontal

(b) Estrutura do LowcostROV



(c) Estrutura do LowcostROV

Figura 3.5: Estrutura do LowcostROV(a), (b) e (c). Fonte: Autor

Materiais utilizados para a construção.

- 1 - Tubo PVC 150mm;
- 4 - Bomba De Porão Seaflo 1100gph - 4164lph 12v;
- 2 - Tampas PVC 150mm com borracha;

- 4 - Suportes impressos em PLA 3d;
- 4 - Hélices de plástico;
- 1 - Arduino Mega 2560;
- 2 - Ponte H Monster Motor Shield VNH2SP30 30a (2 Motor);
- 40 - Parafusos de aço inox com porca;
- 15 - Metros de fios awg 22;

3.2.1 Motores de Tração

De acordo com (SEAFLO, 2018), as bombas de esgoto não-automáticas 1100GPH da SEAFLO oferecem evacuação de água ativada por um painel ou interruptor de bóia além de Incluir proteção de bloqueio anti-ar e vedações herméticas exclusivas, fiação de bloco de grau marítimo. Disponível para tensão 12 e 24 tensão. O modelo 1100GPH 12V, atende as especificações do projeto devido ao seu alto torque e selagem hermética, possibilitando uma submersão de até 20 pés (6,096) metros.

Figura 3.6: Bomba Porão Seaflo, Fonte: (SEAFLO, 2018)

Principais Características:

- 2 anos de garantia
- Atende ou excede os padrões RoHS, SGS e ISO

- Operação tradicional por interruptor ou interruptor de bóia
- Totalmente submersível
- Base do filtro de encaixe para fácil manutenção
- Bomba e interruptor são produtos independentes
- Motores eficientes e de longa vida útil selados
- Funciona a seco, capaz de cargas de trabalho normais
- Operação silenciosa
- limites de temperatura de 110 °F (43 °C)
- Eixo de aço inoxidável
- Proteção anti-Airlock
- Selos herméticos exclusivos
- Fiação bloqueada de grau marítimo

3.2.2 Cabos e Conexões

O cabo utilizado no LowcostROV foi o modelo Xtra Guard Flex da Alpha Wire 26AWG, resistente e de excelente qualidade, como é mostrado na Figura 3.7, especificações no quadro 3.1 abaixo:

Figura 3.7: Cabo Alpha Wire 26AWG Fonte: (ALPHAWIRE, 2020)

Tabela 3.1: Quadro Cabo Alpha Wire

Propriedades	**Valores**
Modelo	Xtra Guard Flex — Flex Cable — 86503CY — Alpha Wire
Comprimento	15 Metros
Cor	Ardósia
Isolamento	Parede de 0,010, Nom, PVC, Semi-rígido

Continua na próxima página

Tabela 3.1 – Continuação

Propriedades	**Valores**
Condutor	26 (7/34) de cobre estanhado AWG
Componente	105 °C / 300 V RMS
Faixa de temperatura	-10 a 105 °C (estático), +5 a 105 °C (dinâmico)
Classificação de tensão	300 V RMS
Capacitância	Mútua 23 pF / ft @ 1 kHz, Nominal
Impedância característica	83 Ω
Indutância	0,2 μH / ft, nominal

3.2.3 Sensores

O sensor de imagem utilizado no projeto foi uma tp-link NC200 Wifi como é mostrado na Figura 3.8 e especificações de acordo com o quadro 3.2 abaixo:

Figura 3.8: Tp-link NC200 Wifi Fonte: (TPLINK, 2020)

Tabela 3.2: Quadro Tp-Link NC200 Câmera na nuvem, 300 Mbps Wi-Fi

Propriedades	**Valores**
Modelo	Tp-Link NC200 Câmera na nuvem, 300 Mbps Wi-Fi
Certificação	CE, FCC, RoHS
Dimensões	96*61*24 mm
Taxas de dados sem fio	IEEE 802.11 b/g/n, até 300 Mbps
Frequência	2,4 ~ 2,4835 GHz
Sensor de imagem	Sensor CMOS progressivo de 1/4”
Resolução	0,3 megapixels (640 x 480)

Continua na próxima página

Tabela 3.2 – Continuação

Propriedades	**Valores**
Lentes	F:2,8, f:3,85 mm
Ângulo de visão	FOV = 64 °
Resolução e taxa de quadros	Máx. 20 fps a 640x480 (VGA)
	máx. 20 fps a 320x240 (QVGA)
Interface de rede	RJ-45 para Ethernet 10/100 Base-T

3.2.4 Microcontroladores

O sistema embarcado utilizado para controlar as entradas e saídas de comandos lógicos do LowcostROV, foram o Arduíno e Raspberry Pi, que serão abordados logo abaixo:

3.2.4.1 Arduino

Segundo (GOMES, 2014), "Arduíno", é nome próprio italiano de origem germânica. É composto pelas palavras *hard/hart (forte - brave, hardy, strong) e win* (amigo em saxão antigo) formando *Hardwin* (Grande Amigo), que foi latinizado para Ardovinus, e depois para o italiano Arduíno.

O Arduíno é uma plataforma de prototipagem eletrônica de hardware livre que funciona devido a ação de um microcontrolador o qual através de uma conexão com um computador, funciona com uma linguagem de programação baseado em Wiring podendo se comunicar com computadores ou Shields de expansão, como é mostrado na Figura 3.9, especificações no quadro 3.3 abaixo:

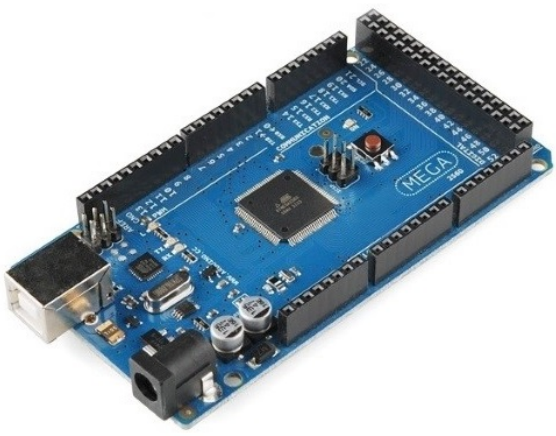


Figura 3.9: Arduino Mega 1280. Fonte: (GOMES, 2014)

Tabela 3.3: Quadro Microcontrolador ATmega1280

Propriedades	**Valores**
Microcontrolador	ATmega 1280
Tensão de Alimentação	5V
Tensão de Entrada (recomendado)	7 ∼ 12V
Tensão de Entrada (limites)	6 ∼ 20V
Portas digitais I/O	54 (das quais 14 podem ser usadas como saída PWM)
Portas analógicas de entrada	16
Corrente DC por pino	40 mA
Corrente DC por pino 3.3V	50 mA
Memória Flash	256 KB
SRAM	8 KB
EEPROM	4 KB
Clock	16 MHz

3.2.4.2 Raspberry Pi

De acordo com (PI, 2019), O Raspberry Pi é um minicomputador, possuindo todos os componentes numa pequena placa do tamanho de um cartão de crédito, possuindo um baixo custo, cerca de US$ 35 para a qualidade e recursos aos quais ele dispõe, como é mostrado na Figura 3.10, especificações no quadro 3.4 abaixo:

Figura 3.10: Raspberry Pi. Fonte: (PI, 2019)

Tabela 3.4: Quadro Raspberry Pi 2

Propriedades	**Valores**
Modelo	Raspberry Pi 2 Modelo B
Ano Lançamento	Fevereiro de 2016
Preço oficial	US$ 35
Processador	CPU Broadcom BCM2837 Quad Core de 1.2GHz 64 bits
Memória RAM	1GB RAM
Portas USB	4 portas USB 2.0
Internet	Wi-Fi 2,4 GHz BCM43438 e Ethernet
Conectividade	HDMI
	Conector de câmera CSI
	Conector de display DSI
	Saída de áudio P2
	Micro SD
	Conector GPIO de 40 pinos
Bluetooth	Bluetooth Low Energy (BLE)
Fonte de alimentação	Micro USB 5V/2.5A DC.

3.2.5 Ponte-H

De acordo com (INSTRUCTABLES, 2016), O VNH2SP30 é um driver de motor de ponte que incorpora um driver monolítico duplo alto e dois interruptores laterais baixos, os pinos do sensor de corrente (CS) produzirão aproximadamente 0,13 volts por amp de corrente de saída. Sendo uma ótima escolha já que o mesmo é capaz de controlar até dois motores com corrente de pico de 30A, controlando o sentido de rotação e velocidade do motor, como é mostrado na Figura 3.11, especificações no quadro 3.5 abaixo:

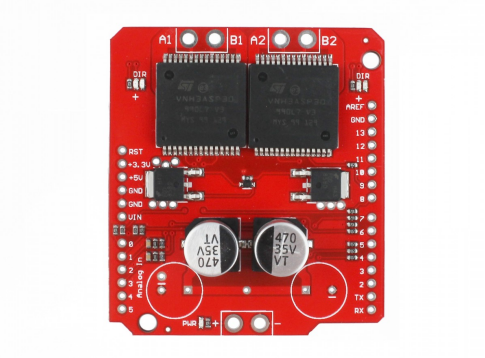


Figura 3.11: Monster Motor Shield. Fonte: (INSTRUCTABLES, 2016)

Tabela 3.5: Quadro Monster Motor Shield

Propriedades	**Valores**
Tensão máxima	30V
Máxima corrente	30A
Corrente nominal de operação	14A
Medição de corrente disponível para o Arduíno	Pinos analógicos
MOSFET sobre resistência	19 mΩ
Frequência PWM máxima	20 kHz
Proteção térmica	Sim
Proteção contra sobretensão e subtensão	Sim

3.2.6 Fonte de Alimentação

Para a correta alimentação do sistema foi utilizado uma fonte de computador convencional, modelo ST-550P-AG Black Edition Seventeam 550W. como é mostrado na Figura 3.12, de acordo com (HARDSTORE, 2019) segue as especificações no quadro 3.6 abaixo:

Figura 3.12: Fonte Seventeam 550w. Fonte: (HARDSTORE, 2019)

Tabela 3.6: Quadro Fonte Seventeam 550W

Propriedades	**Valores**
Modelo	Seventeam ST-550P-AG Black Edition
Potência	550W (Reais)
Filtro de Proteção	PFC ATIVO

Continua na próxima página

Tabela 3.6 – Continuação

Propriedades	**Valores**
Vida útil	100.000 horas (Com energia 100% estabilizada)
Cor	Preta
Padrão	ATX 12V, V2.3
Tensão	103V $\sim$ 263V
Frequência	47 $\sim$ 63 Hz
Temperatura operacional	0ºC $\sim$ 50ºC
Eficiência	¿78% (Carga Máxima)
Tamanho	(C x A x L)
	150 x 86 x 140mm
Certificações	80 PLUS Bronze
	CE
	TUV
	CB
	FCC
	CUL
	BSMI
Peso aprox. c/ embalagem	2.2kg
Comprimento dos cabos	$\sim$ 50cm
Garantia	1 ano

3.3 Interface de Controle

A interface de controle do sistema é composto basicamente por um *Joystick* mostrado na figura 3.13 sendo o responsável pelo controle manual do ROV e todas as interações são controladas pelo sistema ROS que recebe a informação oriunda do *Joystick* e envia um sinal para o Arduíno completando assim a manipulação do ROV.

Para controlar o LowcostROV é necessário pressionar o botão LB localizado na parte superior esquerda do controle ativando assim os motores, seguido da direção que desejar, segue alguns exemplos; para submergir o ROV é necessário pressionar o botão **LB** + $\downarrow$ (analógico esquerdo), emergir **LB** + $\uparrow$ (analógico esquerdo), frente **LB** + $\uparrow$ (analógico direito), ré **LB** + $\downarrow$ (analógico direito), direita **LB** + $\rightarrow$ (analógico direito) e esquerda **LB** + $\leftarrow$ (analógico direito), é possível travar a velocidade dos motores responsáveis por submergir e emergir pressionando o botão **LT** após o comando **LB** + $\downarrow$ (analógico esquerdo) mantendo uma maior estabilidade em seu nível de profundidade, para o desligamento completo dos motores é necessário pressionar o botão **B**.

Figura 3.13: F310 Gamepad - Logitech. Fonte: (LOGITECHG, 2020)

3.4 Visão Geral

É possível observar as funcionalidades que o sistema deve oferecer como representado na figura 3.14, estando dividido em 4 etapas, a primeira é parte de inicialização da interface contendo sensores e sistema, na etapa seguinte é a de controle e trajetória demonstrada no tópico anterior, na terceira é possível observar a forma de obtenção das imagens e a trajetória gerada a partir da mesma, a última etapa representa um comparativo da trajetória gerada pelo algoritmo de odometria e o trajeto do mapa real.

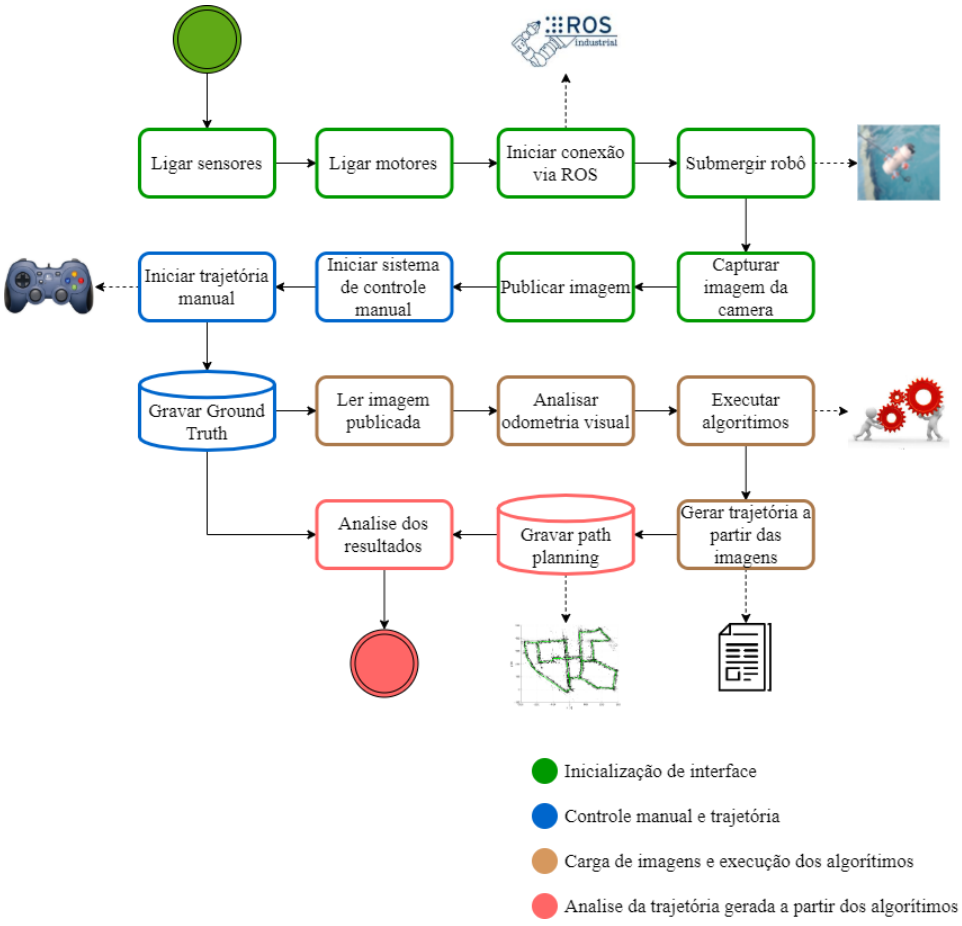


Figura 3.14: Caso de uso do LowcostROV indicando as funcionalidades que o sistema deve oferecer. Fonte: Autor

3.5 Visão Geral dos Algoritmos

3.5.1 Visão Geral do SVO (Semi-Direct Visual Odometry)

Podemos observar na figura 3.15 abaixo, uma análise detalhada do fluxo do algoritmo SVO, composto por duas threads, sendo uma responsável pelo rastreamento de posicionamento da câmera e a outra se encarrega de criar pontos no mapa, diferente do ORB_Slam que utiliza o método de correspondência de ponto de recurso, o SVO Semi-Direct usa o método de correspondência em escala de cinza de acordo com a diferença de brilho.

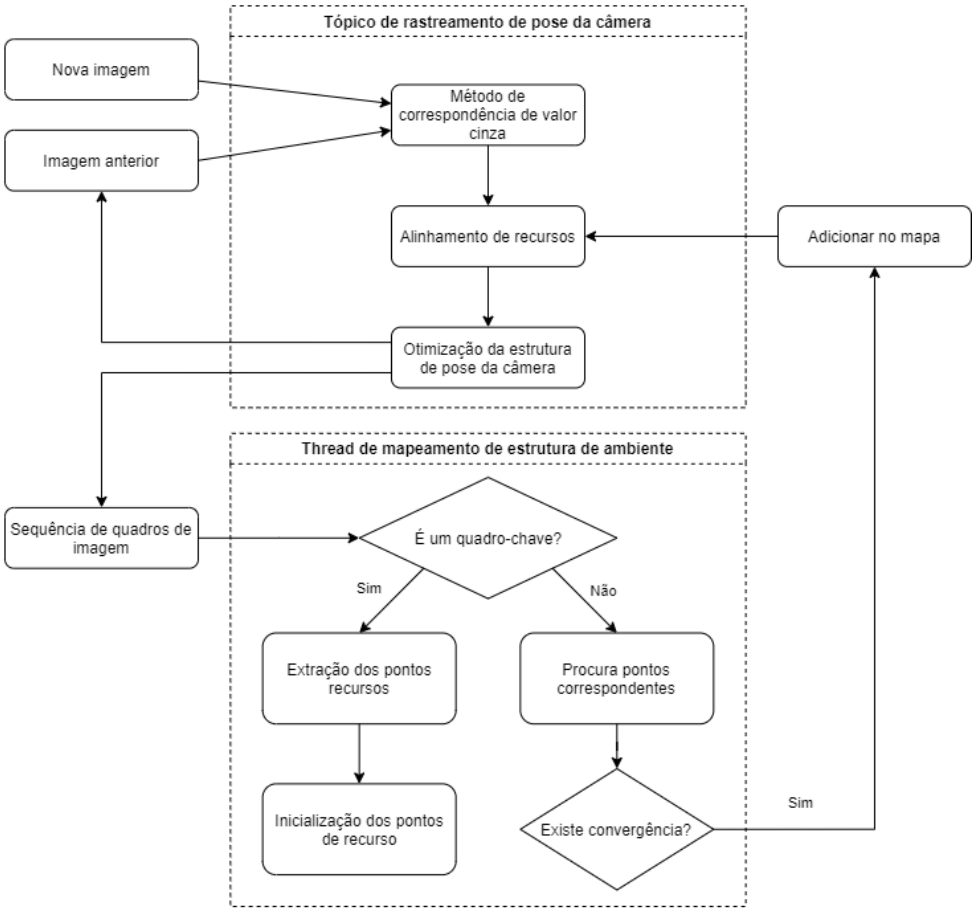


Figura 3.15: Fluxograma do SVO (Semi-Direct Visual Odometry), indicando as funcionalidades que o sistema deve oferecer. Fonte: Autor

Quando uma nova imagem é capturada é iniciada a execução do método de correspondência de valor no qual é analisado a diferença dos tons em cinza em uma área de 4*4 ao redor do ponto de recurso, em seguida do alinhamento de recursos que responsável por analisar a posição da câmera e a estimativa de profundidade da imagem usando distribuição gaussiana responsável por extrair os valores mínimos e máximos, a partir do qual é

feita uma otimização da estrutura de posse da câmera, então é iniciado o sequenciamento do quadro de imagem onde é verificado se é um quadro chave, caso seja, são extraídos os pontos de recursos, onde é utilizado uma geometria epipolar para encontrar a menor diferença de brilho entre os quadros-chave, em seguida é utilizado um filtro de profundidade aumentando ainda mais a precisão dos pontos do mapa, caso contrário, procura-se pontos correspondentes, caso seja realizado o passo de extração dos pontos de recursos, que é responsável por analisar a média da disparidade entre o quadro atual e o quadro anterior, é feita a inicialização dos pontos de recurso.

3.5.2 Visão Geral do ORB_Slam

Na figura 3.16, é mostrada uma visão geral do ORB_Slam sendo composto por três segmentos principais: rastreamento, mapeamento local e fechamento de loop, que pode criar um quarto segmento para realizar um ajuste de pacote completo após o fechamento de um loop.

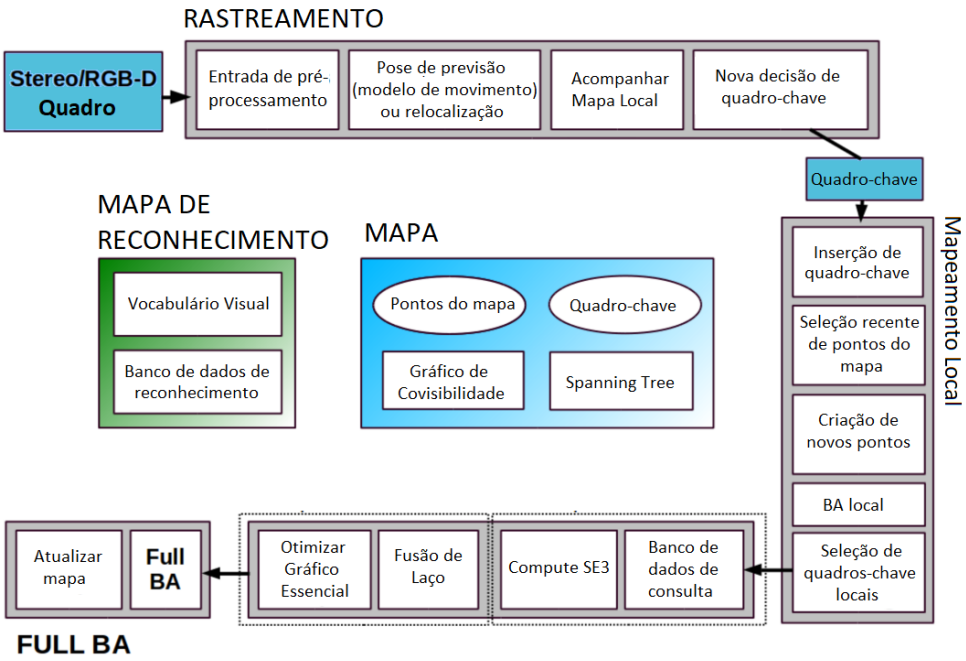


Figura 3.16: Visão geral do ORB_Slam, indicando as funcionalidades que o sistema deve oferecer. Fonte: (MUR-ARTAL; TARDÓS, 2017)

O encadeamento de traços realiza um pré-processo que distingue o tipo de entrada (estéreo ou RGB-D), tornando o resto do sistema independente do sensor de entrada, conforme mostrado na figura 3.17.

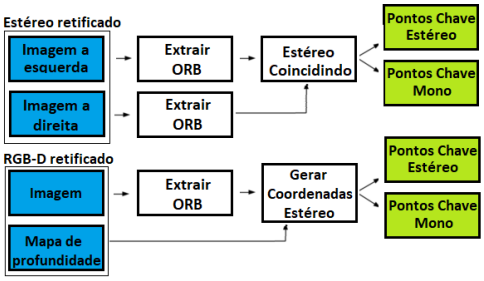


Figura 3.17: Visão de Pré-processamento de entrada ORB_Slam. Fonte: (MUR-ARTAL; TARDÓS, 2017)

Conforme já mencionado acima e mostrado na figura 3.16, três segmentos estão rodando em paralelo, que serão descritos abaixo:

A primeira etapa é o processo de rastreamento, ele é responsável por extrair recursos de ORB da imagem, encontrar a posição da câmera em cada quadro e decidir quando inserir um novo quadro-chave, baseado na velocidade de movimento é possível diminuir erros de ponto 3D, além de analisar as correspondências de pontos do mapa determinando novos quadros-chave por meio de critérios de decisão. na segunda etapa é feito o mapeamento de localização onde o mesmo processa novos quadros-chave e executa um ajuste de agrupamento local (Local BA) para otimizar poses e pontos do mapa e, em seguida é feito uma inserção de quadro no mapa onde é processado a segmentação da imagem, depois é feito a criação de pontos que será salvo em uma árvore de vocabulário determinando novas propriedades do ponto do mapa e, finalmente, reduz-se a complexidade de BA e limita o número de quadros-chave removendo assim quadros redundantes. A terceira e última etapa é o fechamento de loop, onde o mesmo é responsável por informar que um veículo retornou ao local visitado anteriormente por meio da correção de loop fechado.

3.5.3 Visão Geral do Lsd_Slam

Na figura 3.18, é mostrada uma visão geral do LSD_Slam sendo composto por três segmentos principais: rastreamento de imagem, estimativa de profundidade e otimização de mapa.

Após uma nova imagem ser capturada é iniciado o rastreamento de imagem sendo responsável por calcular a transformação relativa entre o quadro atual e o quadro de referência, além de utilizar o método Gauss-Newton de otimização para resolver problemas de mínimos quadrados não lineares, ligando a nova imagem a posição do quadro-chave atual e assim calculando a relação de transformação de coordenadas, em seguida é feito uma estimativa do mapa de profundidade sendo responsável pelo processo de decisão da

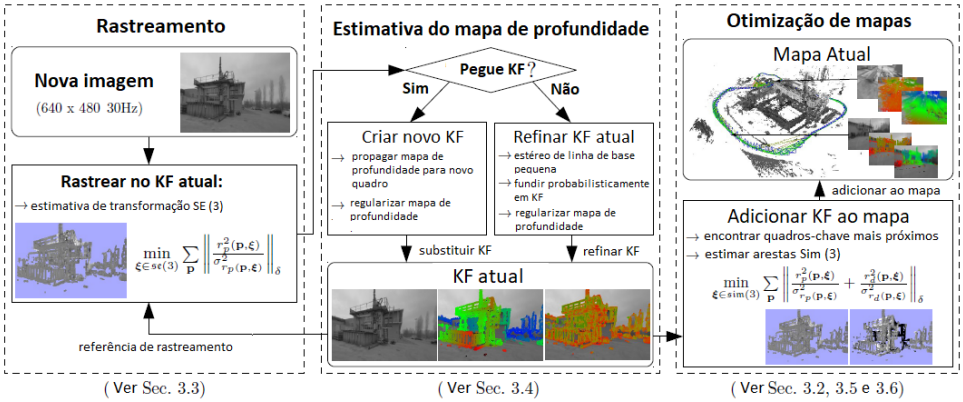


Figura 3.18: Visão geral do algoritmo LSD_SLAM. Fonte: (ENGEL; SCHÖPS; CREMERS, 2014)

criação de um novo quadro-chave, onde a estimativa de profundidade do quadro-chave é passada pela transformação Sim3, se o número de correspondências entre o quadro-chave e o candidato for muito alta, um novo quadro-chave é obtido, e finalmente é iniciado o processo de otimização do mapa que inclui um mapa de posição composto de quadros-chave e um mapa de profundidade semi-denso correspondente, vários quadros-chave que estão próximos o suficiente para detectar um fechamento de loop são utilizados.

Capítulo Quatro

Materiais e métodos

Neste capítulo apresentaremos os detalhes da estruturação do ambiente e materiais utilizados para o desenvolvimento do projeto.

4.1 Ambiente de testes

O ambiente utilizado para a execução dos testes foi uma piscina de 8m x 4m x 1.70m, nela foram feitas alguns gravações preliminares na resolução de 640x480P a uma altura média de 1 metro dentro da água utilizando a câmera *Tp-Link NC200* mostrado na figura 3.8, O critério de escolha adotado pela utilização da piscina foi de efetuar testes em um ambiente simulado de forma controlada, possibilitando mapear a mesma de uma forma mais detalhada, após as filmagens a mesma foi processada em alguns algoritmos de odometria visual e foi constatado que não era possível gerar nenhum mapa pois os algoritmos não conseguiam identificar nenhuma característica da imagem (*features*) como cor, profundidade dos pixels da imagem, forma e textura.

Para solucionar o problema foi decidido mapear a piscina com *tags*, aumentando assim a quantidade de *features* do ambiente, uma das possibilidades para a criação dessas *tags* era a impressão seguida de plastificação e adição de pesos para a completa imersão, já que era preciso que as imagens dessas *tags* ficassem rentes e bem possessionadas ao fundo da piscina, no entanto devido a possibilidade de descolamento da plastificação foi decidido a criação dos pisos *tags* de forma artesanal.

Utilizando fita isolante, fita crepe e sobras das pastilhas dos pisos da própria piscina representados na Figura 4.1, onde foram distribuídas de maneira que uma pastilha esteja a uma distância de 1 metro da outra formando quadrados, levando em consideração que o peso da mesma fará o piso *tag* permanecer rente ao solo e sem se mover com qualquer oscilação da água, é possível observar na figura 4.2 que o algoritmo já consegue mapear a piscina de forma esperada com uma resolução de 640x480P, comprovando a eficiência dos pisos *tags*.

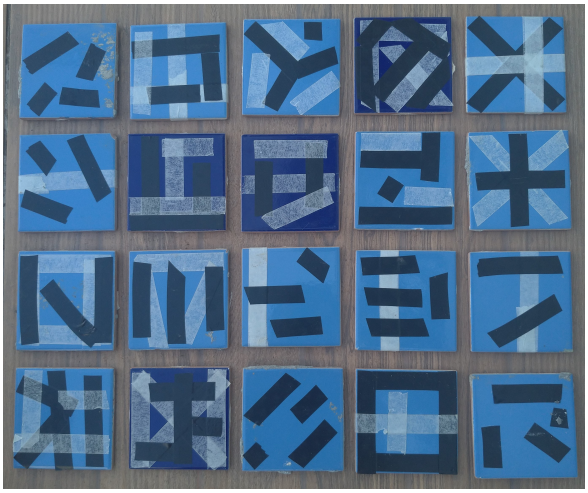


Figura 4.1: Pisos *tags*. Fonte: Autor

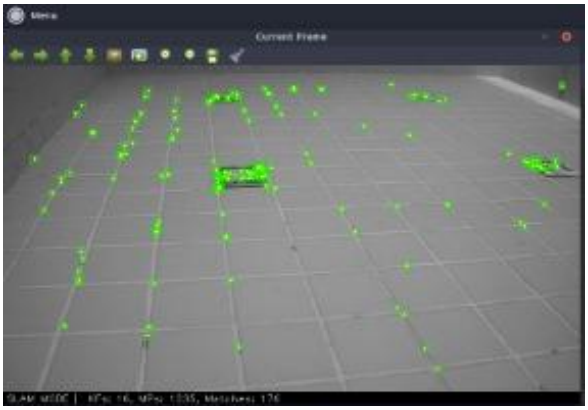


Figura 4.2: Reconhecimento dos pisos *tags* pelo OrbSlam Fonte: Autor

4.2 Algoritmos

O método para execução dos algoritmos utilizado neste projeto foi submergir o Rov em seguida iniciar uma gravação de vídeo em uma trajetória conhecida com resolução de 640x480P, entre os intervalos dos pisos *tags* existe uma distância conhecida de 1 metro formando assim uma grade, para cada direção que o LowcostROV for, é possível saber seu caminho verdadeiro, exemplo observado na figura 4.3.

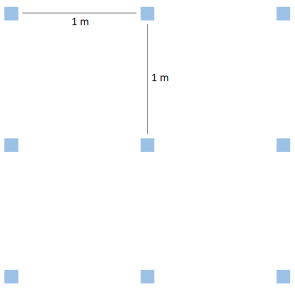


Figura 4.3: Mapa dos pisos *tags*. Fonte: Autor

4.2.1 ORB-Slam

Os recursos necessários para a execução do algoritmo de odometria visual orb_slam é um input de vídeo por meio de um tópico "/camera/image_raw"previamente definido, e a configuração de calibração da câmera por meio do arquivo de extensão .yaml, além de ser responsável por todos os parâmetros de configurações, para a publicação do stream video foi utilizado uma aplicação em C++/ROS "video_stream_opencv", ela contém um nó para publicar qualquer vídeo em um tópico definido, a resolução utilizada nos vídeos foi de 640x480p a 30fps, o comando utilizado para executar o video_stream_opencv é o "roslaunch video_stream_opencv video_file.launch"existindo também a possibilidade de verificar todos os tópicos por meio do comando via terminal Linux "rostopic list".

O comando utilizado para iniciar o orb_slam foi utilizado o "roslaunch orb_slam_ros orbSlamMono.launch", por meio do visualizador RViz mostrado na figura 4.4 é possível acompanhar a trajetória e nuvem de pontos gerado pelo algoritmo.

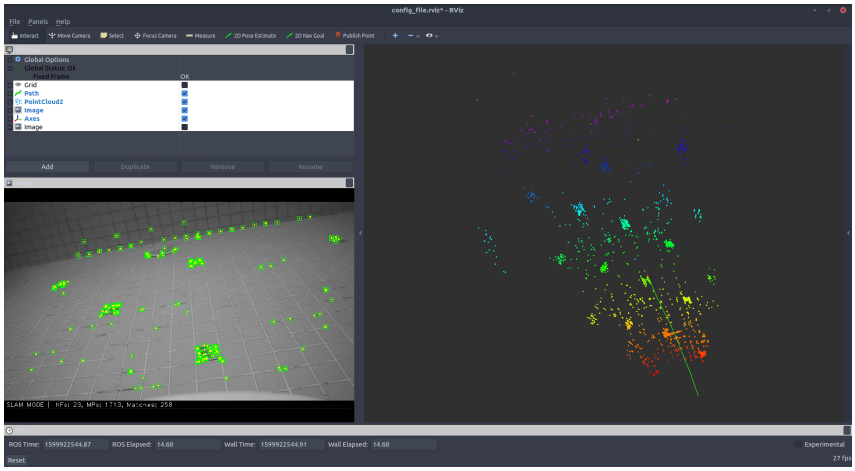


Figura 4.4: Orb_slam em execução através do RViz. Fonte: Autor

A figura 4.3 é possível observar que o "video_stream_opencv"está publicando no tópico "/camera/image_raw"onde logo em seguida o orb_slam está lendo essa publicação, possibilitando assim o transporte da imagem via ROS entre as duas aplicações.

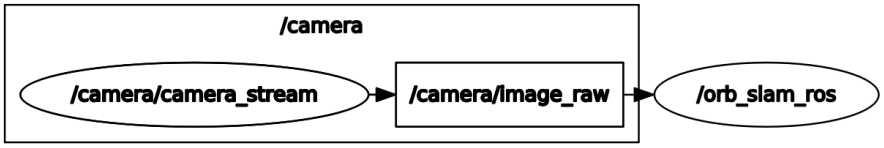


Figura 4.5: rqt_graph ROS, indicando a interação entre os tópicos de stream do vídeo e o orb_slam. Fonte: Autor

4.2.2 SVO Fast Semi-Direct

Para a correta configuração em seus parâmetros assim como o orb_slam citado acima, um input de vídeo também é necessário já que a criação da trajetória depende exclusivamente das informações das imagens, para a publicação do stream de vídeo foi utilizado a mesma aplicação "video_stream_opencv", uma diferença importante em seu script de inicialização é que não é necessário carregar o rviz manualmente já que o mesmo é iniciado de forma automática e parametrizada pelo próprio SVO, é possível observar na figura 4.6 recursos muito interessantes utilizado pelo algoritmo, existe uma interface de apoio ao usuário a qual é possível controlar a inicialização e reinicialização da trajetória criada pelo próprio SVO, exibindo também informações importantes como; total de fps, *features* e status de sua analise.

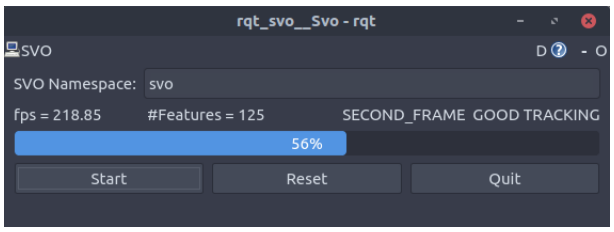


Figura 4.6: Interface de apoio RViz do SVO. Fonte: Autor

Para a configuração e execução do SVO Fast é necessário parametrizar dois arquivos de configuração, o primeiro está relacionado com a configuração de calibração da câmera e o segundo arquivo está relacionado aos parâmetros básicos da própria aplicação, com ele é possível configurar algumas informações como: número de *features*, *keyframe*, rotação inicial da câmera, além de vários outros recursos exibindo assim todas as informações geradas pelo SVO como mostrado na figura 4.7 abaixo.

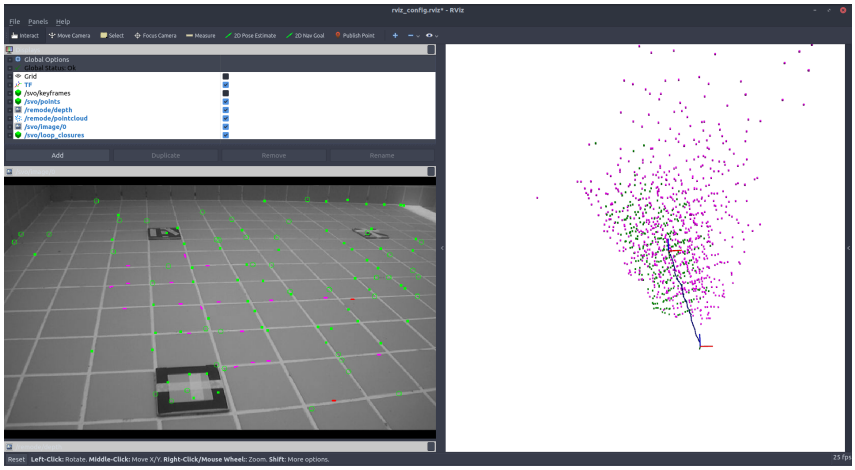


Figura 4.7: SVO Fast em execução através do RViz. Fonte: Autor

4.2.3 Lsd-Slam

Diferente dos algoritmos mencionados acima o Lsd-Slam possui algo bem particular, a sua própria interface de visualização denominada "lsd_slam_viewer", com ele foi possível identificar a nuvem de pontos e a trajetória feita pelo algoritmo demonstrado na figura 4.8 abaixo, a linha em vermelho e verde é referente a trajetória criada a partir das imagens vindas da câmera, é possível perceber os quadros chaves ao longo desta linha, mais acima está a nuvem de pontos gerada pelo algoritmo.

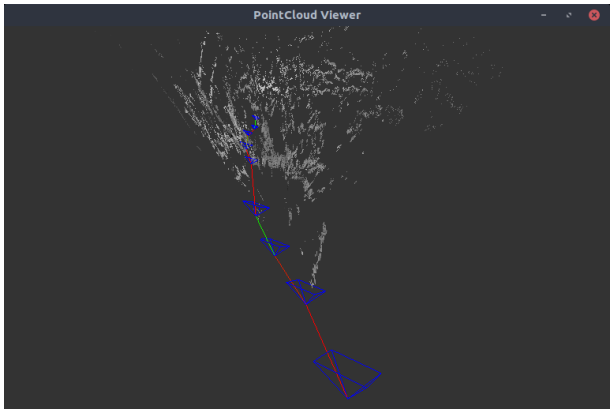
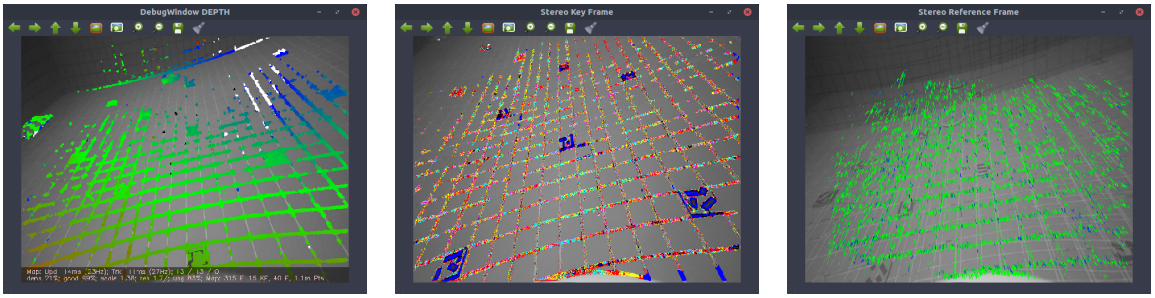


Figura 4.8: Interface própria do Lsd-Slam em execução. Fonte: Autor

Na interface de visualização existem sub-janelas com as quais é possível acompanhar mesmo com um pouco de atraso os processos de mapeamento e criação da nuvem de pontos, é possível observar na figura 4.9(a) a tela de depuração que é responsável por exibir informações como taxa de quadros, frequência, percentual de acerto do algoritmo entre outros, na figura 4.9(a) é exibido informações dos quadros-chaves, e por fim a figura 4.9(c) exibindo os frames de referência sendo muito importante para uma análise mais detalhada do comportamento de varredura da imagem. Para publicação do stream de video foi utilizado a mesma aplicação "video_stream_opencv".



(a) Tela de debug (b) Janela de *key frames* (c) *Frames* de referência

Figura 4.9: Janelas de depuração do Lsd-Slam (a), (b) e (c). Fonte: Autor

Capítulo Cinco

Resultados e Discussão

Para o refinamento da acurácia dos resultados dos experimentos, foram levantadas algumas informações das condições do tempo/clima na localidade de Santo Amaro - BA, por meio do site climatempo, nele foi possível obter informações importantes das condições meteorológicas do momento dos experimentos, é possível acompanhar as informações do clima com intervalos de 2 horas, o site possibilita ter informações detalhadas do clima como: temperatura, previsão de chuva ou sol, velocidade e direção do vento, umidade do ar e pressão atmosférica, para mais detalhes das informações utilizadas na momento do experimento veja na tabela 5.1 abaixo:

É possível observarmos os resultados obtidos da tabela comparativa separando os melhores valores por sessão entre os algoritmos ORB_Slam, *SVO Semi-Direct Monocular Visual Odometry*, evidenciando as faixas de trajetória em metros sendo elas: (valor real: 3.43m, média: 3.43m, desvio padrão: 0.04, erro -0.29%) para o ORB_Slam, (valor real: 2.26m, média: 2.26m, desvio padrão: 0.03, erro -0.15%) SVO Semi-Direct, (valor real: 4.68m, média: 4.71m, desvio padrão: 0.11, erro 0.57%) Lsd-Slam. Para mais detalhes veja as especificações na tabela 5.2 abaixo:

Tabela 5.1: Informações das condições dos experimento, clima em Santo Amaro - BA

Dia Turno	**Data Hora**	**Temperatura Celsius (°C)**	**Chuva**	**Vento**	**Umidade Do Ar**	**Pressão Atmosférica**
01 Manha	12/06/2021 10:00hs	28 °C	00mm 00%	E 9 km/h	84%	1.015 hPa
01 Tarde	12/06/2021 14:00hs	31 °C	00mm 00%	ESE 8 km/h	55%	1.016 hPa
02 Manha	13/06/2021 10:00hs	28 °C	0.63mm 90%	E 7 km/h	84%	1.016 hPa
02 Tarde	13/06/2021 14:00hs	29 °C	0.63mm 90%	ESE 15 km/h	52%	1.017 hPa
03 Manha	14/06/2021 10:00hs	28 °C	0.33mm 90%	ESE 23 km/h	75%	1.017 hPa
03 Tarde	14/06/2021 14:00hs	28 °C	0.33mm 90%	SE 24 km/h	53%	1.018 hPa

Tabela 5.2: Tabela comparativa dos algoritmos em metros

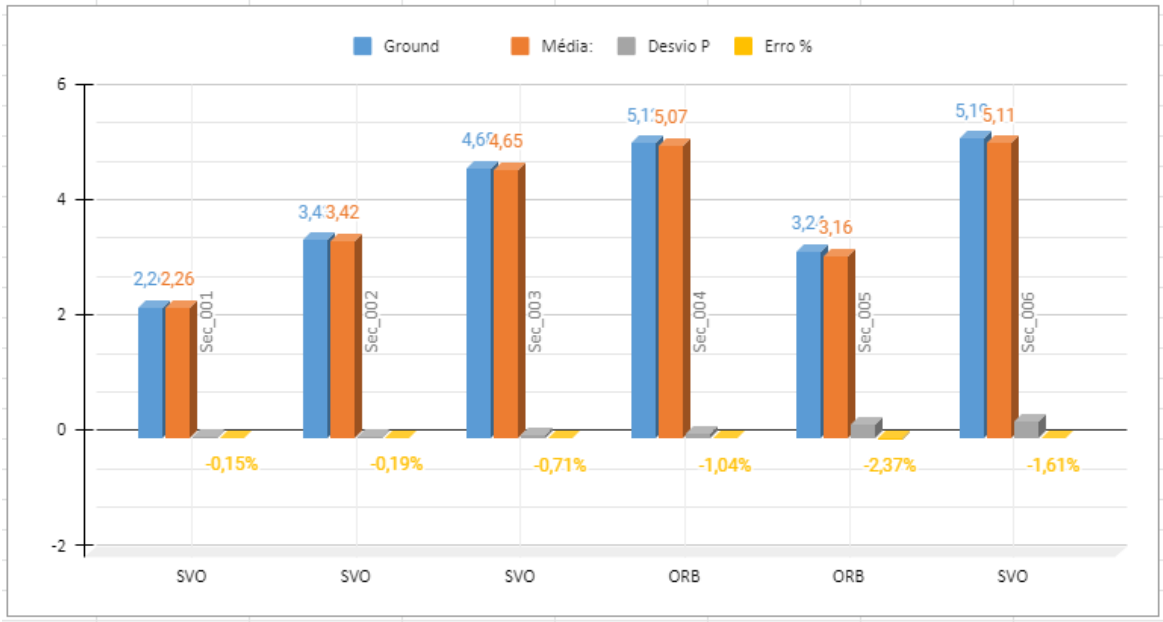
		ORB_Slam	**SVO_Slam**	**LSD_Slam**
Sequência Dia do Experimento	Ground Truth	Média Desvio/Erro	Média Desvio/Erro	Média Desvio/Erro
Sec_001 Dia_01 - Manha	**2.26m**	2.23m 0.04/-1.33%	**2.26m** 0.03/-0.15%	2.29m 0.19/1.33%
Sec_002 Dia_01 - Tarde	**3.43m**	**3.42m** 0.04/-0.29%	**3.42m** **0.02/-0.19%**	3.37m 0.34/-1.75%
Sec_003 Dia_02 - Manha	**4.68m**	4.59m 0.03/-1.85%	**4.65m** 0.05/-0.71%	4.71m 0.11/0.57%
Sec_004 Dia_02 - Tarde	**5.12m**	**5.07m** 0.07/-1.04%	5.19m 0.01/1.37%	5.28m 0.05/3.06%
Sec_005 Dia_03 - Manha	**3.24m**	**3.16m** 0.23/-2.37%	3.40m 0.08/4.94%	3.17m 0.34/-2.26%
Sec_006 Dia_03 - Tarde	**5.19m**	5.34m 0.05/2.89%	**5.11m** 0.29/-1.61%	5.38m 0.13/3.66%

É possível observar nos experimentos demonstrados na tabela 5.2, que para cada gravação denominada "Sec_xxx", executamos cada sequência 3 vezes em busca de uma precisão da trajetória estimada, extraindo assim as informações necessárias como média, desvio padrão, taxa de erro, ao longo dos testes foi observado que o *SVO Semi-Direct Monocular Visual Odometry* se saiu melhor nos experimentos Sec_001, Sec_002 e Sec_003, com sua faixa de erro muito inferior em relação aos demais algoritmos, tornando assim o mais assertivo na média geral com o erro de 0,73%, seguido pelos algoritmos Orb_Slam com sua faixa de erro de -0,80% e LSD_Slam com erro de 0,92%, porém em uma análise mais criteriosa foi notado que o Orb_Slam conseguiu se equiparar ao *SVO Semi-Direct Monocular Visual Odometry* no experimento Sec_002, possuindo o mesmo valor médio com uma diferença no seu desvio padrão: 0.02 e erro de 0.10%, Foi observado que nos experimentos Sec_005 e Sec_006 houve uma oscilação maior entre os resultados obtidos, aumentando o erro e desvio padrão exibida na figura 5.1(b), isso foi devido às mudanças climáticas no momento dos testes, com a chuva e ventos intensos por volta dos 24 km/h, houve um grande impacto na qualidade do experimento devido a grande variação de luz gerada pelas ondas na superfície da piscina impactando diretamente qualidade da imagem. para mais detalhes das informações utilizadas na momento do experimento veja na tabela 5.1 acima:

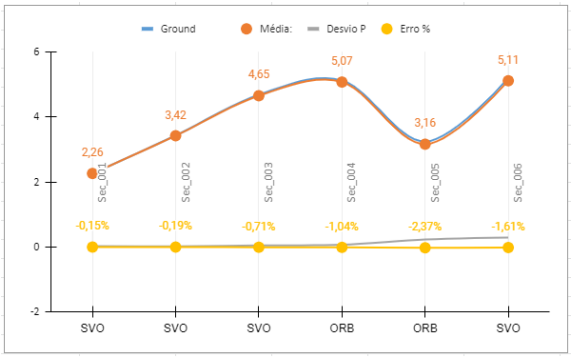
Na figura 5.1(a) abaixo é representada em forma de gráfico os valores dos algoritmos em suas respectivas sessões, na figura 5.1(b) é mostrada de forma bastante objetiva uma linha tênue entre a medida real e a média dos valores alcançados pelo algoritmo em cada sessão,

exibida também na figura 5.1(c) demonstrada em forma de radar.

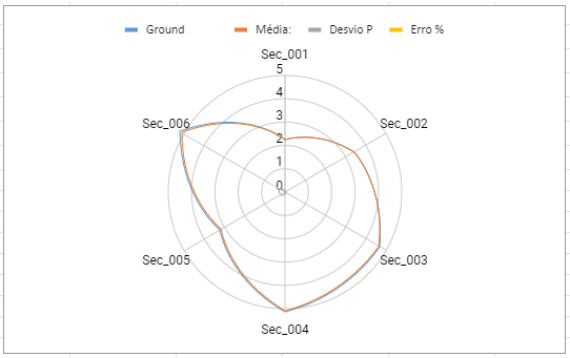
Para executar a sequência de testes de pós processamento foi utilizado um laptop tradicional com as seguintes especificações: processador Intel Core i7-4500U @1.80GHz ∼ @2.40GHz, 8GB de memória RAM, placa de video Nvidia GeForce GT820M com 2GB de RAM contendo 96 cores Cuda e um SSD SanDisk 120GB.



(a) Representação dos resultados em colunas



(b) Representação dos resultados em linhas



(c) Representação dos resultados em radar

Figura 5.1: Gráfico de representação dos resultados (a), (b) e (c). Fonte: Autor

Capítulo Seis

Considerações Finais

Este trabalho teve como objetivo implementar um equipamento de baixo custo para estudos envolvendo visão computacional, navegação e coletas de dados em ambiente subaquático, além de realizar experimentos envolvendo odometria visual utilizando o ROV, mantendo as informações acuradas das condições climáticas durante os experimentos, para isso foi necessário implementar os algoritmos selecionados, construir o ROV para atender aos requisitos específicos e comparar os resultados, além de determinar os algoritmos que apresentaram os melhores resultado nos experimentos.

O maior desafio para a construção do LowcostROV foi desenvolve-lo com o menor curso em sua produção possível, o seu corpo é todo feito de tubo PVC 150mm e suportes impressos em PLA 3d, as hélices são feitas de plástico mostrado na figura 3.5(a). Ao longo desta pesquisa conseguimos superar alguns desafios, o primeiro problema que enfrentamos foi com os algoritmos que não conseguiam identificar nenhuma característica da imagem (cor, forma ou textura), para solucionar este problema foi necessário mapear a piscina com *tags*, aumentando assim a quantidade de *features* do ambiente, no entanto no desenvolvimento dos pisos *tags* existiu a possibilidade do descolamento da plastificação, onde era preciso que as imagens dessas *tags* ficassem rentes e bem possessionadas ao fundo da piscina, como solução foram feitos pisos *tags* de forma artesanal, utilizando fita isolante, fita crepe e sobras das pastilhas dos pisos da própria piscina como mostrado na figura 4.1.

Um dos quesitos importantes para a navegabilidade do LowcostROV era encontrar um motor subaquático com custo/beneficio, pois muitos modelos possuem um custo elevado, foi utilizado uma bomba de porão da Seaflo 1100gph figura 3.6, sendo que a mesma foi desmontada com adição de uma hélice com intensão de impulsionar LowcostROV sento o quesito chave em sua navegação. Para tornar a tp-link NC200 Figura 3.8 hermeticamente fechada, já que existia a possibilidade da mesma se danificar ao longo dos testes foi desenvolver uma proteção hermeticamente fechada em vidro com dimensões apropriadas para a tp-link NC200.

Diante do exposto ao longo deste trabalho, é possível concluirmos que o algoritmo que obteve um melhor resultado em termos de acurácia e precisão em um ambiente simulado controlado foi o *SVO (Semi-Direct Monocular Visual Odometry)* que conseguiu alcançar os melhores resultados em 4 testes de um total de 6 por sessão demonstrando a sua superioridade em relação ao LSD_Slam e o ORB_Slam, este que por sua vez na média geral conseguiu o melhor resultado no desvio padrão médio cerca de 0.09, indicando que

dentre os outros possui menos variações entre um teste e outro.

6.1 Propostas para trabalhos futuros

Como propostas para trabalhos futuros, sugerimos a utilização de uma câmera estéreo a fim de melhorar a precisão da trajetória estimada, devido a um fator extra que é a profundidade da imagem certamente os algoritmos de odometria visual terão um resultado mais preciso e satisfatório, tendo em vista que o LowcostROV não será descartado, pois o mesmo deve ser utilizado e melhorado recebendo a nova câmera com segurança e tendo seus recursos de hardware estendidos como exemplo o cabo umbilical.

Referências Bibliográficas

ALEXANDRESCU, A. *Modern C++ design: generic programming and design patterns applied*. [S.l.]: Addison-Wesley, 2001. 2.4

ALPHAWIRE. *Cabo Alpha Wire 26AWG*. 2020. Disponível em: ⟨http://www.alphawire.com/en/Products/Cable/Xtra-Guard-Performance-Cable/Xtra-Guard-Flex/86503CY⟩. 3.7

BAPPY, D.; RAHMAN, M. H. *A Study in 3D Structure Detection Implementing Forward Camera Motion*. [S.l.]: Omniscriptum Gmbh & Co. Kg., 2012. 2.3.2, 2.5

BAY, H.; TUYTELAARS, T.; GOOL, L. V. Surf: Speeded up robust features. In: SPRINGER. *European conference on computer vision*. [S.l.], 2006. p. 404–417. 2.3.1

BELLAVIA, F.; FANFANI, M.; COLOMBO, C. Selective visual odometry for accurate auv localization. *Autonomous Robots*, Springer, v. 41, n. 1, p. 133–143, 2017. 2.3, 2.3.1, 2.3.2

CANONICAL. *Deliver, maintain,secure and sustain*. 2018. Disponível em: ⟨https://canonical.com⟩. 3.1.1

CHEN, L.-H.; CHIANG, K.-W. The performance analysis of stereo visual odometry assisted low-cost ins/gps integration system. *Smart Science*, Taylor & Francis, v. 3, n. 3, p. 148–156, 2015. 2.3.2, 2.7, 2.8, 2.9

CORKE, P. et al. Experiments with underwater robot localization and tracking. In: IEEE. *Proceedings 2007 IEEE International Conference on Robotics and Automation*. [S.l.], 2007. p. 4556–4561. 2.3.1

CREUZE, V. Monocular odometry for underwater vehicles with online estimation of the scale factor. In: *IFAC 2017 World Congress*. [S.l.: s.n.], 2017. 2.3.1

ENGEL, J.; SCHÖPS, T.; CREMERS, D. Lsd-slam: Large-scale direct monocular slam. In: SPRINGER. *European conference on computer vision*. [S.l.], 2014. p. 834–849. 2.4.1, 2.10, 3.18

EUSTICE, R. M.; PIZARRO, O.; SINGH, H. Visually augmented navigation for autonomous underwater vehicles. *IEEE Journal of Oceanic Engineering*, IEEE, v. 33, n. 2, p. 103–122, 2008. 2.3.1

FORSTER, C.; PIZZOLI, M.; SCARAMUZZA, D. Svo: Fast semi-direct monocular visual odometry. In: IEEE. *2014 IEEE international conference on robotics and automation (ICRA)*. [S.l.], 2014. p. 15–22. 2.4.3

FRAUNDORFER, F.; SCARAMUZZA, D. Visual odometry: Part i: The first 30 years and fundamentals. *IEEE Robotics and Automation Magazine*, v. 18, n. 4, p. 80–92, 2011. 2.3, 2.3.3

GAZEBO. *Gazebo*. 2013. Disponível em: ⟨http://gazebosim.org/⟩. 2.2.1

GEPHI.ORG. *Gephi - The Open Graph Viz Platform*. 2017. Disponível em: ⟨https://gephi.org/⟩. 3.1.3

GOMES, M. *Conceitos, referências e programações básicas com Arduino*. [S.l.]: Assessoria de Inclusão Digital SMED, Porto Alegre - RS, 2014. 3.2.4.1, 3.9

HARDSTORE. Disponível em:¡ https://www.hardstore.com.br/fonte/seventeam-550w-atx12v-v2-2-series-st-550p-ag-black-edition¿. *Acessado em: setembro de*, 2019. 3.2.6, 3.12

HUANG, A. S. et al. Visual odometry and mapping for autonomous flight using an rgb-d camera. In: *Robotics Research*. [S.l.]: Springer, 2017. p. 235–252. 2.3.1

INSTRUCTABLES. *Monster Motor Shield VNH2SP30*. 2016. 2–9 p. Disponível em: ⟨https://www.instructables.com/id/Monster-Motor-Shield-VNH2SP30/⟩. 3.2.5, 3.11

KIM, A.; EUSTICE, R. Pose-graph visual slam with geometric model selection for autonomous underwater ship hull inspection. In: IEEE. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.], 2009. p. 1559–1565. 2.3.1

KIM, A.; EUSTICE, R. M. Real-time visual slam for autonomous underwater hull inspection using visual saliency. *IEEE Transactions on Robotics*, IEEE, v. 29, n. 3, p. 719–733, 2013. 2.3.2

LOGITECHG. *F310 Gamepad - Logitech*. 2020. Disponível em: ⟨https://www.logitechg.com/pt-br/products/gamepads/f310-gamepad.940-000110.html⟩. 3.13

LOWE, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, Springer, v. 60, n. 2, p. 91–110, 2004. 2.3.1

MUR-ARTAL, R.; TARDÓS, J. D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, IEEE, v. 33, n. 5, p. 1255–1262, 2017. 2.4.2, 2.11, 3.16, 3.17

NAWAF, M. et al. Towards guided underwater survey using light visual odometry. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Copernicus GmbH, v. 42, p. 527, 2017. 2.3, 2.3.3

NISTÉR, D.; NARODITSKY, O.; BERGEN, J. Visual odometry. In: IEEE. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* [S.l.], 2004. v. 1, p. I–I. 2.3, 2.3.3

OLIVEIRA, J. et al. Modelo computacional para revisÃo sistemÁtica baseado na teoria de rede complexa. In: . [S.l.: s.n.], 2019. p. 690–696. 1.6

PI, R. Disponível em:¡ http://www.raspberrypi.org/¿. *Acessado em: agosto de*, 2019. 3.2.4.2, 3.10

ROS.ORG. *ROS/Introduction - ROS Wiki*. 2018. Disponível em: ⟨http://wiki.ros.org/ROS/Introduction⟩. 2.1, 3.1.2

SÀNCHEZ, N. D. G.; VARGAS, P. A.; COUCEIRO, M. S. A darwinian swarm robotics strategy applied to underwater exploration. In: IEEE. *2018 IEEE Congress on Evolutionary Computation (CEC)*. [S.l.], 2018. p. 1–6. 1.1

SEAFLO. *Bilge Pump 12v — SEAFLO 1100 GPH DC Submersible Marine Water Pumps'*. 2018. Disponível em: ⟨http://www.seaflo.com/en-us/product/detail/619.html⟩. 3.2.1, 3.6

SOLIDWORKS. *Software de desenho mecânico e CAD 3D*. 2019. Disponível em: ⟨https://www.solidworks.com/⟩. 3.1.5

STIVANELLO, M. E. et al. Correspondência densa para sistemas de visão estereoscópica para robótica móvel. In: *Congresso Brasileiro de Automática*. [S.l.: s.n.], 2008. 2.3.2, 2.4, 2.6, 2.3.2

STRASDAT, H.; MONTIEL, J. M.; DAVISON, A. J. Visual slam: why filter? *Image and Vision Computing*, Elsevier, v. 30, n. 2, p. 65–77, 2012. 2.3.3

TPLINK. 2020. Disponível em: ⟨https://www.tp-link.com/pt/home-networking/cloud-camera/nc200/⟩. 3.8

UBUNTU. *What is Ubuntu*. 2018. Disponível em: ⟨https://help.ubuntu.com/lts/installation-guide/s390x/ch01s01.html?_ga=2.104634883.2038114713.1582800789-242135103.1582800789⟩. 3.1.1

ULTIMAKER. *Cura Software*. 2016. Disponível em: ⟨https://ultimaker.com/en/products/cura-software⟩. 3.1.6

WIRTH, S.; CARRASCO, P. L. N.; CODINA, G. O. Visual odometry for autonomous underwater vehicles. In: IEEE. *2013 MTS/IEEE OCEANS-Bergen*. [S.l.], 2013. p. 1–6. 2.3, 2.3.1, 2.3.1, 2.3.3

YOUSIF, K.; BAB-HADIASHAR, A.; HOSEINNEZHAD, R. An overview to visual odometry and visual slam: Applications to mobile robotics. *Intelligent Industrial Systems*, Springer, v. 1, n. 4, p. 289–311, 2015. 2.3, 2.2, 2.3.1, 2.3, 2.3.2, 2.3.3

ZHOU, Q. et al. a robust method for stereo visual odometry based on multiple euclidean distance constraint and ransac algorithm. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Copernicus GmbH, v. 42, p. 219, 2017. 2.3.2

Um ROV de baixo custo como instrumento para pesquisas de algoritmos e métodos aplicáveis à robótica subaquática

Jótelly Barros Oliveira

Salvador, 2021.