Renato Afonso da Silva Fontão

Navegação de um veículo robótico subaquático em ambiente estruturado baseada em visão monocular

Navigation of an underwater robotic vehicle in a structured environment based on monocular vision

Universidade de Aveiro
2021

Renato Afonso da Silva Fontão

Navegação de um veículo robótico subaquático em ambiente estruturado baseada em visão monocular

Navigation of an underwater robotic vehicle in a structured environment based on monocular vision

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Doutor José Nuno Panelas Nunes Lau, Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, do Doutor Francisco José Curado Mendes Teixeira, Investigador (Nível 2) do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e colaboração científica do Doutor Paulo Artur Pinto de Oliveira Lopes, Professor Auxiliar do Departamento de Física da Universidade de Aveiro.

Dedico este trabalho aos meus pais e ao meu irmão.

o júri / the jury

presidente / president

Professor Doutor António José Ribeiro Neves
Professor Auxiliar, Universidade de Aveiro

vogais / examiners committee

Doutor Nuno Gracias
Investigador, Universidade de Girona

Professor Doutor José Nuno Panelas Nunes Lau
Professor Associado, Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

Esta dissertação não só é este documento apresentado, como também é o culminar de vários anos de trabalho e dedicação, mas nunca teria sido possível fazê-lo sozinho.

Quero agradecer aos meus orientadores, Dr. Nuno Lau e Dr. Francisco Curado que, mesmo num ano atípico em que se trabalhou à distância, sempre se mostraram disponíveis para me guiarem e para me darem a sua valiosa ajuda na realização deste trabalho. Agradeço também a ajuda do Dr. Paulo Lopes na fase inicial desta dissertação.

Aos meus amigos, tanto os que conheço desde sempre como os que conheci em Aveiro, um grande obrigado pela vossa amizade, companheirismo e pelas histórias que juntos criámos e experienciámos, fossem em ambiente académico ou não, ao longo destes anos. A conclusão desta etapa também não seria possível sem vocês.

Finalmente, mas não menos importante, um grande obrigado à minha família, especialmente aos meus pais, que sempre acreditaram em mim e me deram todas as condições e todo o apoio necessários para que eu pudesse alcançar os meus objetivos. Um grande obrigado ao meu irmão por sempre me ouvir falar de algo mais técnico, apesar de não estar a perceber. Amo-vos.

Palavras Chave

robótica subaquática, ROV, odometria visual, odometria visual-inercial, visão monocular

Resumo

Durante as últimas décadas, o desenvolvimento de veículos subaquáticos não tripulados (na sigla em inglês – UUV) permitiu a execução de atividades subaquáticas onde a presença humana não era possível. Para auxiliar a operação destes veículos e dotá-los de uma maior autonomia, existe a necessidade de determinar a sua localização no espaço 3-D. Entre as diferentes técnicas existentes para a localização subaquática, as técnicas baseadas em visão computacional são bastante atrativas pois as câmaras fazem parte do equipamento padrão de um veículo robótico subaquático, permitindo obter soluções de localizações de baixo custo. Nesta dissertação, é abordado o tema da localização subaquática com recurso a uma única câmara, a odometria visual monocular, e com recurso a uma câmara auxiliada por uma Unidade de Medição Inercial (na sigla em inglês - IMU), a odometria visual-inercial monocular. A IMU é um sensor composto por um acelerómetro, um giroscópio e um magnetómetro, os quais se encontram no veículo utilizado nesta dissertação, o Pro4 ROV da VideoRay. São explorados diferentes métodos de odometria visual-inercial e odometria visual, e a sua aplicação ao meio subaquático. Devido às condições difíceis do meio, são exploradas duas formas de melhorar a visibilidade das imagens adquiridas com o objetivo de melhorar o desempenho dos algoritmos avaliados. Como a aquisição de dados no ambiente subaquático não é trivial, não existe muita informação sobre o desempenho dos métodos utilizados no meio em estudo. Deste modo, antes de os aplicar no Pro4 ROV, juntamente com as melhorias propostas, os algoritmos foram aplicados sobre um *dataset* subaquático público. Devido a fatores que condicionaram a utilização do Pro4 num ambiente real e a dificuldades técnicas na leitura dos dados da IMU, foram definidos um conjunto de testes em ambiente terrestre utilizando apenas os métodos de odometria visual, com os objetivos de validar o processo de calibração efetuado e demonstrar a aplicação dos algoritmos no veículo utilizado. Os resultados obtidos com o *dataset* demonstram que a utilização de uma câmara e de uma IMU no meio subaquático permite obter uma solução de localização de baixo custo com uma precisão submétrica. Em particular, a visibilidade da imagem revela ser um fator determinante para o aumento dessa precisão. Relativamente ao resultados obtidos com o ROV, estes destacam a importância do aproveitamento do largo campo de visão da câmara para o desempenho da odometria visual.

Keywords underwater robotics, ROV, visual odometry, visual-inertial odometry, monocular vision

Abstract

During the past few decades, the development of Unmanned Underwater Vehicles (UUV) has allowed underwater activities to be carried out where human presence was not possible. To assist the operation of these vehicles and provide them with greater autonomy, there is a need to determine their position in the 3-D space. Among the different existing techniques for underwater localization, techniques based on computer vision are quite attractive because cameras are part of the standard equipment of an underwater robotic vehicle, allowing to obtain low cost localization solutions. In this dissertation, the theme of underwater localization is addressed using a single camera, called monocular visual odometry, and using a camera aided by an Inertial Measurement Unit (IMU), called monocular visual-inertial odometry. The IMU is a sensor composed of an accelerometer, a gyroscope and a magnetometer, which are part of the vehicle used in this dissertation, the VideoRay Pro4 ROV. Different methods of visual-inertial odometry and visual odometry are explored, including their application to the underwater environment. Due to the difficult conditions of the environment, two ways of improving the visibility of the acquired images are explored in order to improve the performance of the evaluated algorithms. As the acquisition of data in the underwater environment is not trivial, there is not much information on the performance of the methods used in the study environment. Thus, before applying them to the Pro4 ROV, as well as the proposed improvements, the algorithms were applied over a public underwater dataset. Due to factors that conditioned the use of Pro4 in a real environment and technical difficulties in reading the IMU data, a set of tests in terrestrial environment were defined using only the methods of visual odometry, with the purpose of validating the calibration process performed and demonstrate the application of the algorithms in the vehicle used. The results obtained with the dataset demonstrate that the use of a camera and an IMU in the underwater environment allows to obtain a low-cost localization solution with submetric precision. In particular, the visibility of the image proves to be a determinant factor for increasing this accuracy. Regarding the results obtained with the ROV, these highlight the importance of taking advantage of the wide field of view of the camera.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AUV	*Autonomous Underwater Vehicle*
BA	*Bundle Adjustment*
CLAHE	*Contrast-Limited Adaptive Histogram Equalization*
CPU	*Central Processing Unit*
DR	*Dead Reckoning*
DVL	*Doppler Velocity Log*
EKF	*Extended Kalman Filter*
GPS	*Global Positioning System*
GPU	*Graphical Processing Unit*
IEKF	*Iterated Extended Kalman Filter*
INS	*Inertial Navigation System*
IMU	*Inertial Measurement Unit*
LBL	*Long Baseline*
PGO	*Pose-Graph Optimisation*
PnP	*Perspective-n-Point*
RGB	*Red, Green, Blue*
RGB-D	RGB-*Depth*
RMSE	*Root Mean-Square Error*
ROV	*Remotely Operated Vehicle*
SBL	*Short Baseline*
SLAM	*Simultaneous Localisation and Mapping*
USBL	*Ultrashort Baseline*
UUV	*Unmanned Underwater Vehicle*
VIO	Odometria Visual-Inercial
VO	Odometria Visual
vSLAM	*Visual* SLAM

CAPÍTULO 1

Introdução

1.1 Enquadramento e motivação

Com o rápido desenvolvimento da tecnologia que ocorre atualmente, a robótica é uma das áreas de estudo que mais tem vindo a beneficiar disso mesmo. Este campo multidisciplinar envolve a conceção, construção e operação de máquinas (robôs) com o objetivo de auxiliar a ação humana ou até substituí-la. Entre as diversas aplicações que existem, no âmbito desta dissertação insere-se a robótica marítima.

Efetivamente, o oceano constitui uma importante fonte de alimentação, emprego ou recursos minerais e energéticos, cobrindo cerca de 70% da superfície terrestre e albergando diversas formas de vida. No entanto, o oceano ainda não foi devidamente explorado, principalmente em profundidade, em que o ambiente é mais adverso para o ser humano. Fatores como a elevada pressão, baixas temperaturas e limite de oxigénio tornam difícil a missão de um mergulhador.

Nesse sentido, existem os veículos robóticos que podem operar nestes ambientes, sem a necessidade de transportar tripulantes, denominados *Unmanned Underwater Vehicle* (UUV). Estes veículos estão divididos em duas categorias: ROV e *Autonomous Underwater Vehicle* (AUV). Os AUV são autónomos, pelo que não necessitam do controlo de um operador, e percorrem trajetórias pré-programadas. Para tal, este veículo é dotado de vários sensores, bateria e um computador integrado que permite que o sistema robótico possa navegar de forma autónoma. Relativamente ao ROV, este é comandado pelo operador através de um cabo umbilical, que serve como fonte de energia e para envio de sinais de controlo e observar as imagens captadas pela câmara. Ao contrário do AUV, que guarda toda a informação da câmara e dos sensores, no ROV é possível aceder a estes dados em tempo real. Por estes motivos, o AUV é normalmente usado em missões que não requerem uma supervisão constante, ao contrário do ROV. Este veículo é utilizado em operações mais complexas, as quais podem incluir a navegação num cenário industrial, como em plataformas petrolíferas, ou num cenário de catástrofe como o naufrágio de navios. Estes são alguns dos cenários em que o ROV representa uma alternativa bastante viável a um mergulhador e um AUV.

Contudo, existem problemas associados a este tipo de operações, que consistem na dificuldade da pilotagem ou no posicionamento do veículo, bem como o mapeamento, apenas com base na informação visual, devido às propriedades do meio subaquático que diminuem a visibilidade.

Por conseguinte, seria importante ter um sistema de navegação que auxiliasse na pilotagem do ROV, através da determinação da posição do agente robótico no espaço 3-D. Isto permitiria, por exemplo, localizar ou mapear certas regiões de interesse.

Deste modo, os métodos de odometria baseados em visão e fusão de dados sensoriais, por exemplo, obtidos a partir de uma IMU, são bastante atrativos. Isto deve-se ao facto de que a câmara e os sensores inerciais fazem parte do equipamento padrão de um ROV, o que permitiria ter uma solução de baixo custo sem o uso de instrumentação externa.

1.2 Objetivos

O objetivo principal da dissertação é a contribuição para o desenvolvimento da temática da navegação subaquática, com recurso a sensores de baixo custo, nomeadamente a câmara monocular e a *Inertial Measurement Unit* (IMU). As principais soluções de localização para o meio subaquático são baseadas em sensores dispendiosos, pelo que se pretende avaliar a viabilidade dos sensores mencionados como única fonte para a localização do veículo robótico subaquático utilizado, o VideoRay Pro4 ROV.

Como as principais soluções de localização baseadas em visão foram desenvolvidas e avaliadas para o meio terrestre ou aéreo, pretende-se estudar de que forma os algoritmos do estado da arte estão preparados para atuar num ambiente subaquático e como é possível adaptá-los para o meio em estudo.

1.3 Estrutura da Dissertação

No **Capítulo 2** são abordadas as diferentes técnicas de navegação utilizadas no meio subaquático, com um foco na navegação ótica. É apresentado o estado da arte atual da odometria visual e odometria visual-inercial, assim como são revistas as principais perspetivas relacionadas com a navegação ótica no meio subaquático.

O **Capítulo 4** apresenta as principais técnicas e noções matemáticas utilizadas nos algoritmos do estado da arte. Neste capítulo ainda é descrita toda a implementação associada ao trabalho experimental desenvolvido.

No **Capítulo 5** são apresentados e analisados os resultados obtidos com um *dataset* público e com o VideoRay Pro4 ROV.

O **Capítulo 6** apresenta a conclusão desta dissertação e são discutidas futuras linhas de trabalho.

No **Anexo** são apresentados resultados suplementares sobre o *dataset* utilizado.

CAPÍTULO 2

Revisão da literatura

2.1 Problema da localização

De forma a tornar autónoma a tarefa de navegação, o agente tem de localizar-se a si próprio no ambiente em que se encontra. Para atingir este objetivo, o robô necessita de sensores para obter informação seja do ambiente que o rodeia (exterocetivos) ou do seu estado atual (propriocetivos).

No geral, para qualquer veículo, pretende-se que a localização de um robô móvel seja precisa. Isto é especialmente importante em UUV, de modo a permitir uma navegação segura e a recolha do veículo do mar, assim como um registo correto dos dados observados.

Stutters, Liu, Tiltman et al. [1] dividem a navegação subaquática em três métodos distintos, aos quais se pode adicionar um quarto método que tem vindo a ter um interesse crescente, o ótico. Estes métodos estão representados na figura 2.1.

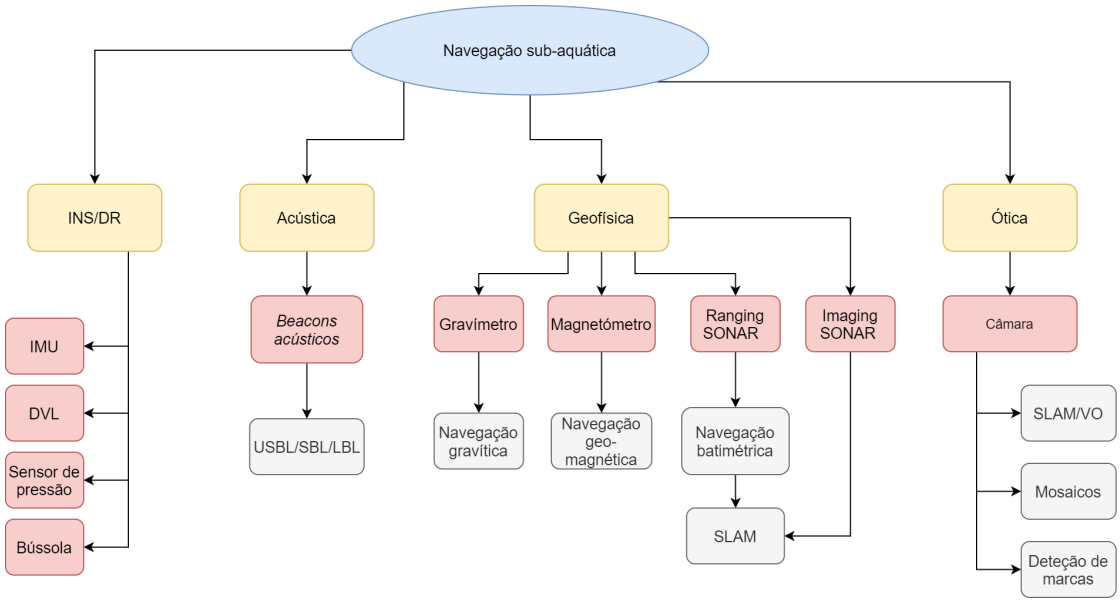


Figura 2.1: Tipos de navegação no meio subaquático. Adaptado de [2]

.

2.2 Sistema de Navegação Inercial (INS) e *Dead Reckoning*

De acordo com Leonard, Bennett, Smith et al. [3], *Dead Reckoning* (DR) é o método mais simples de obter a posição. A técnica consiste na integração da velocidade do veículo relativamente ao tempo para obter uma nova posição. A forma mais básica de obter a velocidade e a orientação do veículo é através do uso de um sensor de velocidade da água e uma bússola.

Contudo, a corrente do oceano adiciona uma componente de velocidade ao veículo que não é detetada pelo sensor de velocidade. Isto provoca erros de posicionamento, especialmente quando o robô se desloca a baixas velocidades, como ilustrado na figura 2.2.

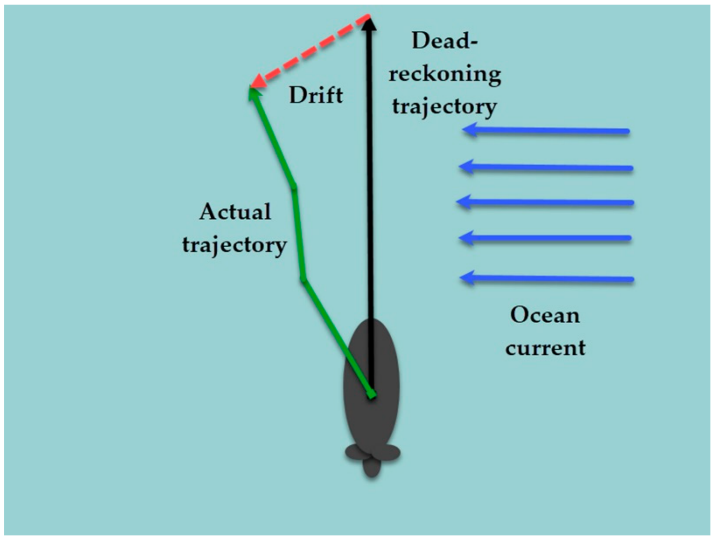


Figura 2.2: Efeito das correntes oceânicas na trajetória estimada em técnicas de DR. Fonte: [2].

Os sensores inerciais podem ser usados para melhorar a qualidade e fiabilidade dos métodos de *Dead Reckoning*, a partir de valores calculadas por uma IMU. A IMU é normalmente constituída por um giroscópio e um acelerómetro tri-axiais que medem, a velocidade angular e a aceleração linear do veículo, respetivamente. A integração destes valores, permite obter uma estimativa da posição e da orientação do veículo.

A principal vantagem de um *Inertial Navigation System* (INS) é que é um sistema independente. Um INS não depende da receção ou envio de sinais, logo é imune a interferências, sejam intencionais ou não.

No entanto, as IMU não resolvem totalmente o problema do DR. As correntes oceânicas quando constantes e de baixa velocidade podem não ser detetadas. Além disto, como qualquer outro sensor, as IMU têm fontes de erro intrínsecas [4]. Ainda assim, mesmo que o instrumento fosse perfeito, as estimativas calculadas por um INS seriam afetadas no decorrer da navegação [5]. Este facto deve-se a existirem distúrbios na gravidade da Terra (defleções verticais) que o INS não tem em conta.

Os valores lidos estão, portanto, sujeitos a erros e são contínuamente integrados ao longo do tempo, o que faz com que a posição estimada se vá afastando cada vez mais da posição

real.

Além da IMU, outro sensor que é possível utilizar em técnicas de DR é o sensor de *Doppler Velocity Log* (DVL). Este sensor envia 4 ondas acústicas para superfícies estáticas, e mede as diferenças de frequências dos ecos dos sinais recebidos. Com base no efeito de Doppler, fornece uma estimativa da velocidade do veículo respetivamente à superfície. Para reduzir a degradação da posição ao longo do tempo, Leonard, Bennett, Smith et al. [3] sugerem usar um DVL em longas missões. Porém, o DVL tem um alcance limitado, pelo que só é útil quando se encontra perto da superfície de reflexão. Esta pode ser uma superfície estática, como o relevo oceânico, ou a camadas de água refletivas, que têm o problema de serem móveis.

McEwen, Thomas, Weber et al. [6] numa operação com um AUV no Ártico destacam as limitações das técnicas de *Dead Reckoning* e do DVL. Para corrigir o erro crescente, o AUV voltava à superfície para receber o sinal GPS. Além disto, o DVL teve de ser colocado em direção ao gelo, visto que o relevo oceânico não estava dentro do alcance do equipamento.

Em conjunto com os sensores mencionados, também podem ser incorporado num INS o sensor de pressão, uma girobússola e uma bússola, o que permite ter melhores estimativas da localização.

O sensor de pressão permite obter uma estimativa da profundidade a que o veículo se encontra, o que permite diminuir o erro na estimativa da componente vertical do veículo. Todavia, fatores ambientais como as correntes oceânicas, a salinidade e a temperatura influenciam as medidas lidas pelo sensor de pressão.

Uma girobússola determina a orientação do veículo relativamente ao norte geográfico e a bússola determina a orientação relativamente ao norte magnético. No entanto, o custo, o tamanho e o consumo de energia de uma girobússola tornam a aplicação deste instrumento exclusiva a operações militares, como indicam Kinsey, Eustice e Whitcomb [7]. Por sua vez, a aplicação da bússola é suscetível a erros em zonas com fortes anomalias magnéticas na zona.

2.3 Navegação acústica

De acordo com LaPointe [8] a navegação acústica consiste em técnicas de posicionamento de um veículo que envolvem a medição do tempo de voo de sinais acústicos, utilizando para o efeito dispositivos (*beacons*) recetores, denominados hidrofones, e um transmissor acústico. Ao contrário das ondas eletromagnéticas e rádio, na água a atenuação do som é bastante menor, especialmente em baixas frequências, o que permite que as ondas acústicas percorram distâncias maiores.

Contudo, este tipo de navegação envolve a instalação de equipamento externo, seja na embarcação ou debaixo de água, o que torna a sua implementação mais complexa. Além disto, os sistemas de posicionamento acústicos dão estimativas da localização com baixa frequência e com pouca precisão.

Como Vickery [9] referem, os métodos de navegação acústica mais comuns podem ser agrupados consoante a distância entre os recetores: *Ultrashort Baseline* (USBL) (menor que 10cm), *Short Baseline* (SBL) (20cm até 50m) e *Long Baseline* (LBL) (100m até 6km).

2.3.1 *Ultrashort Baseline*

Nos sistemas de posicionamento acústico USBL existe apenas um hidrofone colocado na embarcação, constituído por vários elementos recetores, e um dispositivo transmissor que acompanha o veículo subaquático. A posição é medida através da diferença de fase do sinal acústico que passa pelos diferentes elementos do hidrofone. No entanto, esta posição é estimada relativamente à embarcação que se encontra à superfície, pelo que normalmente se tira partido da estação à superfície e se utiliza o sinal GPS para referenciar as medidas obtidas pelo sistema USBL relativamente a um sistema de coordenadas global.

Devido à baixa frequência de operação, à imprecisão e ao ruído presente nas medidas, o USBL é normalmente combinado com outros sensores para tornar a navegação mais robusta e precisa.

Neste sentido, Rigby, Pizarro e Williams [10] utilizaram um filtro de partículas para fusão das observações do USBL com as velocidades obtidas por um DVL. O modelo do movimento é baseado em *Dead Reckoning*, integrando as medidas do DVL. As posições estimadas pelo sistema USBL, quando disponíveis, servem para corrigir o peso das partículas do filtro, dando um maior peso àquelas com maior probabilidade de indicarem a verdadeira trajetória do veículo. Estes testes foram realizados *offline* com dados obtidos de uma operação com um AUV, pelo que não se aplicam os problemas de sincronização das medidas do USBL numa operação em tempo real, como é discutido por Font, Bonin-Font, Negre et al. [11].

2.3.2 *Short Baseline*

Nos sistemas SBL são utilizados pelo menos três hidrofones, colocados nas extremidades da embarcação, e a posição estimada é derivada a partir das diferenças dos tempos de chegada da onda acústica a cada recetor. Tal como no USBL, a posição calculada não é global.

Contudo, como indicam Christ e Wernli [12], o desempenho do SBL é degradado quando a distância entre os *beacons* recetores é inferior a 10m, devido ao efeito do caminho múltiplo. Por isto, a implementação deste sistema em embarcações pequenas não é possível. Além disto, implica a instalação de mais hidrofones relativamente ao USBL. Por estas razões, o USBL é normalmente o escolhido para ser implementado.

2.3.3 *Long Baseline*

Nos sistemas LBL são colocados diversos hidrofones (4-12) debaixo de água em localizações conhecidas. Sabendo as coordenadas geográficas do conjunto de hidrofones e com recurso a técnicas de multilateração, é possível calcular a posição do veículo subaquático. Como é conhecida a localização dos recetores, a posição calculada pelo sistema LBL é global e, portanto, não é necessário incluir medidas de referência globais. Apesar de este sistema ter uma precisão maior que os outros sistemas de posicionamento acústico, a sua implementação acarreta elevados custos monetários e é demorada.

Neste sentido, LaPointe [8] introduziu o conceito de LBL virtual, utilizando apenas um hidrofone com a localização fixa e conhecida. A trajetória é estimada com base nas leituras de um DVL e de uma bússola. Após quatro medidas de distância entre o veículo subaquático

e o hidrofone, é definida uma posição virtual deste hidrofone para cada instante, segundo a distância medida e a trajetória estimada por *Dead Reckoning*. Após se conhecer a geometria da rede de hidrofones virtuais, a posição é corrigida de forma semelhante a uma rede LBL clássica.

Devido à implementação complexa e difícil calibração destes sistemas, Thomas [13] propôs usar uma configuração invertida do LBL, através do uso de sistemas GIB (*GPS Intelligent Buoys*). Estes sistemas utilizam entre 4 e 12 bóias equipadas com recetores GPS e hidrofones submersos, assim como um *pinger* acústico colocado no veículo subaquático que opera a uma determinada frequência. Uma unidade central recebe, via rádio, os tempos de chegada registados em cada bóia e a profundidade a que estas se encontram. Desta forma, esta unidade central, normalmente colocada numa embarcação à superfície, calcula a posição do veículo através de técnicas de trilateração.

2.4 Navegação geofísica

Segundo Melo e Matos [14], a navegação geofísica baseia-se no uso de caraterísticas físicas do ambiente em que o veículo subaquático se encontra para produzir uma estimativa da localização, através de um mapa previamente disponibilizado. Nesta técnica de navegação são explorados parâmetros geofísicos como o campo geomagnético, a topografia do terreno ou as anomalias gravíticas contidas no mapa.

As vantagens deste método residem no facto de ter um alcance elevado e numa maior facilidade de aplicação relativamente aos sistemas de posicionamento acústico, visto que não é necessária a instalação de equipamento adicional.

No entanto, também apresentam limitações, das quais se destacam a necessidade de obter um mapa *a priori* do ambiente, o qual nem sempre está disponível para as regiões em que a missão se realiza, a complexidade de encontrar uma correlação entre o mapa e as informações lidas pelos sensores. Além disso, o ambiente em que decorre a operação tem de ser suficientemente informativo, isto é, deve apresentar anomalias gravíticas ou magnéticas, assim como também um terreno irregular.

Deste modo, a navegação por gravidade explora o facto de o campo gravitacional da Terra não ser uniforme, utilizando para o efeito um gravímetro ou um gradiómetro de gravidade. Estes sensores têm de ser montados numa plataforma estável e isolada de vibrações, o que é difícil de concretizar num ROV ou AUV mais pequenos. Este fator limita a aplicação desta navegação em UUV, conhecendo-se apenas a sua implementação numa classe de submarinos militares, como indicam Rice, Kelmenson e Mendelsohn [5].

Relativamente à navegação geomagnética, esta tira partido das anomalias do campo magnético da Terra para estimar a sua posição. De acordo com Zhao, Wang e Wang [15], ao contrário da navegação baseada na gravidade e na topografia do terreno, um mapa geomagnético tem variações maiores, o que diminui a probabilidade de haver um erro na associação das medidas com o mapa.

No entanto, o campo magnético terrestre varia diariamente ainda que de forma previsível, o que permite remover os efeitos dessas variações, como foi demonstrado por Kato e Shigetomi

[16]. Ainda assim, como referido por Teixeira e Pascoal [17], a maior desvantagem da navegação geomagnética prende-se com o facto de ser dependente da profundidade a que o mapa é registado. Isto deve-se à intensidade do campo magnético variar com a distância a que os sensores magnéticos se encontram da fonte magnética.

Por último, surge a navegação baseada no terreno, ou batimétrica, que consiste em estimar uma posição com base em medições da topografia do terreno, e compará-las com o mapa disponível. Conforme Melo e Matos [14] afirmam, a implementação deste método de navegação requer que se tenha informação acerca da profundidade a que o veículo subaquático se encontra relativamente à superfície, bem como a altitude relativamente ao fundo oceânico. Para medir a profundidade, pode ser usado um sensor de pressão utilizada uma fórmula de conversão de pressão para profundidade como a apresentada por Fofonoff e Millard [18]. Para obter o perfil do terreno são utilizados sonares numa configuração ativa. Neste tipo de configuração é emitida uma onda acústica e a reflexão é recebida pelo sonar. Existem diferentes tipos de sonares, consoante o número de pulsos acústicos enviados, o que influencia a precisão das medidas obtidas e o custo do equipamento.

Contudo, se não houver um mapa disponível, é possível adotar uma estratégia de SLAM baseada em sonares. No caso dos sonares que permitem apenas obter o perfil do terreno, as medidas lidas servem para construir um mapa batimétrico do ambiente que limita o erro de navegação do veículo, como foi demonstrado por Barkby, Williams, Pizarro et al. [19]. Ainda existem outro tipo de sonares, que constroem uma imagem do terreno observado. Estes sonares, para além de medirem o tempo de voo da onda acústica, medem a intensidade da onda refletida. Emitindo vários pulsos acústicos, este tipo de sonares permite criar uma imagem. Deste modo, um mapa do terreno observado vai sendo criado, no qual o veículo se localiza, como abordado por Romagós et al. [20].

2.5 Navegação ótica

Entre os sensores que fornecem informação acerca do ambiente externo, denominados exterocetivos, a câmara é uma alternativa bastante promissora. Excluindo as câmaras RGB-*Depth* (RGB-D), as câmaras tradicionais são sensores passivos que medem a reflexão da luz nos objetos e não dependem do envio de sinais. Além disto, a câmara é um equipamento acessível e compacto, pelo que podem ser facilmente integradas num UUV.

Efetivamente, existem diferentes tipos de câmaras, dependendo do número de lentes, ou da utilização de um sensor para medir a profundidade:

- Câmara monocular: As imagens são captadas a partir de um único sensor e lente. Tem como vantagens o facto de ser uma solução de baixo custo, de pequenas dimensões, e também por ser de fácil calibração. No entanto, a desvantagem de utilizar uma só câmara é que não é possível observar a escala da imagem, o que faz que com métodos de navegação baseados em visão monocular sofram de *scale drift* e tenham uma inicialização mais complexa.
- Câmaras *stereo*: Em visão estereoscópica, o sistema é normalmente composto por duas lentes, colocadas de forma a ter campos de visão sobrepostos. Isto permite que seja

possível retirar a escala das imagens observando os pixeis que são correspondentes entre as imagens capturadas pelas duas câmaras e conhecendo a distância entre as lentes. Assim, o problema de inicialização do método de navegação não se verifica neste caso. No entanto, as câmaras têm de ser calibradas com maior cuidado e a distância entre as duas câmaras deve ser constante.

- Câmaras RGB-D: Este tipo de câmaras consiste na combinação de uma câmara *Red,Green,Blue* (RGB) com um sensor de profundidade baseado em luz estruturada ou tempo de voo. Isto permite diretamente criar mapas detalhados com a distância a que a câmara se encontra relativamente ao ambiente ao redor. Contudo, não é possível utilizar este tipo de câmaras debaixo de água, porque utilizam luz infravermelha para determinar a escala da imagem, o que não é praticável no meio em estudo.

2.5.1 Odometria Visual e SLAM Visual

O processo de odometria visual consiste em, através de uma sequência de imagens, estimar o movimento de um agente, seja este um humano, um robô ou um veículo. Considere-se um conjunto de imagens adquiridas consecutivas:

$$\mathcal{I} = \{I_{k-1}, I_k, I_{k+1}, \ldots, I_n\} \tag{2.1}$$

De acordo com Scaramuzza e Fraundorfer [21], o problema da odometria visual pode ser definido formalmente pelo seguinte. A cada imagem $\mathcal{I}_k$ pretende-se calcular a pose da câmara $\mathcal{C}_k$, a qual é definida por:

$$C_k = \begin{bmatrix} p_k & q_k \end{bmatrix}, \;\; p_k \in \mathbb{R}^3, \;\; q_k \in \mathbb{SO}(3) \tag{2.2}$$

onde p_k representa a posição da câmara e q_k é a sua orientação, ambas relativamente a um sistema de coordenadas fixo.

Duas poses da câmara $\mathcal{C}_{k-1}$ e $\mathcal{C}_k$ são relacionadas por uma transformação $\mathcal{T}_k \in \mathbb{SE}(3)$ definida por:

$$\mathcal{T}_k = \begin{bmatrix} R_k & t_k \\ 0 & 1 \end{bmatrix}, \;\; R_k \in \mathbb{SO}(3), \;\; t_k \in \mathbb{R}^{3x1} \tag{2.3}$$

onde R_k é a matriz de rotação e t_k o vetor de translação.

A pose da câmara $\mathcal{C}_k$ é, portanto, dada por:

$$C_k = C_{k-1} T_k \tag{2.4}$$

Deste modo, o objetivo da odometria visual é estimar uma trajetória, a qual é definida por um conjunto de poses da câmara $\mathcal{C} = \{\mathcal{C}_0, \ldots, \mathcal{C}_n\}$, através do cálculo de um conjunto de transformações $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ que relacionam as poses da câmara estimadas no conjunto de imagens $\mathcal{I}$.

O termo «odometria visual» foi introduzido por Nister, Naroditsky e Bergen [22] devido à sua semelhança com o conceito de *wheel odometry*, na qual a posição de um agente é estimada integrando o número de voltas das rodas relativamente ao tempo do percurso. De forma análoga, os métodos de odometria visual estimam o movimento de forma incremental a partir de imagens consecutivas. Helmick, Yang Cheng, Clouse et al. [23] mostram que a odometria visual apresenta resultados significativamente melhores do que a *wheel odometry*. Para além do facto de o desempenho da *wheel odometry* ser bastante afetado por terrenos irregulares, os quais provocam a derrapagem do robô, a *wheel odometry* não é um método geral. Robôs como os subaquáticos ou aéreos, não têm a estrutura necessária para que se aplique esse método. Por outro lado, estes robôs normalmente têm uma câmara integrada, o que torna a odometria visual uma solução mais atrativa.

Contudo, a odometria visual estima o movimento de forma incremental, pelo que a posição calculada acumula erros a longo prazo. Esta técnica permite precisão a curto prazo, pelo que seria desejável ter informação sobre o ambiente que rodeia o agente, de forma a situá-lo no espaço e a reduzir o erro crescente. Com base em dados sensoriais é possível criar um mapa, mas a criação de um mapa envolve estimar a localização do ambiente relativamente ao agente, o qual também tem de se localizar relativamente ao ambiente observado.

Portanto, é necessário ter um mapa para o agente se localizar, mas para o mesmo se localizar é necessário criar um mapa. Este processo é conhecido por SLAM onde como o nome indica, a localização e o mapa são resolvidos simultaneamente. Esta técnica permite que o agente seja capaz de navegar sem conhecimento prévio do ambiente em que se encontra, através da estimação do seu movimento e da construção de um mapa ao longo da navegação.

A forma como um sistema de vSLAM ou VO explora a informação contida nas imagens permite distinguir o método utilizado, podendo inferir-se, portanto, a existência de métodos indiretos e diretos. As principais diferenças estão ilustradas na figura 2.3.

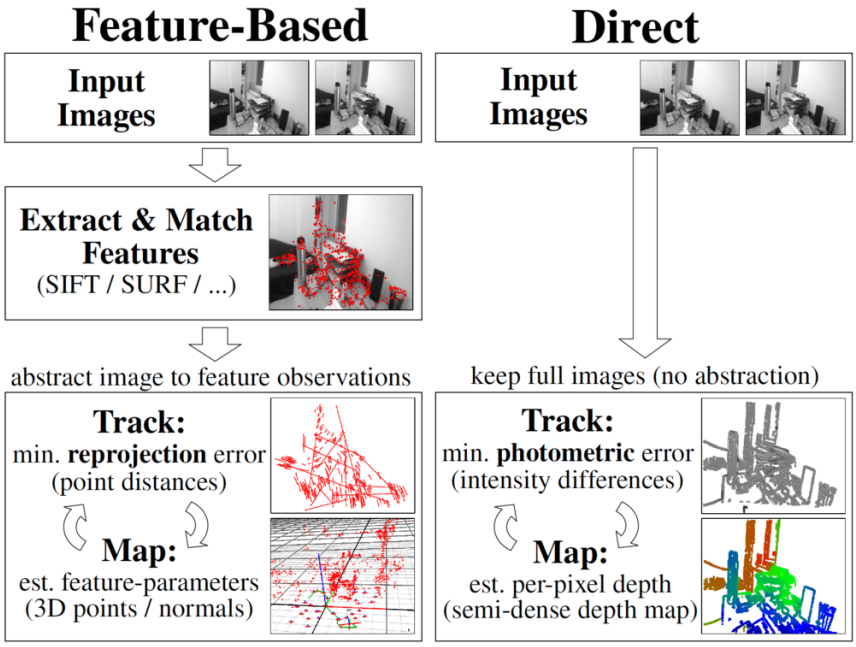


Figura 2.3: Diferença entre os métodos baseados em extração de caraterísticas e os métodos diretos. Fonte: [24].

2.5.1.1 Métodos Indiretos

Assim, os métodos indiretos processam a imagem de modo a extrair pontos de interesse (*keypoints*) distintos, os quais podem ser detetados em imagens consecutivas, idealmente, sob diferentes pontos de vista ou mudanças de iluminação.

Para esse efeito, segundo Tuytelaars e Mikolajczyk [25] um algoritmo de deteção deve ter as seguintes propriedades:

- *Precisão da localização*: um detetor deve identificar com rigor a localização do ponto na imagem, bem como a escala;
- *Repetibilidade*: dada uma sequência de imagens da mesma cena, os mesmos pontos, ou uma grande percentagem deles, detetados na primeira imagem, devem ser detetados nas imagens seguintes;
- *Distintivo*: os *keypoints* detetados devem conter alguma informação que os diferencie entre si;
- *Eficiência computacional*: devido a aplicações de tempo-real, em que existem restrições temporais e computacionais, o algoritmo de deteção deve ter o menor tempo de processamento possível;
- *Robustez*: um detetor deve identificar os mesmos pontos em imagens seguintes, independentemente do ruído, artefactos de compressão e desfoque. Além destes fatores, em situações de mudanças de escala, rotação ou iluminação, o algoritmo deve detetar os mesmos pontos.

Figura 2.4: Estrutura de um método indireto de odometria visual.

As componentes fundamentais de um método indireto de VO estão ilustradas na figura 2.4.

Após a deteção dos pontos de interesse, a região à volta de cada um é compactada num vetor, seja com valores binários ou valores reais dos pixeis, criando um descritor. A comparação entre diferentes descritores permite que seja possível fazer corresponder os *keypoints* entre diferentes imagens.

Em vez de descrever a região, uma alternativa é fazer o *tracking* dos pontos de interesse. Para o efeito, por métodos de fluxo ótico como o de Lucas e Kanade [26], é possível calcular o deslocamento dos pixeis. Por não ser necessário usar descritores, este método é mais leve computacionalmente quando aplicado a um conjunto de pixeis.

Deste modo, após as caraterísticas serem associadas em duas imagens consecutivas, é calculada uma transformação que descreve o movimento entre essas mesmas duas imagens. Esta transformação pode ser estimada através da resolução de problemas geométricos como a geometria epipolar ou o *Perspective-n-Point* (PnP). Na geometria epipolar o movimento é estimado a partir de caraterísticas extraídas em imagens consecutivas, pelo que a escala do movimento é desconhecido. O movimento estimado com PnP consiste em associar caraterísticas no espaço 3-D e as caraterísticas da imagem. Este processo implica que se saiba a posição de pontos 3-D, estimados a partir da triangulação de pontos.

Cada transformação estimada tem uma incerteza associada que se irá propagar para a seguinte, afetando a qualidade da trajetória estimada. Especialmente em visão monocular, este efeito é mais acentuado, visto que a incapacidade de extrair a profundidade da imagem torna bastante mais complexa a resolução dos problemas geométricos, pelo que as soluções

podem não ser ótimas.

Assim, é necessário diminuir a incerteza por técnicas de otimização não linear que visam diminuir o erro de reprojeção entre um *keypoint* de uma imagem e a reprojeção de um ponto 3-D correspondente na imagem. De acordo com Fraundorfer e Scaramuzza [27], existem duas formas de diminuir o erro da trajetória, utilizando um método de mínimos quadrados não linear como o de Gauss-Newton ou como o de Levenberg-Marquadt. Na primeira, a técnica de *Bundle Adjustment* (BA) [28] consiste em otimizar as posições dos pontos do mapa criado e as poses da câmara, de modo a minimizar o erro de projeção respetivamente às caraterísticas extraídas. Devido ao custo computacional que é necessário despender, os métodos de odometria visual aplicam esta técnica apenas a um conjunto de *keyframes* mais recentes. A segunda técnica, *Pose-Graph Optimisation* (PGO), é uma simplificação da anterior, em que se descarta a estrutura do mapa e apenas se otimizam as poses da câmara, minimizando o erro de transformação entre poses consecutivas. Por sua vez, a minimização do erro de transformação envolve a minimização do erro de reprojeção, respetivamente às caraterísticas extraídas. No entanto, a aplicação de PGO implica uma abordagem *Graph-Based* SLAM, interpretando o problema de SLAM como um gráfico, no qual os nós são as poses e as ligações são representadas por restrições visuais. Uma descrição detalhada sobre este tema pode ser encontrada no livro de Dellaert e Kaess [29].

Uma vantagem dos métodos indiretos está relacionada com os algoritmos detetores de caraterísticas, como refere Torr e Zisserman [30]. Como os *keypoints* detetados têm invariância geométrica e fotométrica, estes métodos são mais robustos em diversos cenários.

No entanto, em ambientes que não possuam textura suficiente esta abordagem não é a mais indicada. Além disso, como apenas considera as caraterísticas extraídas, o resto da imagem é descartada, apesar de potencialmente ter informação que pode ser explorada. Isto, por consequência, significa que o mapa criado é esparso, pelo que não tem grande utilidade para além da tarefa de localização.

2.5.1.2 Métodos Diretos

A abordagem direta, tal como o nome indica, trabalha diretamente sobre a intensidade dos pixeis da imagem, o que permite eliminar a deteção e associação de caraterísticas. Um método direto pode ser denso, se usar todos os pixeis da imagem, semidenso se considerar apenas os pixeis que apresentam um elevado gradiente, ou esparso se apenas um conjunto de pixeis for considerado. Independentemente da quantidade de pixeis utilizados, é estimada a profundidade de cada *pixel* para criar um mapa.

Assim, as componentes fundamentais de um método direto de VO estão ilustradas na figura 2.5.

Figura 2.5: Estrutura de um método direto de odometria visual.

De modo a estimar o movimento, os métodos diretos calculam uma transformação que minimiza o erro fotométrico, ou seja, a diferença de intensidades dos pixeis considerados entre duas imagens.

Tal como em métodos indiretos, existe uma incerteza associada à transformação, pelo que é necessário diminuir o seu efeito. Neste caso, a principal diferença para os métodos indiretos está no erro que pretendem minimizar. Na otimização dos métodos diretos por *Bundle Adjustment*, o mapa de profundidade e das poses da câmara são otimizados de forma a diminuir o erro fotométrico. Relativamente à *Pose-Graph Optimisation*, é feita de forma semelhante aos métodos indiretos, mas a minimização do erro de transformação visa diminuir o erro das intensidades.

Assim, como não dependem da deteção de um tipo de *keypoint* ou de descritores, os métodos diretos são esperados que tenham maior precisão e robustez em ambientes com pouca textura. Além disso, relativamente à reconstrução do ambiente observado, estes métodos constroem mapas muito mais detalhados e, por isto, muito mais informativos que os mapas construídos por métodos indiretos.

Porém, como referem Aqel, Marhaban, Saripan et al. [31], esta abordagem é sensível ao efeito de *rolling shutter* e a baixas frequências de captura de imagem. Ademais, especialmente em métodos densos e semidensos, o custo computacional é maior.

2.5.1.3 Visão Monocular

A visão *stereo* permite uma maior precisão nos sistemas de odometria visual devido à capacidade de extrair a escala da imagem. No entanto, em situações em que a distância entre a câmara e o ambiente ao redor é significativamente maior que a distância entre as duas câmaras, este tipo de visão tem o seu desempenho seriamente afetado [21]. Além disso, devido a restrições de custos ou de dimensões dos veículos, a integração de uma só câmara é mais simples de ser implementada. Por estas razões, a utilização de uma só câmara tem sido um tema de estudo ao longo dos anos, em ambas as temáticas de odometria e SLAM, como é descrito na tabela 2.1.

Inicialmente, o SLAM monocular foi desenvolvido através de abordagens por filtragem [32], [33]. O MonoSLAM, proposto por Davison, Reid, Molton et al. [34] em 2007, é considerado

o método mais representativo dos métodos de vSLAM baseados em filtragem. O algoritmo utiliza um *Extended Kalman Filter* (EKF) para estimar o movimento da câmara e a estrutura 3-D de um ambiente desconhecido. Como o vetor de estado do filtro contém as caraterísticas extraídas, o algoritmo funciona apenas em ambientes restritos, para que o tamanho do vetor não se torne demasiado grande e não se torne computacionalmente pesado. Além disto, existe desperdício de processamento por adicionar ao mapa imagens que não contêm informação relevante, assim como acumulação de erros de linearização.

No mesmo ano, com o objetivo de resolver a eficiência computacional, Klein e Murray [35] apresentaram o trabalho *Parallel Tracking And Mapping* (PTAM). No PTAM a tarefa de *tracking* e de mapeamento é dividida em duas *threads* paralelas, tirando partido do facto de os processadores possuírem mais que um núcleo. No PTAM, apenas são adicionadas *frames* ao mapa se forem relevantes segundo determinados critérios de visibilidade e de caraterísticas extraídas. Deste modo, como o mapeamento do ambiente pode ser feito a uma frequência mais baixa que a de aquisição de imagens, o processamento poupado serve para otimizar o mapa e a trajetória, usando a técnica de BA [28]. Este algoritmo possui um mecanismo de relocalização, baseado na correlação entre *thumbnails* de baixa resolução das *keyframes*, pouco robusto a diferentes ângulos de visão sobre o mesmo ambiente.

Em comparação, o PTAM superou o MonoSLAM tanto em termos de precisão como de robustez, o que mudou o paradigma do vSLAM, a favor das técnicas baseadas na utilização de *keyframes* e paralelização da localização e do mapeamento. No entanto, o PTAM não é adequado para mapeamento de larga escala. Como executa um *Bundle Adjustment* sobre toda a trajetória, o custo computacional que a tarefa de otimização requer aumenta com o decorrer da sessão.

Ambos, o MonoSLAM e o PTAM, são algoritmos de vSLAM baseados em deteção de pontos de interesse, criando mapas esparsos. De modo a criar mapas com mais informação, foi proposto o *Dense Tracking and Mapping* (DTAM) [36] em 2011. O DTAM utiliza todos os pixeis da imagem para estimar o movimento da câmara, através do alinhamento direto entre as imagens, e para efetuar o mapeamento 3D do ambiente. Esta abordagem torna o algoritmo bastante mais pesado computacionalmente do que os anteriores, o que o torna dependente do uso de uma *Graphical Processing Unit* (GPU).

Com o objetivo de tornar os métodos diretos capazes de serem executados utilizando apenas a *Central Processing Unit* (CPU), foi proposto por Engel, Sturm e Cremers [37] um algoritmo de odometria visual semidenso. Ao invés do DTAM, o sistema proposto não utiliza toda a informação contida na imagem, mas apenas as regiões que apresentem um gradiente não negligenciável. Isto deve-se ao facto de ser difícil estimar a profundidade de superfícies com pouca textura. Com base neste trabalho, em 2014 Engel, Schöps e Cremers propuseram o *Large-Scale Direct Monocular SLAM* (LSD-SLAM) [24], o primeiro método direto de vSLAM capaz de ser executado em tempo real por uma CPU. Ao contrário dos seus antecessores, o LSD-SLAM permite o mapeamento em grande escala, construindo um mapa semi-denso. Também introduziu um mecanismo de deteção de um lugar, utilizando o OpenFABMAP [38] para reconhecer se nas últimas *keyframes* a *frame* atual corresponde a um sítio anteriormente

visitado. Em paralelo, e quando é detetado um *loop*, é executada a *Pose-Graph Optimisation*.

Ainda em 2014, foi proposto por Forster, Pizzoli e Scaramuzza [39] o *Semi-Direct Visual Odometry* (SVO). Este é um método híbrido que combina as vantagens dos métodos indiretos e diretos, com o objetivo de maximizar o binómio eficiência e precisão. O algoritmo extrai pontos de interesse com o detetor de cantos FAST [40], mas o *tracking* dos pontos é feito de forma direta, sem o uso de descritores. Como a extração de *features* é feita apenas em *keyframes*, torna o algoritmo mais rápido.

Em 2015 Mur-Artal, Montiel e Tardós [41] apresentaram o ORB-SLAM, um método indireto que combina as ideias principais do PTAM, o método de *loop-closure* apresentado em [42], e as noções de escalabilidade de um sistema vSLAM em [43]. O reconhecimento de um lugar anteriormente visitado é feito através de técnicas de *bag of words*, baseado no DBoW2 [44]. Através deste módulo de reconhecimento, é feita a relocalização, caso o *tracking* falhe, e caso um *loop* seja detetado, ocorre otimização do mapa e da trajetória por PGO seguida por um BA. Além disto, introduziu um processo de inicialização mais robusto baseado na seleção de um modelo mais adequado a um ambiente plano ou a um ambiente irregular. Os resultados apresentados indicam que o ORB-SLAM tem um desempenho e uma robustez bastante superiores ao PTAM e ao LSD-SLAM. Mur-Artal e Tardós [45] adicionaram o suporte a câmaras *stereo* e RGB-D ao ORB-SLAM.

Na sequência do trabalho SVO, em 2017 Forster, Zhang, Gassner et al. [46] publicaram uma nova versão [46] em que para além dos cantos FAST, extrai regiões de elevado gradiente para dotar o algoritmo de uma maior robustez em ambientes com poucos cantos. Além disso, adiciona um modelo de previsão do movimento para lidar com variações rápidas e ambientes dinâmicos. Contudo, por ser um algoritmo de odometria visual não existe *loop closure*, pelo que a sua precisão ao longo da trajetória irá, irremediavelmente, diminuir.

Motivados pelas vantagens de uma formulação direta e dispersa de um algoritmo de odometria visual, em 2017 Engel, Koltun e Cremers [47] propuseram o *Direct Sparse Odometry* (DSO). Por ser um método direto, não depende tanto do ambiente exterior e permite utilizar mais informação visual. De modo a modelar a imagem de uma forma mais precisa, o DSO incorpora uma calibração fotométrica que permite corrigir a imagem tendo em conta os tempos de exposição, a atenuação da lente e a correção de gama. A opção de abordar por uma solução que opere apenas em pixeis de elevado gradiente permite realizar uma otimização fotométrica por *bundle adjustment* nas *keyframes* mais recentes, em tempo-real e num CPU. O desempenho deste algoritmo é especialmente robusto e preciso quando é feita a calibração fotométrica. Gao, Wang, Demmel et al. [48] propuseram a integração de uma técnica *loop-closure* no DSO. Para tal, a escolha de pontos feita pelo DSO é adaptada para favorecer cantos na imagem, pois a sua repetibilidade permite a identificação de um sítio através de *bag of words*. Quando é detetado um lugar previamente visitado, é executada a *Pose-Graph Optimisation*.

Recentemente, mais concretamente em 2019, foi apresentado um novo método direto de vSLAM por Zubizarreta, Aguinaga e Montiel [49], o *Direct Sparse Mapping* (DSM). Este sistema tira partido das ideias do DSO mas ao contrário deste último que utiliza um mapa temporário, constrói um mapa global persistente que permite lidar com reobservações do

ambiente. Estas reobservações são processadas através da associação de *keyframes* por um critério de covisibilidade, ou seja, *keyframes* que partilham pontos do mapa em comum. No entanto, apesar da utilização de um mapa que permite reduzir o erro da odometria, o DSM não possui um módulo de reconhecimento, que permita a relocalização ou o *loop-closure*, o que se justifica pela dificuldade em aplicar as *features* utilizadas na odometria para efeitos de reconhecimento.

Recentemente, em 2020, Campos, Elvira, Rodríguez et al. [50] (artigo em revisão à data da escrita desta dissertação) apresentaram o ORB-SLAM3, reforçando o suporte a câmaras do tipo *fisheye* que têm um campo de visão muito maior. Também é aperfeiçoado o método de reconhecimento de um lugar de modo a tornar mais eficaz a fusão de mapas de diferentes sessões. A integração de um sistema multi-mapas também dá uma maior robustez ao algoritmo numa só sessão. Isto acontece em casos que se perca o *tracking*. Em casos de perda de *tracking*, um novo mapa é criado. Quando se relocaliza num sítio comum ao mapa que estava a ser criado antes de perder o *tracking*, é possível combinar os dois mapas. Os resultados apresentados no artigo mostram melhorias de desempenho comparado com o ORB-SLAM e outros sistemas do estado da arte.

Método	**SLAM/VO**	**Direto/Indireto**	**Estimativa da pose**	**Módulo de reconhecimento e aplicação**
Mono-SLAM	SLAM	Indireto	EKF	-
PTAM	SLAM	Indireto	BA sobre toda a trajetória	*Thumbnail* -relocalização
DTAM	SLAM	Direto	Modelo 3D	-
LSD-SLAM	SLAM	Direto	PGO	FABMAP – *loop-closure (PGO)*
SVO	VO	Híbrido	BA local	-
ORB-SLAM	SLAM	Indireto	BA local	*Bag of words* - relocalização, *loop-closure* (PGO+BA)
DSO	VO	Direto	BA local	-
LC-DSO	VO	Direto	BA local	*Bag of words* de caraterísticas ORB - *loop-closure (PGO)*
DSM	SLAM	Direto	BA local	-

Tabela 2.1: Síntese dos métodos de vSLAM e VO monocular.

2.5.2 Odometria Visual-Inercial

Os métodos de odometria baseados em visão computacional são afetados por condições externas, como a iluminação, desfoque devido ao movimento, ou mesmo a textura do ambiente observado. Por outro lado, como já foi abordado na secção 2.2, os sistemas de navegação inercial não são afetados por estas condições, apesar de haver um erro crescente com o tempo. Assim, para mitigar os eventuais decréscimos de qualidade, utiliza-se normalmente uma IMU por ser um equipamento pequeno e já acessível no mercado a baixos preços e com um bom desempenho. Os dois sensores complementam-se, com a IMU a fornecer uma boa precisão do movimento a curto prazo, ao passo que a câmara reduz o erro crescente a longo prazo.

Reunidas que estão a IMU e a câmara, importa referenciar as diferentes formas de combinar

os dados extraídos destes dois elementos. Neste sentido, deve registar-se que os sistemas de VIO podem ser divididos consoante a forma como as informações visual e inercial interagem de modo a conduzir a um resultado final. Para o efeito, deve apontar-se as diferentes formas de estimar a localização a partir dos dados, nomeadamente a filtragem e a otimização.

Assim, a abordagem por filtragem pode ser interpretada como um método *Maximum A Posteriori* (MAP), em que as medidas obtidas através da IMU são utilizadas para calcular a probabilidade *a priori* da posição, o que corresponde ao passo de previsão num EKF. Relativamente à informação visual, esta é usada para calcular a função de verosimilhança, como referem Gui, Gu, Wang et al. [51], o que corresponde ao passo de atualização do filtro.

Apesar de as abordagens mais clássicas utilizarem, normalmente, um EKF, este não é escalável. Ou seja, o número de caraterísticas extraídas não pode ser muito elevado, sob pena de a computação se tornar demasiado pesada e não permitir a operação em tempo real. Por conseguinte, como referem Strasdat, Montiel e Davison [52], esta desvantagem afeta a precisão do filtro, por ser imposto um limite ao número de caraterísticas, fazendo-se valer de medidas heurísticas para selecionar as *landmarks* que cumprem determinados critérios. Ainda assim, mesmo que a complexidade computacional não seja um problema, existem duas principais fontes de erro na utilização de um filtro, como indicam Scaramuzza e Zhang [53]. Em primeiro lugar, como o estado atual depende da informação obtida através dos estados anteriores, e estes últimos são descartados permanentemente, erros de linearização ou a presença de *outliers* vão permanecer no filtro. Em segundo lugar, o erro de linearização torna o filtro inconsistente.

Por outro lado, existem as técnicas de otimização que, de acordo com Mohamed, Haghbayan, Westerlund et al. [54], resolvem um problema de otimização não-linear, minimizando uma função de custo baseada nos erros de reprojeção entre pontos da imagem projetados no espaço 3-D e os erros das medidas inerciais.

Como sugerem Gui, Gu, Wang et al. [51] , esta abordagem é um método de *Maximum Likelihood* (ML), em que o estado para o qual a probabilidade das medidas é maior é calculado de forma iterativa, o que permite refinar o ponto de linearização e diminuir o erro. Assim, uma possibilidade é otimizar todos os estados, o que garante a maior precisão. Cada estado é tipicamente constituído pela pose do sensor da IMU associada a uma determinada imagem, a respetiva velocidade e valores de ruído. Porém, como a trajetória aumenta com o tempo, a otimização de todos os estados não é exequível para operações em tempo real. Uma outra possibilidade é adotar o conceito de *keyframes*, partindo do princípio de que nem todas as *frames* são relevantes para a otimização da trajetória, havendo desta maneira um balanceamento entre o custo computacional e a precisão. No entanto, como indicam Scaramuzza e Zhang [53], por serem eliminados estados, de forma semelhante à filtragem que elimina os estados antigos, os métodos que utilizam uma janela sofrem de inconsistência e erros de linearização.

Não obstante a presença de algumas fragilidades, comparada com as técnicas de filtragem, a otimização não-linear normalmente garante melhores resultados devido ao facto de reduzir os erros de linearização em estados antigos.

Além destas duas categorias, os sistemas de VIO distinguem-se pela forma como os dados visuais e inerciais contribuem para a estimativa da localização, podendo resultar de dois

procedimentos distintos: loosely-coupled e tightly-coupled.

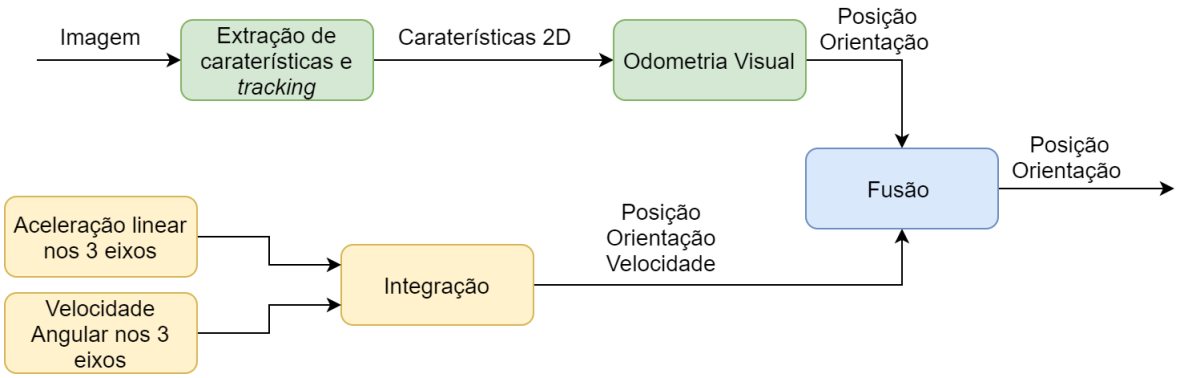


Figura 2.6: Fusão de dados *loosely-coupled*.

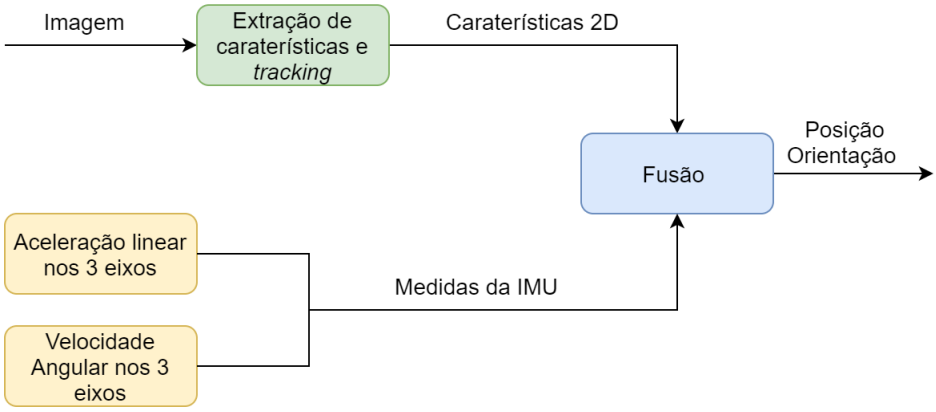


Figura 2.7: Fusão de dados *tightly-coupled*.

De acordo com [54] e como é ilustrado na figura 2.6, nas abordagens *loosely-coupled* as medidas visuais e inerciais são processadas de forma separada, como se de dois módulos independentes se tratassem, resultando em duas estimativas do movimento diferentes. Os resultados são depois combinados de forma a fornecer a posição e orientação. Nas abordagens *tightly-coupled*, por sua vez e como apresentado na figura 2.7, a posição e a orientação são resultado da fusão dos dados visuais (caraterísticas extraídas) e das medidas da IMU a partir de uma única equação, resultando numa só estimativa final.

De acordo com Scaramuzza e Zhang [53], comparativamente, pode afirmar-se que as abordagens *tightly-coupled* são mais precisas relativamente às *loosely-coupled* por duas razões. A primeira, o facto de se utilizar a integração das medidas inerciais para prever a localização das caraterísticas extraídas na imagem seguinte, facilita o processo de *tracking*. A segunda, pelo facto de as abordagens *loosely-coupled* não considerarem a relação que existe entre a informação visual e inercial, o que impede de limitar o *drift* no módulo de visão.

2.5.2.1 Visão Monocular

A combinação de sensores visuais e inerciais permite tornar os métodos de VO/vSLAM mais robustos a condições externas mas, principalmente, possibilita a estimação da escala

em sistemas de visão monoculares, a qual é o maior problema deste tipo de visão. Por estes motivos, a utilização de uma IMU para auxiliar este tipo de sistemas tem sido bastante estudada nos últimos anos, como é descrito na tabela 2.2.

Tal como nos métodos de visão, os métodos de visão-inercial foram inicialmente resolvidos por filtragem. O *Multi-State Constraint Kalman Filter* (MSCKF), apresentado por Mourikis e Roumeliotis [55], foi uma das principais abordagens. No trabalho foi proposto alterar o modelo do filtro convencional do EKF, de modo a expressar as restrições geométricas que acontecem com reobservações do ambiente. Para tal, os autores propõem retirar do vetor de estado do EKF as posições no espaço 3D das caraterísticas extraídas. Com esta proposta, também reduzem a complexidade quadrática relativamente ao número de caraterísticas para uma complexidade linear, o que diminui também o custo computacional que uma solução baseada num EKF normalmente implica quando o número de pontos 3-D aumenta.

Contudo, como indicam Li e Mourikis [56], o MSCKF tem problemas de inconsistência que se devem a problemas de observabilidade no modelo linearizado do sistema usado, no qual a rotação sobre o vetor da gravidade (*yaw*) aparenta ser observável, resultando na sub-estimativa da incerteza do *yaw* e por consequência na sub-estimativa de todos os estados. Através das medidas do acelerómetro apenas é possível calcular os ângulos em *pitch* e *roll*. O eixo do ângulo *yaw* está alinhado sobre o eixo do vetor da gravidade, pelo que uma rotação em torno deste eixo não alteraria os valores medidos pelo sensor, o que torna o ângulo *yaw* não observável. Outra fragilidade deve-se à precisão finita dos parâmetros de calibração extrínsecos entre a câmara e a IMU. Como a incerteza associada aos parâmetros de calibração não é modelada no filtro, a covariância dos estados é sub-estimada. De modo a resolver estas inconsistências, Li e Mourikis [56] publicaram o MSCFK 2.0 [56].

Com base em trabalhos que indicam a superioridade da otimização não-linear sobre *keyframes* relativamente à filtragem, e a superioridade da fusão de dados visuais e inerciais por uma abordagem *tightly-coupled* sobre *loosely-coupled*, Leutenegger, Lynen, Bosse et al. [57], [58] apresentaram o OKVIS. Em vez de otimizar sobre uma janela temporal constituída por *frames* consecutivas, são propostos diferentes critérios que visam eliminar estados antigos e minimizar a perda da informação visual associada Apesar de não ser o ideal de informação, isto permite limitar o custo computacional da otimização e manter a operação exequível em tempo real. Os erros de reprojeção e inerciais são combinados numa função de custo, a qual a otimização não linear pretende minimizar.

Bloesch, Omari, Hutter et al. propuseram, em 2015, o ROVIO [59], um algoritmo de odometria visual-inercial monocular baseado num EKF. Nele, os autores introduzem uma formulação robocêntrica, em que a localização das *landmarks* são estimadas relativamente à posição da câmara, o que permite saltar o processo de inicialização do algoritmo, comum em sistemas de visão monocular, e diminuir o erro de linearização, típico em abordagens por filtragem. É proposto um *tracking* direto, no qual é extraída a intensidade das regiões à volta de cantos FAST. O erro fotométrico é incluído na atualização do filtro, como termo de inovação, o que permite o *feedback* fotométrico. Em [60], o ROVIO é adaptado para um *Iterated Extended Kalman Filter* (IEKF).

No início do ano de 2017, Mur-Artal e Tardós [61] adaptaram ORB-SLAM, para uma versão visual-inercial [61]. Por ser um sistema de vSLAM, ao contrário dos outros sistemas de VIO apresentados, esta versão do ORB-SLAM tem um mecanismo de *loop-closure* e utiliza o mapa para reduzir o erro do módulo de odometria. Porém, como é referido pelos autores, o principal problema do sistema consiste na inicialização dos dados inerciais, que depende da inicialização do sistema de visão monocular. O método de inicialização era lento, o que afetava a robustez e precisão do sistema. No ORB-SLAM3 [50], o método de inicialização foi melhorado.

Qin, Li e Shen apresentaram, em 2018, um novo sistema de odometria visual-inercial, o VINS-Mono [62]. Estes autores propõem um novo método de inicialização da IMU, adotando uma abordagem *loosely-coupled*, baseando no facto de que a câmara monocular permite, por si só, uma inicialização ótima. Após a inicialização do módulo de visão, os dados pré-integrados da IMU são alinhados com os dados visuais, permitindo recuperar a escala, a gravidade, velocidade e os erros de *offset*. O módulo de odometria visual-inercial funciona por uma abordagem *tightly-coupled*, e por otimização não linear sobre uma janela de *keyframes*, de forma semelhante ao que foi proposto no OKVIS. Apesar de ser um algoritmo de VIO, tem capacidade de relocalização e de executar *Pose-Graph Optimisation* em quatro graus de liberdade (x, y, z, *yaw*) após deteção de um *loop*, usando DBoW2 [44].

Von Stumberg, Usenko e Cremers [63], por seu turno, publicaram um novo sistema de odometria visual-inercial, o VI-DSO, uma extensão do DSO. Neste algoritmo é proposto um novo modo de inicialização em que são incluídos o vetor da gravidade, a escala, os *offsets* e as velocidades no modelo a ser otimizado. Isto permite que estes valores possam ser inicializados com um valor arbitrário e que sejam otimizados por *bundle adjustment*. Na função a ser minimizada, para além do erro fotométrico, é incluído o erro inercial, pelo que é uma abordagem *tightly-coupled*.

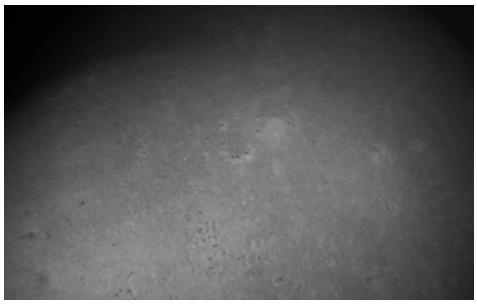
Método	SLAM/VIO	Direto/Indireto	Estimativa da pose	Fusão de dados visuais e inerciais	Módulo de reconhecimento e aplicação
MSCKF	VIO	Indireto	EKF	TC	-
OKVIS	VIO	Indireto	BA local	TC	-
ROVIO	VIO	Direto	EKF	TC	-
VI-ORBSLAM	SLAM	Indireto	BA local	TC	*Bag of words* - relocalização, *loop-closure* (PGO+BA)
VINS-Mono	VIO	Indireto	BA local	TC (LC na inicialização)	*Bag of words* - relocalização, *loop-closure* (PGO 4 DoF)
VI-DSO	VIO	Direto	BA local	TC	
ORB-SLAM3	SLAM	Indireto	BA local	TC	*Bag of words* - relocalização, *loop-closure* (PGO+BA)

Tabela 2.2: Síntese dos métodos de vSLAM visual-inercial e VIO monoculares. A nomenclatura «TC» refere-se a *tightly-coupled* e «LC» a *loosely-coupled*.

2.6 Navegação ótica no meio subaquático

A navegação ótica em meio subaquático é ainda um campo pouco explorado, pelo que não foi produzida ainda uma extensa literatura sobre a utilização de sensores de forma a que a câmara seja o módulo principal de localização e, em simultâneo, se revele financeiramente mais favorável.

(a) Imagem captada pelo ROV Pro 4 da Videoray no fundo de uma piscina, sendo percetível o efeito de perda de visibilidade por efeito de *backscattering*.



(b) As zonas arenosas são parcas em caraterísticas que se possa extrair. Fonte: [65]

Figura 2.8: Ilustração das dificuldades de extrair informação relevante no meio subaquático.

No contexto subaquático, o SLAM divide-se entre as abordagens com recurso a um sonar, ou a partir de câmaras. Esta última alternativa tem ganho um interesse crescente nos últimos anos devido às soluções que são apresentadas para os meios aéreo e terrestre. As imagens fornecidas pelo sonar não são tão detalhadas e precisas como as imagens obtidas por uma câmara, que é essencial nos cenários em que o ROV opera. Especialmente em ambientes que seja necessário o veículo estar perto de uma estrutura ou do relevo oceânico, as vantagens de usar visão são amplificadas.

Contudo, devido às particularidades do meio em estudo, a localização por visão é mais complexa do que nos meios aéreo e terrestre, como ilustrado na figura 2.8. A absorção da luz por parte da água reduz fortemente a perceção visual, o que obriga incluir nos veículos uma luz artifical. No entanto, a luz artificial também pode trazer uma desvantagem, visto que a fauna subaquática é atraída pela mesma, criando um ambiente dinâmico e oclusões na câmara. Também as partículas que flutuam na água refletem a luz, o que resulta numa imagem turva, reduzindo a visibilidade. Além disto, o ambiente em questão pode revelar cenas com pouca textura, ou com padrões repetitivos, como nos corais. Devido a estas condições adversas, os algoritmos do estado da arte de VO/vSLAM revelam dificuldades, como se conclui pelo estudo de Joshi, Rahman, Kalaitzakis et al. [64].

Uma abordagem para estimar o movimento de um veículo subaquático com uma câmara monocular baseia-se na criação de mosaicos, como são os estudos realizados no início do século de Garcia, Cufi e Carreras [66] e Gracias e Santos-Victor [67]. Partindo da sobreposição existente entre imagens consecutivas, as imagens adquiridas vão sendo combinadas de modo a formar um mapa 2D da cena.

Numa das primeiras aplicações da abordagem num UUV, Gracias, van der Zwaan, Bernardino et al.[68] propuseram um método de localização e mapeamento visuais baseado em mosaicos para um AUV. O mapa dos mosaicos é criado *offline* e depois usado para a navegação em tempo real. De forma a controlar o erro do módulo de odometria visual, a imagem é aproximada a um mosaico que, por sua vez, é comparado com o mapa previamente obtido.

Por decomposição de uma homografia, a posição é corrigida. Para o efeito, é assumido que o relevo oceânico pode ser aproximado a um plano e que a câmara está num plano paralelo ao ambiente observado.

A condição de que se pode aproximar um ambiente 3D para um 2D é a maior limitação da localização e mapeamento por mosaicos. A maior parte das regiões ou objetos de interesse não são planos, e em casos em que a câmara se encontra a curta distância de uma estrutura, como num cenário de inspeção, este princípio não funciona. Além disto, o movimento do veículo está restringido a um movimento em duas dimensões.

Outra abordagem consiste em utilizar *landmarks* conhecidas, naturais ou artificiais, para localizar o veículo num ambiente estruturado. Nesta perspetiva, Romagós, Ridao, Carreras et al. [69] utilizaram um padrão binário (preto e cinzento) no fundo de uma piscina, que codificava 32 posições. Palomeras, Nagappa, Ribas et al. [70], por seu lado, propõem um *template matching* com caraterísticas já conhecidas no ambiente e utilizaram um *software* externo para identificar, detetar e rastrear marcas artificiais, colocadas em posições conhecidas. Jung, Li, Choi et al. [71] utilizam marcas artificiais com um padrão, que fornece um ID único. Estas abordagens, todavia, implicam um conhecimento da área prévio à missão, pelo que limitadas ao ambiente em que são aplicadas.

No seguimento do estudo nesta temática, Eustice, Pizarro e Singh [72] apresentaram, em 2008, uma proposta de integração da visão monocular num veículo subaquático. Devido às caraterística das câmara do AUV, a frequência de aquisição de imagem era baixa, resultando em imagens que exibiam pouca sobreposição entre elas. Com o objetivo de colmatar este problema, foi proposto calcular a matriz essential, que descreve o movimento da câmara com base na associação das caraterísticas em coordenadas da câmara (2D-2D). Como não é possível obter a escala por este método, a mesma é calculada a partir das variações de pressão e da integração das medidas de velocidade de um DVL. Para processar a informação dos sensores de *Dead Reckoning* e a informação visual, é utilizado um *Extended Information Filter* (EIF).

Posteriormente, Warren, Corke, Pizarro et al. [73] implementaram um método de odometria visual *stereo* para um AUV, assumindo que o veículo possuía apenas 4 graus de liberdade (x, y, z, *yaw*). Como a frequência de aquisição de imagens era baixa, possuindo pouca sobreposição entre imagens consecutivas, e o AUV tem um movimento negligenciável em *pitch* e *roll*, é proposta a utilização de um magnetómetro para fornecer a orientação global do veículo. A trajetória é otimizada por *bundle adjustment* [28], introduzindo um termo rotacional à função a ser minimizada, para além do erro de reprojeção.

Kim e Eustice [74] aprofundaram o trabalho de Eustice, Pizarro e Singh [72], introduzindo uma formulação baseada no *Graph*-SLAM [75], em que as posições antigas são guardadas no vetor de estado do filtro. No seguimento do trabalho de Eustice, Pizarro e Singh [72] e Kim e Eustice [74], Kim e Eustice [76] refinaram o módulo de localização da câmara, melhorando a associação de dados. Para tal utilizaram *bag of words*, com o vocabulário a ser criado durante a missão de modo a estar adaptado às imagens que iam sendo registadas. A métrica utilizada para criar o vocabulário foi segundo a saliência visual, o que permite definir um critério para adicionar uma *keyframe*. Assim, uma imagem com elevada saliência permite dar um peso

maior às estimativas da câmara, pois a probabilidade de estar certa é maior. Além disto, o sistema possui um mecanismo de deteção de *loop-closure*.

Em 2015, Carrasco, Bonin-Font, Campos et al. [77] propôs uma abordagem por uma otimização não-linear sobre um conjunto de posições e *landmarks* ligadas entre si, numa formulação que se baseia no *Graph-SLAM*, apresentada por Thrun e Montemerlo [75]. A cada 0.5m eram capturadas imagens *stereo*, as quais eram associadas ao resultado da odometria, adicionando um novo nó ao gráfico. Para estimar o resultado da posição do robô, é utilizado um EKF para fusão de dados obtidos por um *Global Positioning System* (GPS) (quando o veículo está à superfície), um DVL, uma IMU e um sensor de pressão.

Contudo, os sistemas baseados no trabalho de Eustice, Pizarro e Singh [72] utilizam uma baixa frequência de captura de imagens (1/2 Hz), e consequentemente, os sistemas de visão servem como complemento para reduzir o *drift* da trajetória estimada por *Dead Reckoning*. No trabalho de Warren, Corke, Pizarro et al. [73], para além da baixa frequência de imagem, os valores do magnetómetro podem ser seriamente afetados em zonas com fortes anomalias magnéticas e assim perturbar a trajetória estimada. Além disto, assume um movimento com apenas 4 graus de liberdade. No sistema proposto por Carrasco, Bonin-Font, Campos et al. [77], as imagens são capturadas apenas para a tarefa de otimização, dependendo de outros sensores para realizar a odometria.

Na sequência do desenvolvimento tecnológico, as câmaras passaram a ter uma capacidade de aquisição de imagem que permite estimar o movimento de um agente dando origem a vários trabalhos que visam discutir a melhor forma de explorar a informação visual numa imagem capturada debaixo de água.

Deste modo, a deteção, descrição e associação das *features* é bastante afetada pelas condições do meio, como estudaram Oliver, Hou e Wang [78].

Aulinas, Carreras, Llado et al. [79], por seu turno, avaliaram o desempenho de caraterísticas SURF [80], de modo a limitar o erro da trajetória estimada por técnicas de DR. A imagem é pré-processada de modo a eliminar o efeito de *flickering* do sol em águas superficiais. Em vez de considerar toda a área da imagem, a imagem é segmentada de modo a identificar regiões de interesse, sobre as quais são detetadas caraterísticas SURF. No caso de não haver uma região de interesse, é considerada toda a imagem como área de extração. Apesar das melhorias dos resultados, este método não é adequado para operações em tempo real.

Numa outra perspetiva, Garcia e Gracias [81] comparam diversos detetores de modo a testar a sua robustez em imagens com diferentes graus de turbidez, segundo uma métrica em que um ponto detetado numa imagem turva, pode ter um desvio máximo de 0.5 pixeis em relação à imagem original. Os resultados indicam que os detetores de *blobs*, como o Hessiano, têm um melhor desempenho do que os deterores de cantos e de *edges*. No entanto, é apenas examinada a repetibilidade dos detetores, o que não é suficiente para avaliar se o *matching* entre os pontos detetados pode ser feito sem ambiguidades.

Também Shkurti, Rekleitis e Dudek [82] testaram diversas combinações de detetores e descritores, de modo a testar a robustez dos mesmos a falsos positivos. Os resultados indicam que *keypoints* SURF e Shi-Tomasi [83] associados através de ZNCC (*Zero-Normalized*

Cross-Correlation) são os mais robustos a *outliers*. No entanto, as combinações efetuadas não incluem o uso de descritores binários como ORB [84], BRISK [85] ou BRIEF [86] (à data do artigo estes não tinham sido apresentados) e os métodos de fluxo ótico, que são usados em diferentes aplicações.

Hidalgo e Braunl [87], por sua vez, avaliaram o desempenho de diferentes detetores e descritores em diversas condições, como áreas arenosas e outras com a presença de rochas e algas, turbidez ou iluminação, assim como a robustez dos mesmos a falsos positivos. De acordo com os resultados, verifica-se que SIFT [88], o ORB e o BRISK destacam-se ligeiramente de SURF e AKAZE [89] quando o número máximo de caraterísticas a serem detetadas aumenta. O AKAZE, porém, é o que apresenta um rácio melhor entre o número de *inliers* e número de caraterísticas detetadas, indicando que o seu detetor é restrito e que o descritor é robusto. Além disto, verifica-se que padrões luminosos devido à reflexão da água e *backscatter* afetam seriamente o número de associações corretas, mas não o número de caraterísticas detetadas. Por outro lado, os ambientes arenosos e a turbidez diminuem o número de caraterísticas detetadas, enquanto que a presença de rochas e algas beneficia o desempenho dos detetores e descritores. No geral, os detetores de caraterísticas ORB e BRISK são os que apresentam melhor desempenho, com o ORB a ter o menor tempo de processamento.

Num outro estudo, Ferrera, Moras, Trouvé et al. [90] fizeram uma avaliação extensiva sobre o desempenho de diferentes combinações entre detetores e descritores. Além destes, ainda avaliaram o KLT [91], um método que deteta cantos *Harris* pelo algoritmo de Shi-Tomasi [83] e faz o *tracking* destes cantos pelo método de fluxo ótica implementado por Lucas-Kanade Lucas e Kanade. Os testes são efetuados sobre um conjunto de imagens, em que a turbidez vai aumentando e em imagens capturadas num cenário real nas quais é evidente a pouca textura e os padrões repetitivos. Os resultados sobre estes dois conjuntos de imagens indicam que o desempenho do KLT é superior às outras combinações avaliadas.

Outra contribuição para a temática em questão foi fornecida por Shkurti, Rekleitis, Scaccia et al. [92] e Burguera, Bonin-Font e Oliver [93] com o desenvolvimento de um método de localizado baseado em visão monocular. Em ambos os estudos, é utilizado um EKF para fusão de dados visuais e inerciais, juntamente com os dados lidos por um sensor de pressão. Assim, o método proposto por Shkurti, Rekleitis, Scaccia et al. [92] deteta *keypoints* SURF que são associados por *Approximate Nearest Neighbour* (ANN) nas imagens consecutivas. O sensor de pressão tem como objetivo diminuir o erro nas estimativas da posição na componente vertical. No entanto, o método não funciona em tempo real. No trabalho Burguera, Bonin-Font e Oliver [93], é utilizado um altímetro para medir a distância ao fundo, permitindo reduzir a complexidade do problema. A imagem é pré-processada, de modo a realçar caraterísticas observadas, e nela são detetados pontos SIFT, o que representa a maior parte do processamento requerido juntamente com o processo remoção de *outliers*. De modo a reduzir os erros de linearização do EKF é guardado no vetor de estado a posição relativa entre duas posições consecutivas, o que permite criar uma dependência entre os estados e diminuir a covariância. Utiliza uma abordagem geométrica para deteção de um *loop* e baseada na sobreposição entre o campo de visão da câmara de duas imagens. Porém, o método assume que o fundo do local

seja plano e que a câmara esteja virada para baixo.

Uma outra proposta surgiu de Ferrera, Moras, Trouvé et al. [90] sob a forma de um sistema de odometria visual baseado em visão monocular. Fundamentando-se na avaliação feita para aferir a melhor forma de fazer o *tracking* das caraterísticas, o método utiliza o KLT. De maneira a lidar com oclusões que podem ocorrer num ambiente dinâmico, é proposto manter uma janela das imagens mais recentes com a lista das *features* que se perderam na aplicação do fluxo ótico. Assim, a cada iteração pretende-se voltar a fazer o *tracking* das caraterísticas perdidas. O método otimiza a trajetória e os pontos do mapa sobre a janela de *keyframes* mais recentes, por *bundle adjustment*. Os resultados indicam que o sistema é mais robusto do que outros algoritmos do estado da arte neste tipo de ambientes, e mostram a vantagem da janela de *retracking*.

Perante o exposto, pode concluir-se que a investigação no âmbito da navegação ótica em meio aquático tem vindo a desenvolver-se com maior acuidade a partir dos primeiros anos deste século, de forma gradual e sustentada. No entanto, ainda existe uma necessidade de desenvolvimento, nomeadamente a nível de propostas para a navegação ótica adaptadas para o meio em estudo, assim como soluções de processamento de imagem com o objetivo de serem usadas para a localização. O trabalho apresentado nesta dissertação pretende ser um pequeno contributo para o avanço do conhecimento nestas áreas.

CAPÍTULO 3

Conceitos teóricos

3.1 Câmara *pinhole*

Como estudado por Hartley e Zisserman [94] e como ilustrado na figura 3.1, o modelo mais básico de uma câmara *pinhole* assume que o sistema de coordenadas do mundo *World Frame* ($\mathcal{W}$) tem origem no centro de projeção da câmara, por onde passam todos os feixes óticos. Um qualquer feixe de luz com origem num ponto 3-D que passa centro de projeção, interseta o plano da imagem, ou plano de projeção.

Assim, o ponto de interseção entre o plano da imagem e o feixe de luz é a projeção do ponto 3-D na imagem. O plano da imagem permite definir o sistema de coordenadas da câmara, representado em píxeis. A origem deste sistema de coordenadas terá origem no ponto principal da imagem, o qual é o ponto de interseção da linha perpendicular ao plano de projeção, denominada eixo ótico, com origem no centro de projeção da câmara. A distância entre o ponto principal e o centro de projeção é dado pela distância focal, normalmente definida em milímetros. No sistema de coordenadas da figura mencionada, o eixo Z tem a mesma direção que o eixo ótico.

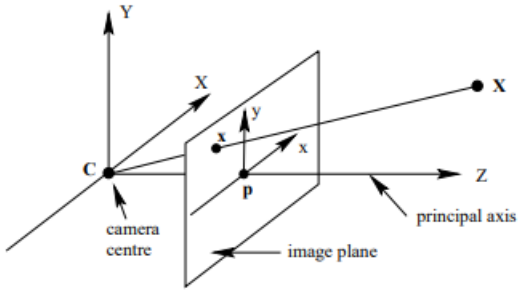


Figura 3.1: Modelo de uma câmara *pinhole* com uma projeção perspetiva. Fonte: [94]

Considere-se o ponto $X_{\mathcal{W}} = [X, Y, Z]^T$ com coordenadas referentes ao sistema de coordenadas do mundo, $\mathcal{W}$. O ponto $x = [u, v]^T$ é a projeção do ponto $X_{\mathcal{W}}$ no plano da imagem, considerando uma projeção central pela origem de coordenadas. Assumindo que o plano da

imagem a uma distância $z = f$ do centro de projeção, por semelhança de triângulos verifica-se que x tem coordenadas $[f\frac{X}{Z}, f\frac{Y}{Z}, f]^T$, referentes ao sistema de coordenadas da câmara. Como qualquer ponto projetado na imagem terá a mesma terceira coordenada, então conclui-se que:

$$X_{\mathcal{W}} = [X, Y, Z]^T \longmapsto x = [u, v]^T = [f\frac{X}{Z}, f\frac{Y}{Z}] \tag{3.1}$$

No entanto, cada câmara tem propriedades geométricas dependentes do equipamento utilizado e que são descritas pelos parâmetros intrínsecos da câmara. Estas propriedades influenciam a projeção do ponto $X_{\mathcal{W}}$ na imagem e devem ser tidos em conta.

A origem do sistema de coordenadas da câmara nem sempre é o ponto principal, pelo que é necessário ter em conta esse *offset*. Outro fator a ter em conta, é o de que os eixos u e v não têm a mesma escala. As escalas axiais são dependentes de m_u e m_v, que indicam o número de píxeis por unidade de distância (píxeis/mm). Finalmente, os píxeis não são exatamente quadrados, ou seja, os eixos u e v não são ortogonais. Apesar de na maioria das câmaras o ângulo entre os eixos ser bastante próximo de 90º, é um parâmetro que pode ser tido em conta.

Deste modo, assumindo que não existe rotação ou translação da câmara, é possível definir a projeção de um ponto $X_{\mathcal{W}} = [X, Y, Z]^T$ no sistema de coordenadas $\mathcal{W}$, no plano da imagem em $x = w[u, v, 1]^T$, através de uma matriz K que define os parâmetros intrínsecos da câmara:

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KX_{\mathcal{W}} = \begin{bmatrix} f_u & s & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.2}$$

- w: fator de escala
- $(u, v, 1)^T$: coordenadas homogéneas do ponto **x** no sistema de coordenadas da câmara, correspondentes à projeção do ponto $X_{\mathcal{W}}$ na imagem;
- K: matriz 3x3 dos parâmetros intrínsecos da câmara;
- s: parâmetro de distorção dos eixos u e v. Este é igual a zero quando os eixos são ortogonais;
- (f_u,f_v): distâncias focais em píxeis nas direções u e v, respetivamente. Estes parâmetros são dados pelas expressões $f_u = f.m_u$ e $f_v = f.m_v$.
- (c_u,c_v): coordenadas do ponto principal da projeção da imagem, definidas em *pixels*. Representam o *offset* que existe em relação ao ponto principal.

Pela multiplicação de matrizes, verifica-se que para este caso, as coordenadas do ponto na imagem são dadas por:

$$\begin{cases} wu = f_uX + sY + c_uZ \\ wv = f_vY + c_vZ \\ w = Z \end{cases} \Leftrightarrow \begin{cases} u = f_u\frac{X+sY}{Z} + c_u \\ v = f_v\frac{Y}{Z} + c_v \end{cases} \tag{3.3}$$

Note-se que é possível derivar a equação definida anteriormente em (3.1), a partir de (3.3), assumindo que os eixos do sistema de coordenadas da câmara são ortogonais, que o número de píxeis por unidade de distância é igual nos dois eixos e que a origem do sistema de coordenadas da câmara coincide com o ponto principal. Isto equivale a colocar $s = 0$, $m_u = m_v$ e $[c_x, c_y]^T = [0, 0]^T$

Partindo de (3.3), pode definir-se então uma função de projeção para uma câmara *pinhole* sem distorção, expressa por $\pi_{pin} : (\mathbb{R}^3 \setminus z = 0) \mapsto \mathbb{R}^2$:

$$p = \pi_{pin}(X_{\mathcal{W}}) = \begin{bmatrix} f_u(\frac{X}{Z} + sY) \\ f_u v \frac{Y}{Z} \end{bmatrix} + \begin{bmatrix} c_u \\ c_v \end{bmatrix}, \; p = [u, v]^T, \; X_{\mathcal{W}} = [X, Y, Z]^T \tag{3.4}$$

Até agora assumiu-se que a origem do sistema de coordenadas do mundo coincidia com o centro de projeção da câmara. Porém, a câmara movendo-se, o seu sistema de coordenadas também se move. Assim, é necessário definir os parâmetros extrínsecos da câmara, que transformam os pontos 3-D no sistema de coordenadas do mundo no sistema de coordenadas da câmara. A partir de (3.2), temos:

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = PX_{\mathcal{W}} = K \begin{bmatrix} R_{3x3} & t_{3x1} \end{bmatrix} X_{\mathcal{W}} \tag{3.5}$$

onde P é uma matriz 3x4, $R \in \mathcal{SO}(3)$ e $t \in \mathbb{R}^3$ representam a rotação e a translação de um sistema de coordenadas Euclidiano fixo para o sistema de coordenadas da câmara.

Por sua vez, também se pode modificar (3.4) a partir de R e t, o que permite obter uma função de projeção representada por:

$$p = \pi_{pin}(R.X_{\mathcal{W}} + t) \tag{3.6}$$

O modelo ideal de uma câmara *pinhole* não considera o uso de uma lente ótica, a qual modifica a alteração da direção dos feixes luminosos. A presença de uma lente causa uma distorção, que é aparente na imagem resultante ao apresentar linhas curvas que, idealmente, deveriam ser retas. Especialmente em câmaras do tipo *fisheye*, a mesma apresentam um elevado campo de visão, pelo que apresentam uma elevada distorção de forma a representar o ambiente observado no plano da imagem.

Entre diversas formas de representação dos modelos de distorção de uma câmara *fisheye*, o modelo que é apresentado nesta secção refere-se à proposta de Kannala e Brandt [95]. Neste trabalho, os autores apresentam um modelo no qual a distância entre o ponto principal da imagem e o ponto projetado por uma câmara *pinhole* é proporcional a uma função polinomial dependente do ângulo entre o mesmo ponto e o eixo principal, como ilustrado na figura 3.2. Note-se que apenas é apresentado o modelo mais reduzido, visto que os autores também apresentam um modelo com 23 parâmetros, onde a função polinomial não é só dependente do ângulo em que o feixe de luz atravessa o eixo principal.

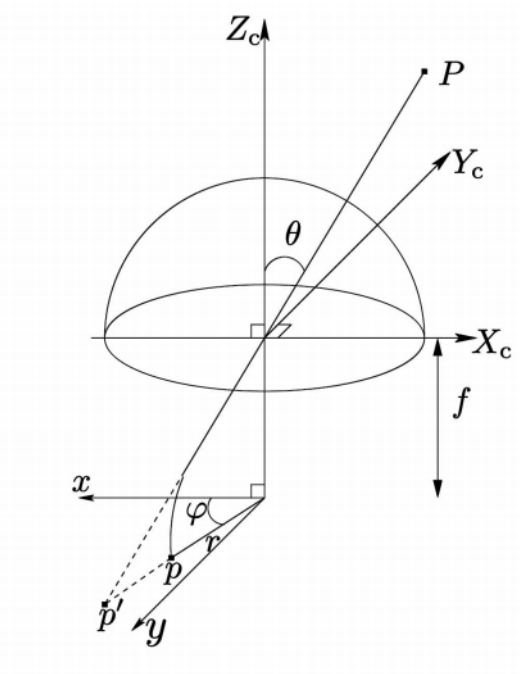


Figura 3.2: Modelo de uma lente *fisheye*. O ponto **p'** corresponde a uma projeção perspetiva de uma câmara *pinhole* do ponto *P*. O ponto *p* corresponde à projeção real numa câmara *fisheye*. A distância entre **p** e o ponto principal é dada por uma função polinomial r(θ). Fonte: [95]

Desta forma, considere-se o modelo de uma câmara *pinhole*, de acordo com (3.1). Um ponto $X_{\mathcal{W}} = [X, Y, Z]^T$ é projetado no plano de projeção $z = 1$, ou seja, no qual a distância focal, **f**, é igual a 1 milímetro. Obtém-se então a projeção do ponto $X_{\mathcal{W}}$ num ponto $x_u = [a, b, 1]^T$ (subscrito **u** refere um ponto sem distorção):

$$a = \frac{X}{Z}, \;\; b = \frac{Y}{Z} \tag{3.7}$$

A projeção perspetiva de uma câmara *pinhole* é descrita pela seguinte fórmula:

$$d = f \tan(\theta) \tag{3.8}$$

em que d é a distância entre o ponto principal da imagem e o ponto projetado, f é a distância focal e θ é o ângulo de incidência do feixe de luz.

Sabendo as coordenadas do ponto x_u e $f = 1$:

$$\begin{cases} d^2 = a^2 + b^2 \\ d = f \tan(\theta) \end{cases} \Leftrightarrow \begin{cases} d = \sqrt{a^2 + b^2} \\ \theta = \arctan(d) \end{cases} \tag{3.9}$$

De acordo com Kannala e Brandt [95], a distância entre o centro de projeção da imagem e o ponto projetado pode ser aproximado por uma função polinomial de grau nove:

$$r(\theta) = \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9 = \theta(k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \tag{3.10}$$

A distância $r(\theta)$ é proporcional à distância entre o ponto principal da imagem e X_u, o que permite definir as coordenadas de um ponto x_d (subscrito d para indicar um ponto após distorção) através de uma função de distorção D:

$$x_d = D(x_u) = \frac{r(\theta)}{d} X_u = \begin{pmatrix} \frac{r(\theta)}{d} a \\ \frac{r(\theta)}{d} b \\ 1 \end{pmatrix} \tag{3.11}$$

Assim, é possível definir uma função de projeção para uma câmara *fisheye*, expressa por $\pi_{fish} : (\mathbb{R}^3 \setminus [0,0,0]^T) \mapsto \mathbb{R}^2$:

$$p = \pi_{fish}(X_{\mathcal{W}}) = \begin{bmatrix} f_u(\frac{r(\theta)}{r_u} X + sY) \\ f_v \frac{r(\theta)}{r_u} Y \end{bmatrix} + \begin{bmatrix} c_u \\ c_v \end{bmatrix}, \; p = [u,v]^T, \; X_{\mathcal{W}} = [X,Y,Z]^T \tag{3.12}$$

onde $r_u = \sqrt{X^2 + Y^2}$ e se aplicam as expressões definidas em (3.7).

3.2 Transformações de corpo rígido

Como referido na equação (2.3), a transformação entre dois sistemas de coordenadas diferentes pode ser representada por uma matriz de transformação de dimensão 4x4, constituída por uma matriz de rotação R e um vetor de translação t. Esta matriz pertence ao grupo Especial Euclidiano $\mathbb{SE}(3)$, o qual descreve o movimento de um corpo rígido no espaço 3-D, formalmente definido por:

$$\mathbb{SE}(3) = \{R \in \mathcal{SO}(3) \wedge t \in \mathbb{R}^3\} \tag{3.13}$$

As operações do grupo são as seguintes. Considere-se uma matriz de transformação $T_B^A \in \mathbb{SE}(3)$ que transforma um ponto no sistema de coordenadas da *frame* B para o sistema de coordenadas da *frame* A, constituída pela matriz de rotação $R_{BA} \in \mathcal{SO}(3)$, que descreve a rotação do sistema de coordenadas de B para A, e o vetor de translação $t_{BA} \in \mathbb{R}^3$, que move a origem do sistema de coordenadas da *frame* A para o sistema de coordenadas da *frame* B:

$$T_B^A = \begin{bmatrix} R_{AB} & t_{AB} \\ 0_{1x3} & 1 \end{bmatrix} \tag{3.14}$$

A matriz inversa $T_A^B \in \mathbb{SE}(3)$, que transforma um ponto no sistema de coordenadas da *frame* A para o sistema de coordenadas da *frame* B, é definida por:

$$T_A^B = (T_B^A)^{-1} = \begin{bmatrix} R_{AB}^T & -R_{AB}^T t_{AB} \\ 0_{1x3} & 1 \end{bmatrix} \tag{3.15}$$

A outra operação neste grupo consiste na composição de matrizes, o que torna possível transformar um ponto num sistema de coordenadas da A para o sistema de coordenadas da *frame* C, pela transformação $T_A^C \in \mathbb{SE}(3)$, através de uma *frame* B intermédia:

$$T_A^C = T_B^C T_A^B \tag{3.16}$$

A matriz de rotação R pertence ao grupo Especial Ortogonal $\mathcal{SO}(3)$, o qual descreve o grupo de matrizes de rotação no espaço 3-D, formalmente definido por:

$$\mathcal{SO}(3) = \{R \in \mathbb{R}^{3x3} : RR^T = 1 \wedge det(R) = 1\} \tag{3.17}$$

Note-se que esta representação da rotação é sobreparametrizada, visto que uma rotação no espaço 3-D apenas tem três graus de liberdade.

Uma rotação também pode ser representada por um conjunto de rotações sucessivas em torno dos três eixos, definida pelos ângulos de Euler: *roll*, *pitch* e *yaw*. Num sistema de coordenadas com origem no centro de gravidade do agente, é normal considerar-se a seguinte convenção: *roll* indica uma rotação em torno do eixo longitudinal, *pitch* indica uma rotação em torno do eixo lateral e *yaw* indica uma rotação em torno do eixo vertical, como ilustrado na figura 3.3.

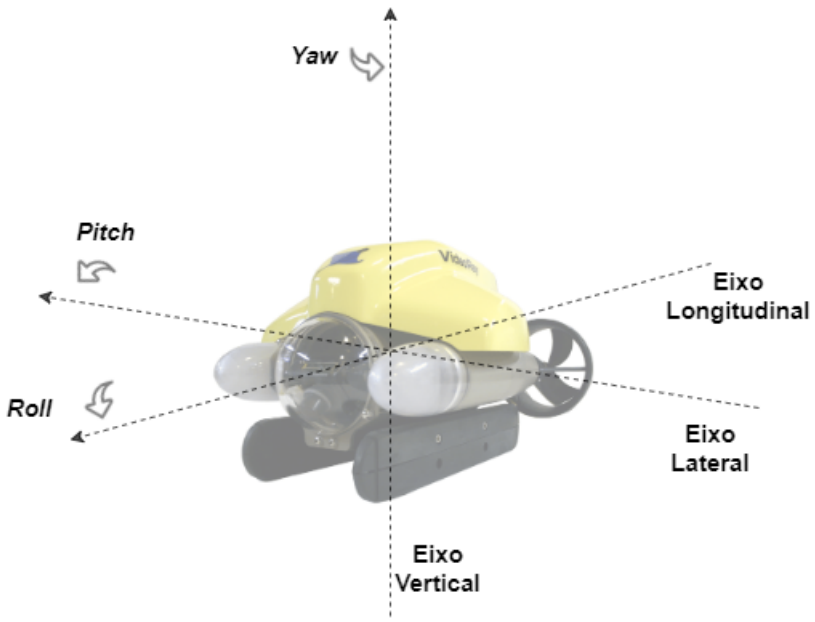


Figura 3.3: Exemplo dos ângulos *Roll*,*Pitch* e *Yaw* num ROV.

Apesar de esta notação dar uma noção mais intuitiva acerca da orientação do corpo rígido e ser a mais simples relativamente ao números de parâmetros, sofre de um problema denominado *gimbal lock*, o qual consiste na perda de um grau de liberdade na rotação em certas condições.

Uma outra alternativa é o uso de quaterniões, os quais não sofrem de *gimbal lock* como os ângulos de Euler e parametrizam uma rotação com um menor número de variáveis relativamente a uma matriz de rotação. Os quaterniões fazem parte do conjunto dos números hipercomplexos $\mathbb{H}$, uma extensão dos números complexos. Seguindo a notação de Hamilton um quaternião é definido por:

$$q = q_w + q_x i + q_y j + q_z k \tag{3.18}$$

onde q_w é a parte real, e as componentes imaginárias i, j e k são relacionadas pela seguinte igualdade:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{3.19}$$

Detalhes matemáticos sobre as conversões entre cada uma das representações, podem ser encontrados na publicação de Blanco [96].

3.3 Geometria de duas poses

3.3.1 Geometria Epipolar

A geometria epipolar descreve a relação entre a observação de um ponto 3-D e a respetiva projeção em duas imagens diferentes. Usando uma câmara monocular, permite estimar o movimento relativo entre essas duas imagens com um fator de escala desconhecido, num processo conhecido por 2D-2D *matching*.

Importa definir três conceitos para clarificar o problema ilustrado na figura 3.4:

- Epipolo (e_L, e_R) é o ponto de interseção da linha que liga os centros de projeção da câmara, em dois instantes de tempo diferentes, com o plano da imagem;
- Plano epipolar é o plano que contém a linha que liga os centros de projeção, o ponto 3-D e respetiva projeção em duas imagens;
- Linha epipolar é a interseção do plano epipolar com o plano da imagem.

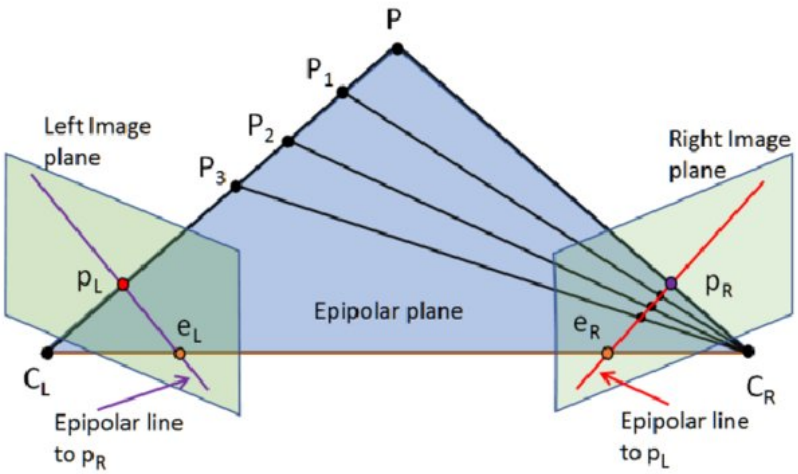


Figura 3.4: Ilustração da geometria epipolar. Fonte: [97]

Considere-se a projeção de um ponto P em duas imagens consecutivas, p_L e p_R em coordenadas homogéneas. A geometria epipolar garante a seguinte igualdade:

$$p_L^T F p_R = 0 \tag{3.20}$$

onde $F \in \mathbb{R}^{3x3}$ é a matriz fundamental e relaciona o ponto p_R com a linha epipolar que contém p_L. A linha epipolar definida por $l_L = Fp_R$ corresponde à linha epipolar na imagem da esquerda, com centro de projeção C_L. Se os pontos p_L e p_R satisfazem a relação significa que são coplanares, pelo que são correspondentes. Para estimar a matriz fundamental é necessário obter no mínimo sete correspondências, mas é normalmente utilizado o algoritmo de 8 pontos [98], de modo a garantir que apenas uma solução é calculada.

O cálculo de uma matriz fundamental não necessita de uma câmara calibrada, isto é, a matriz fundamental pode ser estimada a partir dos píxeis correspondentes entre imagens. No caso de a câmara estar calibrada, ou seja, conhece-se os seus parâmetros intrínsecos, é possível considerar um caso especial da matriz fundamental, a matriz essencial $E \in \mathbb{R}^{3x3}$. A relação entre E e F é dada por:

$$E = K^T F K \tag{3.21}$$

Substituindo (3.21) em (3.20), é possível calcular a restrição epipolar imposta pela matriz essencial:

$$p_L^T K^{-T} E K^{-1} p_R = 0 \tag{3.22}$$

$$p_L^* E p_R^* = 0 \tag{3.23}$$

onde as igualdades $p_L^* = K^{-1} p_L$ e $p_R^* = K^{-1} p_R$ correspondem pontos p_L e p_R em coordenadas normalizadas da imagem, isto é, o efeito da matriz de calibração é removido. Assim, o sistema de coordenadas da imagem tem origem no ponto principal e as distâncias focais $f_u = f_v = 1$. Isto equivale a ter o mesmo modelo definido em 3.1, em que o plano da imagem encontra-se no plano $z = 1$.

A matriz essencial é normalmente estimada a partir do algoritmo de 5 pontos proposto por Nister [99], apesar de também ser possível usar 7 ou 8 pontos. As vantagens no cálculo da matriz essencial relativamente à matriz fundamental residem no facto de que são necessários menos pontos e de que a matriz essencial fornece resultados mais robustos quando o ambiente observado é plano. Como demonstrado por Hartley e Zisserman [94], por decomposição do valor singular da matriz essencial que associa duas imagens é possível obter quatro soluções para o movimento, das quais apenas uma é válida.

3.3.2 Homografia

A homografia também pode descrever o movimento relativo entre duas imagens, mas apenas quando a região observada pelas câmaras é plana. Ao contrário da matriz fundamental, que relaciona o ponto p_R com a linha epipolar que passa em p_L, a homografia descreve diretamente uma relação entre p_L e p_R formulada por:

$$p_L = H.p_R \Leftrightarrow \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} \tag{3.24}$$

Segundo Faugeras e Lustman [100], para obter a matriz H são necessárias, no mínimo, 4 correspondências e fornece 8 soluções para o movimento relativo.

3.4 Triangulação

A triangulação consiste em estimar as coordenadas de um ponto 3-D $X_{\mathcal{W}}$ através da interseção das linhas que ligam o centro de projeção das câmaras à projeção de $X_{\mathcal{W}}$ na imagem.

Idealmente o ponto de interseção das linhas é o ponto 3-D, sobre o qual se pretende estimar as suas coordenadas. No entanto, a câmara não é um equipamento perfeito e existem erros nas medidas obtidas pela geometria epipolar/homografia.

A posição 3-D do ponto $X_{\mathcal{W}}$ é estimada minimizando o erro de reprojeção em todas as observações de $X_{\mathcal{W}}$. O problema de otimização não linear tem como solução $X^*_{\mathcal{W}}$ e pode ser formulado por:

$$X^*_{\mathcal{W}} = \arg\min_{X_{\mathcal{W}}} e_{rep}(P, X_{\mathcal{W}}) \tag{3.25}$$

$$e_{rep} = \sum_k \|\widehat{x}_k(P, X_{\mathcal{W}}) - x_k\|^2 \tag{3.26}$$

onde x_k é a projeção do ponto $X_{\mathcal{W}}$ na imagem k e $\widehat{x}_k(P, X_{\mathcal{W}})$ representa a projeção estimada do ponto $X_{\mathcal{W}}$ na imagem k, através da matriz de projeção da câmara P.

3.5 Perspective-n-Point (PnP)

Após o cálculo de uma estimativa da estrutura observada, composta pelo conjunto de pontos triangulados $X_{\mathcal{W}} = (X_1 X_2, \ldots, X_n)$, e as suas projeções na imagem $x = (x_1, x_2, \ldots, x_n)$, pode-se estimar a pose da câmara $\mathcal{C}_k$.

Como se assume que a câmara está calibrada, a associação de pontos 3D-2D é um problema conhecido por PnP, no qual a partir de n associações 3D-2D a pose da câmara é estimada. O mínimo de associações são $n = 3$, o que indica que tem de haver pelo menos a triangulação de três pontos, levando ao problema P3P ilustrado na figura 3.5.

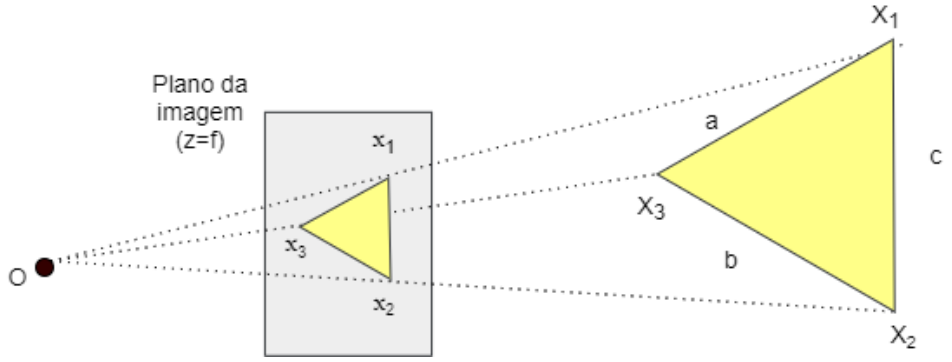


Figura 3.5: Ilustração do problema P3P. O ponto O é o centro ótico da câmara. x_1, x_2 e x_3 são as projeções dos pontos 3-D X_1, X_2 e X_3 respetivamente. Os lados do triângulo formado pelos pontos 3-D são conhecidos, mas a distância de cada ponto ao centro ótico não.

Através dos triângulos $\triangle OX_1X_2$, $\triangle OX_1X_3$ e $\triangle OX_2X_3$ e da lei do coseno é possível derivar o sistema de equações do problema P3P:

$$\begin{cases} a^2 = x^2 + y^2 - xyr \\ b^2 = z^2 + x^2 - xzq \\ c^2 = y^2 + z^2 - yzp \end{cases} \tag{3.27}$$

onde $x = \overline{OX_1}$, $y = \overline{OX_2}$, $z = \overline{OX_3}$, $p = 2\cos(\alpha)$, $q = 2\cos(\beta)$, $r = 2\cos(\gamma)$, $a = \overline{X_1X_2}$, $b = \overline{X_1X_3}$, $c = \overline{X_2X_3}$, $\alpha = \angle x_2Ox_3$, $\beta = \angle x_1Ox_3$ e $\gamma = \angle x_1Ox_2$.

Através da resolução de (3.27), pretende-se calcular o valor de x, y e z e o valor dos ângulos α, β e γ, o que possibilita estimar a pose da câmara.

A resolução deste sistema de equações é já um problema relativamente antigo, como revela o trabalho apresentado em 1994 por Haralick, Lee, Ottenberg et al. [102]. Ainda assim, outras soluções e abordagens têm aparecido mais recentemente, como por exemplo por Kneip, Scaramuzza e Siegwart [103] e Ke e Roumeliotis [102].

No entanto, a resolução do PnP não está limitado a três pontos, pelo que outras publicações como a de Lepetit, Moreno-Noguer e Fua [104] ou Kneip, Li e Seo [105] resolvem o problema de uma forma mais geral.

3.6 RANSAC

RANdom SAmple Consensus, inicialmente publicado por Fischler e Bolles [106] é um algoritmo iterativo que estima os parâmetros de um modelo matemático a partir de um conjunto de dados observados mesmo na presença de *outliers*.

Essencialmente, no RANSAC, em cada iteração um modelo é construído para pequenos conjuntos de dados extraídos aleatoriamente de um conjunto maior. Cada modelo parametrizado é aplicado ao resto dos dados, permitindo construir o conjunto de *inliers*, os quais são os que encaixam no modelo proposto, e descartar os *outliers*. O modelo que reunir o maior consenso, isto é, o que reúne o maior número de *inliers*, é o escolhido como solução, e um novo modelo é parametrizado com base no conjunto de *inliers*.

Tirando partido dos modelos geométricos anteriormente apresentados, o RANSAC é normalmente utilizado em conjunto com a estimativa das matrizes essencial, fundamental e de homografia, bem como na estimativa da pose relativamente à estrutura 3-D. A presença de *outliers* afeta a robustez da estimação, visto que os métodos de minimização dos mínimos quadrados são bastante sensíveis à presença de *outliers* nas medidas. Assim, o RANSAC permite que as estimações com base na minimização dos mínimos quadrados operem apenas sobre os *inliers*, o que garante uma solução ótima.

3.7 *Bundle Adjustment*

A aplicação da técnica de BA visa refinar as poses da câmara $\mathcal{C}$ e pontos 3-D $X_{\mathcal{W}}$ observados pelas câmaras em imagens I, minimizando o erro de reprojeção para cada observação da câmara, adaptando a definição apresentada em (3.26).

Bundle Adjustment é uma técnica de otimização não linear formalmente definido por:

$$\theta^* = \arg\min_{\theta} \sum_{k \in I} \sum_{j \in \zeta_k} E_{rep}(P_k, \zeta_j) \tag{3.28}$$

$$E_{rep}(P_k, \zeta_j) = \left\| \widehat{x}_k^j(P_k, \zeta_j) - x_k^j \right\|^2 \tag{3.29}$$

onde $\theta = \{\mathcal{C}, X_{\mathcal{W}}\}$ representa o conjunto de poses da câmara e as *landmarks* a serem otimizadas, ζ_k é o conjunto de pontos 3-D X observados na k-ésima imagem, $\widehat{x}_k^j$ a projeção estimada do j-ésimo ponto 3-D na k-ésima pose C_k, através da matriz de projeção P_k, e x_k^j é a projeção real.

Aumentando a dimensão do vetor θ, rapidamente se verifica que a resolução da regressão de mínimos quadrados não linear se torna bastante exigente em termos de processamento. Por esta razão, a dimensão do vetor é normalmente limitada pelo número de imagens que se considera, aplicando o conceito de *keyframe*.

3.8 Métodos diretos

Como os métodos diretos operam diretamente sobre os valores de intensidade da imagem, minimizam o erro fotométrico. Ao contrário, os métodos indiretos selecionam a informação que é extraída e representam-na geometricamente, minimizando o erro de reprojeção. No âmbito desta dissertação, como apenas se utilizam métodos diretos que operam sobre um conjunto de píxeis pequeno, não serão considerados os métodos densos e semidensos.

3.8.1 Alinhamento da imagem

Os métodos diretos estimam a transformação que relaciona duas poses da câmara que minimiza a diferença de intensidades entre duas imagens. De forma a aumentar a robustez, em cada píxel considerado de elevado gradiente, é extraída a região à sua volta. Assim, o erro fotométrico entre duas regiões da imagem pode ser definido pela soma da diferença de intensidade dos píxeis extraídos:

$$e_{foto} = \sum_{x \in N_p} (I_k(x') - I_{k-1}(x)) \tag{3.30}$$

onde N_p é o conjunto de píxeis vizinhos de um píxel x, x' é a projeção da posição do ponto x na imagem I_k, sabendo o valor de profundidade d.

Sabendo o valor d associado a x e recorrendo à equação (3.4) mas assumindo uma função de mapeamento para qualquer câmara, π, as coordenadas do ponto x' em I_k a partir do ponto x de I_{k-1} são dadas por:

$$x' = \pi(R\pi^{-1}(x, d) + t) \tag{3.31}$$

onde

$$\begin{bmatrix} R_{3x3} & t_{3x1} \\ 0_{1x3} & 1 \end{bmatrix} = T_k^w \tag{3.32}$$

Assim, uma pose da câmara C_k é estimada por alinhamento direto de duas imagens I_k e I_{k-1}, método este que consiste em estimar a transformação relativa T_k que minimiza o erro fotométrico entre duas imagens. Minimizando a soma dos erros quadráticos, temos:

$$T_k^* = \arg\min_{T_k} \sum_{p \in \mathcal{P}} {e_{foto}}^2 \tag{3.33}$$

onde $T_k \in \mathbb{SE}(3)$ é a matriz de transformação entre a pose C_k e C_{k-1} e $\mathcal{P}$ é o conjunto de *patches* considerados.

3.8.2 Estimação da profundidade

A maior parte das implementações conhecidas, utilizam o inverso da profundidade, o que permite representar melhor a incerteza associada à estimativa da profundidade, assim como apresenta melhores resultados quando a distância à observação é elevada. Este tipo de parametrização da distância baseia-se no trabalho de Civera, Davison e Montiel [107], no qual são explicadas as vantagens da representação da profundidade pelo seu inverso.

Independentemente do método direto desenvolvido, em todos eles é aplicado um modelo probabilístico para estimar a profundidade.

Apesar de existirem diferentes modelos propostos, todos partilham a ideia de que através geometria epipolar, é possível estimar a profundidade. Isto é efetuado ao procurar ao longo da linha epipolar o ponto para o qual a disparidade minimiza o erro fotométrico. Conforme a figura 3.4, a disparidade é a distância entre os pontos p_L e p_R.

Partindo da figura referida, o processo da procura pela disparidade λ que minimiza o erro fotométrico pode ser definido por:

$$\lambda^* = \arg\min_{\lambda} (I_L(p_L) - I_R(\lambda, p_R))^2 \tag{3.34}$$

onde $I_L(p_L)$ corresponde à intensidade do *patch* centrado em p_L na imagem I_L, com centro de projeção em C_L, e $I(\lambda, p_R)$ é a intensidade da região centrada num ponto de disparidade λ contido na linha epipolar que passa por p_R, na imagem I_R com centro de projeção em C_R.

Obtida a disparidade, o passo de estimação da profundidade assume uma baixa rotação entre as poses da imagem, tirando partido de uma elevada frequência de aquisição de imagem. Esta assunção permite relacionar a disparidade com a profundidade, de forma semelhante ao que acontece em visão *stereo*. A incerteza associada ao cálculo da disparidade é propagada de modo a reduzir o intervalo de procura da linha epipolar e refinar a estimativa da profundidade dos píxeis ao longo das imagens seguintes.

3.8.3 *Bundle Adjustment* Fotométrico

De forma análoga à formulação anterior, através do erro fotométrico é possível otimizar a estrutura 3-D constituído pelos X observados em imagens I, bem como a trajetória constituída pelas poses da câmara $\mathcal{C}$, segundo a expressão apresentada por Engel, Koltun e Cremers [47]:

$$\theta^* = \arg\min_{\theta} \sum_{k \in \mathcal{F}} \sum_{p \in P_k} \sum_{j \in obs(p)} {e_{foto}}^2 \tag{3.35}$$

onde $\mathcal{F}$ é o conjunto de *frames* a serem otimizadas, P_k é o conjunto de pontos na k-ésima *frame* e $obs(p)$ o conjunto de *frames* que observam o ponto p.

3.9 Cinemática baseada em dados da IMU

O uso de uma IMU implica a definição de, pelo menos, um novo sistemas de coordenadas e a redefinição do sistema de coordenadas do Mundo ($\mathcal{W}$, para além do sistema de coordenadas da câmara:

- *Body Frame* ($\mathcal{B}$): sistema de coordenadas fixo na IMU com origem na junção dos três eixos do acelerómetro.
- *World Frame* ($\mathcal{W}$): sistema de coordenadas fixo, alinhado com a gravidade. Se o eixo z aponta para cima, o vetor da gravidade é $g = (0, 0, -g)^T$.

A IMU é constituída por um acelerómetro, giroscópio e por um magnetómetro. Porém, no âmbito desta dissertação, o interesse recai nos dois primeiros, pelo facto de que o magnetómetro pode estar sujeito a erros quando na presença fortes anomalias magnéticas.

A leitura de ambos os sensores é corrompida por ruído e por um valor de *offset* conhecido como *bias*, que varia com o tempo. É possível modelar o ruído como sendo Gaussiano, e o *bias* como sendo um processo *random walk*. De acordo com Kok, Hol e Schön [108], os valores medidos pelo acelerómetro e pelo giroscópio, ${}_{\mathcal{B}}\widetilde{a}(t)$ e ${}_{\mathcal{B}}\widetilde{\omega}_{\mathcal{WB}}(t)$ são então definidos por:

$${}_{\mathcal{B}}\widetilde{a}(t) = R^T_{\mathcal{WB}}(t)({}_{\mathcal{W}}a(t) - {}_{\mathcal{W}}g) + b_a(t) + n_a(t) \tag{3.36}$$

$${}_{\mathcal{B}}\widetilde{\omega}_{\mathcal{WB}}(t) = {}_{\mathcal{B}}\omega_{\mathcal{WB}}(t) + b_w(t) + n_w(t) \tag{3.37}$$

onde os prefixos $\mathcal{B}$ e $\mathcal{W}$ indicam os sistemas de coordenadas em que os valores estão expressos. $n_w \sim \mathcal{N}(0, \sigma_w^2)$ e $n_a \sim \mathcal{N}(0, \sigma_a^2)$ são os ruídos Gaussianos do giroscópio e do acelerómetro, respetivamente. b_w e b_a são os ruídos aditivos do giroscópio, processos *random walk* modelados com ruído Gaussiano $n_{b_w} \sim \mathcal{N}(0, \sigma_{b_w}^2)$ e $n_{b_a} \sim \mathcal{N}(0, \sigma_{b_a}^2)$. ${}_{\mathcal{B}}\widetilde{\omega}_{\mathcal{WB}}(t)$ é a velocidade angular instantânea de $\mathcal{B}$ relativamente a $\mathcal{W}$, expressa no sistema de coordenadas $\mathcal{B}$. ${}_{\mathcal{W}}a(t)$ é a aceleração do sensor e ${}_{\mathcal{W}}g$ é o vetor de gravidade. $R_{\mathcal{WB}}$ representa a rotação de um vetor expresso no sistema de coordenadas $\mathcal{B}$ para $\mathcal{W}$, pelo que a sua transposta representa o inverso.

Segundo Chatfield [109], as cinemáticas dos dados da IMU são então definidas pelas seguintes equações diferenciais:

$$ {}_{\mathcal{W}}\dot{p}(t) = {}_{\mathcal{W}}v(t) \tag{3.38} $$

$$ {}_{\mathcal{W}}\dot{v}(t) = {}_{\mathcal{W}}a(t) = R_{\mathcal{WB}}(t)({}_{\mathcal{W}}\widetilde{a}(t) - b_a(t) - n_a(t)) + {}_{\mathcal{W}}g \tag{3.39} $$

$$ \dot{q}_{\mathcal{WB}}(t) = \frac{1}{2}\Omega({}_{\mathcal{B}}\omega_{\mathcal{WB}}(t)) \otimes q_{\mathcal{WB}}(t) = \frac{1}{2}\Omega({}_{\mathcal{B}}\widetilde{\omega}_{\mathcal{WB}}(t) - b_w - n_w) \otimes q_{\mathcal{WB}}(t) \tag{3.40} $$

$$ \dot{b}_w = n_{b_w} \tag{3.41} $$

$$ \dot{b}_a = n_{b_a} \tag{3.42} $$

onde:

- ${}_{\mathcal{W}}\dot{p}(t)$ é a posição do sensor IMU no sistema de coordenadas $\mathcal{W}$
- ${}_{\mathcal{W}}\dot{v}(t)$ é a velocidade do sensor IMU no sistema de coordenadas $\mathcal{W}$
- $q_{\mathcal{WB}}$ é o quaternião que descreve a rotação de $\mathcal{B}$ para $\mathcal{W}$
- n_w e n_a são os valores de *bias* do giroscópio e do acelerómetro respetivamente
- R_{WB} descreve a rotação que transforma um vetor de $\mathcal{B}$ para $\mathcal{W}$
- $\Omega(w) = \begin{bmatrix} \lfloor w \rfloor_\times & w \\ -w^T & 0 \end{bmatrix}$, $\lfloor w \rfloor_\times = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix}$
- $w = [w_x, w_y, w_z]^T$ é a velocidade angular medida no sistema de coordenadas $\mathcal{B}$

CAPÍTULO 4

Abordagem Experimental

4.1 Métodos de Odometria Visual-Inercial

Entre os algoritmos de odometria visual-inercial referidos na tabela 2.2 e disponíveis *open-source*, pretendeu-se mostrar as diferenças de desempenho entre a abordagem por filtragem e otimização, o que levou à decisão de escolher o ROVIO [59], [60], o qual faz parte do MAPLAB [110], o OKVIS [57], [58] e o VINS-Mono [62].

4.1.1 MAPLAB

Apresentado por Schneider, Dymczyk, Fehr et al. [110], o MAPLAB é um sistema SLAM visual-inercial direcionado à construção e processamento de mapas para serem usados em diferentes sessões. Ao invés de outros sistemas SLAM, o MAPLAB distingue-se pelas suas funcionalidades que permitem, em *offline*, a fusão de mapas, a reconstrução do ambiente e *loop-closure*, entre outras.

No âmbito desta dissertação, o interesse por esta ferramenta baseia-se, fundamentalmente, no seu sistema de odometria visual-inercial e mapeamento, o ROVIOLI. O ROVIOLI é uma adaptação do módulo de odometria visual-inercial, o ROVIO [60], ao qual são adicionadas capacidades de mapeamento e de localização que permitem criar um mapa para pós-processamento ou situar um agente com base num mapa previamente obtido. Como não se utilizaram estas funcionalidades adicionais, importa explicar o funcionamento, ainda que de uma forma simplificada, do algoritmo de odometria visual-inercial utilizado.

O ROVIO, e como referido na secção 2.5.2.1, é um método direto de odometria visual-inercial com uma abordagem *tightly-coupled* para fusão dos dados visuais e inerciais, através da implementação de um *Iterated Extended Kalman Filter*.

Efetivamente, apesar de ser um método direto, o algoritmo depende, numa primeira fase, da deteção de cantos FAST [40], o que permite ter um número elevado de *keypoints*. Devido à falta da escalabilidade do filtro, é necessário eliminar progressivamente os *keypoints* de modo a ter um conjunto mais restrito:

i. O primeiro critério consiste na eliminação de *keypoints* que estão localizados perto de *landmarks* já incluídas no filtro;
ii. O segundo critério visa avaliar a informação visual contida na região à volta de cada *keypoint*. Para tal, cada região é representada por uma pirâmide, em que cada nível representa a imagem sub-amostrada por um fator de dois. Em cada nível é calculada a matriz Hessiana, a qual permite avaliar o gradiente da região nesse nível. Assim, a região é caraterizada por um valor que combina os diferentes gradientes de cada nível e as regiões são eliminadas se o seu valor estiver abaixo de *threshold* pré-determinado e configurável;
iii. O terceiro critério tem como objetivo distribuir as *landmarks* que restam ao longo da imagem, de modo a que estas não se encontrem todas numa pequena área da imagem;

Nesta fase, os dados inerciais são usados para a propagação do estado do filtro, o que corresponde ao passo de previsão. Como a frequência de amostragem de uma IMU é normalmente alta, os autores propuseram efetuar a média das medidas que são recebidas durante o período entre duas imagens consecutivas, o que permite reduzir o custo computacional desta fase do algoritmo.

De seguida, para cada *landmark*, é calculada uma matriz de distorção linear que a reprojeta na imagem seguinte, visto que o movimento da câmara altera a perspetiva em que a *landmark* é observada. Assim, é possível calcular o erro fotométrico entre a região prevista e a região projetada. Contudo, este processo assume que a frequência de aquisição de imagem é elevada, de modo a que haja uma grande área de sobreposição entre duas imagens consecutivas.

Quando uma nova imagem é capturada, é executado o passo de atualização do filtro. Devido à implementação de um IEKF, em cada atualização, iterativamente o erro de linearização vai sendo diminuído até atingir um valor mínimo, o que permite diminuir a incerteza dos estados e melhorar a sua convergência e precisão. Neste caso, para cada *landmark*, o objetivo é minimizar o erro fotométrico, aproximando a região prevista da região projetada.

Por fim, o fluxo do algoritmo recomeça com a deteção de novas *landmarks*. A partir da segunda iteração do filtro, de modo a aferir a qualidade do *tracking*, para cada *landmark*, é implementada uma medida heurística que permite eliminar ou adicionar caraterísticas ao filtro, segundo os seguintes critérios:

i. Qualidade global: quantas vezes o *tracking* de uma *landmark* foi executado com sucesso;
ii. Qualidade local: nas imagens mais recentes, quantas vezes o *tracking* foi bem sucedido quando é previsto que a *landmark* esteja na imagem;
iii. Visiblidade local: nas imagens mais recentes, quantas vezes a *landmark* esteve, de facto, na imagem.

Algorithm 1 ROVIO

Input: conjunto de imagens e dados da IMU sincronizados
Output: pose da câmara
for k = 0, 1, 2, . . . , numImagens **do**
  Deteção de cantos FAST
  Seleção das melhores landmarks e extração dos patches
  Previsão do filtro através dos dados da IMU
  for i = 0, 1, 2, . . . , numPatches **do**
    Alinhamento do patch i e cálculo do erro fotométrico
  end for
  Eliminação de *outliers*
end for

4.1.2 OKVIS

O OKVIS [58] [57] divide o seu algoritmo em duas *threads* paralelas: uma para o processamento de imagem e outra para a otimização da trajetória.

Começando pela sua *frontend*, quando uma imagem é recebida, são detetados cantos Harris [111], os quais são descritos pelo BRISK [85]. Os dados da IMU servem nesta fase para propagar os estados.

Durante a inicialização, são utilizadas as medidas do acelerómetro e do giroscópio para estabelecer uma primeira estimativa da escala e estados. Além dos dados inerciais, os *keypoints* são associados e calcula-se a pose relativa entre duas *frames* com base no *matching* 2D-2D. A escala desta transformação é fornecida pelas medidas lidas da IMU. Juntamente com este processo, é utilizado o RANSAC para eliminar os *outliers*, garantindo a integridade da transformação.

Quando o algoritmo inicializa com sucesso e assumindo que já existe um mapa construído que apresenta as *landmarks* com as suas posições em 3D definidas, a cada imagem recebida, os dados da IMU propagam os estados e é feita a associação dos pontos 3D aos pontos 2D (*matching* 3D-2D) detetados. As correspondências efetuadas passam depois por um esquema de rejeição de *outliers*.

Logo após, os *keypoints* da imagem que não tiveram uma correspondência a uma *landmark* são associados aos *keypoints* da imagem anterior. De seguida, é feita a triangulação das caraterísticas extraídas para extrair os pontos 3D, mas apenas são adicionados ao mapa os pontos que apresentam uma baixa incerteza em relação à profundidade. Os *keypoints* que restam são utilizados para calcular a pose relativa entre a imagem atual e *keyframe* mais recente.

Durante este processo, o OKVIS mantém uma janela de otimização que contém os últimos estados, os quais têm em comum certas *landmarks*. Nesta janela são guardadas as frames e *keyframes* mais recentes, podendo estar espaçadas no tempo. Assim, a *frontend* também é responsável por decidir se uma *frame* é, efetivamente, uma *keyframe*, segundo o cumprimento dos seguintes critérios:

- Se o rácio entre a área que envolve as *landmarks* com associações e a área que envolve o

número total de *landmarks* é inferior a 60 %

- Se o rácio entre o número de *keypoints* associados e o número total de *keypoints* detetados for inferior a 20 %

Relativamente à *backend* do OKVIS, é responsável pela otimização de uma função de custo que contém os erros de reprojeção e os erros inerciais.

Para manter o problema da otimização praticável em tempo real, os autores propuseram uma estratégia de eliminação de estados antigos. Assim, dentro de uma janela composta por X *frames* e Y *keyframes*, é criada uma janela de eliminação composta pelas Y+1 *frames* mais antigas. Deste modo, sempre que uma *frame* é adicionada à janela de otimização, um dos dois cenários pode acontecer:

- Se o último estado está associado a uma *frame*, este é eliminado juntamente com a informação visual associada;
- Se o último estado está associado a uma *keyframe*, são eliminadas as *landmarks* que não estão visíveis em *keyframes* mais recentes.

Algorithm 2 OKVIS

Input: conjunto de imagens e dados da IMU sincronizados
Output: pose da câmara
for k = 0, 1, 2, . . . , numImagens **do**
  if !Initialized **then**
    Propagação do vetor de estado pelos dados da IMU
    Cálculo do movimento relativo
    Bundle Adjustment Visual-Inercial
  else
    Deteção e descrição de caraterísticas BRISK
    Propagação do vetor de estado pelos dados da IMU
    Matching 3D-2D
    Deteção de outliers por RANSAC
    Associação de caraterísticas BRISK (Matching 2D-2D)
    Triangulação
    Seleção de Keyframe
  end if
end for
{**Otimização da *sliding window* em paralelo**} Gestão das keyframes da janela Bundle Adjustment Visual-Inercial

4.1.3 VINS-Mono

O módulo de processamento de imagem no VINS-Mono [62] é composto pela receção de imagens e dos dados IMU. Em cada imagem, é aplicado o algoritmo de KLT [91] para fazer o *tracking* das caraterísticas entre imagens consecutivas. Em cada imagem são detetados novos cantos Shi-Tomasi [83], para garantir um número mínimo de *keypoints* em cada imagem. Os novos pontos detetados e os rastreados passam por um esquema de rejeição de *outliers*, composto pelo cálculo de uma matriz fundamental junto com o RANSAC [106]. Além disto,

neste módulo também é tomada a decisão de selecionar uma *keyframe*. Esta decisão baseia-se em dois critérios independentes:

i. A média do movimento aparente das *features* (paralaxe) é menor que um valor pré-determinado;
ii. O número de *features* rastreadas é menor que um determinado valor.

Relativamente aos dados da IMU, entre cada imagem os mesmos são pré-integrados, isto é, são combinadas várias medidas de modo a obter um só valor e evitar o processamento à frequência da IMU.

Deste modo, apesar de o VINS-Mono utilizar uma abordagem *tightly-coupled* para estimar o movimento, na inicialização é utilizada uma abordagem *loosely-coupled*. No método de inicialização proposto,se o *tracking* for bem sucedido e houver movimento suficiente, é recuperada a transformação relativa entre duas *frames*, pelo algoritmo de 5-pontos [99]. As *features* associadas entre as duas imagens são trianguladas com uma escala arbitrária e um método de PnP permite recuperar as poses das próximas imagens. Para terminar a inicialização dos dados visuais, é aplicado um BA para minimizar os erros de reprojeção.

Nesta fase, a estrutura visual tem como referência a primeira *frame* e o movimento da câmara estimado não está à escala, isto é, não há uma referência, por exemplo, métrica. Assim, os dados da IMU são pré-integrados, o que permite propagar o estado e estimar a escala. Além disto, o cálculo de vetor de gravidade permite estabelecer a rotação existente entre o sistema de coordenadas de referência da primeira *frame* e o sistema de coordenadas do mundo, alinhando o vetor de gravidade com o eixo z.

Após a inicialização, é utilizada uma janela temporal com as *frames* e *keyframes* mais recentes. Assim, é aplicado um BA que minimiza uma função que contém os erros de reprojeção e os erros das medidas da IMU.

A gestão desta janela tem como objetivo tornar a tarefa de otimização praticável em tempo real. Sempre que se pretende adicionar uma *keyframe*, existem duas possibilidades:

- Se a penúltima (a segunda mais antiga) imagem não for uma *keyframe*, são eliminados do seu estado as informações sobre as *features* mas são mantidas as medidas da IMU;
- Se a penúltima imagem for uma *keyframe*, esta mantém-se e elimina-se a última imagem juntamente com as informações contidas no estado associado.

O VINS-Mono também permite a capacidade de relocalizar e *loop-closure*. Estas funcionalidades baseiam-se na capacidade de deteção de um sítio anteriormente visitado, utilizando para o efeito uma abordagem por *bag of words*, através do DBoW2 [44].

Algorithm 3 VINS-Mono

```
Input: conjunto de imagens e dados da IMU sincronizados
Output: pose da câmara
for k = 0, 1, 2, ..., numImagens do
  if !Initialized then
    if !VisualInit then
      KLT Tracking
      Cálculo da matriz essencial para recuperação do movimento relativo
      Triangulação das features observadas
      PnP
      Bundle Adjustment visual
    else
      Inicialização dos valores de bias da IMU
      Recuperação da escala das medidas visuais da inicialização
    end if
  else
    KLT tracking
    Deteção de outliers por RANSAC
    Propagação do vetor de estado
    Seleção da keyframe
  end if
end for
{Otimização da sliding window em paralelo}
Gestão das keyframes da janela
Bundle Adjustment Visual-Inercial
if LoopDetected then
  4 DoF PGO global
end if
```

4.2 Métodos de Odometria Visual

Entre os algoritmos de odometria visual referidos na tabela 2.1, a escolha recaiu sobre dois métodos com abordagens opostas, um indireto e outro direto. Assim, foi escolhido o ORB-SLAM [41] e o DSO [47]. Como recentemente foi disponibilizado ao público o ORB-SLAM3 [50], este também foi utilizado, servindo para verificar se apresenta melhorias significativas nesta nova versão.

4.2.1 ORB-SLAM3

O ORB-SLAM3 [50] é um sistema de vSLAM indireto e o seu nome é derivado da utilização de descritores ORB. Mur-Artal, Montiel e Tardós [41] descrevem o ORB como uma «alternativa eficiente» a outros descritores populares como o SIFT e o SURF.

Tal como no PTAM, o ORB-SLAM3 divide a tarefa de mapeamento e localização em *threads* paralelas. No ORB-SLAM3, o algoritmo divide-se em três: *tracking*, mapeamento local e *loop-closure*/fusão de mapas.

A *thread* de *tracking* é, essencialmente, responsável pelo processamento da imagem de forma a localizar a câmara e pela inicialização do algoritmo. Assim, cada imagem recebida é

representada sob a forma de uma pirâmide e são extraídos cantos FAST em várias escalas da imagem e de forma uniforme, o que garante que os *keypoints* detetados não se encontram localizados na mesma área. Estes pontos são então descritos utilizando o descritor ORB.

De modo automatizar o processo de inicialização do algoritmo, é proposto o cálculo de dois modelos geométricos, a homografia que descreve uma cena plana e a matriz fundamental que descreve o oposto. Com base numa medida heurística um dos modelos é escolhido e as *features* associadas para a cálculo desse modelo são trianguladas. Se o erro de reprojeção for baixo e o movimento aparente (paralaxe) de uma das soluções for aceitável, então efetua-se um BA para refinar o mapa e a trajetória iniciais. Caso contrário, o processo recomeça.

Após a inicialização e assumindo que o *tracking* foi bem sucedido, uma primeira estimativa da pose da câmara é calculada através de um modelo de movimento constante. Os pontos triangulados na última imagem foram adicionados ao mapa e procura-se efetuar a correspondência entre estes e as caraterísticas detetadas na imagem atual. Se o *tracking* não tiver sido bem sucedido, ou seja, perdeu-se o estado da câmara, então é efetuada a relocalização. Neste processo, procura-se encontrar uma correspondência entre a imagem atual e os mapas ativo e inativo (se este existir). Se a relocalização for possível, o mapa atual passa para o estado inativo e um novo mapa é criado.

Se até esta fase o *tracking* tiver sido bem sucedido, o mapa local ativo será usado para encontrar correspondências com *features* da imagem atual que não foram associadas. Este mapa contém um conjunto de *keyframes* que partilham pontos 3D em comum com a imagem atual. Uma nova pose é estimada com base nas novas correspondências efetuadas. O último passo do módulo de *tracking* passa por decidir se uma *frame* é considerada uma *keyframe*.

O módulo de mapeamento local é responsável pela atualização do mapa ativo, o que inclui a integração de uma *keyframe* no mapa e no gráfico de covisibilidade, o qual indica que *keyframes* partilham observações. Também é responsável pela decisão de eliminar pontos do mapa que foram adicionados recentemente e de adicionar de novos pontos ao mapa, Em cada inserção de uma *keyframe*, esta e todas as que estão ligadas a si pelo gráfico de covisibilidade, bem como os pontos do mapa observados por todas estas *keyframes* serão otimizados por BA. No final, ocorre um processo que permite remover *keyframes* redundantes, isto é, partilham mais de 90 % dos pontos 3D com três *keyframes* anteriores. A remoção destas *keyframes* permite manter a tarefa de otimização praticável em tempo real.

Em paralelo, o módulo de *loop-closure* e fusão de mapas procura encontrar regiões anteriormente visitadas, através de *bag of words*. Se esta região pertencer ao mapa ativo é executado, a trajetória e o mapa são otimizados de uma forma global, isto é, sobre todas as *keyframes* e respetivos pontos do mapa associados contidos no mapa ativo. Se a região encontrada pertencer a um mapa inativo, então os dois mapas são combinados e otimizados.

Algorithm 4 ORB-SLAM3

```
Input: conjunto de imagens
Output: pose da câmara
for k = 0, 1, 2, ..., numImagens do
  if !Initialized then
    Extração e Associação de caraterísticas ORB
    Cálculo das matrizes de homografia e fundamental
    Seleção do modelo e cálculo do movimento relativo
    Bundle Adjustment Visual
  else
    Extração de caraterísticas ORB
    Matching 2D-2D / 3D-2D
    Seleção da keyframe
  end if
end for
{Tarefa paralela de mapeamento}
Inserção da keyframe no mapa
Gestão do mapa
Triangulação de caraterísticas ORB entre Keyframes
Bundle Adjustment sobre as keyframes que partilham informação visual
Gestão das keyframes
{Tarefa paralela de reconhecimento de um lugar e fusão de mapas}
if LoopDetected then
  PGO + BA
end if
if Place Detected in 2 maps then
  Fusão de mapas
end if
```

4.2.2 *Direct Sparse Odometry* (DSO)

O DSO é um método de odometria visual direto, pelo que opera diretamente sobre a intensidade dos píxeis, e disperso, isto é, apenas opera sobre píxeis que apresentem um elevado gradiente.

Como noutros métodos, o DSO apresenta uma *front-end* responsável pela tarefa de odometria visual e uma *back-end* encarregue da minimização de erro fotométrico sobre um conjunto de *keyframes* mais recentes.

No DSO, as poses estimadas são calculadas relativamente à última *keyframe* adicionada à janela de otimização. A criação de *keyframes* é determinada por três critérios:

- Quando o ambiente observado muda significativamente desde a *keyframe* mais recente;
- Quando a translação da câmara provoca oclusões;
- Se o tempo de exposição da câmara variar significativamente, o qual é medido através do cálculo de um fator de brilho entre duas imagens.

Tal como acontece em outros algoritmos, determinadas *keyframes* são eliminadas. No DSO, a estratégia de eliminação consiste em manter as duas *keyframes* mais recentes e eliminar as *keyframes* que apresentam menos de 5% pontos visíveis na *keyframe* mais recente. Além disso,

se o limite da janela de otimização for alcançado, é realizada uma medida heurística baseada na distância Euclidiana, com o objetivo de manter as *keyframes* mais próximas da *keyframe* mais recente.

Cada imagem adquirida é dividida em blocos de iguais dimensões, para garantir uma distribuição uniforme. Em cada bloco, é calculada a mediana do gradiente sobre todos os pixeis para definir um valor de *threshold*. Os pixeis selecionados são aqueles que apresentem uma magnitude do gradiente superior ao valor de *threshold*. Para incluir píxeis de regiões de baixo gradiente, o processo é repetido mais duas vezes, aumentando o tamanho do bloco em cada iteração.

O *tracking* dos pixeis selecionados é concretizado através da procura ao longo da linha epipolar do píxel correspondente que minimiza o erro fotométrico. No entanto, nem todos os píxeis selecionados são utilizados para estimar o movimento. O algoritmo tenta manter um número de pixeis constante, distribuídos ao longo das *keyframes* que compõem a janela de otimização e distribuídos igualmente pelas regiões das imagens, pelo que quando necessário, novos pontos são ativados para serem otimizados.

Apesar de o algoritmo fazer o *tracking* de pontos que não serão usados no imediato para otimização, são detetados os *outliers* para eliminá-los antes de serem ativados.

Em paralelo, na *back-end* é otimizado o mapa e as poses, minimizando o erro fotométrico sobre a *sliding window* através do algoritmo de Gauss-Newton. Os pontos do mapa são projetados na *host frame* (imagem onde o ponto foi observado), e projetados nas *keyframes* consecutivas de modo a calcular termos residuais. O conjunto dos residuais com as *keyframes* incluídas na janela constitui um problema de otimização não linear de uma função de energia.

Algorithm 5 DSO

Input: conjunto de imagens
Output: pose da câmara
for k = 0, 1, 2, . . . , numImagens **do**
  Extração de pontos localizados numa área de elevado gradiente
  Alinhamento direto da imagem
  Decisão sobre a seleção da keyframe
  Gestão de keyframes
end for
{**Tarefa paralela de otimização da *sliding window***}
Gestão dos pontos utilizados no *tracking*
Estimação da profundidade
BA Fotométrico

4.3 *Datasets*

Os *datasets* constituem uma vertente importante no desenvolvimento de soluções de odometria visual. Estes permitem testar os métodos e avaliá-los antes de serem aplicados em situações reais e servir como *benchmarking*, o que permite comparar diferentes soluções.

Porém, quando comparado com os meios terrestre e aéreos, verifica-se que o meio subaquático carece deste tipo de informação. Este facto deve-se principalmente a toda a logística

e equipamento que são necessários para aquisição de dados. Para além de uma sequência de imagens, também seria importante ter um *ground-truth* para poder servir de referência e comparar com os resultados obtidos.

Tendo em conta estas restrições, existem alguns *datasets* disponíveis publicamente com o objetivo de localização subaquática. Alguns adquiridos a partir de simuladores, como os de Duarte, Zaffari, da Rosa et al. [112] e Wang, Zhu, Wang et al. [113]. Relativamente aos que foram adquiridos em ambientes reais, existem opções como o *Marine Robotics Dataset* do grupo *Australian Centre For Field Robotics* (ACFR) pertencente à Universidade de Sydney, apresentado por Bender, Williams e Pizarro [114]. Este *dataset* foi obtido num coral em águas australianas e fornece diversos dados sensoriais da navegação, de um sonar e das duas câmaras que compõem a visão esteoroscópica. Outra opção é o *Underwater Caves Sonar And Vision Data Set* da Universidade de Girona, apresentado por Mallios, Vidal, Campos et al. [115]. Este conjunto de dados foi obtido numa operação com um AUV equipado com um DVL, IMU, entre outros sensores, navegou numa cave subaquática localizada na costa da província de Girona. Durante as imagens capturadas por uma câmara monocular aparecem 6 cones que servem como fonte de localização para fornecer um *ground-truth*. Por último, o *dataset* denominado AQUALOC, fornecido por Ferrera, Creuze, Moras et al. [65], no qual um ROV navega num porto e em zonas arqueológicas na costa da Córsega, e são fornecidas as imagens das operações, capturadas por uma câmara monocular, assim como os dados da IMU e do sensor de pressão obtidos durante as operações.

Assim, foi necessário tomar uma decisão sobre qual destes conjuntos de dados seria utilizado para realizar os testes. Uma condição foi a de que os dados teriam sido adquiridos em ambientes reais, visto que mostram melhor o efeito da degradação da imagem e os tipos de ambientes em que os veículos navegam. Outro requisito consistiu na frequência de aquisição de imagem. De facto, nos *datasets* divulgados por Bender, Williams e Pizarro e Mallios, Vidal, Campos et al., as imagens são capturadas a uma frequência demasiado baixas (menores que 5 Hz) para serem testados em métodos de odometria visual, os quais necessitam de uma frequência mais alta pois baseiam-se na sobreposição elevada entre imagens consecutivas. A decisão recaiu então no *dataset* proposto por Ferrera, Creuze, Moras et al., o AQUALOC, que cumpre todos os requisitos mencionados e apresenta muitas gravações em cenários distintos.

4.3.1 AQUALOC

O AQUALOC, apresentado por Ferrera, Creuze, Moras et al. [65], é um *dataset* com diversos vídeos capturados no meio subaquático e destinado ao teste de algoritmos de VIO e vSLAM. Para além do meio em questão, as gravações mostram um dos cenários em que é possível aplicar um ROV, isto é, numa missão de exploração.

Deste modo, as sequências foram gravadas por dois ROVs, ambos equipados com uma câmara monocular e monocromática, uma IMU e um sensor de pressão. Uma particularidade das gravações efetuadas, é que acabam e terminam no mesmo sítio, o que permite verificar numa primeira análise se a trajetória estimada está de acordo com o esperado.

O primeiro grupo, constituído por sete sequências, foi capturado num porto em poucos

metros de profundidade. Nas filmagens verificam-se as degradações visuais do meio subaquático, bem como a presença de zonas arenosas, de turbidez da água e o fenómeno de *backscattering*. Além disso, em duas sequências existem colisões do veículo subaquático com rochas, o que permite testar a robustez dos algoritmos de VIO à ausência de informação visual.

O segundo grupo, constituído por dez sequências, foi capturado em dois sítios arqueológicos a 270m e a 380m, na costa da Córsega. Nestas gravações as degradações visuais são ainda mais intensas e, em algumas, o ambiente observado torna-se bastante dinâmico com a presença de peixes a entrarem no campo de visão da câmara. Além destas caraterísticas, em duas das sequências verifica-se a presença de um braço robótico a pegar num objeto.

Como é esperado, gerar um *ground-truth* neste tipo de ambientes é bastante complicado. Para superar esta dificuldade, Ferrera, Creuze, Moras et al. utilizaram o *software* COLMAP [1], apresentado por Schönberger e Frahm [116], para fazer uma reconstrução 3-D do ambiente e estimar a trajetória. Como os autores deste *dataset* referem, o mesmo teve de ser feito *offline* e o *ground-truth*, apesar de não ser perfeito, é um bom ponto de partida para a avaliação dos algoritmos.

Relativamente ao conjunto de ficheiros disponibilizados, este *dataset* também possibilita uma rápida integração com o ROS [117] ao fornecer um ficheiro ROS *bag* que contém vários tópicos com as seguintes informações:

- Imagens captadas a uma frequência de 20Hz;
- Medidas da IMU lidas a uma frequência de 200Hz;
- Medidas do sensor de pressão a uma frequência de 5Hz

Além do ficheiro ROS *bag*, estes dados também são disponibilizados sob a forma de um ficheiro CSV, com a respetiva *timestamp* associada. Por último, os ficheiros de calibração da câmara e da IMU também estão disponíveis. Estes ficheiros contêm os parâmetros íntrinsecos da câmara, os parâmetros extrínsecos que relacionam a câmara e o sensor IMU, bem como os valores de ruído e *bias* do acelerómetro e do giroscópio.

4.4 Processamento de imagem

O processamento de uma imagem com o objetivo de melhorar a visibilidade pode ajudar à identificação de informação a ser extraída, o que por sua vez, pode melhorar a qualidade dos algoritmos de odometria visual.

A maior limitação da implementação de um módulo de pré-processamento é o seu tempo de execução. Um algoritmo de odometria visual espera uma frequência elevada de aquisição de imagens, pelo que a soma dos tempos de execução do algoritmo e do módulo de pré-processamento deve ser inferior ao período de aquisição de uma imagem. Muitas das abordagens, no entanto, estão focadas na recuperação da cor ou na eliminação do efeito de *backscattering*, pelo que recorrem a processos de filtragem da imagem intensivos.

Outra restrição, mas esta enquadrada no *dataset* escolhido, consiste no facto de que as imagens do fornecidas pelo *dataset* do AQUALOC são em *grayscale*, pelo que os métodos foram escolhidos também com base neste factor.

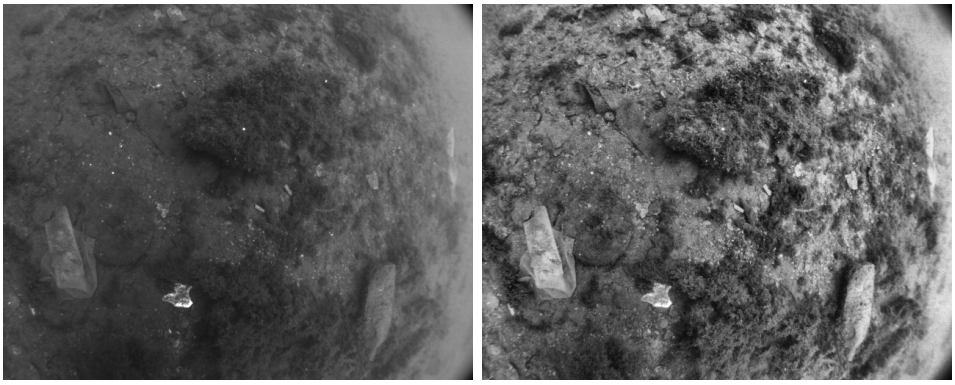
[1] https://github.com/colmap/colmap

Deste modo, foram utilizados o CLAHE e um processo por decomposição da imagem, proposto por Cho, Jeong e Kim [118]. Note-se que este último, não preenche totalmente o requisito de ser aplicado em métodos de odometria visual, visto que o seu tempo de processamento é, em média, 350ms por imagem. Porém, quando comparado com outras propostas como a apresentada Tarel e Hautière [119] ou por Marques e Branzan Albu [120], as quais têm um tempo médio de execução de por imagem por volta dos 6 e 30 segundos respetivamente, 350ms é um tempo de execução razoável para ser aplicado *offline* em milhares de imagens.

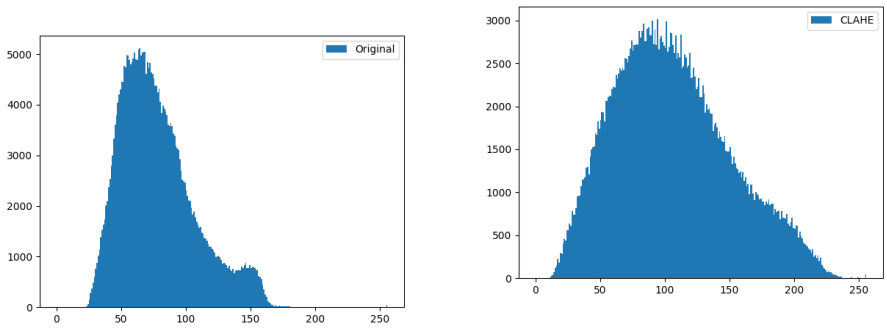
4.4.1 Contrast-Limited Adaptive Histogram Equalization

O CLAHE [121] é uma versão alternativa da equalização do histograma para aumentar o contraste existente na imagem. Ao contrário do último, que aplica a equalização sobre toda a imagem, no CLAHE a equalização é realizada em regiões da imagem, denominados *tiles*, o que permite evitar diminuir o contraste de certas regiões para aumentar o contraste global da imagem.

Efetivamente, a equalização de um histograma consiste em ajustar a intensidade dos *pixels* da imagem de modo a aumentar o contraste. As imagens com baixo contraste concentram as suas intensidades numa reduzida gama, pelo que o objetivo é estender o gráfico do histograma, como ilustrado na figura 4.1.



(a) Imagem original. **(b)** Resultado da aplicação do CLAHE.



(c) Histograma da imagem original. **(d)** Histograma da imagem após aplicar o CLAHE.

Figura 4.1: Resultado do processamento da imagem com o CLAHE. A imagem original foi retirada do *dataset* do AQUALOC.

Este processo, todavia, pode aumentar o ruído presente na imagem, algo que é limitado no CLAHE impondo um limite para o contraste. Para remover limites artificiais entre as regiões vizinhas, estas são combinadas através de interpolação.

Espera-se que a aplicação do CLAHE aumente o número *keypoints* detetados e o número de associações de *features* entre imagens, de modo a tornar os métodos de odometria mais robustos a cenários em que a imagem capturada tem pouco contraste ou tenha demasiada luminosidade.

4.4.2 Decomposição Multi-Banda

Cho, Jeong e Kim [118] propuseram um método de melhoria da visibilidade da imagem baseado na decomposição da imagem após filtragens consecutivas, através de um *guided filter* [122]. Este filtro é conhecido pelas suas propriedades de preservar as bordas/*edges* contidas na imagem com uma complexidade computacional inferior à do filtro bilateral.

A imagem é filtrada sequencialmente, variando o fator de *smoothing* ϵ. Este fator indica um nível de *threshold*, em que píxeis contidos numa região da imagem com uma variância muito menor que ϵ têm o seu valor de intensidade alterado para a média das intensidades dos píxeis da região considerada. Uma *edge* é uma zona da imagem que é caraterizada por mudanças de gradiente, pelo que é uma área de maior variância. Se o valor for maior que ϵ, então o valor do *pixel* mantém-se.

Assim, o método proposto guarda as imagens residuais, as quais são o resultado da diferença entre filtragens de uma imagem com valores de ϵ diferentes.

A imagem filtrada com o maior ϵ serve como base para o resto do algoritmo. Sobre esta imagem é estimado um mapa da iluminação do ambiente, assumindo que esta luz é caraterizada por uma elevada intensidade e onde a região da imagem tem pouca textura. Para tal, os 0.1% de píxeis para os quais a probabilidade de pertencerem à iluminação do ambiente são selecionados como candidatos e a mediana da intensidade deste conjunto de píxeis é escolhida como valor da intensidade da iluminação do ambiente.

Após a obtenção do valor da intensidade da iluminação do ambiente, a imagem é dividida em várias regiões e é construído o mapa de transmissão inicial da luz na imagem para cada bloco, através de uma função de custo na qual é maximizado o contraste e minimizado o ruído. Este mapa visa estimar a luz que é atenuada pelo meio ambiente em que se propaga. Finalmente, este mapa é refinado através de um *guided filter*.

Figura 4.2: Resultado do processamento da imagem por decomposição multi-banda. A imagem original no lado esquerdo e a imagem processada no lado direito. Nota-se um aumento do detalhe na imagem processada.

Cada imagem residual passa por um filtro Laplaciano para realçar os detalhes contidos na imagem e eliminar certos artefactos provocados pelo *guided filter*.

Na última etapa do método a imagem é reconstruída a partir da imagem de base, dos mapas criados e das imagens residuais. Um exemplo da aplicação deste método é visível na figura 4.2.

4.5 VideoRay Pro4 ROV

4.5.1 Descrição do equipamento

O VideoRay Pro 4 ROV é um ROV de inspeção adquirido à empresa VideoRay e disponibilizado pelo Instituto de Electrónica e Engenharia Informática de Aveiro (IEETA). As suas especificações técnicas encontram-se na tabela 4.1. Juntamente com o ROV, é facultado um *software* de operação do veículo para Windows que permite comandar o ROV, verificar o seu estado geral e adquirir dados. Também foi fornecido um cabo *tether* que é responsável pela alimentação e comunicação entre o ROV e um computador.

O veículo pode mover-se em seis graus de liberdade, três graus de movimento de translação (*surge*, *sway* e *heave*) e três graus de rotação (*roll*, *pitch* e *yaw*), como descrito na figura 4.1. Tal é possível devido a estar equipado com três propulsores, dos quais dois horizontais estão localizados na parte traseira e são responsáveis pelo movimento em *surge* e *yaw*, e um propulsor vertical localizado na parte de cima e dedicado ao movimento em *heave*, o que torna possibilita o controlo de três graus de liberdade do veículo.

Dimensões (C x L x A)	37.5 x 28.9 x 22 cm
Profundidade máxima	300m
Velocidade máxima	4.2 nós (7.78 km/h)
Câmara	Resolução (L x A): 640x480 Frequência: 25 fps Formato: PAL Ângulo de visão: 90º na horizontal/140º na diagonal Tilt: 180º graus
Luz	2 conjuntos de LED de 3600 lúmens
Sensores	Bússola Acelerómetro Giroscópio Temperatura interna Temperatura externa Tensão interna Profundidade

Tabela 4.1: Especificações técnicas do VideoRay Pro4 ROV.

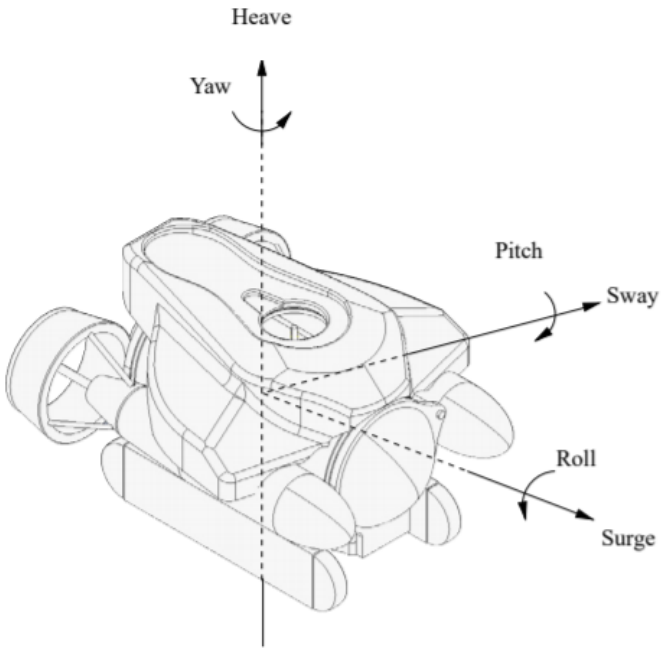


Figura 4.3: Ilustração dos graus de liberdade do ROV. Adaptado do trabalho de Mai, Pedersen, Hansen et al. [123].

A comunicação com o ROV neste momento está totalmente integrada no ROS, substituindo, no âmbito do presente trabalho, o *software* para *Windows* disponibilizado pelo fabricante.. Isto facilita o desenvolvimento de novas funcionalidades e a utilização dos algoritmos já mencionados, que possuem uma interface para o ROS.

4.5.2 Entrelaçamento do vídeo

Como é referido na tabela 4.1, a imagem capturada pela câmara do ROV está no formato PAL (*Phase Alternated Line*). O PAL é um *standard* de vídeo que codifica o espaço de cor e que foi muito utilizado na Europa nas televisões analógicas. Neste formato, a imagem observada é o resultado de dois campos transmitidos em instantes de tempo diferentes. Um campo contém as linhas pares da imagem e outro outro as linhas ímpares, como descrito na figura 4.4.

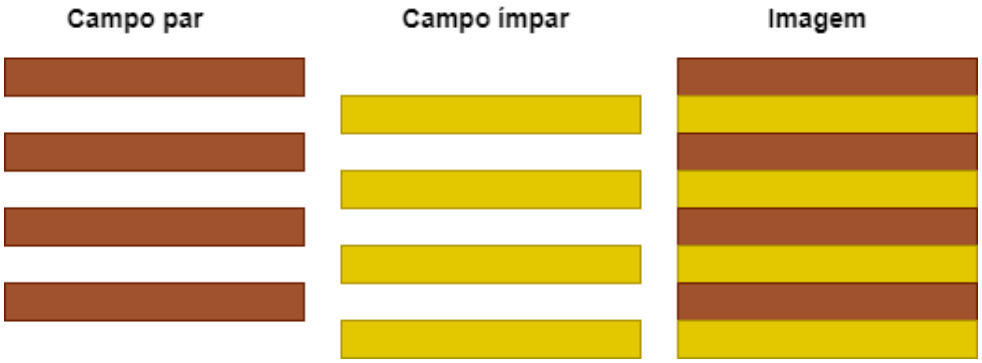
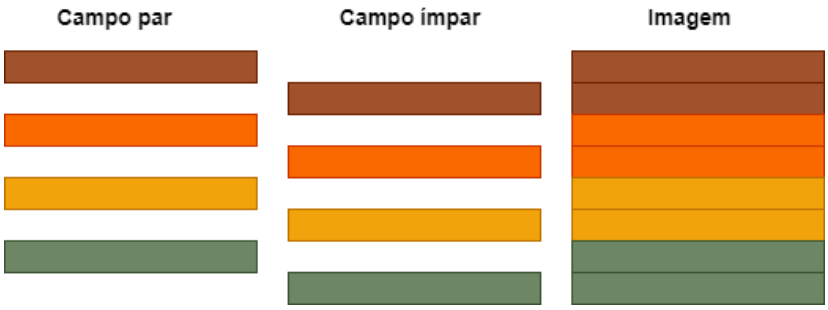


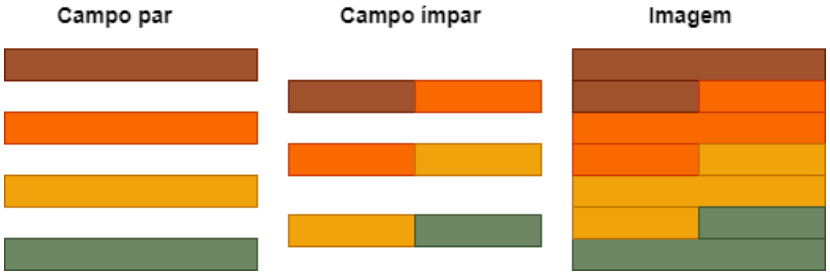
Figura 4.4: Processo de entrelaçamento da imagem.

Apesar de este processo permitir duplicar a frequência de imagem percetível sem aumentar a largura de banda, tem um impacto negativo nos métodos visuais aplicados no presente trabalho.

Assim, duas soluções foram avaliadas para resolver o problema do *interlacing* mantendo a *framerate*. A primeira consistiu na duplicação das linhas pares. O segundo método consistiu substituir as linhas ímpares pela média entre as linhas pares adjacentes. Ambas as técnicas estão ilustradas na figura 4.5.



(a) Duplicação das linhas pares.



(b) As linhas ímpares correspondem à média entre as linhas pares adjacentes.

Figura 4.5: Ilustração dos dois processos utilizados para retirar o *interlacing*.

Na figura 4.6 é ilustrado o resultado das duas técnicas de desentrelaçamento numa imagem capturada pelo veículo, onde se verifica que o efeito do entrelaçamento desapareceu.

(a) Imagem original ampliada.

(b) Desentrelaçamento por duplicação de linhas.

(c) Desentrelaçamento pela média entre as linhas pares.

Figura 4.6: Resultado do desentrelaçamento.

4.5.3 Calibração da câmara

A calibração da câmara é um processo crucial para poder estimar o movimento de um agente com base em imagens. As imagens formadas representam informação do ambiente num formato 2-D, pelo que é necessário modelar a câmara de forma a conseguir projetar os pontos da imagem num espaço 3-D. Além disto, a lente de uma câmara não é perfeita, pelo que as imagens também apresentam uma distorção que pode ser modelada.

Para o efeito, utilizou-se o módulo de calibração de câmaras *fisheye* da biblioteca OpenCV [124], em que é aplicado o modelo de uma projeção perspetiva de uma câmara *pinhole* e utiliza o modelo proposto Kannala e Brandt [95] para corrigir a distorção, como foi descrito na secção 3.1.

Para calibrar a câmara segundo o modelo descrito foi utilizado um tabuleiro *ChArUco*, composto por um padrão de xadrez e marcadores ArUco [125], como ilustrado na figura 4.7.

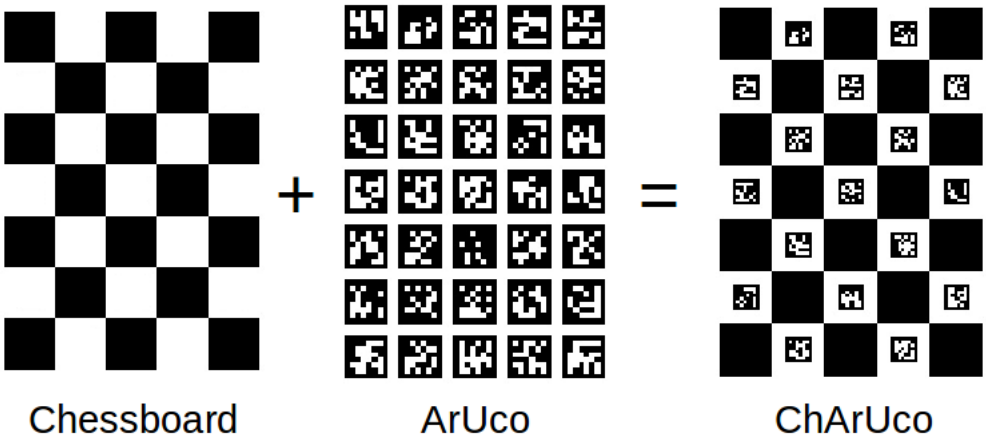


Figura 4.7: Representação do tabuleiro ChArUco. Fonte:[2]

O padrão de xadrez é bastante utilizado para efeitos de calibração, no entanto a calibração com este tabuleiro é sensível à luz e o padrão tem de estar totalmente visível na imagem.

Um marcador ArUco é composto por um marcador quadrado preto e um padrão binário que representa o seu número identificador (ID). Para a identificação é utilizado um dicionário que é definido pelo número de bits e pelo número de marcadores contidos na lista. Assim, uma calibração com estes marcadores permite que haja oclusões ou visibilidade parcial do tabuleiro. Apesar disto, a deteção dos cantos dos marcadores não é tão precisa como acontece nos padrões axadrezado.

Portanto, os tabuleiros ChArUco conjugam as vantagens e suprimem as desvantagens dos outros dois tabuleiros, o que permite uma calibração tão versátil como a que é feita com marcadores ArUco e tão precisa como quando se usa um tabuleiro de xadrez.

Para implementar a deteção e reconhecimento do tabuleiro ChArUco foi utilizado o OpenCV, que tem um módulo que lida com estas tarefas, como ilustrado na figura 4.8.

[2] `https://docs.opencv.org/3.4/df/d4a/tutorial_charuco_detection.html`

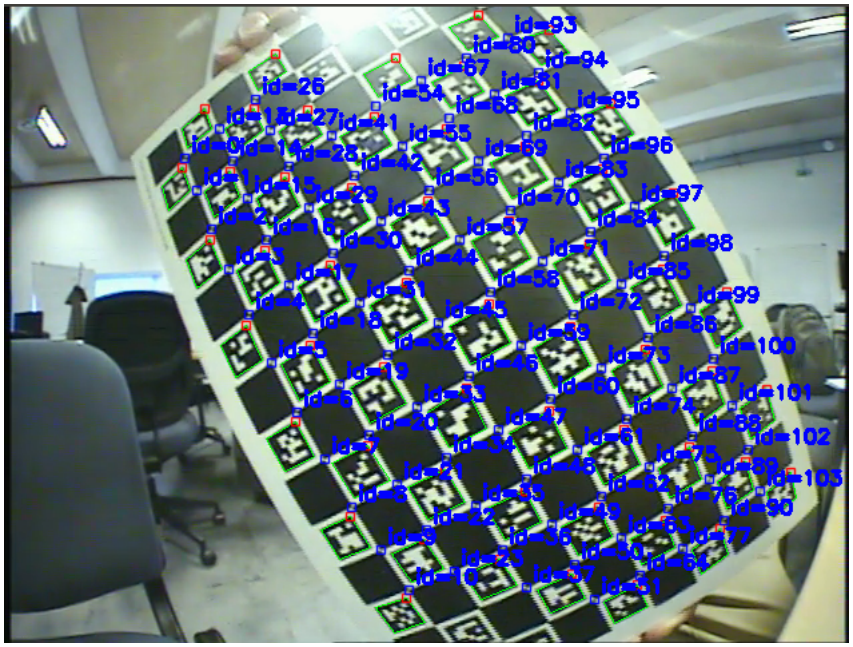


Figura 4.8: Deteção dos marcadores ArUco e dos cantos do padrão de xadrez.

O resultado da calibração de uma câmara *fisheye* utilizando um tabuleiro ChArUco é ilustrado na figura 4.9. Obtém-se então uma matriz composta pelos parâmetros intrínsecos da câmara K juntamente com os parâmetros de distorção ($k1, k2, k3$ e $k4$) e o valor de erro de reprojeção em *pixels*, definido pela medida do *Root Mean-Square Error* (RMSE).

$$K = \begin{bmatrix} 316.30 & 0 & 325.56 \\ 0 & 324.65 & 252.71 \\ 0 & 0 & 1 \end{bmatrix}, D_k = \begin{pmatrix} k1 \\ k2 \\ k3 \\ k4 \end{pmatrix} = \begin{pmatrix} -6.0810^{-2} \\ 1.32.10^{-1} \\ -1.35.10^{-1} \\ 4.72.10^{-2} \end{pmatrix}, RMSE = 0.68\, píxeis \quad (4.1)$$

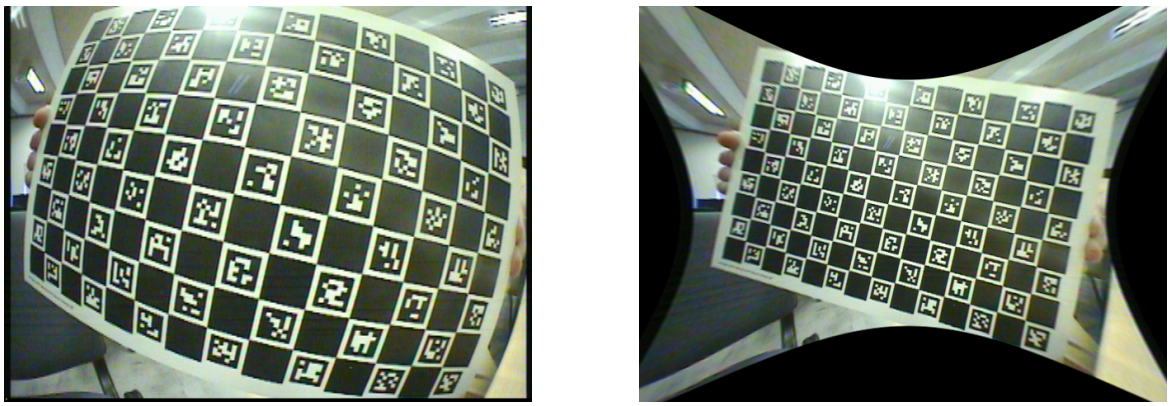


Figura 4.9: Resultado da calibração. À esquerda é a imagem distorcida, onde se nota o efeito de barril. À direita a imagem após calibração da câmara onde as linhas já estão retas e o efeito de barril foi eliminado.

4.5.4 Testes efetuados

Várias trajetórias foram realizadas para testar o desempenho dos algoritmos de odometria e a sua aplicabilidade no ROV.

Infelizmente não foi possível operar o ROV num cenário subaquático devido à conjuntura social que o vírus Covid-19 gerou, o que tornou o acesso a locais como piscinas ou tanques mais restrito e difícil. No entanto, os ambientes subaquáticos artificiais disponíveis para teste também não constituem um cenário ideal para esta aplicação, devido à falta de *landmarks* que possam ser usadas para a odometria visual. Também existem fortes limitações à colocação de marcas artificiais nas estruturas referidas por serem usados na biologia marinha, estando destinadas à criação e proteção de espécies marinhas. Além disto, o veículo em questão é uma versão para uso académico que não passou os testes de estanquidade requeridos para a sua operação em maior profundidade, como implicariam cenários de arqueologia ou exploração náuticas.

Por estes motivos, de forma a demonstrar que é exequível a utilização da visão computacional no veículo em estudo, são propostos dois circuitos dentro do edifício do *IRIS Lab* (*Intelligent Robotics and Systems Laboratory*) da Universidade de Aveiro.

O primeiro circuito está ilustrado na figura 4.10. Este circuito consiste na descida de duas rampas consecutivas, do *IRIS Lab* com o veículo transportado numa cadeira rolante. No final da descida da segunda rampa do trajeto, o veículo é levantado e colocado de novo na posição inicial para dar uma volta completa.

Por ter um conjunto de *landmarks* bastante próximas no percurso que é efetuado pelo veículo e o efeito do entrelaçamento ser mais percetível em movimento, o objetivo deste circuito é avaliar as diferenças que existem com as técnicas de desentrelaçamento da imagem testadas, assim como o efeito do entrelaçamento no desempenho dos algoritmos.

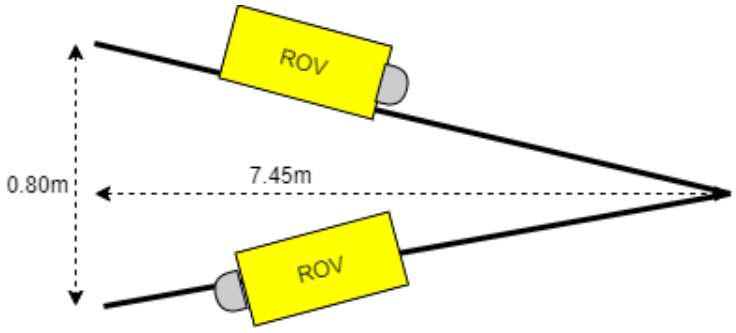


Figura 4.10: Vista lateral do primeiro circuito de teste.

O segundo circuito está ilustrado no esquema da figura 4.11, sobreposto à planta do cenário interior utilizado. Neste circuito foram colocadas diversas marcas artificiais, não para servir como referência para um *ground-truth* de localização, mas sim para adicionar caraterísticas ao cenário, visto que o chão e as paredes que delimitam o circuito têm pouca textura e, portanto, poucas caraterísticas distintas, como é visível na figura 4.12.

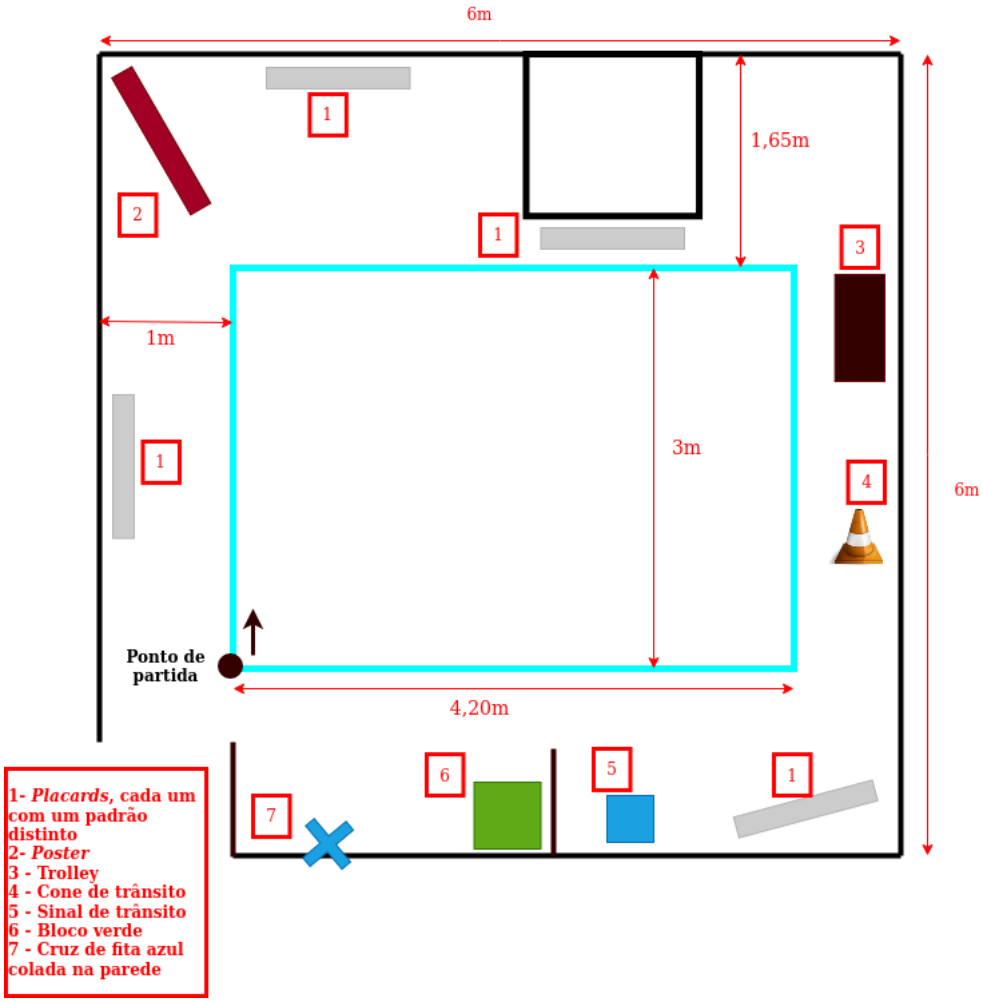


Figura 4.11: Representação gráfica do circuito com as medidas do trajeto realizado pelo ROV.

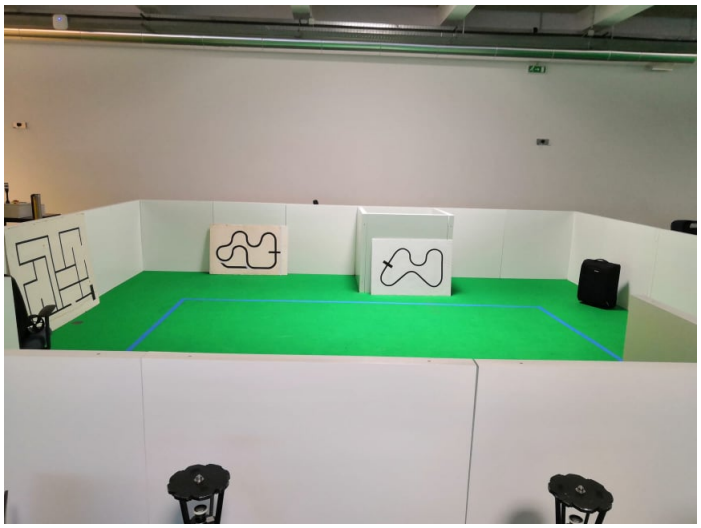


Figura 4.12: Fotografia do circuito.

Deste modo, seguindo a linha azul do trajeto proposto, foram executados quatro percursos em que o veículo dá uma volta completa e termina no canto superior esquerdo. As diferenças entre cada percurso são a altura a que o veículo se encontra e a variação da sua orientação:

i. O primeiro caminho efetuado está ilustrado na figura 4.13, onde o ROV foi transportado a uma altura constante numa cadeira de escritório com rodas. Em cada canto do trajeto definido, o veículo roda 90º;
ii. No segundo caminho, o ROV também foi transportado a uma altura constante mas sem mudanças de direção, como ilustrado na figura 4.14;
iii. Num terceiro teste, a trajetória realizada foi baseada no primeiro cenário, mas com o veículo a ser transportado à mão, de modo a variar a altura do deslocamento do veículo. Além disto, tirando partido do campo de visão amplo da câmara do ROV, esta foi apontada ligeiramente para baixo para que o seu campo de visão não abrangesse as estruturas e materiais observáveis no teto e apenas incluísse o circuito delimitado pelas paredes brancas. Como no primeiro cenário, em cada canto o veículo roda 90º;
iv. O quarto trajeto efetuado é semelhante ao segundo, na medida em que a orientação mantém-se constante ao longo do trajeto, mas com a diferença de que o ROV foi transportado à mão com objetivo de variar a altura do movimento. Ao contrário do trajeto anterior, não houve qualquer restrição ao campo de visão da câmara.

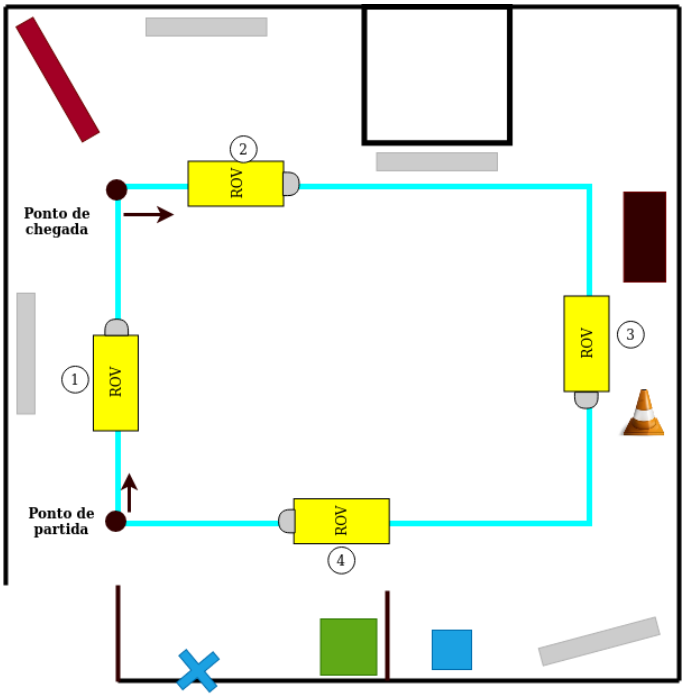


Figura 4.13: Representação do primeiro percurso efetuado. As setas colocadas junto dos pontos de partida e chegada, indicam a orientação inicial e final do veículo, respetivamente. Os números indicam a sequência de posições ao longo do trajeto.

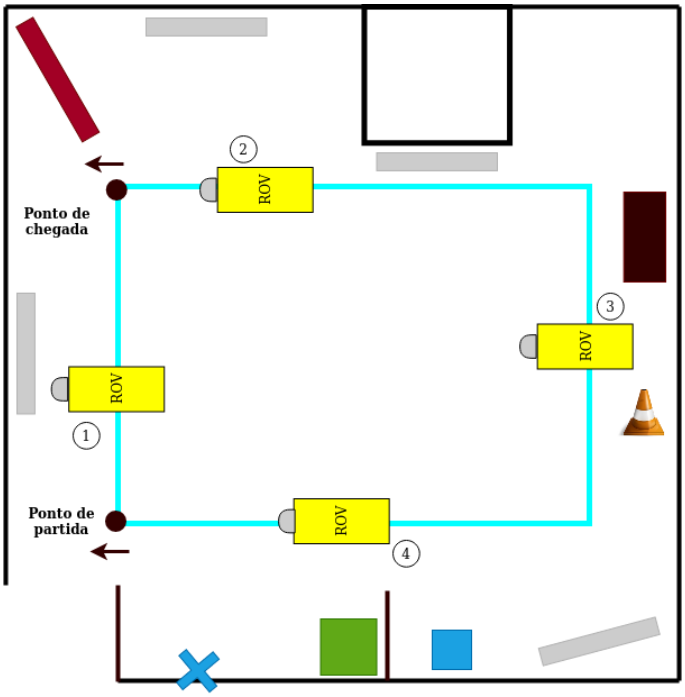


Figura 4.14: Representação do segundo percurso efetuado, em que a orientação é constante.

CAPÍTULO 5

Resultados

Neste capítulo são apresentados os resultados obtidos nas experiências definidas no capítulo 4.

Através da observação direta dos resultados disponibilizados sob a forma de uma trajetória não é possível obter informação qualitativa acerca da trajetória estimada. Para poder verificar o desempenho dos diferentes algoritmos avaliados é necessário comparar as trajetórias calculadas com um *ground-truth* para calcular a precisão dos algoritmos de VIO/VO.

As métricas de erro mais utilizadas para calcular o erro das trajetórias estimadas são o erro absoluto da trajetória (ATE) e o erro relativo da pose (RPE) através da medida do RMSE, como sugerido por Sturm, Engelhard, Endres et al. [126].

Considere-se uma trajetória estimada constituída por um conjunto de poses $E_1, \ldots, E_n \in \mathbb{SE}(3)$ e um *ground-truth* constituído pelas poses $G_1, \ldots, G_n \in \mathbb{SE}(3)$. Assume-se que o número de poses em ambas as sequências é igual e que as sequências estão sincronizadas.

O erro absoluto mede a consistência global da trajetória através da comparação da distância absoluta entre duas trajetórias. Como a trajetória estimada e o *ground-truth* podem estar expressos em sistemas de coordenadas diferentes, é necessário proceder ao alinhamento da trajetória. Para este alinhamento foi utilizado o método de mínimos quadrados segundo a abordagem proposta por Umeyama [127], a qual permite calcular uma transformação $\mathbf{T}$ entre a trajetória estimada $\mathbf{E}$ e o *ground-truth* $\mathbf{G}$ que minimiza os erros quadráticos. Além disto, é possível estimar a escala que minimiza o erro da transformação, o que é útil para analisar o erro de escala nos métodos de odometria visual-inercial. A cada instante de tempo i, o erro absoluto é dado por:

$$A_i = G_i^{-1} T E_i \in \mathbb{SE}(3) \tag{5.1}$$

Para a sequência de erros absolutos em cada instante de tempo i, A_i, o cálculo do RMSE permite calcular o ATE de translação, expresso em metros:

$$ATE = \sqrt{\frac{\sum_{i=1}^{n} \|t_{A_i}\|^2}{n}} \tag{5.2}$$

onde $t_{A_i} = [A_{i_3}, A_{i_7}, A_{i_{11}}]^T \in \mathbb{R}^{3x1}$ corresponde ao vetor do erro de translação da matriz $A_i \in \mathbb{SE}(3)$ no instante de tempo i e $A_{i_3}, A_{i_7}, A_{i_{11}}$ são os elementos dessa matriz.

Relativamente ao erro relativo, esta métrica serve para avaliar a consistência local da trajetória estimada num intervalo fixo Δ, dividindo a trajetória em subtrajetórias, o que permite calcular o *drift* da trajetória. Em cada instante de tempo i, o erro relativo é definido por:

$$R_i = (G_i^{-1} G_{i+\Delta})^{-1} (E_i^{-1} E_{i+\Delta}) \tag{5.3}$$

Tal como no cálculo do ATE, é possível calcular o RMSE sobre a sequência de erros relativos R_i. Utilizando os erros de translação, o erro relativo da pose é definido, em metros, por:

$$RPE = \sqrt{\frac{\sum_{i=1}^{n} \|t_{R_i}\|^2}{n}} \tag{5.4}$$

onde $t_{R_i} = [R_{i_3}, R_{i_7}, R_{i_{11}}]^T \in \mathbb{R}^{3x1}$ corresponde ao vetor do erro de translação da matriz $R_i \in \mathbb{SE}(3)$ no instante de tempo i e $R_{i_3}, R_{i_7}, R_{i_1 1}$ são os elementos dessa matriz.

Pelo facto de os algoritmos de VO estudados devolverem a trajetória final sobre um conjunto disperso de *keyframes*, o erro relativo será pouco expressivo. Por este motivo, foi apenas utilizado a métrica ATE (5.2) das trajetórias como métrica de avaliação da precisão dos algoritmos por ser invariante ao tamanho da trajetória e resolução da mesma.

A ferramenta utilizada para a representação gráfica e avaliação das trajetórias foi o «evo», disponibilizado por Grupp [128].

Com o objetivo de maximizar o desempenho dos algoritmos utilizados, foram alterados alguns parâmetros de configuração predefinidos. Estas modificações efetuadas consistiram, principalmente, no aumento da informação visual utilizada e no aumento do tempo de processamento dedicado a tarefas que exigem mais processamento, como a otimização.

- MAPLAB
  i. O número máximo de iterações do IEKF foi aumentado de 20 para 40;
  ii. O número de píxeis mínimo entre os quais as *landmarks* devem estar separadas foi diminuído de 100 para 50 píxeis. O objetivo desta alteração é manter um número mínimo de *landmarks* quando as regiões de interesse estão localizada apenas numa determinada área da imagem.
- OKVIS
  i. O número de *keyframes* incluídas na janela de otimização foi aumentado de 5 para 10;
  ii. *threshold* de deteção do BRISK foi diminuído de 40 para 25.
- VINS-Mono
  i. O número máximo de caraterísticas extraídas foi aumentado de 150 para 300
  ii. O limite temporal máximo dedicado à otimização foi aumentado de 40ms para 200ms e o número de iterações aumentado de 8 para 12.

- ORB-SLAM/ORB-SLAM3
  i. O número máximo de caraterísticas extraídas foi aumentado de 1000 para 4000;
  ii. O *threshold* de deteção foi diminuído de 20 para 10.
- DSO
  i. O número de píxeis incluídos no *tracking* foi aumentado de 1000 para 4000;
  ii. O número de *keyframes* incluídas na janela de otimização foi aumentado de 7 para 10.

As modificações foram aplicadas sobre a sequência n.º 1 do *dataset* em estudo, em que são comparados os desempenhos dos algoritmos com as configurações predefinida e modificada, como é ilustrado na figura 5.1. A melhoria mais significativa é da parte do DSO que, ainda assim, apresenta uma trajetória com um erro elevado. O desempenho dos restantes algoritmos também melhora apesar de não ser tão significativo como no DSO. Como o tempo de processamento não é um problema que se coloca nesta dissertação, a avaliação dos algoritmos utilizou esta versão *tuned* dos parâmetros.

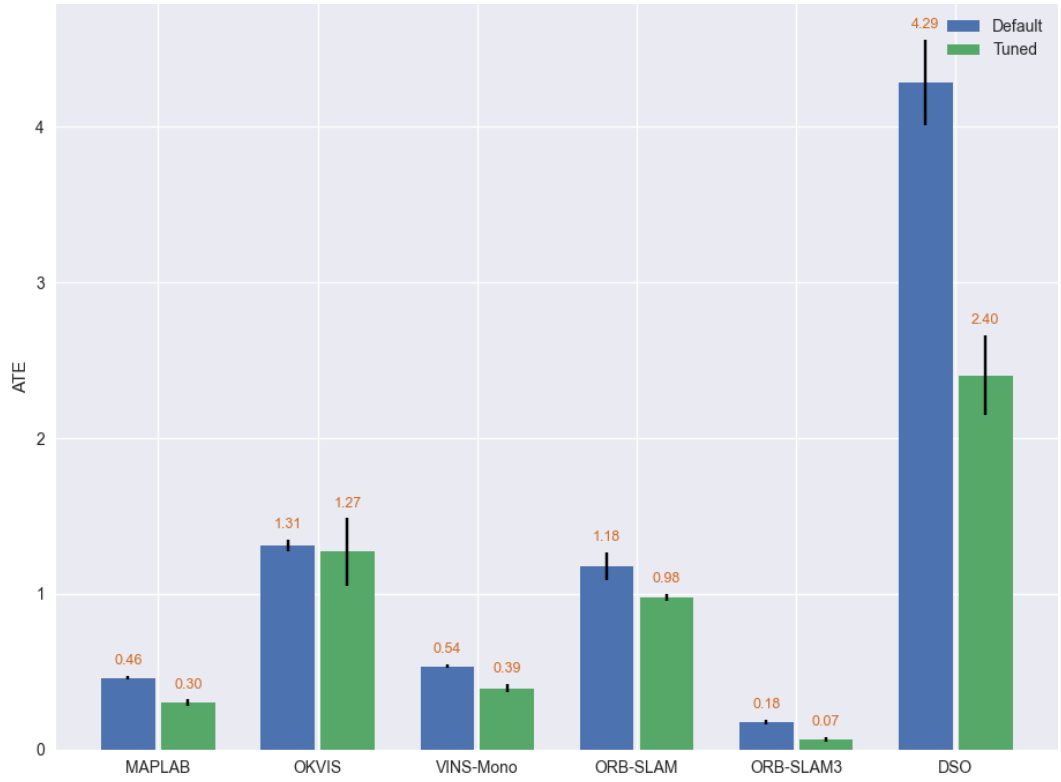


Figura 5.1: Erro das trajetórias estimadas pelos diferentes algoritmos com duas configurações diferentes.

5.1 Resultados no *dataset* AQUALOC

Os resultados apresentados sobre o *dataset* do AQUALOC são relativos ao grupo de sequências gravadas no *Harbor Site*, descritas na tabela 5.1, a poucos metros de profundidade, perfazendo um total de sete sequências gravadas e testadas com os métodos anteriormente citados.

Nº. da Sequência	Duração	Distância Percorrida
1	3'49"	39.3m
2	6'47"	75.6m
3	4'17"	23.6m
4	3'26"	55.8m
5	2'52"	28.5m
6	2'06"	19.5m
7	1'53"	32.9m

Tabela 5.1: Detalhes das sequências avaliadas.

Para cada sequência, os algoritmos foram executados três vezes, pelo que os valores apresentados dizem respeito à média do erro sobre três execuções. A figura 5.2 representa o mapa de calor de todas as execuções dos algoritmos estudados em função da métrica do ATE (5.2). No anexo deste documento é apresentado um conjunto de gráficos nos quais estão representadas as trajetórias com o menor ATE juntamente com o *ground-truth*.

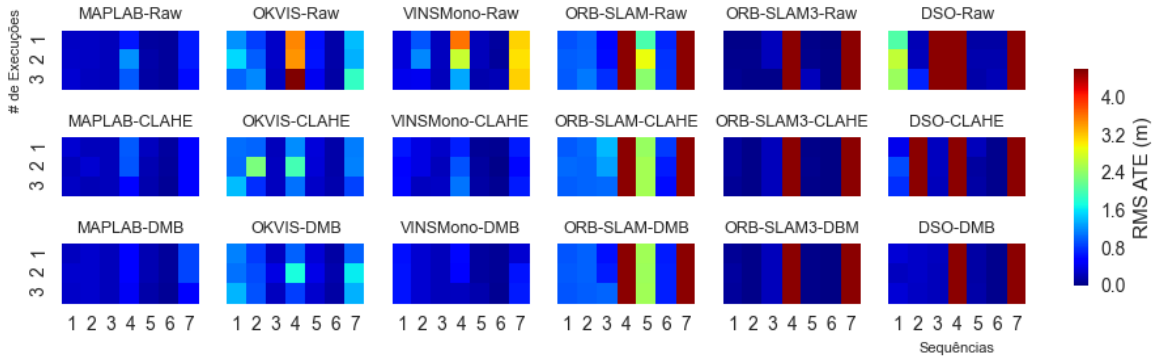


Figura 5.2: Resultados totais registados. Cada quadrado colorido representa o ATE para as três execuções de cada sequência avaliada do *dataset* AQUALOC.

De seguida são apresentados os resultados registados para o *dataset*. A figura 5.3 contém os resultados para cada sequência com as imagens originais, onde não foi aplicado nenhum processamento de imagem. Além disto, também contém a classificação média dos 6 algoritmos estudados ao longo das 7 sequências, ou seja, para cada sequência os algoritmos são ordenados consoante o erro.

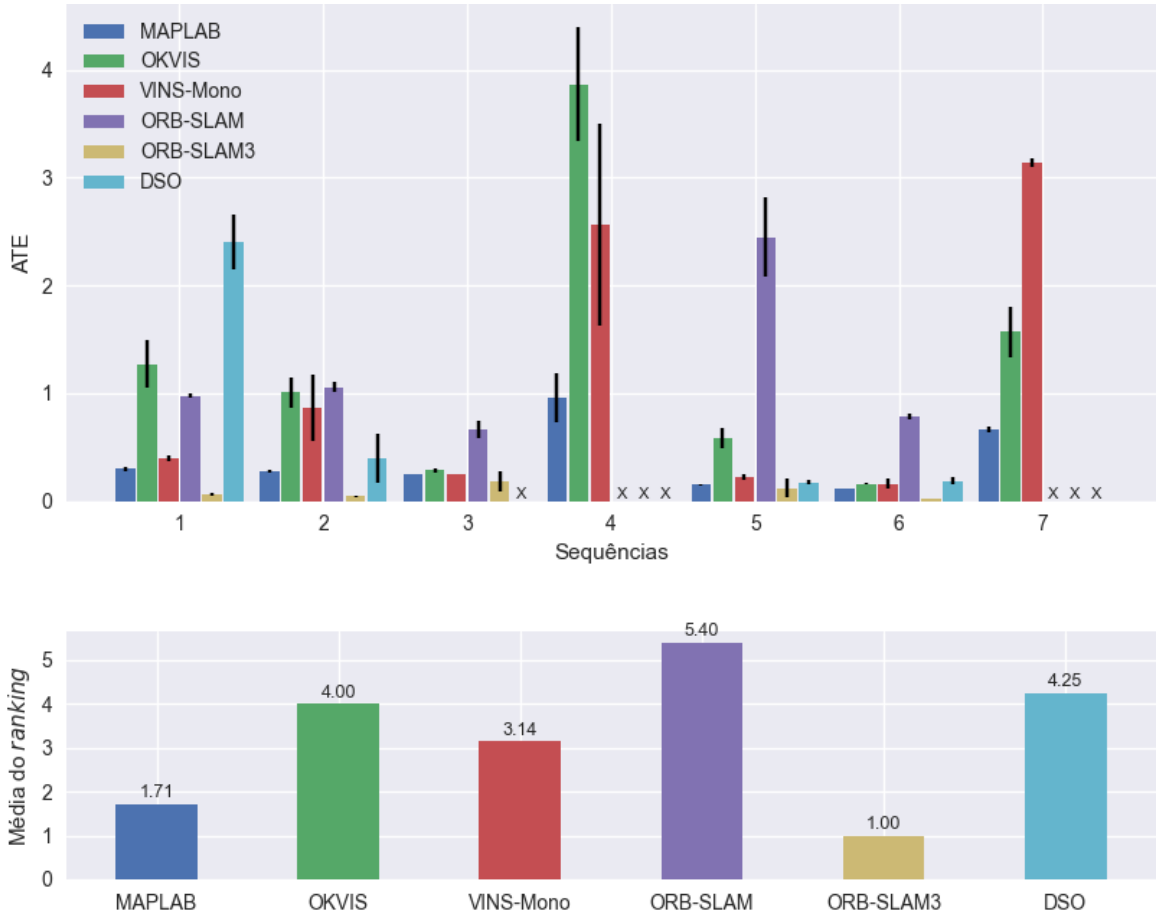


Figura 5.3: Resultados dos algoritmos de VIO no *dataset* do AQUALOC. No gráfico de cima, é apresentado o erro em cada uma das sete sequências avaliadas. Um X indica que o algoritmo não conseguiu completar essa sequência. No gráfico em baixo é apresentada a classificação média dos algoritmos. Um valor mais próximo de 1 significa que, em média, o algoritmo indicado foi o melhor entre os 6. Note-se que os algoritmos de VO não completam todas as sequências, pelo que a sua média correspondente engloba apenas as sequências que completam.

As figuras 5.4 e 5.5 apresentam os resultados após processamento da imagem, onde é apresentada a variação do erro. Esta variação corresponde à subtração do valor médio obtido entre as 3 execuções para imagens não processadas, com o valor médio entre as 3 execuções para as imagens processadas:

$$\Delta_{ATE} = \overline{ATE}_{raw} - \overline{ATE}_{proc} \pm \sqrt{\sigma^2_{ATE_{raw}} + \sigma^2_{ATE_{proc}}} \tag{5.5}$$

onde o sufixo *raw* indica as imagens originais, o sufixo *proc* indica as imagens processadas e σ representa o desvio padrão entre as 3 execuções. Um valor de Δ_{ATE} negativo indica que o erro diminuiu com imagens processadas, ao passo que um valor positivo indica que o erro aumentou. Além disto, também é apresentada a classificação média dos algoritmos consoante a variação do erro entre as 7 sequências. Isto significa que, para cada sequência, os algoritmos são ordenados de forma crescente de acordo com a variação apresentada. Numa

sequência, o algoritmo ao qual é atribuído o 1º lugar significa que foi o que mais beneficiou do processamento de imagem.

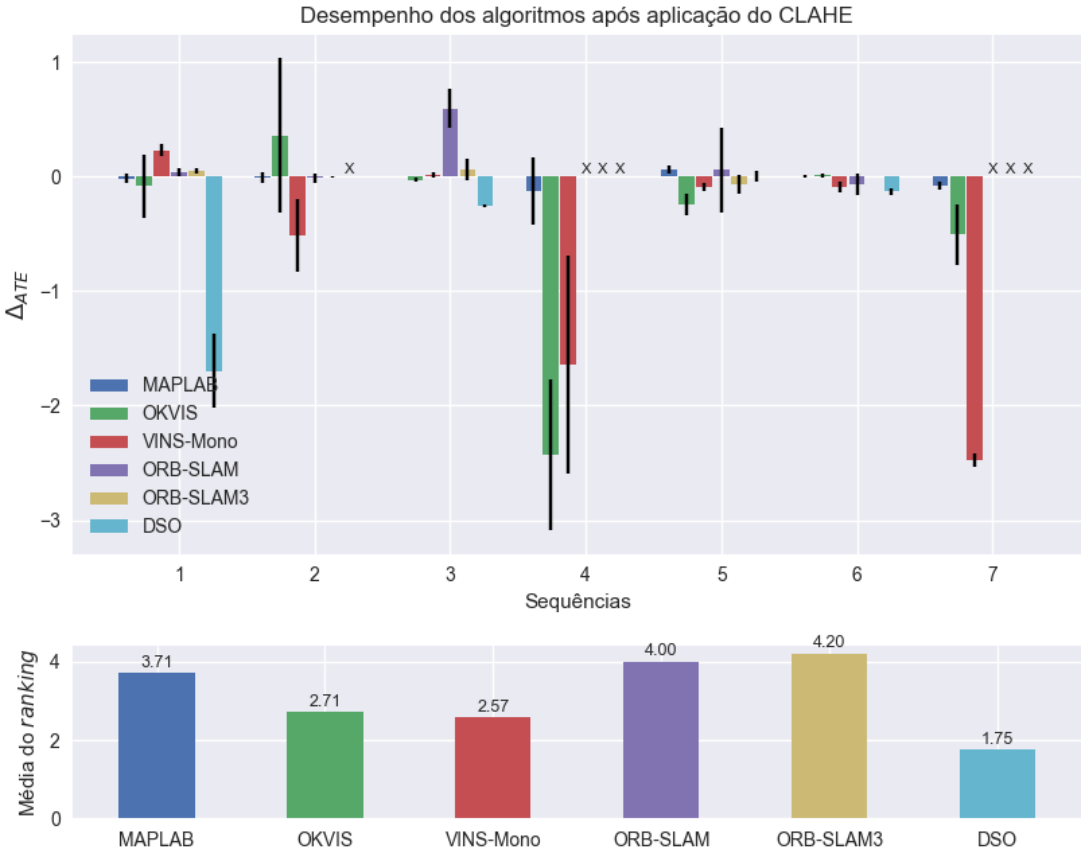


Figura 5.4: Resultados dos algoritmos *dataset* do AQUALOC com as imagens pré-processadas com o CLAHE. O gráfico de cima representa a variação do erro em cada uma das sequências. Uma variação negativa indica que o erro diminiu na quantidade indicada, ao passo que uma variação positiva mostra que o erro aumentou. No gráfico em baixo é apresentada a classificação média dos algoritmos entre as 7 sequências. Um valor mais próximo de 1, significa que, em média, o algoritmo indicado foi o que mais beneficiou do processamento de imagem por CLAHE.

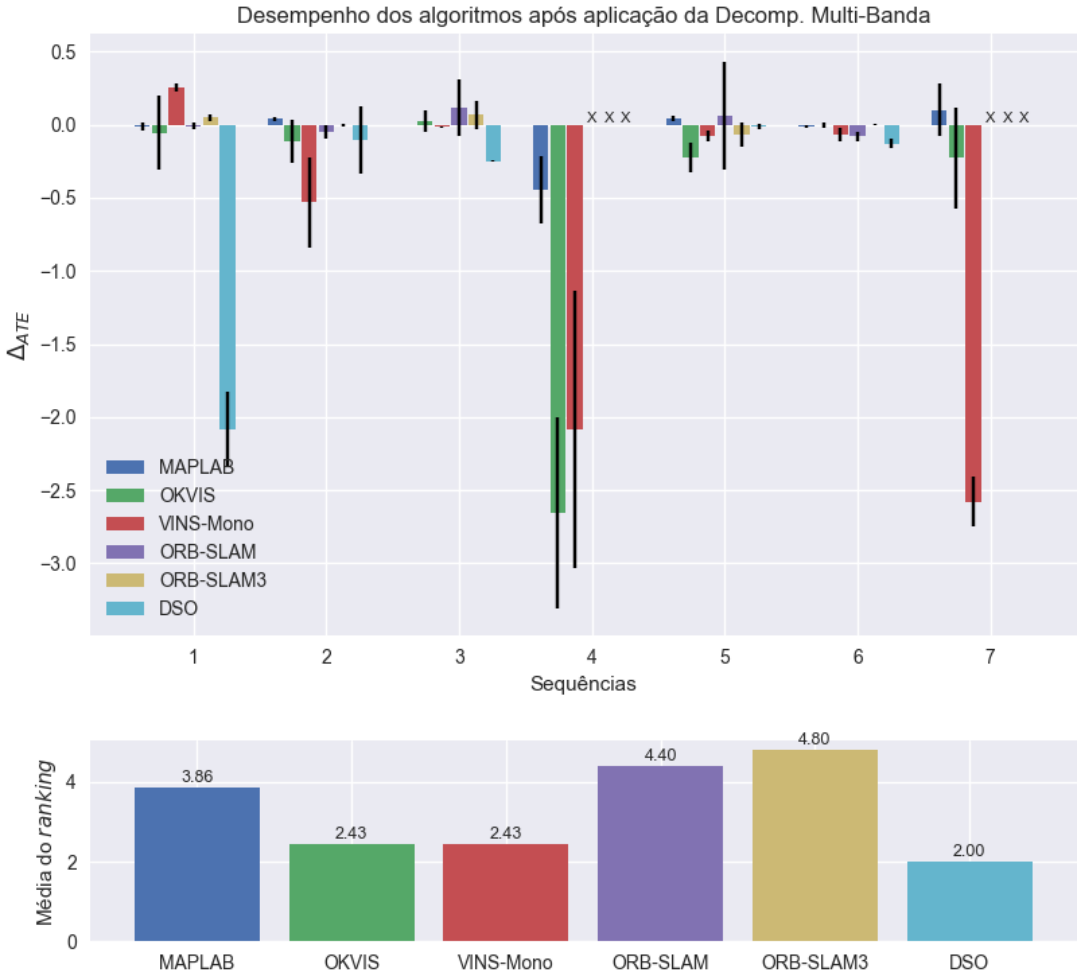


Figura 5.5: Resultados dos algoritmos no *dataset* do AQUALOC com as imagens pré-processadas com a técnica de Decomposição Multi-Banda.

Para além do valor de erro, também é necessário quantificar a escala das trajetórias estimadas pelos algoritmos. Como não é possível obter a escala da trajetória estimada somente através das imagens capturadas por um sistema de visão monocular, na figura 5.6 as escalas apresentadas apenas abrangem o conjunto de algoritmos de VIO avaliados, MAPLAB, OKVIS e VINS-Mono.

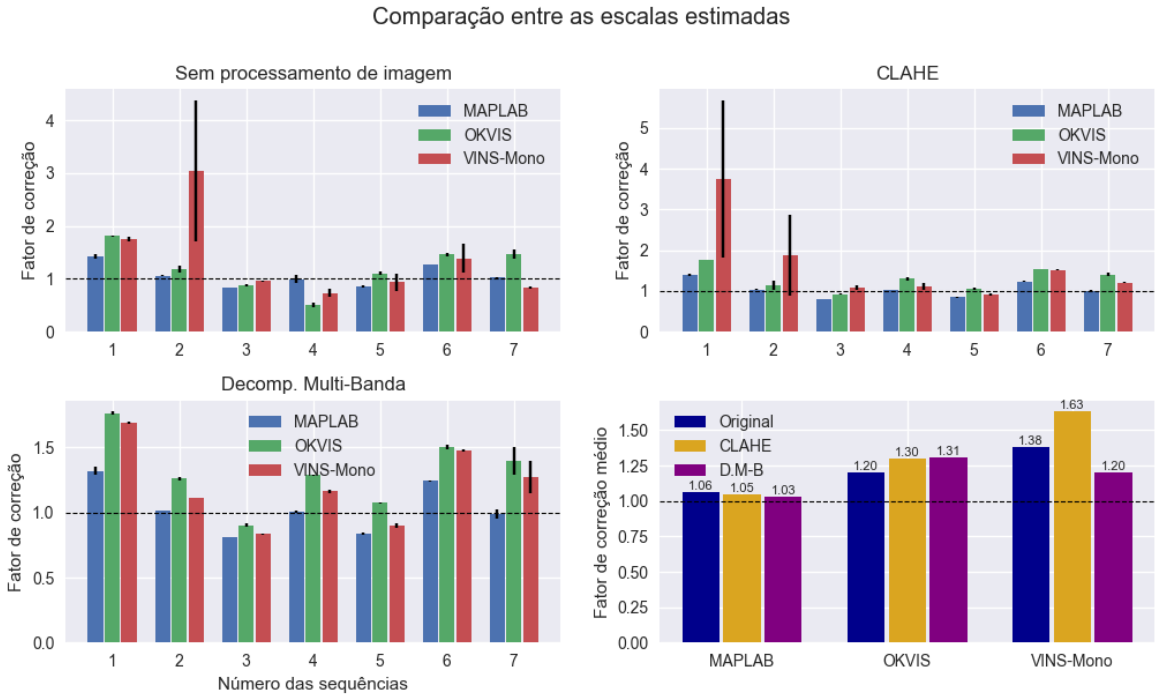


Figura 5.6: Fatores de correção da escala da trajetória estimada para os diferentes casos em estudo: sem pré-processamento da imagem e após o processamento da imagem com as técnicas utilizadas. O último gráfico indica a média do fator de correção sobre as sete sequências. A linha horizontal tracejada com valor igual a 1 indica o fator de correção ideal. Um valor superior a 1 indica que a escala da trajetória foi subestimada, enquanto que um valor inferior indica que foi sobreestimada.

Análise

Os resultados apresentados foram executados num portátil equipado com um processador Intel Core™ i5 (4 núcleos/8 *threads* @2.4GHz / 8GB RAM) sem restrições de funcionamento em tempo real.

Com base nos resultados obtidos, ao longo das sete sequências verifica-se que no cômputo geral o algoritmo de VIO com melhor desempenho a nível de precisão é o MAPLAB, suportado pela versão do ROVIO com o IEKF. Tal pode dever-se aos critérios robustos de seleção das *landmarks* e do *tracking* ao longo de vários níveis dos *patches* minimizando o erro fotométrico. Apesar de a ideia geral apresentada na literatura ser a de que os métodos baseados em filtragem têm uma menor precisão que os métodos de otimização, pelo menos a utilização de um IEKF parece indicar que este pode ser tão eficaz como as técnicas de otimização.

O VINS-Mono é o segundo melhor método de VIO em termos de erro nestas sequências, revelando um pior desempenho nas sequências em que ocorrem oclusões da câmara (sequências 4 e 7), devido ao embate do veículo com uma superfície, e na sequência mais longa (tabela5.1). O *tracking* no VINS-Mono utiliza o KLT e, portanto, é minimizado o erro fotométrico entre os cantos detetados. A principal diferença para o MAPLAB pode estar relacionada com a possibilidade de no VINS-Mono haver um número mais elevado de *outliers* presente no processo de *tracking*, ao passo que no MAPLAB existe um controlo de *landmarks* utilizadas. Outro resultado que se verificou é que o VINS-Mono não deteta nenhum *loop*, o que indica

que o vocabulário de *bag of words* treinado neste algoritmo não está preparado para este tipo de ambientes.
O OKVIS apresentou um bom desempenho nas sequências mais curtas (3, 5 e 6), mas nas restantes apresenta um valor de erro superior a 1m. O processo de *tracking*, através da extração e associação de caraterísticas BRISK, é bastante afetado pela turbidez da água, pela diminuição da iluminação e por áreas com pouca textura, o que levou a um pior desempenho do algoritmo.

Relativamente às escalas estimadas pelos algoritmos VIO ilustradas na figura 5.6, o mais consistente volta a ser o MAPLAB. O OKVIS e o VINS-Mono na maior parte das sequências fazem uma boa estimação, embora algo subestimadas. No VINS-Mono, existem certas sequências em que o fator de correção é elevado, possivelmente devido ao seu processo de inicialização menos preciso, através de uma abordagem *loosely-coupled*. Desta forma, uma inicialização menos robusta pode levar a que em determinadas sequências a estimação da escala seja afetada, como é visível pelo desvio padrão da escala estimada na sequência 2 sem processamento de imagem e na sequência 1 com pré-processamento por CLAHE.

Sobre o efeito do pré-processamento da imagem nas trajetórias estimadas pelos algoritmos de VIO, ilustrado nas figuras 5.4 e 5.5, verifica-se que tem um efeito bastante positivo principalmente no OKVIS e no VINS-Mono. Principalmente nas sequências mais difíceis, onde existe uma colisão e consequente ausência de informação visual, o pré-processamento das imagens permitiu a deteção de um maior número de *features* e estabilizar o processo de *tracking* mais rapidamente, evitando que os erros de integração da IMU fossem mais salientes, como aconteceu quando se utilizou a imagem original. O MAPLAB, por sua vez, provavelmente por impor um número máximo de *landmarks* e fazer uma seleção das mesmas, não beneficiou tanto das técnicas utilizada, apesar de o erro ter diminuído para cerca de metade na sequência 4 quando se utilizou uma imagem pré-processada pelo algoritmo de decomposição multibanda. Entre as duas técnicas, a decomposição multibanda é a que levou a uma maior diminuição do erro, mas o efeito de ambas as técnicas nas escalas estimadas pelos algoritmos não parece ser significativo. Ainda assim, quando a imagem foi pré-processada com o algoritmo de decomposição multibanda, as escalas estimadas pelo MAPLAB, OKVIS e VINS-Mono foram mais constantes ao longo das sequências, não se registando um valor demasiado elevado ou desvio padrão elevado, ao contrário das imagens original e pré-processada com o CLAHE.

Em relação aos métodos de VO, o que teve um melhor desempenho foi claramente o ORB-SLAM3. No entanto, é necessário ressalvar que tanto o ORB-SLAM como o ORB-SLAM3 são algoritmos de SLAM e, portanto, guardam um mapa global do trajeto e efetuam *loop-closures*. O DSO, apesar de ser um algoritmo puro de odometria visual, em certas sequências tem um desempenho bastante satisfatório. O desempenho superior do ORB-SLAM3 relativamente ao ORB-SLAM, pode estar relacionada com o facto de no ORB-SLAM3 poder ter suporte para câmaras *fisheye*, enquanto que o ORB-SLAM apenas utiliza câmaras *pinhole*, o que no ORB-SLAM3 permite que haja um maior número de associações com o mapa criado. Verificou-se que os três algoritmos falham nas sequências 4 e 7, onde ocorre uma colisão do veículo subaquático com uma rocha, devido à perda de informação visual que resulta da

colisão. Os algoritmos de VIO compensam esta ausência de informação visual com as leituras da IMU, pelo que são mais robustos nestas situações.

Os percursos efetuados pelo veículo utilizado para capturar o *dataset* cruzam-se várias vezes. Contudo, observou-se que em algumas sequências não são detetados os *loops* por parte do ORB-SLAM e do ORB-SLAM3. Nas sequências avaliadas, os *loops* que ambas as versões do ORB-SLAM detetam ocorrem apenas no final de cada sequência, onde existe a presença de uma *AprilTag* que serve como uma marca para o ponto de partida e de chegada. O ORBSLAM3 apenas não deteta esta marca na sequência 2, ao passo que o ORBSLAM não deteta nas sequências 2, 5 e 6. Tal como o VINS-Mono, também o ORBSLAM e ORBSLAM3 assentam o reconhecimento de um lugar em *bag of words*. Apesar de o vocabulário no ORBSLAM3 aparentar estar melhor treinado, ainda não consegue reconhecer os *loops* existentes a meio do percurso, por não serem zonas com caraterísticas distintas.

O efeito do pré-processamento de imagem nas trajetórias estimadas pelos algoritmos de VO é notório apenas no DSO. O erro das trajetórias estimadas pelo DSO diminuíram e o pré-processamento permitiu que o DSO completasse a sequência 3. No entanto, com a aplicação do CLAHE o DSO não conseguiu terminar a sequência 2. Relativamente às duas versões do ORB-SLAM, a diferença é residual na maior parte das sequências, exceto a execução do ORB-SLAM na sequência 3 com imagem processada por CLAHE. Verificou-se ainda que com a aplicação das duas técnicas de pré-processamento da imagem, o ORB-SLAM3 apenas deteta um *loop* na sequência 5 com ambas as técnicas, ao passo que o ORB-SLAM deteta apenas nas sequências 2 e 3 após aplicação da técnica de decomposição multibanda. Isto indica que o aumento de ruído na imagem, inerente a estas técnicas de pré-processamento, prejudica a abordagem de reconhecimento de um lugar por *bag of words*. Apesar disto, o erro não aumentou significativamente, e inclusivé, na sequência 5, onde o ORB-SLAM3 deteta o *loop* independentemente da imagem original ou processada, o erro diminuiu.

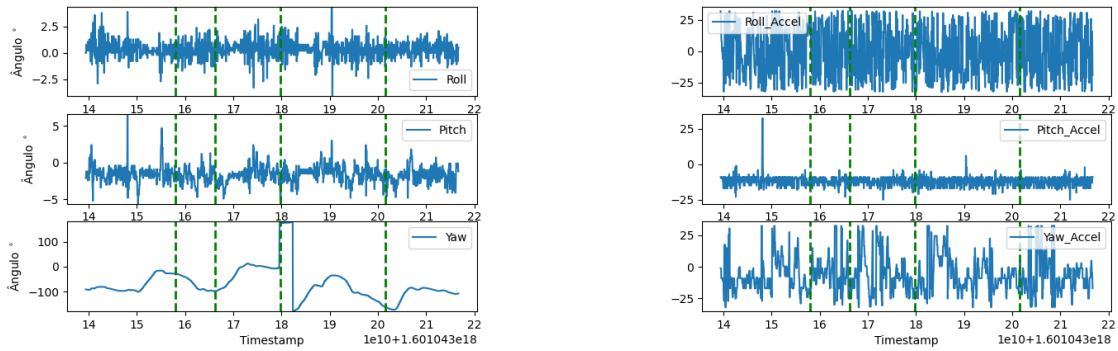
5.2 Resultados obtidos com o Pro4 ROV

A avaliação do desempenho dos algoritmos com as sequências gravadas com o Pro4 ROV cingiu-se apenas ao grupo dos algoritmos de VO. Tal deveu-se ao facto de que a leitura dos dados da IMU do ROV não foi efetuada com sucesso pelas razões a seguir apresentadas.

O protocolo de comunicação do VideoRay Pro4 ROV utiliza uma arquitetura *Control and Status Register* (CSR) entre o *host* (e.g. computador) e o Pro4 ROV, na qual a informação obtida por diferentes sensores está localizada num mapa de memória. O *host* envia um pacote de controlo para aceder a um dispositivo, ao qual o veículo responde. Estes dispositivos podem ser, por exemplo, as luzes do veículo, a câmara ou os dados lidos pelos diversos sensores que fazem parte do veículo. A comunicação utiliza o barramento físico RS-485, e é *half-duplex*.

A VideoRay fornece uma documentação que contém o mapa de memória do Pro4 ROV. Porém, esta documentação não está totalmente correta pois alguns registos a que se tem acesso atualmente são diferentes daqueles que estão referidos. Além disto, de acordo com a documentação, verificou-se que não existe um acesso aos dados *raw* do giroscópio do Pro4

ROV. Os dados fornecidos são os das acelerações angulares, os quais foram analisados em diversos testes e concluiu-se que não estavam corretos.



(a) **(b)**

Figura 5.7: Valores dos ângulos *roll*, *pitch* e *yaw* e das acelerações angulares registados no primeiro percurso do segundo circuito. No gráfico em a) existe alguma flutuação de valores do *yaw* entre determinados instantes de tempo devido à presença de anomalias magnéticas no local onde foi realizado o trajeto. O gráfico em b) não apresenta unidades no eixo vertical pois na documentação da Videoray não existe informação sobre as unidades destes valores. As linhas verde a tracejado correspondem, aproximadamente, aos instantes de tempo em que o Pro4 ROV se preparava para mudar de direção.

No conjunto de gráficos da figura 5.7 são expostos os valores da orientação do veículo, segundo os ângulos de Euler, e os valores das acelerações angulares registados no primeiro trajeto, que consistiu num trajeto com 4 mudanças de direção com o veículo estabilizado numa cadeira rolante. Como se verifica pelo gráfico a) e conforme o esperado, as mudanças de orientação consistiram apenas em mudanças de ângulo *yaw*. No entanto, pelo gráfico b), os valores de *roll acceleration* parecem estar corrompidos por ruído.

Como outros dados a que se tem atualmente acesso estão localizados em registos contíguos e são lidos corretamente, pensa-se que a causa do erro pode estar relacionada com uma falha física do sensor. Ainda assim, a causa do erro continua desconhecida.

Assim, os resultados apresentados nas seguintes secções dizem respeito aos testes efetuados com o Pro4 ROV nos dois circuitos propostos na secção 4.5.4, e ilustrados nas figuras 4.10 e 4.11.

Como não existe um *ground-truth* devidamente marcado com a *timestamp* associada à posição e à orientação do veículo, não é possível fazer o alinhamento das trajetórias. Os algoritmos testados consideram que o eixo Z aponta para a frente, o eixo X aponta para a direita. No ORB-SLAM e no ORB-SLAM3 o eixo Y aponta para baixo e no DSO o eixo Y aponta para cima. Em todo o caso, a primeira pose corresponde à posição (0,0,0) nos algoritmos avaliados.

Circuito 1: Rampa

Com o objetivo de avaliar o desempenho dos algoritmos consoante os três tipos de imagem fornecidos, os algoritmos foram executados 5 vezes cada tipo de imagem. O primeiro tipo

consiste na imagem original, onde existe o efeito de *interlacing*. Os segundo e terceiro tipos são as imagens desentrelaçadas pelas técnicas de duplicação de linhas pares, *doubling*, e média entre linhas pares, *blending*.

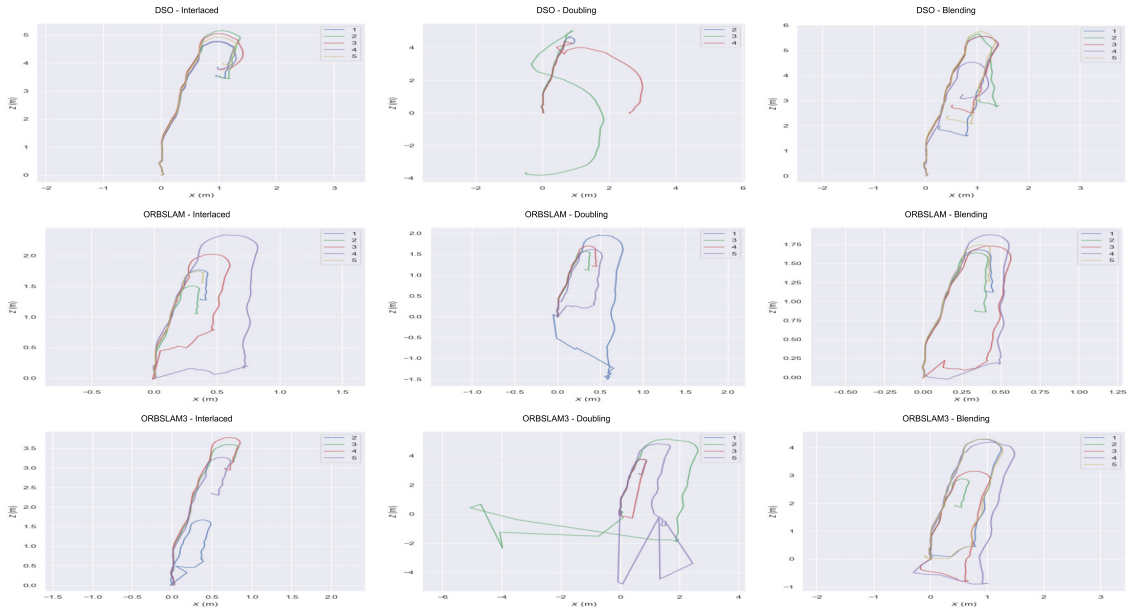


Figura 5.8: Representação das trajetórias estimadas no plano XZ, com imagens entrelaçadas e desentrelaçadas.

De acordo com as trajetórias apresentadas na figura 5.8, observou-se uma diferença de desempenho notável quando o desentrelaçamento foi efetuado pela técnica de duplicação das linhas pares. Esta queda de desempenho pode dever-se ao facto de que a duplicação de linhas cria artefactos visuais quando a câmara está em movimento. Como é visível na figura 4.6, esta técnica provoca na imagem um efeito de *combing*, no qual as bordas parecem recortadas, o que pode levar a um processo de *tracking* pouco eficaz. Assim, nenhum dos algoritmos completou as 5 execuções com sucesso ou, no caso do DSO e do ORB-SLAM3, estimam trajetórias com uma forma que não corresponde ao movimento efetuado.

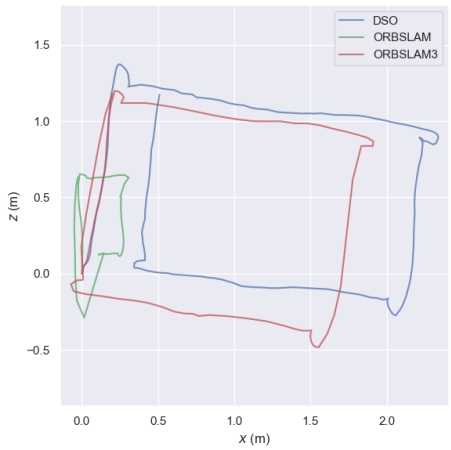
Relativamente aos restantes dois tipos de imagem, verificou-se que com a imagem desentrelaçada pela técnica de *blending* todas as execuções terminaram com sucesso, enquanto que com a imagem original apenas o ORB-SLAM3 perdeu o *tracking* numa das execuções. No caso do DSO com a imagem entrelaçada (DSO-*Interlaced*), observa-se que o efeito de *scale drift* ocorre logo após a rotação do veículo e de forma mais severa que no caso da imagem desentrelaçada por *blending*.

Entre o ORB-SLAM com a imagem original e a imagem após *blending* não existiram grandes diferenças, visto que as trajetórias são semelhantes e em ambos os casos foi detetado o mesmo número de *loops*. A diferença entre a aplicação destes dois tipos de imagem está mais visível nas trajetórias estimadas pelo ORB-SLAM3. Com a imagem original, o ORB-SLAM3 detetou apenas 1 *loop*, enquanto que com a imagem desentrelaçada apenas não detetou o *loop* em uma das execuções.

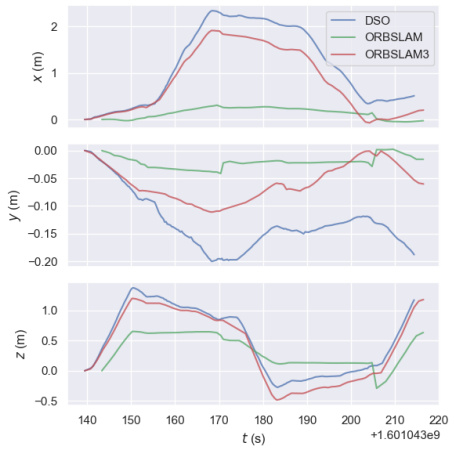
Conforme os resultados obtidos, o resto da avaliação foi feita com imagens desentrelaçadas pela técnica de *blending*.

De acordo com o segundo circuito proposto em 4.5.4, são apresentados os resultados das estimações das trajetórias referentes aos percursos definidos nas figuras 4.13 e 4.14.

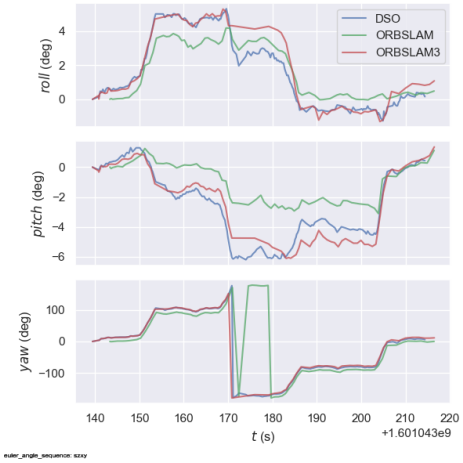
Caminho 1: 4 mudanças de orientação com altura constante



(a) Representação das trajetórias estimadas no plano XZ.



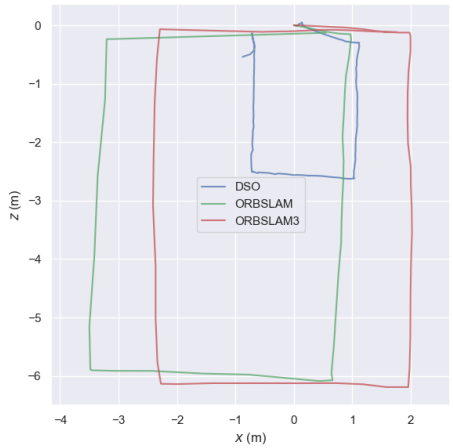
(b) Representação das trajetórias estimadas segundo as coordenadas X, Y e Z.



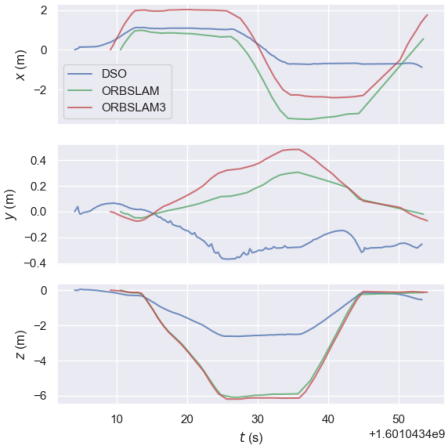
(c) Representação da evolução da orientação do veículo ao longo do percurso.

Figura 5.9: Trajetórias e orientação estimadas pelos algoritmos de VO no primeiro caminho percorrido.

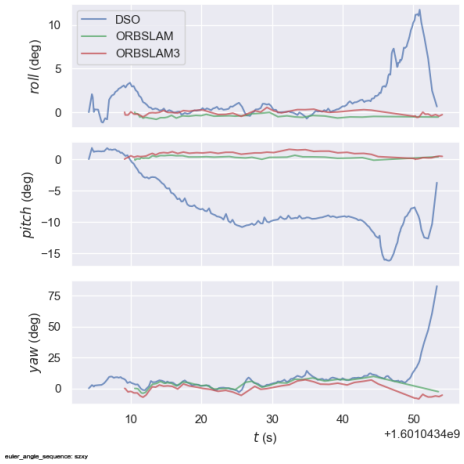
Caminho 2: Altura constante sem mudança de orientação



(a) Representação das trajetórias estimadas no plano XZ.



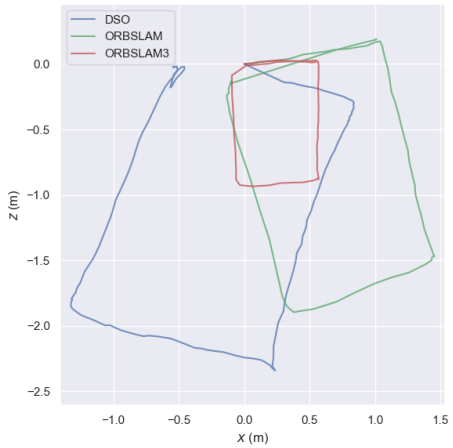
(b) Representação das trajetórias estimadas segundo as coordenadas X, Y e Z.



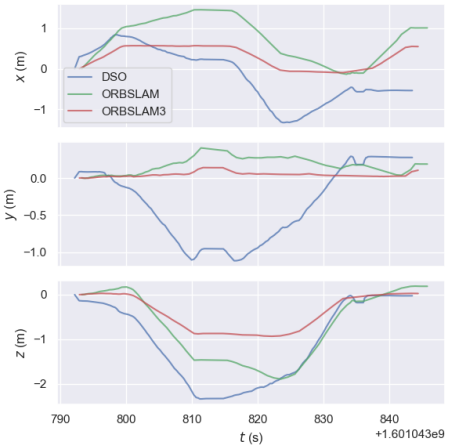
(c) Representação da evolução da orientação do veículo ao longo do percurso.

Figura 5.10: Trajetórias e orientação estimadas pelos algoritmos de VO no segundo caminho percorrido.

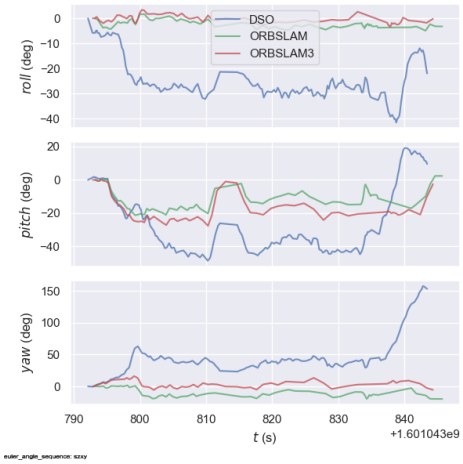
Caminho 4: Altura variável sem mudança de orientação



(a) Representação das trajetórias estimadas no plano XZ.



(b) Representação das trajetórias estimadas segundo as coordenadas X, Y e Z.



(c) Representação da evolução da orientação do veículo ao longo do percurso.

Figura 5.11: Trajetórias e orientação estimadas pelos algoritmos de VO no quarto caminho percorrido.

Análise

De acordo com os resultados no caminho 1, ilustrados na figura 5.9, verifica-se que o ORB-SLAM não conseguiu estimar um percurso consistente com o que era previsto. Provavelmente tal deve-se ao facto de que este algoritmo apenas opera sobre modelos de câmara *pinhole*, o que diminui o campo de visão. A escala local estimada pelos algoritmos é afetada pelos algoritmos, visto que após cada movimento de rotação ocorre o efeito de *scale drift*. A odometria visual monocular depende de imagens da mesma cena capturadas sob diferentes pontos de vista, pelo que os algoritmos são ineficazes a processar rotações rápidas do veículo. Ainda assim, nota-se que um modelo de câmara apropriado para o caso em estudo permite que o ORB-SLAM3 consiga estimar uma trajetória parecida com a forma de um retângulo. A diferença entre as trajetórias estimadas, antes da deteção do *loop* é visível na figura 5.12.

Figura 5.12: Trajetórias estimadas pelos algoritmos ORB-SLAM (esquerda) e ORB-SLAM3 (direita) antes da deteção do *loop*. As marcas azuis representam a pose da câmara em cada *keyframe*. Em cada rotação os algoritmos necessitam de adicionar um maior número de *keyframes*.

No caso do DSO, observa-se que por não possuir um módulo de reconhecimento ou um mapa global, acumulou erros. Ainda assim, tanto o DSO e como o ORB-SLAM3 estimam uma trajetória ligeiramente distorcida, algo que pode ter sido causado pelo movimento irregular efetuado com a cadeira usada para transportar o veículo na reta inicial do percurso. Como o ORB-SLAM inicializa mais tarde, este movimento irregular não é tão expressivo na sua trajetória. De acordo com a orientação estimada em 5.9c, verifica-se que a variação dos ângulos *roll* e *pitch* é mínima, e que o *yaw* varia cerca de 90º em cada rotação, voltando à orientação inicial no final da trajetória.

No caminho 2, pelos resultados indicados na figura 5.10 verifica-se que sem movimentos puros de rotação, ambas as versões do ORB-SLAM são capazes de estimar uma trajetória e orientações de acordo com o previsto. A diferença entre as trajetórias está na inicialização dos algoritmos, sendo que o ORB-SLAM inicializa mais tarde pelo que a sua trajetória não é tão simétrica quanto a do ORB-SLAM3. No movimento inicial do veículo, a câmara está muito próxima da parede branca, pelo que o ORB-SLAM3, por permitir a utilização do modelo de calibração aplicado na secção 4.5.3, deteta mais rapidamente as *features* necessárias para inicializar o algoritmo. Este comportamento é visível na figura 5.13. O DSO apresenta uma quebra de pontos utilizados no *tracking* na reta final da trajetória, pelo que não consegue fechar o *loop* e acaba por ter um pior desempenho. Em todo o caso, os três algoritmos falham na fase final, como ilustra a figura pelo que a diferença está nos mecanismos de *loop-closure* que o ORB-SLAM e ORB-SLAM3 possuem.

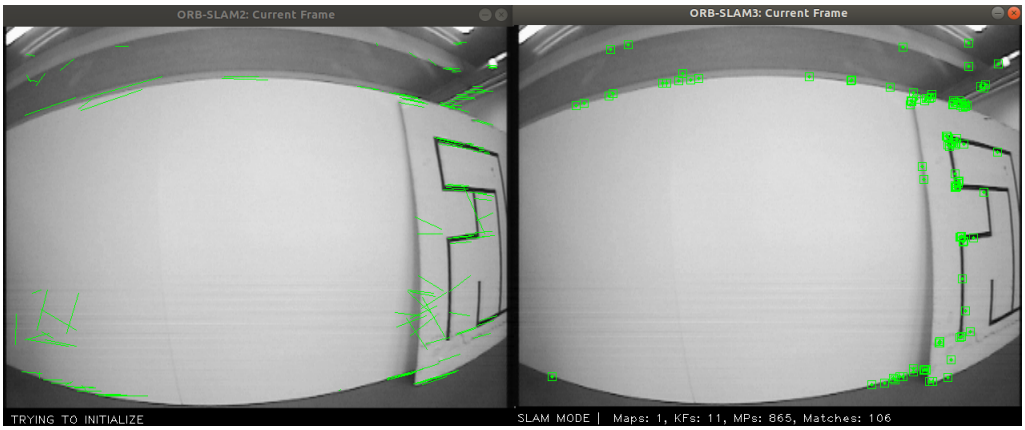


Figura 5.13: Inicialização do ORB-SLAM e do ORB-SLAM3.

Nenhum dos algoritmos conseguiu concluir o caminho 3 com sucesso. Tal deveu-se ao facto de que o campo de visão da câmara era maioritariamente o chão do circuito, que é parco em caraterísticas que se possam extrair, o que torna difícil a execução dos métodos indiretos. Relativamente ao método direto, o DSO, apesar de conseguir detetar zonas de elevado gradiente, perde o *tracking* após o ROV executar uma rotação, provavelmente pela rotação efetuada ter sido demasiado rápida. Esta quebra de *features* está ilustrada na figura.

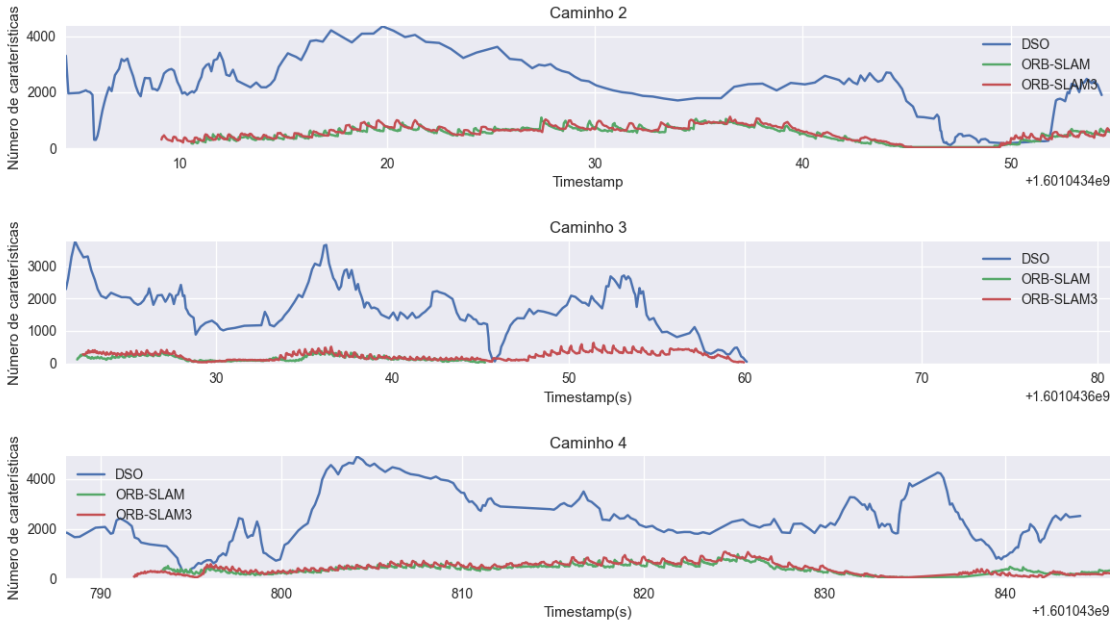


Figura 5.14: Evolução do número de caraterísticas utilizadas para o *tracking* pelos três algoritmos ao longo dos percursos 2, 3 e 4. Note-se que as duas versões do ORB-SLAM utilizam caraterísticas ORB e foi registado o número de *inliers*, isto é, aqueles pontos em que o *matching* foi efetuado com sucesso. O DSO não utiliza caraterísticas, mas sim pontos.

No último trajeto efetuado, considerando a figura 4.14, a altura do veículo foi variada no vértice superior direito e no vértice inferior ao dar a volta ao circuito. A variação da altura consistiu em colocar o ROV no chão e deixar o veículo estático nessa posição durante um pequeno período de tempo, antes de voltar a transportá-lo. Porém, na segunda vez em que se coloca o veículo no chão, a câmara está muito próxima da parede e o os algoritmos falham, de forma semelhante com o que aconteceu na segunda experiência. Na estimação do percurso, ilustrada na figura 5.11, tanto o ORB-SLAM como o ORB-SLAM3 fecham o *loop*, apesar de perderem o *tracking* na segunda vez que se coloca o veículo no chão, ao passo que o DSO diminui bastante o número de pontos extraídos quando o veículo pára pela segunda vez, pelo que não consegue fechar o *loop*. Ainda assim, as trajetórias são conforme o esperado, apesar de as trajetórias estimadas pelo DSO e pelo ORB-SLAM estarem um pouco distorcidas, devido a erros na inicialização dos algoritmos. Verifica-se na figura 5.11b que a altura, representada pelo eixo Y, é constante durante um certo período de tempo, na primeira vez que se coloca o ROV no chão. Em relação à orientação estimada, o *yaw* varia bastante pouco no ORB-SLAM e ORB-SLAM3, assim como o *roll*. A variação em *pitch* deve-se à

irregularidade do movimento ao colocar o ROV no chão. No DSO, os ângulos *yaw* e *pitch* são um pouco elevados desde o ínicio, talvez devido ao seu algoritmo de inicialização que, segundo os seus autores, tem algumas debilidades. No entanto, o ângulo *pitch*, apesar de ser elevado, parece ter uma variação consistente com a estimação efetuada pelos outros dois algoritmos.

CAPÍTULO 6

Conclusão e Trabalho Futuro

6.1 Conclusão

O trabalho desenvolvido nesta dissertação abordou vários tópicos relacionados com a temática da navegação ótica no meio subaquático. Apesar de não ter sido possível desenvolver um sistema de navegação para o veículo utilizado, foi demonstrado através da aplicação de diversos algoritmos do estado da arte num cenário subaquático que a utilização de uma só câmara pode complementar outros sensores e ser uma solução viável debaixo de água. Também foi explorado de que forma a melhoria da visibilidade de uma imagem capturada no meio em estudo pode melhorar o desempenho dos algoritmos estudados. O trabalho desenvolvido com o Pro4 ROV permitiu destacar a importância de um modelo de câmara apropriado para uma câmara *fisheye*, a influência do entrelaçamento das imagens nos algoritmos de odometria visual e as limitações dos mesmos.

Inicialmente, foram revistos os vários tipos de navegação existentes para um veículo que atua no meio em estudo, o que permitiu identificar em que situações a utilização da câmara debaixo de água é exequível. Foi abordado o problema que envolve a localização e mapeamento a partir de uma câmara e as soluções que têm sido apresentadas ao longo dos anos, tanto nos meios terrestre e aéreo, para os quais as diversas soluções têm sido desenvolvidas, como também para o meio subaquático de modo a aferir o que tem sido feito para contornar as dificuldades inerentes ao meio. Esta revisão possibilitou o reconhecimento de que não existe na literatura uma avaliação quantitativa dos métodos de VIO/VO aplicados num meio subaquático e que uma imagem capturada debaixo de água apresenta desafios para a localização a partir de uma só câmara.

Para suprir esta lacuna, foram estudados diferentes algoritmos de odometria visual e visual-inercial, os quais foram aplicados a um *dataset* subaquático. Os resultados apresentados mostram a viabilidade da aplicação de uma solução de localização e mapeamento baseada em visão computacional num veículo subaquático. Nas diferentes sequências do *dataset* sobre o qual foi realizada a avaliação dos algoritmos, os algoritmos de VIO são mais robustos que os de VO, como era expectável. Na categoria dos algoritmos de odometria visual-inercial,

o MAPLAB foi o que teve um melhor desempenho e que mostrou uma maior robustez nas sequências em que existe a perda de informação visual durante a colisão. A robustez é essencial pois em ambientes mais constritos, a operação com um ROV pode levar a choques com as estruturas que rodeiam o veículo. Entre os algoritmos de odometria visual, verifica-se que a grande valia dos métodos de SLAM está no mapa global que é construído e que auxilia à localização do veículo. Além disto, os métodos indiretos possibilitam mais facilmente a implementação de um módulo de reconhecimento que, porém, não estão adaptados ao ambiente em estudo. O DSO, considerando a sua natureza de um algoritmo puro de odometria visual, também conseguiu atingir precisões bastantes elevadas pelo que seria interessante testar a adição de módulos de reconhecimento para relocalização ou *loop-closure* ao algoritmo, assim como a utilização de um mapa global.

Relativamente às duas propostas apresentadas para o pré-processamento de imagem, é possível concluir que melhoram as trajetórias estimadas, com exceção do ORB-SLAM e ORB-SLAM3. Tanto o CLAHE como o algoritmo de decomposição multi-banda utilizado melhoraram, principalmente, a odometria visual efetuada pelo DSO, o que permite que este atinja resultados com erros pouco superiores aos do ORB-SLAM3. A utilização do CLAHE parece ser uma boa aposta, visto que o algoritmo de decomposição multi-banda tem um tempo de processamento que impede a utilização destes algoritmos em tempo real. Outro resultado bastante positivo, é que o pré-processamento de imagem permitiu que os resultados obtidos com o VINS-Mono fossem semelhantes aos que foram obtidos com o MAPLAB. Ainda assim, esta aparente melhoria da visibilidade da imagem não foi suficiente para melhorar o desempenho do módulo de reconhecimento implementados no ORB-SLAM/ORB-SLAM3 e no VINS-Mono, piorando até no caso das duas versões do ORB-SLAM.

Os resultados obtidos a partir do VideoRay Pro4 ROV permitem concluir que a grande vantagem dos métodos indiretos de visão monocular está no módulo de reconhecimento, o qual possibilitou a correção do *scale drift* causado pelas rotações, assim como a correção global da trajetória após relocalização. Além disto, verifica-se que um maior campo de visão permite uma inicialização mais rápida e melhora o processo de estimação da trajetória. Relativamente ao entrelaçamento da imagem, verificou-se que o entrelaçamento das imagens prejudica a estimação do movimento, mas que o desentrelaçamento por duplicação de linhas é ainda mais prejudicial, pelo que a solução da média é uma boa solução.

Apesar das limitações de índole experimental observadas neste trabalho, com base nos resultados obtidos é possível concluir que os métodos baseados em visão apresentam um elevado potencial para resolver o problema da localização subaquática de um veículo. Entre os métodos avaliados, o ORB-SLAM3 e o MAPLAB destacaram-se e o pré-processamento de imagem teve um impacto positivo na tarefa de localização. Também se conclui que o entrelaçamento das imagens prejudica a estimação do movimento, mas que o desentrelaçamento por duplicação de linhas é ainda mais prejudicial. No entanto, ainda é necessário resolver a forma o pré-processamento da imagem pode beneficiar a localização sem afetar a deteção de um lugar previamente visitado, o que permite a otimização da trajetória global e, assim, diminuir o *drift*. Além disto, não obstante os resultados satisfatórios dos métodos de VO e

VIO, em operações de um veículo subaquático mais longas, o erro irá aumentar com o tempo dessa operação, pelo que, um método de SLAM por criar um mapa global, como acontece no ORB-SLAM/ORB-SLAM3, permite uma melhor estimativa do movimento e obter um mapa otimizado.

6.2 Trabalho Futuro

Para a continuação deste trabalho, existem diversas linhas de investigação que podem ser seguidas.

Nas condições atuais do veículo, é necessário resolver o problema da leitura dos dados das acelerações angulares. Uma alternativa seria usar uma IMU externa, de modo a obter os valores da velocidade angular, como é comum e que iria reduzir o erro de integração da aceleração angular. Também existe a necessidade de realizar um conjunto de testes com o VideoRay Pro4 ROV em cenários mais realistas. Juntamente com estes testes, poderiam ser criados novos *datasets* que ajudem a comunidade científica a desenvolver métodos mais adequados para as condições específicas do meio subaquático.

De forma a tornar a navegação do Pro4 ROV mais robusta, a utilização de outros sensores já existentes no veículo, como o sensor de pressão, permitiria limitar o *drift* e o erro de escala das trajetórias estimadas, ou a bússola magnética que iria melhorar as estimativas da orientação. No entanto, utilização da bússola requer precauções, como já foi discutido. A integração de uma nova câmara no veículo também permitiria aumentar a robustez e melhorar a estimativa da escala da localização baseada em visão.

Relativamente aos algoritmos avaliados, uma forma de obter um *feedback* acerca da qualidade da trajetória ou do mapa estimados permitiria auxiliar o operador do veículo. Por exemplo, a perceção de que a zona para a qual o veículo se move estará a prejudicar odometria visual possibilitaria a alteração da trajetória do veículo. Um bom ponto inicial seria avaliar o número de *matches* efetuados, que está associado à qualidade do *tracking*.

Concretamente sobre os métodos diretos, seria interessante dotá-los de um módulo de reconhecimento. O ideal seria utilizar os pontos utilizados durante a estimação do movimento embora a associação destes pontos se torne bastante complexa sem a utilização de descritores. A tendência nos últimos anos tem sido a utilização de *bag of words*, mas como se observou, os vocabulários treinados não estão adaptados ao meio em estudo. Assim, o desenvolvimento de uma solução que elabore um vocabulário de deteção no decorrer da missão poderá ser uma solução. Outra abordagem possível é a identificação de regiões/*clusters* da imagem distintas/os, como indicam Negre, Bonin-Font e Oliver [129], que aplicam este método e demonstram a sua vantagem em imagens adquiridas debaixo de água.

Finalmente, dentro do tema do processamento de imagem, uma possível abordagem é através do *deep learning*. Exemplos desta abordagem demonstrados no meio subaquático são os trabalhos de Park, Han e Ko [130], Li, Skinner, Eustice et al. [131] e Hu, Wang, Zhao et al. [132]. A vantagem desta abordagem é que não estaria limitada ao uso de imagens *grayscale*, como o CLAHE, e pode apresentar um melhor desempenho relativamente às tecnicas estudadas nesta dissertação. No entanto, relativamente às redes neuronais, as mesmas necessitam de

um elevado de dados até fornecerem uma boa precisão, o que não é simples de obter no meio subaquático. Neste sentido, as GAN (*Generative Adversarial Network*, num processo competitivo entre duas redes neuronais de modo a conhecer a distribuição da informação fornecida às redes, permitem mitigar o problema da falta de dados.

Referências

[1] L. Stutters, H. Liu, C. Tiltman e D. J. Brown, «Navigation Technologies for Autonomous Underwater Vehicles», *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, n.º 4, pp. 581–589, 2008. DOI: 10.1109/TSMCC.2008.919147.

[2] A. Gómez-Espinosa, E. Cuan-Urquizo e J. González-García, «Autonomous Underwater Vehicles: Localization, Navigation, and Communication for Collaborative Missions», *Applied Sciences*, vol. 10, p. 1256, fev. de 2020. DOI: 10.3390/app10041256.

[3] J. J. Leonard, A. A. Bennett, C. M. Smith, H. Jacob e S. Feder, «Autonomous Underwater Vehicle Navigation», em *MIT Marine Robotics Laboratory Technical Memorandum*, 1998.

[4] Q. Zhang, X. Niu e C. Shi, «Impact Assessment of Various IMU Error Sources on the Relative Accuracy of the GNSS/INS Systems», *IEEE Sensors Journal*, vol. 20, n.º 9, pp. 5026–5038, 2020. DOI: 10.1109/JSEN.2020.2966379.

[5] H. Rice, S. Kelmenson e L. Mendelsohn, «Geophysical navigation technologies and applications», em *PLANS 2004. Position Location and Navigation Symposium (IEEE Cat. No.04CH37556)*, 2004, pp. 618–624. DOI: 10.1109/PLANS.2004.1309051.

[6] R. McEwen, H. Thomas, D. Weber e F. Psota, «Performance of an AUV navigation system at Arctic latitudes», *IEEE Journal of Oceanic Engineering*, vol. 30, n.º 2, pp. 443–454, 2005. DOI: 10.1109/JOE.2004.838336.

[7] J. Kinsey, R. Eustice e L. Whitcomb, «A Survey of Underwater Vehicle Navigation: Recent Advances and New Challenges», *IFAC Conference of Manoeuvering and Control of Marine Craft*, jan. de 2006.

[8] C. LaPointe, «Virtual Long Baseline (VLBL) Autonomous Underwater Vehicle Navigation Using a Single Transponder», tese de mestrado, Massachusetts Institute of Technology, USA, 2006.

[9] K. Vickery, «Acoustic positioning systems. A practical overview of current systems», em *Proceedings of the 1998 Workshop on Autonomous Underwater Vehicles (Cat. No.98CH36290)*, 1998, pp. 5–17. DOI: 10.1109/AUV.1998.744434.

[10] P. Rigby, O. Pizarro e S. B. Williams, «Towards Geo-Referenced AUV Navigation Through Fusion of USBL and DVL Measurements», em *OCEANS 2006*, 2006, pp. 1–6. DOI: 10.1109/OCEANS.2006.306898.

[11] E. G. Font, F. Bonin-Font, P.-L. Negre, M. Massot e G. Oliver, «USBL Integration and Assessment in a Multisensor Navigation Approach for AUVs.», *IFAC-PapersOnLine*, vol. 50, n.º 1, pp. 7905–7910, 2017, 20th IFAC World Congress, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2017.08.754. URL: http://www.sciencedirect.com/science/article/pii/S2405896317312028.

[12] R. Christ e R. Wernli, «The ROV Manual: A User Guide for Remotely Operated Vehicles: Second Edition», *The ROV Manual: A User Guide for Remotely Operated Vehicles: Second Edition*, pp. 1–679, jan. de 2013.

[13] H. G. Thomas, «GIB buoys: an interface between space and depths of the oceans», em *Proceedings of the 1998 Workshop on Autonomous Underwater Vehicles (Cat. No.98CH36290)*, 1998, pp. 181–184. DOI: 10.1109/AUV.1998.744453.

[14] J. Melo e A. Matos, «Survey on advances on terrain based navigation for autonomous underwater vehicles», *Ocean Engineering*, vol. 139, pp. 250–264, 2017, ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2017.04.047. URL: http://www.sciencedirect.com/science/article/pii/S002980181730241X.

[15] J. Zhao, S. Wang e A. Wang, «Study on underwater navigation system based on geomagnetic match technique», em *2009 9th International Conference on Electronic Measurement Instruments*, 2009, pp. 3-255-3–259. DOI: 10.1109/ICEMI.2009.5274338.

[16] N. Kato e T. Shigetomi, «Underwater Navigation for Long-Range Autonomous Underwater Vehicles Using Geomagnetic and Bathymetric Information», *Advanced Robotics*, vol. 23, n.º 7-8, pp. 787–803, 2009. DOI: 10.1163/156855309X443016. eprint: https://doi.org/10.1163/156855309X443016. URL: https://doi.org/10.1163/156855309X443016.

[17] F. C. Teixeira e A. M. Pascoal, «Geophysical Navigation of Autonomous Underwater Vehicles Using Geomagnetic Information*», *IFAC Proceedings Volumes*, vol. 41, n.º 1, pp. 178–183, 2008, 2nd IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles, ISSN: 1474-6670. DOI: https://doi.org/10.3182/20080408-3-IE-4914.00032. URL: http://www.sciencedirect.com/science/article/pii/S1474667015355221.

[18] N. P. Fofonoff e R. C. Millard, «Algorithms for Computation of Fundamental Properties of Seawater. Endorsed by Unesco/SCOR/ICES/IAPSO Joint Panel on Oceanographic Tables and Standards and SCOR Working Group 51. Unesco Technical Papers in Marine Science, No. 44.», 1983.

[19] S. Barkby, S. Williams, O. Pizarro e M. Jakuba, «An efficient approach to bathymetric SLAM», em *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 219–224. DOI: 10.1109/IROS.2009.5354248.

[20] D. Romagós, P. Ridao, J. Tardos e J. Neira, «Underwater SLAM in a marina environment», dez. de 2007, pp. 1455–1460. DOI: 10.1109/IROS.2007.4399222.

[21] D. Scaramuzza e F. Fraundorfer, «Visual Odometry [Tutorial]», *IEEE Robot. Automat. Mag.*, vol. 18, pp. 80–92, dez. de 2011. DOI: 10.1109/MRA.2011.943233.

[22] D. Nister, O. Naroditsky e J. Bergen, «Visual odometry», em *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1, 2004, pp. I–I. DOI: 10.1109/CVPR.2004.1315094.

[23] D. M. Helmick, Yang Cheng, D. S. Clouse, L. H. Matthies e S. I. Roumeliotis, «Path following using visual odometry for a Mars rover in high-slip environments», em *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No.04TH8720)*, vol. 2, 2004, 772–789 Vol.2. DOI: 10.1109/AERO.2004.1367679.

[24] J. Engel, T. Schöps e D. Cremers, «LSD-SLAM: Large-Scale Direct Monocular SLAM», em *European Conference on Computer Vision (ECCV)*, set. de 2014.

[25] T. Tuytelaars e K. Mikolajczyk, «Local Invariant Feature Detectors: A Survey», *Foundations and Trends in Computer Graphics and Vision*, vol. 3, pp. 177–280, nov. de 2007. DOI: 10.1561/0600000017.

[26] B. Lucas e T. Kanade, «An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI)», vol. 81, abr. de 1981.

[27] F. Fraundorfer e D. Scaramuzza, «Visual Odometry: Part II - Matching, Robustness, and Applications», *IEEE Robotics Automation Magazine - IEEE ROBOT AUTOMAT*, vol. 19, pp. 78–90, jun. de 2012. DOI: 10.1109/MRA.2012.2182810.

[28] B. Triggs, P. F. McLauchlan, R. I. Hartley e A. W. Fitzgibbon, «Bundle Adjustment — A Modern Synthesis», em *Vision Algorithms: Theory and Practice*, B. Triggs, A. Zisserman e R. Szeliski, eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 298–372, ISBN: 978-3-540-44480-0.

[29] F. Dellaert e M. Kaess, «Factor Graphs for Robot Perception», *Found. Trends Robotics*, vol. 6, pp. 1–139, 2017. DOI: http://dx.doi.org/10.1561/2300000043.

[30] P. H. S. Torr e A. Zisserman, «Feature Based Methods for Structure and Motion Estimation», em *Vision Algorithms: Theory and Practice*, B. Triggs, A. Zisserman e R. Szeliski, eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 278–294.

[31] M. Aqel, M. H. Marhaban, M. I. Saripan e N. Ismail, «Review of visual odometry: types, approaches, challenges, and applications», *SpringerPlus*, vol. 5, dez. de 2016. DOI: 10.1186/s40064-016-3573-7.

[32] Davison, «Real-time simultaneous localisation and mapping with a single camera», em *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, 1403–1410 vol.2. DOI: 10.1109/ICCV.2003.1238654.

[33] E. Eade e T. Drummond, «Scalable Monocular SLAM», em *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, 2006, pp. 469–476. DOI: 10.1109/CVPR.2006.263.

[34] A. J. Davison, I. D. Reid, N. D. Molton e O. Stasse, «MonoSLAM: Real-Time Single Camera SLAM», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, n.º 6, pp. 1052–1067, 2007. DOI: 10.1109/TPAMI.2007.1049.

[35] G. Klein e D. Murray, «Parallel Tracking and Mapping for Small AR Workspaces», em *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234. DOI: 10.1109/ISMAR.2007.4538852.

[36] R. A. Newcombe, S. J. Lovegrove e A. J. Davison, «DTAM: Dense tracking and mapping in real-time», em *2011 International Conference on Computer Vision*, 2011, pp. 2320–2327. DOI: 10.1109/ICCV.2011.6126513.

[37] J. Engel, J. Sturm e D. Cremers, «Semi-dense Visual Odometry for a Monocular Camera», em *2013 IEEE International Conference on Computer Vision*, 2013, pp. 1449–1456. DOI: 10.1109/ICCV.2013.183.

[38] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford e G. Wyeth, «OpenFABMAP: An open source toolbox for appearance-based loop closure detection», em *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 4730–4735. DOI: 10.1109/ICRA.2012.6224843.

[39] C. Forster, M. Pizzoli e D. Scaramuzza, «SVO: Fast semi-direct monocular visual odometry», em *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 15–22. DOI: 10.1109/ICRA.2014.6906584.

[40] E. Rosten, R. Porter e T. Drummond, «Faster and Better: A Machine Learning Approach to Corner Detection», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, n.º 1, pp. 105–119, 2010. DOI: 10.1109/TPAMI.2008.275.

[41] R. Mur-Artal, J. M. M. Montiel e J. D. Tardós, «ORB-SLAM: A Versatile and Accurate Monocular SLAM System», *IEEE Transactions on Robotics*, vol. 31, n.º 5, pp. 1147–1163, 2015. DOI: 10.1109/TRO.2015.2463671.

[42] H. Strasdat, J. Montiel e A. Davison, «Scale drift-aware large scale monocular SLAM», em *Robotics: Science and System*, vol. 2, jun. de 2010. DOI: 10.15607/RSS.2010.VI.010.

[43] H. Strasdat, A. J. Davison, J. M. M. Montiel e K. Konolige, «Double window optimisation for constant time visual SLAM», em *2011 International Conference on Computer Vision*, 2011, pp. 2352–2359. DOI: 10.1109/ICCV.2011.6126517.

[44] D. Galvez-López e J. D. Tardos, «Bags of Binary Words for Fast Place Recognition in Image Sequences», *IEEE Transactions on Robotics*, vol. 28, n.º 5, pp. 1188–1197, 2012. DOI: 10.1109/TRO.2012.2197158.

[45] R. Mur-Artal e J. D. Tardós, «ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras», *IEEE Transactions on Robotics*, vol. 33, n.º 5, pp. 1255–1262, 2017. DOI: 10.1109/TRO.2017.2705103.

[46] C. Forster, Z. Zhang, M. Gassner, M. Werlberger e D. Scaramuzza, «SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems», *IEEE Transactions on Robotics*, vol. 33, n.º 2, pp. 249–265, 2017. DOI: 10.1109/TRO.2016.2623335.

[47] J. Engel, V. Koltun e D. Cremers, «Direct Sparse Odometry», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, n.º 3, pp. 611–625, 2018. DOI: 10.1109/TPAMI.2017.2658577.

[48] X. Gao, R. Wang, N. Demmel e D. Cremers, «LDSO: Direct Sparse Odometry with Loop Closure», em *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2198–2204. DOI: 10.1109/IROS.2018.8593376.

[49] J. Zubizarreta, I. Aguinaga e J. M. M. Montiel, «Direct Sparse Mapping», *IEEE Transactions on Robotics*, vol. 36, n.º 4, pp. 1363–1370, ago. de 2020, ISSN: 1941-0468. DOI: 10.1109/tro.2020.2991614. URL: http://dx.doi.org/10.1109/TRO.2020.2991614.

[50] C. Campos, R. Elvira, J. Rodríguez e J. Montiel, «ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM», jul. de 2020.

[51] J. Gui, D. Gu, S. Wang e H. Hu, «A review of visual inertial odometry from filtering and optimisation perspectives», *Advanced Robotics*, vol. 29, pp. 1–13, set. de 2015. DOI: 10.1080/01691864.2015.1057616.

[52] H. Strasdat, J. Montiel e A. Davison, «Real-time monocular SLAM: Why filter?», mai. de 2010, pp. 2657–2664. DOI: 10.1109/ROBOT.2010.5509636.

[53] D. Scaramuzza e Z. Zhang, *Visual-Inertial Odometry of Aerial Robots*, 2019. arXiv: 1906.03289 [cs.RO].

[54] S. A. S. Mohamed, M. Haghbayan, T. Westerlund, J. Heikkonen, H. Tenhunen e J. Plosila, «A Survey on Odometry for Autonomous Navigation Systems», *IEEE Access*, vol. 7, pp. 97 466–97 486, 2019. DOI: 10.1109/ACCESS.2019.2929133.

[55] A. I. Mourikis e S. I. Roumeliotis, «A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation», em *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3565–3572. DOI: 10.1109/ROBOT.2007.364024.

[56] M. Li e A. Mourikis, «High-precision, consistent EKF-based visual–inertial odometry», *The International Journal of Robotics Research*, vol. 32, pp. 690–711, mai. de 2013. DOI: 10.1177/0278364913481251.

[57] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart e P. Furgale, «Keyframe-based visual–inertial odometry using nonlinear optimization», *The International Journal of Robotics Research*, vol. 34, n.º 3, pp. 314–334, 2015. DOI: 10.1177/0278364914554813.

[58] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige e R. Siegwart, «Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization», jun. de 2013. DOI: 10.15607/RSS.2013.IX.037.

[59] M. Bloesch, S. Omari, M. Hutter e R. Siegwart, «Robust Visual Inertial Odometry Using a Direct EKF-Based Approach», en, 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); Conference Location: Hamburg, Germany; Conference Date: September 28 - October 2, 2015, Zürich: ETH-Zürich, 2015. DOI: 10.3929/ethz-a-010566547.

[60] M. Bloesch, M. Burri, S. Omari, M. Hutter e R. Siegwart, «Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback», en, The International Journal of Robotics Research, vol. 36, n.º 10, pp. 1053–1072, set. de 2017, ISSN: 0278-3649. DOI: 10.3929/ethz-b-000187364.

[61] R. Mur-Artal e J. D. Tardós, «Visual-Inertial Monocular SLAM With Map Reuse», *IEEE Robotics and Automation Letters*, vol. 2, n.º 2, pp. 796–803, 2017. DOI: 10.1109/LRA.2017.2653359.

[62] T. Qin, P. Li e S. Shen, «VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator», *IEEE Transactions on Robotics*, vol. 34, n.º 4, pp. 1004–1020, 2018. DOI: 10.1109/TRO.2018.2853729.

[63] L. Von Stumberg, V. Usenko e D. Cremers, «Direct Sparse Visual-Inertial Odometry Using Dynamic Marginalization», em *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2510–2517. DOI: 10.1109/ICRA.2018.8462905.

[64] B. Joshi, S. Rahman, M. Kalaitzakis, B. Cain, J. Johnson, M. Xanthidis, N. Karapetyan, A. Hernandez, A. Q. Li, N. Vitzilaios e I. Rekleitis, «Experimental Comparison of Open Source Visual-Inertial-Based State Estimation Algorithms in the Underwater Domain», em *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7227–7233. DOI: 10.1109/IROS40897.2019.8968049.

[65] M. Ferrera, V. Creuze, J. Moras e P. Trouvé-Peloux, «AQUALOC: An underwater dataset for visual–inertial–pressure localization», *The International Journal of Robotics Research*, vol. 38, n.º 14, pp. 1549–1559, 2019. DOI: 10.1177/0278364919883346. eprint: https://doi.org/10.1177/0278364919883346. URL: https://doi.org/10.1177/0278364919883346.

[66] R. Garcia, X. Cufi e M. Carreras, «Estimating the motion of an underwater robot from a monocular image sequence», em *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 3, 2001, 1682–1687 vol.3. DOI: 10.1109/IROS.2001.977220.

[67] N. Gracias e J. Santos-Victor, «Underwater Video Mosaics as Visual Navigation Maps», *Computer Vision and Image Understanding*, vol. 79, n.º 1, pp. 66–91, 2000, ISSN: 1077-3142. DOI: https://doi.org/10.1006/cviu.2000.0848. URL: http://www.sciencedirect.com/science/article/pii/S1077314200908488.

[68] N. R. Gracias, S. van der Zwaan, A. Bernardino e J. Santos-Victor, «Mosaic-based navigation for autonomous underwater vehicles», *IEEE Journal of Oceanic Engineering*, vol. 28, n.º 4, pp. 609–624, 2003. DOI: 10.1109/JOE.2003.819156.

[69] D. Romagós, P. Ridao, M. Carreras e X. Cufi, «An EKF vision-based navigation of an UUV in a structured environment», *IFAC Proceedings Volumes*, vol. 36, pp. 287–292, set. de 2003. DOI: 10.1016/S1474-6670(17)37822-9.

[70] N. Palomeras, S. Nagappa, D. Ribas, N. Gracias e M. Carreras, «Vision-based localization and mapping system for AUV intervention», em *2013 MTS/IEEE OCEANS - Bergen*, 2013, pp. 1–7. DOI: 10.1109/OCEANS-Bergen.2013.6608058.

[71] J. Jung, J.-H. Li, H.-T. Choi e H. Myung, «Localization of AUVs using visual information of underwater structures and artificial landmarks», *Intelligent Service Robotics*, vol. 10, pp. 1–10, nov. de 2016. DOI: 10.1007/s11370-016-0210-9.

[72] R. M. Eustice, O. Pizarro e H. Singh, «Visually Augmented Navigation for Autonomous Underwater Vehicles», *IEEE Journal of Oceanic Engineering*, vol. 33, n.º 2, pp. 103–122, 2008. DOI: 10.1109/JOE.2008.923547.

[73] M. Warren, P. Corke, O. Pizarro, S. Williams e B. Upcroft, «Visual sea-floor mapping from low overlap imagery using bi-objective bundle adjustment and constrained motion», nov. de 2012.

[74] A. Kim e R. Eustice, «Pose-graph visual SLAM with geometric model selection for autonomous underwater ship hull inspection», em *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 1559–1565. DOI: 10.1109/IROS.2009.5354132.

[75] S. Thrun e M. Montemerlo, «The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures», *The International Journal of Robotics Research*, vol. 25, n.º 5-6, pp. 403–429, 2006. DOI: 10.1177/0278364906065387. eprint: https://doi.org/10.1177/0278364906065387. URL: https://doi.org/10.1177/0278364906065387.

[76] A. Kim e R. M. Eustice, «Real-Time Visual SLAM for Autonomous Underwater Hull Inspection Using Visual Saliency», *IEEE Transactions on Robotics*, vol. 29, n.º 3, pp. 719–733, 2013. DOI: 10.1109/TRO.2012.2235699.

[77] P. L. N. Carrasco, F. Bonin-Font, M. M. Campos e G. O. Codina, «Stereo-Vision Graph-SLAM for Robust Navigation of the AUV SPARUS II», *IFAC-PapersOnLine*, vol. 48, n.º 2, pp. 200–205, 2015, 4th IFAC Workshop onNavigation, Guidance and Controlof Underwater VehiclesNGCUV 2015, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.06.033.

[78] K. Oliver, W. Hou e S. Wang, «Image feature detection and matching in underwater conditions», *Proceedings of SPIE - The International Society for Optical Engineering*, abr. de 2010. DOI: 10.1117/12.852339.

[79] J. Aulinas, M. Carreras, X. Llado, J. Salvi, R. Garcia, R. Prados e Y. R. Petillot, «Feature extraction for underwater visual SLAM», em *OCEANS 2011 IEEE - Spain*, 2011, pp. 1–7. DOI: 10.1109/Oceans-Spain.2011.6003474.

[80] H. Bay, T. Tuytelaars e L. Van Gool, «SURF: Speeded Up Robust Features», em *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof e A. Pinz, eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417, ISBN: 978-3-540-33833-8.

[81] R. Garcia e N. Gracias, «Detection of interest points in turbid underwater images», em *OCEANS 2011 IEEE - Spain*, 2011, pp. 1–9. DOI: 10.1109/Oceans-Spain.2011.6003605.

[82] F. Shkurti, I. Rekleitis e G. Dudek, «Feature Tracking Evaluation for Pose Estimation in Underwater Environments», em *2011 Canadian Conference on Computer and Robot Vision*, 2011, pp. 160–167. DOI: 10.1109/CRV.2011.28.

[83] Jianbo Shi e Tomasi, «Good features to track», em *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600. DOI: 10.1109/CVPR.1994.323794.

[84] E. Rublee, V. Rabaud, K. Konolige e G. Bradski, «ORB: An efficient alternative to SIFT or SURF», em *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[85] S. Leutenegger, M. Chli e R. Y. Siegwart, «BRISK: Binary Robust invariant scalable keypoints», em *2011 International Conference on Computer Vision*, 2011, pp. 2548–2555. DOI: 10.1109/ICCV.2011.6126542.

[86] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha e P. Fua, «BRIEF: Computing a Local Binary Descriptor Very Fast», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, n.º 7, pp. 1281–1298, 2012. DOI: 10.1109/TPAMI.2011.222.

[87] F. Hidalgo e T. Braunl, «Evaluation of Several Feature Detectors/Extractors on Underwater Images towards vSLAM», *Sensors*, vol. 20, p. 4343, ago. de 2020. DOI: 10.3390/s20154343.

[88] D. Lowe, «Distinctive Image Features from Scale-Invariant Keypoints», *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[89] P. Fernández Alcantarilla, «Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces», set. de 2013. DOI: 10.5244/C.27.13.

[90] M. Ferrera, J. Moras, P. Trouvé e V. Creuze, «Real-Time Monocular Visual Odometry for Turbid and Dynamic Underwater Environments», *Sensors*, vol. 19, p. 687, fev. de 2019. DOI: 10.3390/s19030687.

[91] S. Baker e I. Matthews, «Lucas-Kanade 20 Years On: A Unifying Framework Part 1: The Quantity Approximated, the Warp Update Rule, and the Gradient Descent Approximation», *International Journal of Computer Vision - IJCV*, jan. de 2004.

[92] F. Shkurti, I. Rekleitis, M. Scaccia e G. Dudek, «State estimation of an underwater robot using visual and inertial information», em *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 5054–5060. DOI: 10.1109/IROS.2011.6094680.

[93] A. Burguera, F. Bonin-Font e G. Oliver, «Trajectory-Based Visual Localization in Underwater Surveying Missions», *Sensors (Basel, Switzerland)*, vol. 15, pp. 1708–1735, jan. de 2015. DOI: 10.3390/s150101708.

[94] R. I. Hartley e A. Zisserman, *Multiple View Geometry in Computer Vision*, Second. Cambridge University Press, ISBN: 0521540518, 2004.

[95] J. Kannala e S. S. Brandt, «A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, n.º 8, pp. 1335–1340, 2006. DOI: 10.1109/TPAMI.2006.153.

[96] J.-L. Blanco, «A tutorial on SE(3) transformation parameterizations and on-manifold optimization», University of Malaga, rel. téc. 012010, 2010. URL: http://ingmec.ual.es/~jlblanco/papers/jlblanco2010geometry3D_techrep.pdf.

[97] D. Chotrov, Z. Uzunova, Y. Yordanov e S. Maleshkov, «Mixed-Reality Spatial Configuration with a ZED Mini Stereoscopic Camera», nov. de 2018.

[98] Q.-T. Luong e O. Faugeras, «The Fundamental Matrix: Theory, Algorithms, and Stability Analysis», *International Journal of Computer Vision*, vol. 17, pp. 43–75, jan. de 1996. DOI: 10.1007/BF00127818.

[99] D. Nister, «An efficient solution to the five-point relative pose problem», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, n.º 6, pp. 756–770, 2004. DOI: 10.1109/TPAMI.2004.17.

[100] O. Faugeras e F. Lustman, «Motion and structure from motion in a piecewise planar environment», INRIA, rel. téc. RR-0856, jun. de 1988. URL: https://hal.inria.fr/inria-00075698.

[101] R. Haralick, C.-N. Lee, K. Ottenberg e M. Nölle, «Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Problem», *International Journal of Computer Vision*, vol. 13, pp. 331–356, dez. de 1994. DOI: 10.1007/BF02028352.

[102] T. Ke e S. I. Roumeliotis, «An Efficient Algebraic Solution to the Perspective-Three-Point Problem», em *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4618–4626. DOI: 10.1109/CVPR.2017.491.

[103] L. Kneip, D. Scaramuzza e R. Siegwart, «A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation», em *CVPR 2011*, 2011, pp. 2969–2976. DOI: 10.1109/CVPR.2011.5995464.

[104] V. Lepetit, F. Moreno-Noguer e P. Fua, «EPnP: An accurate O(n) solution to the PnP problem», *International Journal of Computer Vision*, vol. 81, fev. de 2009. DOI: 10.1007/s11263-008-0152-6.

[105] L. Kneip, H. Li e Y. Seo, «UPnP: An Optimal O(n) Solution to the Absolute Pose Problem with Universal Applicability», em *ECCV*, 2014.

[106] M. A. Fischler e R. C. Bolles, «Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography», vol. 24, n.º 6, 1981, ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[107] J. Civera, A. J. Davison e J. M. M. Montiel, «Inverse Depth Parametrization for Monocular SLAM», *IEEE Transactions on Robotics*, vol. 24, n.º 5, pp. 932–945, 2008. DOI: 10.1109/TRO.2008.2003276.

[108] M. Kok, J. D. Hol e T. B. Schön, «Using Inertial Sensors for Position and Orientation Estimation», vol. 11, n.º 1–2, pp. 1–153, nov. de 2017, ISSN: 1932-8346. DOI: 10.1561/2000000094. URL: https://doi.org/10.1561/2000000094.

[109] A. B. Chatfield, «Fundamentals of high accuracy inertial navigation», 1997.

[110] T. Schneider, M. T. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski e R. Siegwart, «maplab: An Open Framework for Research in Visual-inertial Mapping and Localization», *IEEE Robotics and Automation Letters*, 2018. DOI: 10.1109/LRA.2018.2800113.

[111] C. Harris e M. Stephens, «A Combined Corner and Edge Detector», em *Procedings of the Alvey Vision Conference 1988*, Alvey Vision Club, 1988. DOI: 10.5244/c.2.23. URL: https://doi.org/10.5244%2Fc.2.23.

[112] A. C. Duarte, G. B. Zaffari, R. T. S. da Rosa, L. M. Longaray, P. Drews e S. S. C. Botelho, «Towards comparison of underwater SLAM methods: An open dataset collection», em *OCEANS 2016 MTS/IEEE Monterey*, 2016, pp. 1–5. DOI: 10.1109/OCEANS.2016.7761315.

[113] W. Wang, D. Zhu, X. Wang, Y. Hu, Y. Qiu, C. Wang, Y. Hu, A. Kapoor e S. Scherer, «TartanAir: A Dataset to Push the Limits of Visual SLAM», 2020.

[114] A. Bender, S. B. Williams e O. Pizarro, «Autonomous exploration of large-scale benthic environments», em *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 390–396. DOI: 10.1109/ICRA.2013.6630605.

[115] A. Mallios, E. Vidal, R. Campos e M. Carreras, «Underwater caves sonar data set», *The International Journal of Robotics Research*, vol. 36, n.º 12, pp. 1247–1251, 2017. DOI: 10.1177/0278364917732838.

[116] J. L. Schönberger e J.-M. Frahm, «Structure-from-Motion Revisited», em *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[117] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler e A. Ng, «ROS: an open-source Robot Operating System», vol. 3, jan. de 2009.

[118] Y. Cho, J. Jeong e A. Kim, «Model-Assisted Multiband Fusion for Single Image Enhancement and Applications to Robot Vision», *IEEE Robotics and Automation Letters*, vol. 3, n.º 4, pp. 2822–2829, 2018. DOI: 10.1109/LRA.2018.2843127.

[119] J. Tarel e N. Hautière, «Fast visibility restoration from a single color or gray level image», em *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 2201–2208. DOI: 10.1109/ICCV.2009.5459251.

[120] T. P. Marques e A. Branzan Albu, «L2UWE: A Framework for the Efficient Enhancement of Low-Light Underwater Images Using Local Contrast and Multi-Scale Fusion», em *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 2286–2295. DOI: 10.1109/CVPRW50498.2020.00277.

[121] S. M. Pizer, R. E. Johnston, J. P. Ericksen, B. C. Yankaskas e K. E. Muller, «Contrast-limited adaptive histogram equalization: speed and effectiveness», em *[1990] Proceedings of the First Conference on Visualization in Biomedical Computing*, 1990, pp. 337–345. DOI: 10.1109/VBC.1990.109340.

[122] K. He, J. Sun e X. Tang, «Guided Image Filtering», *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, pp. 1397–1409, jun. de 2013. DOI: 10.1109/TPAMI.2012.213.

[123] C. Mai, S. Pedersen, L. Hansen, K. Jepsen e Z. Yang, «Modeling and Control of Industrial ROV's for Semi-Autonomous Subsea Maintenance Services», *IFAC-PapersOnLine*, vol. 50, pp. 13 686–13 691, jul. de 2017. DOI: 10.1016/j.ifacol.2017.08.2535.

[124] G. Bradski, «The OpenCV Library», *Dr. Dobb's Journal of Software Tools*, 2000.

[125] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas e M. Marín-Jiménez, «Automatic generation and detection of highly reliable fiducial markers under occlusion», *Pattern Recognition*, vol. 47, n.º 6, pp. 2280–2292, 2014, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2014.01.005.

[126] J. Sturm, N. Engelhard, F. Endres, W. Burgard e D. Cremers, «A benchmark for the evaluation of RGB-D SLAM systems», em *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580. DOI: 10.1109/IROS.2012.6385773.

[127] S. Umeyama, «Least-squares estimation of transformation parameters between two point patterns», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, n.º 4, pp. 376–380, 1991.

[128] M. Grupp, *evo: Python package for the evaluation of odometry and SLAM.* https://github.com/MichaelGrupp/evo, 2017.

[129] P. L. Negre, F. Bonin-Font e G. Oliver, «Cluster-based loop closing detection for underwater slam in feature-poor regions», em *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 2589–2595. DOI: 10.1109/ICRA.2016.7487416.

[130] J. Park, D. K. Han e H. Ko, «Adaptive Weighted Multi-Discriminator CycleGAN for Underwater Image Enhancement», *Journal of Marine Science and Engineering*, vol. 7, n.º 7, 2019, ISSN: 2077-1312. DOI: 10.3390/jmse7070200.

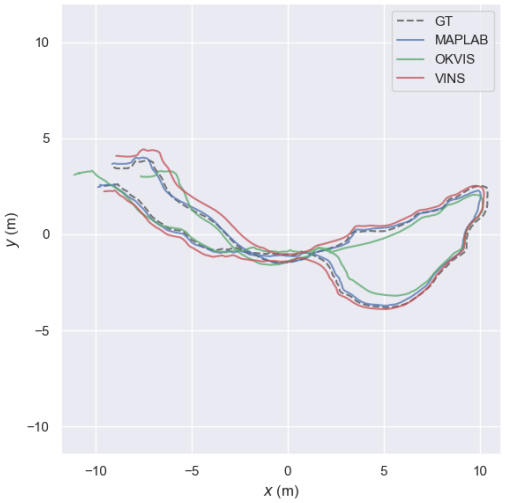
[131] J. Li, K. A. Skinner, R. M. Eustice e M. Johnson-Roberson, «WaterGAN: Unsupervised Generative Network to Enable Real-Time Color Correction of Monocular Underwater Images», *IEEE Robotics and Automation Letters*, vol. 3, n.º 1, pp. 387–394, 2018. DOI: 10.1109/LRA.2017.2730363.

[132] Y. Hu, K. Wang, X. Zhao, H. Wang e Y. Li, «Underwater Image Restoration Based on Convolutional Neural Network», em *Proceedings of The 10th Asian Conference on Machine Learning*, J. Zhu e I. Takeuchi, eds., sér. Proceedings of Machine Learning Research, vol. 95, PMLR, 2018, pp. 296–311.

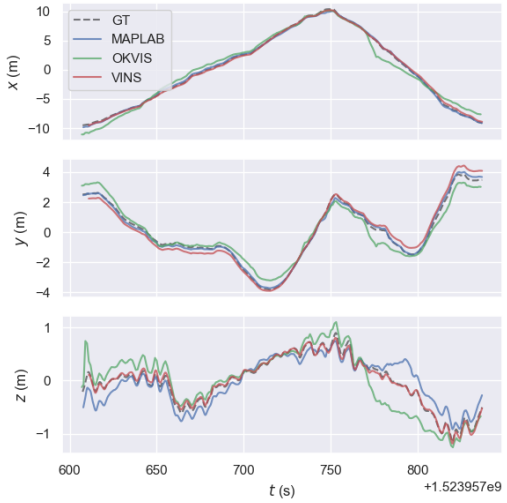
Anexo

A.1 - Trajetórias estimadas para cada sequência

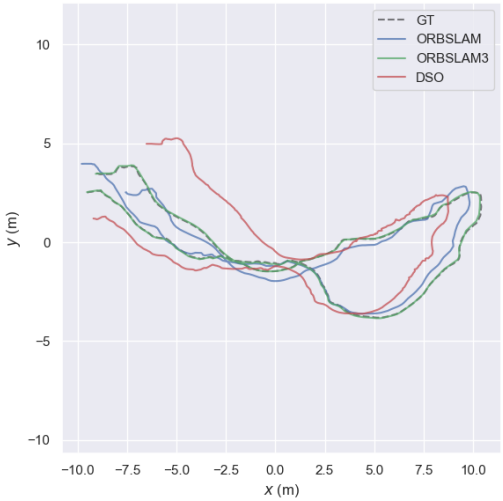
Neste anexo são apresentadas as representações gráficas das trajetórias estimadas pelos algoritmos avaliados com o menor erro entre as três execuções, para cada uma das sequências do *dataset* do AQUALOC, referentes à secção 5.1



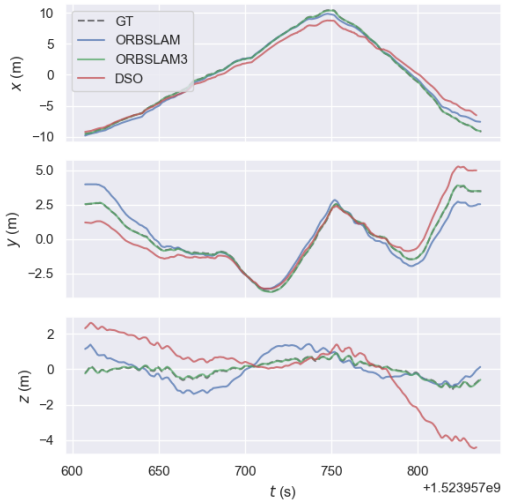
(a) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono no plano XY para a sequência 1.



(b) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono segundo as coordenadas X, Y e Z para sequência 1.

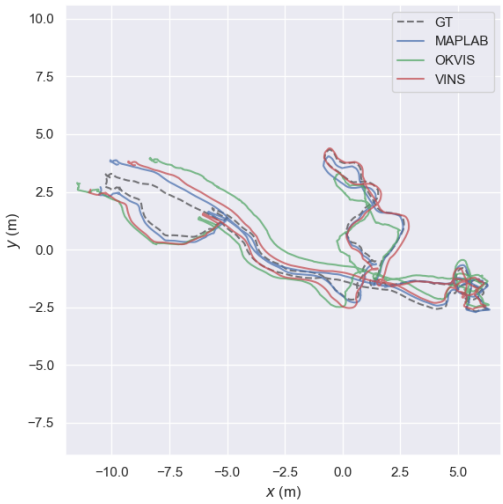


(c) Representação do *ground-truth* e das trajetórias estimadas pelo ORB-SLAM,ORB-SLAM3 e DSO no plano XY para a sequência 1.

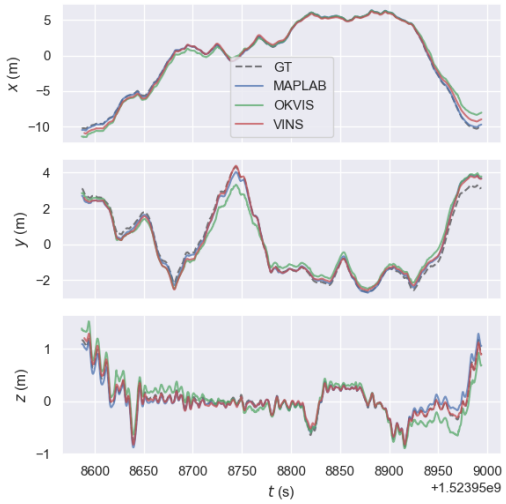


(d) Representação do *ground-truth* e das trajetórias estimadas pelo ORB-SLAM, ORB-SLAM3 e DSO segundo as coordenadas X, Y e Z para sequência 1.

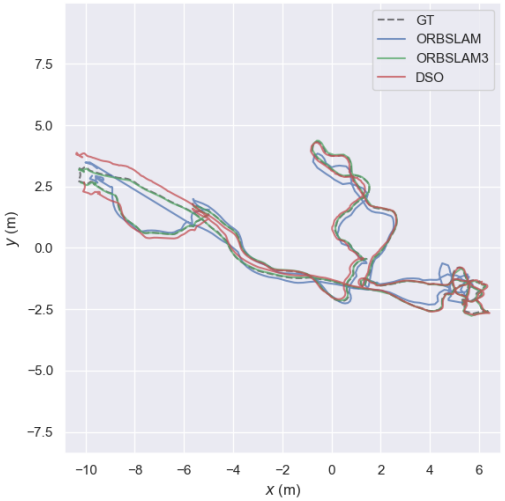
Figura A.1: Representação das trajetórias estimadas com o menor ATE.



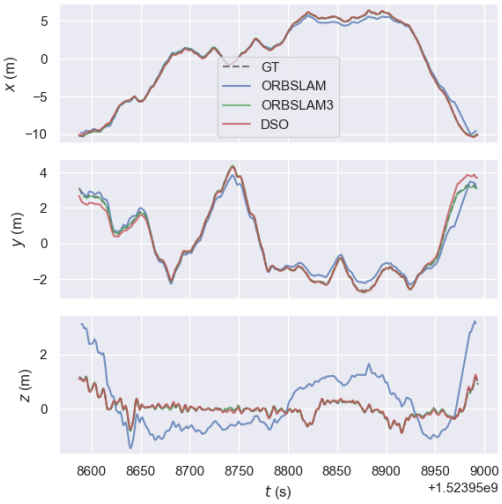
(a) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono no plano XY para a sequência 2.



(b) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono segundo as coordenadas X, Y e Z para sequência 2.

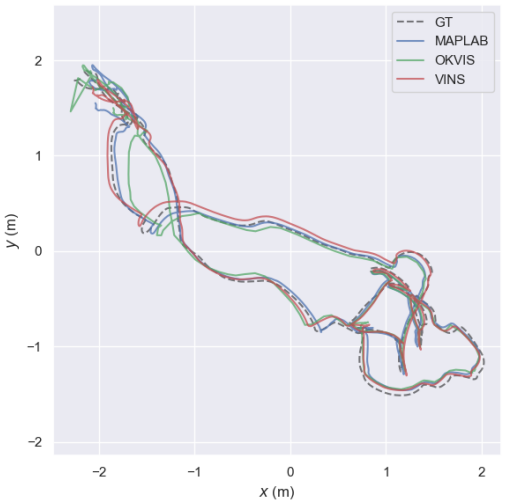


(c) Representação do *ground-truth* e das trajetórias estimadas pelo ORB-SLAM,ORB-SLAM3 e DSO no plano XY para a sequência 2.

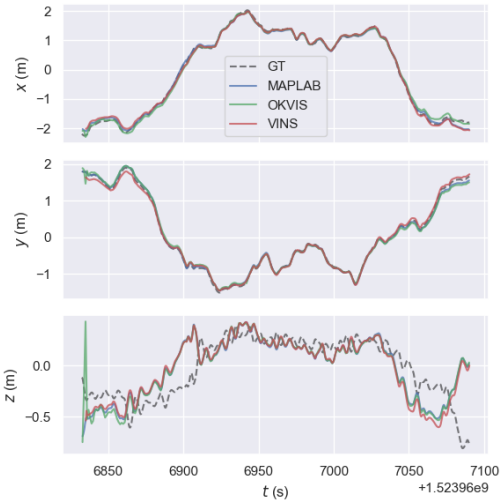


(d) Representação do *ground-truth* e das trajetórias estimadas pelo ORB-SLAM, ORB-SLAM3 e DSO segundo as coordenadas X, Y e Z para sequência 2.

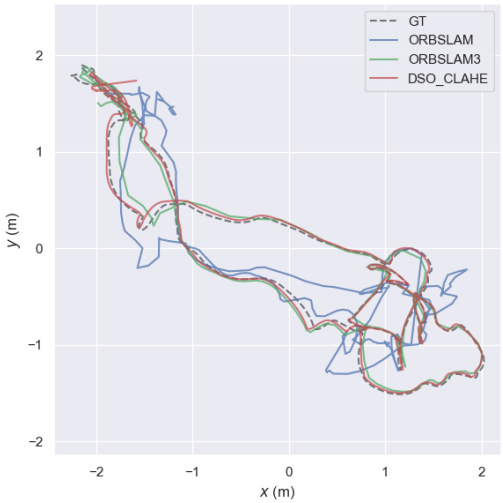
Figura A.2: Representação das trajetórias estimadas com o menor ATE para a sequência 2.



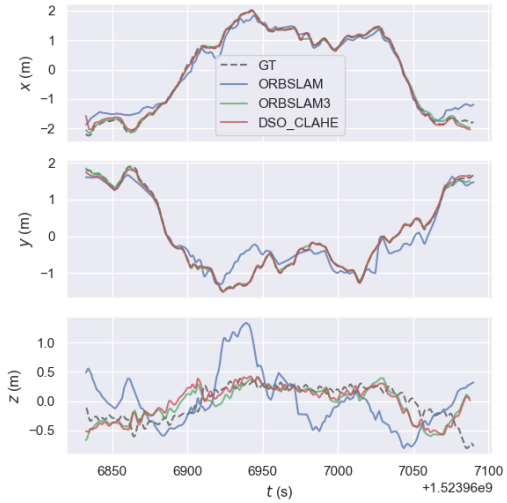
(a) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono no plano XY para a sequência 3.



(b) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono segundo as coordenadas X, Y e Z para sequência 3.

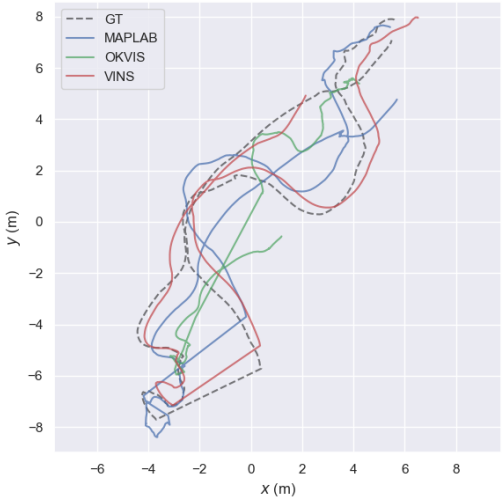


(c) Representação do *ground-truth* e das trajetórias estimadas pelo ORB-SLAM, ORB-SLAM3 e DSO no plano XY para a sequência 3.

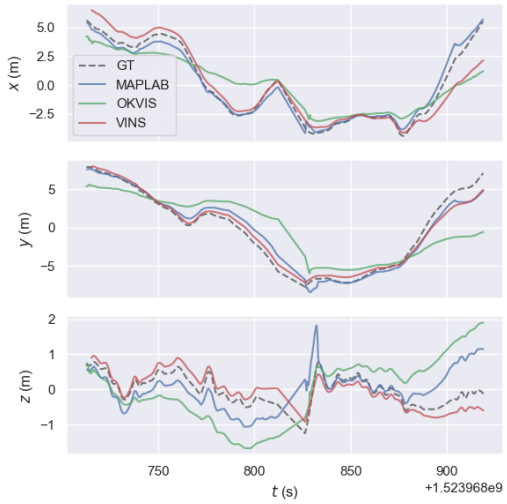


(d) Representação do *ground-truth* e das trajetórias estimadas pelo ORB-SLAM, ORB-SLAM3 e DSO segundo as coordenadas X, Y e Z para sequência 3.

Figura A.3: Representação das trajetórias estimadas com o menor ATE para a sequência 3.

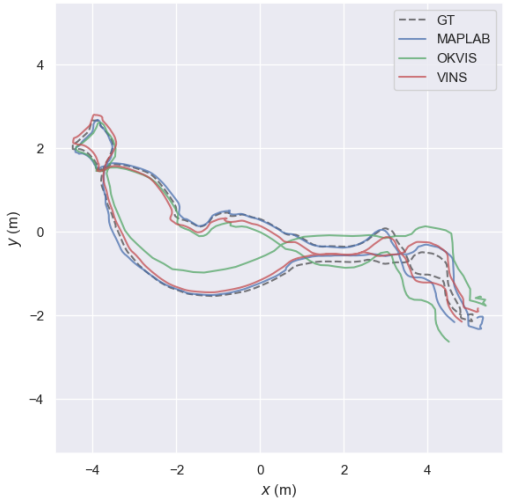


(a) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono no plano XY para a sequência 4.

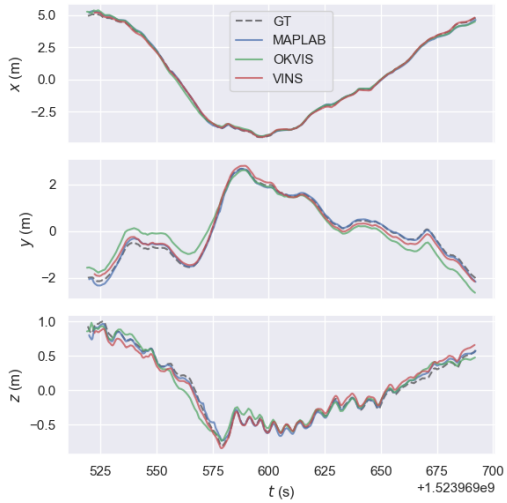


(b) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono segundo as coordenadas X, Y e Z para sequência 4.

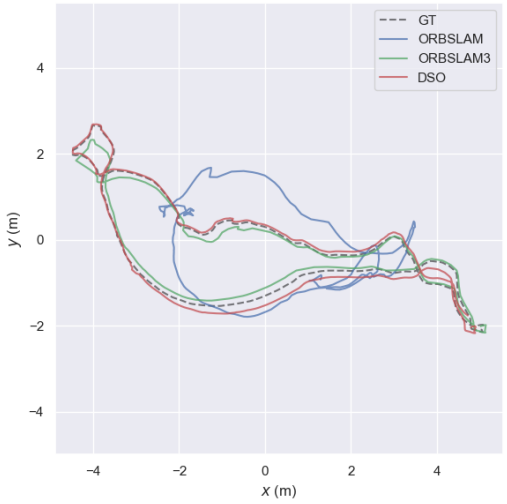
Figura A.4: Representação das trajetórias estimadas com o menor ATE para a sequência 4.



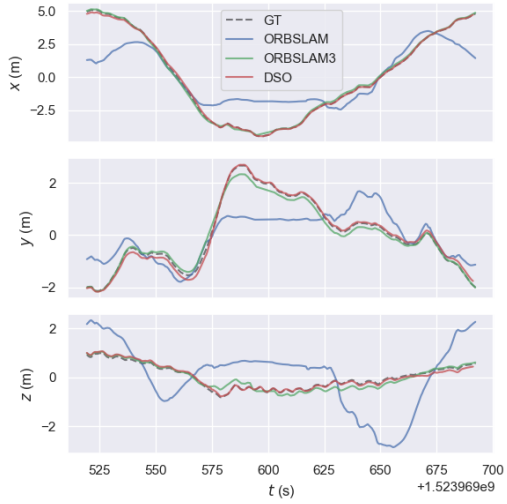
(a) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono no plano XY para a sequência 5.



(b) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono segundo as coordenadas X, Y e Z para sequência 5.

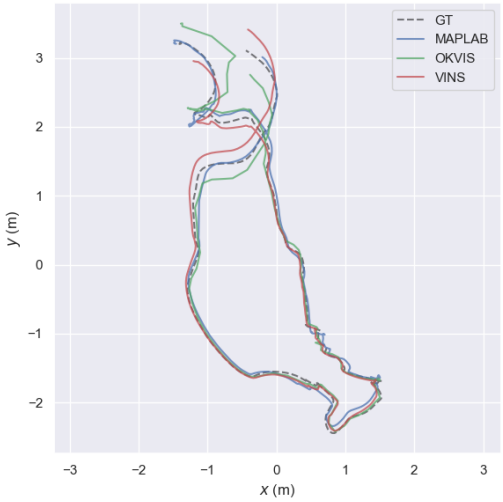


(c) Representação do *ground-truth* e das trajetórias estimadas pelo ORB-SLAM, ORB-SLAM3 e DSO no plano XY para a sequência 5.

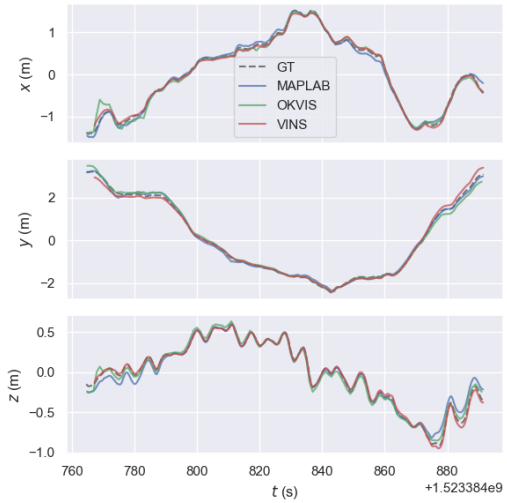


(d) Representação do *ground-truth* e das trajetórias estimadas pelo ORB-SLAM, ORB-SLAM3 e DSO segundo as coordenadas X, Y e Z para sequência 5.

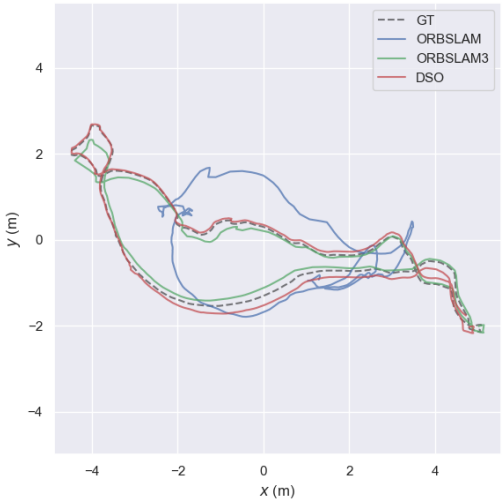
Figura A.5: Representação das trajetórias estimadas com o menor ATE para a sequência 5.



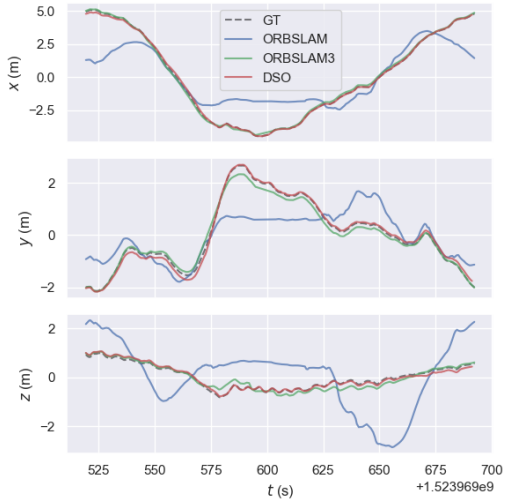
(a) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono no plano XY para a sequência 6.



(b) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono segundo as coordenadas X, Y e Z para sequência 6.

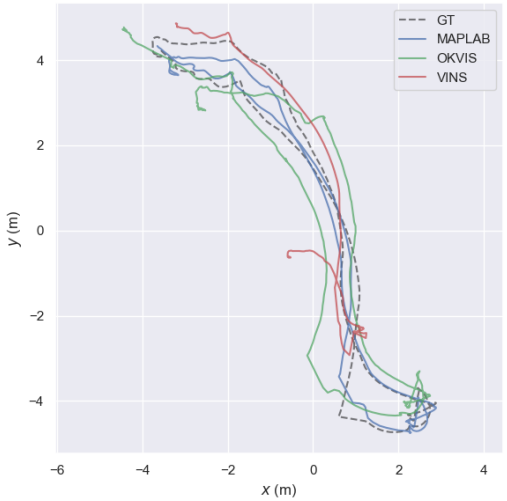


(c) Representação do *ground-truth* e das trajetórias estimadas pelo ORB-SLAM, ORB-SLAM3 e DSO no plano XY para a sequência 6.

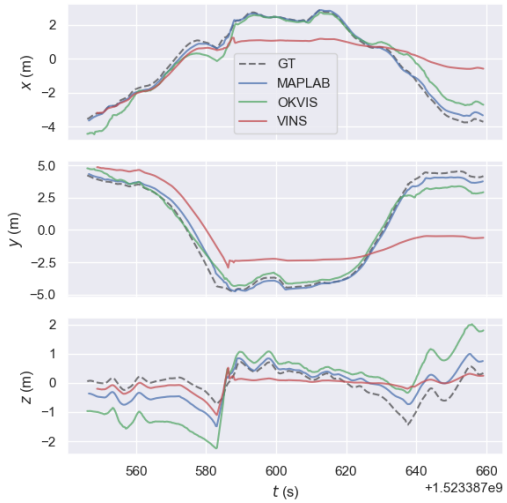


(d) Representação do *ground-truth* e das trajetórias estimadas pelo ORB-SLAM, ORB-SLAM3 e DSO segundo as coordenadas X, Y e Z para sequência 6.

Figura A.6: Representação das trajetórias estimadas com o menor ATE para a sequência 6.



(a) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono no plano XY para a sequência 7.



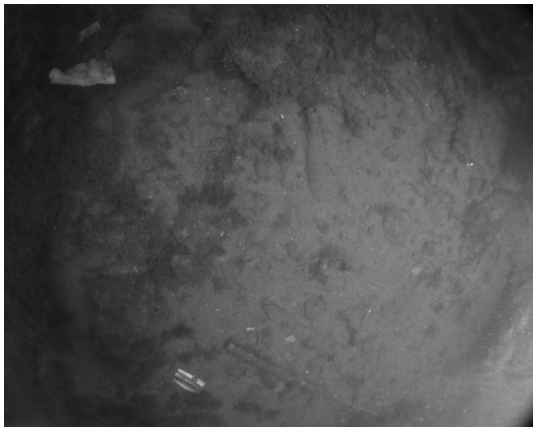
(b) Representação do *ground-truth* e das trajetórias estimadas pelo MAPLAB, OKVIS e VINS-Mono segundo as coordenadas X, Y e Z para sequência 7.

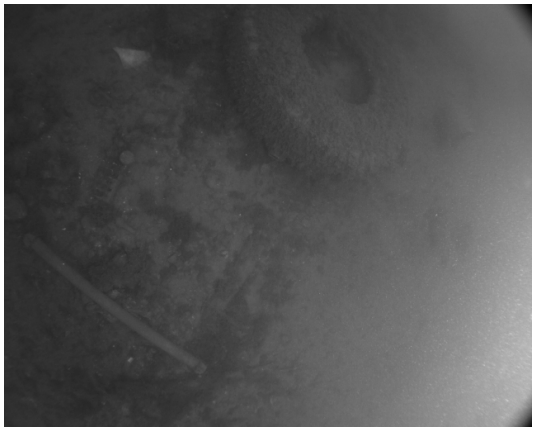
Figura A.7: Representação das trajetórias estimadas com o menor ATE para a sequência 7.

A.2 - Imagens das sequências

Nesta secção são apresentadas algumas imagens exemplificativas das sequências avaliadas, nas quais, em algumas, são visíveis determinadas partes mais difíceis para a extração da informação visual.

Sequência 1





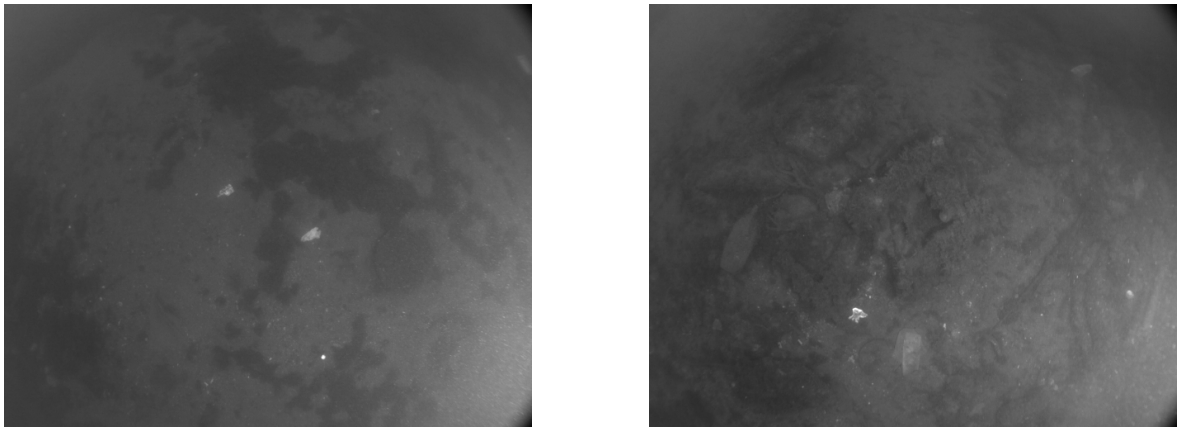


Figura A.8: Imagens da sequência 1.

Sequência 2

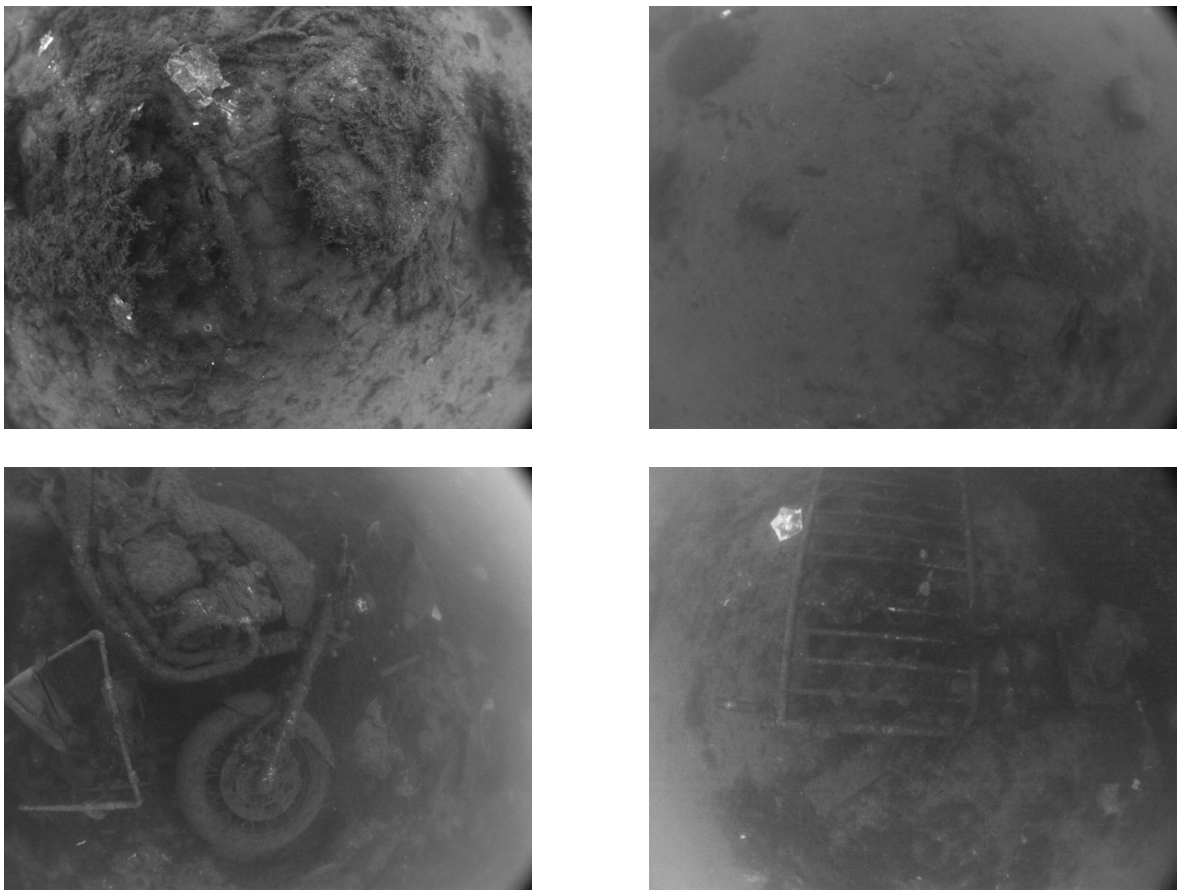


Figura A.9: Imagens da sequência 2.

Sequência 3

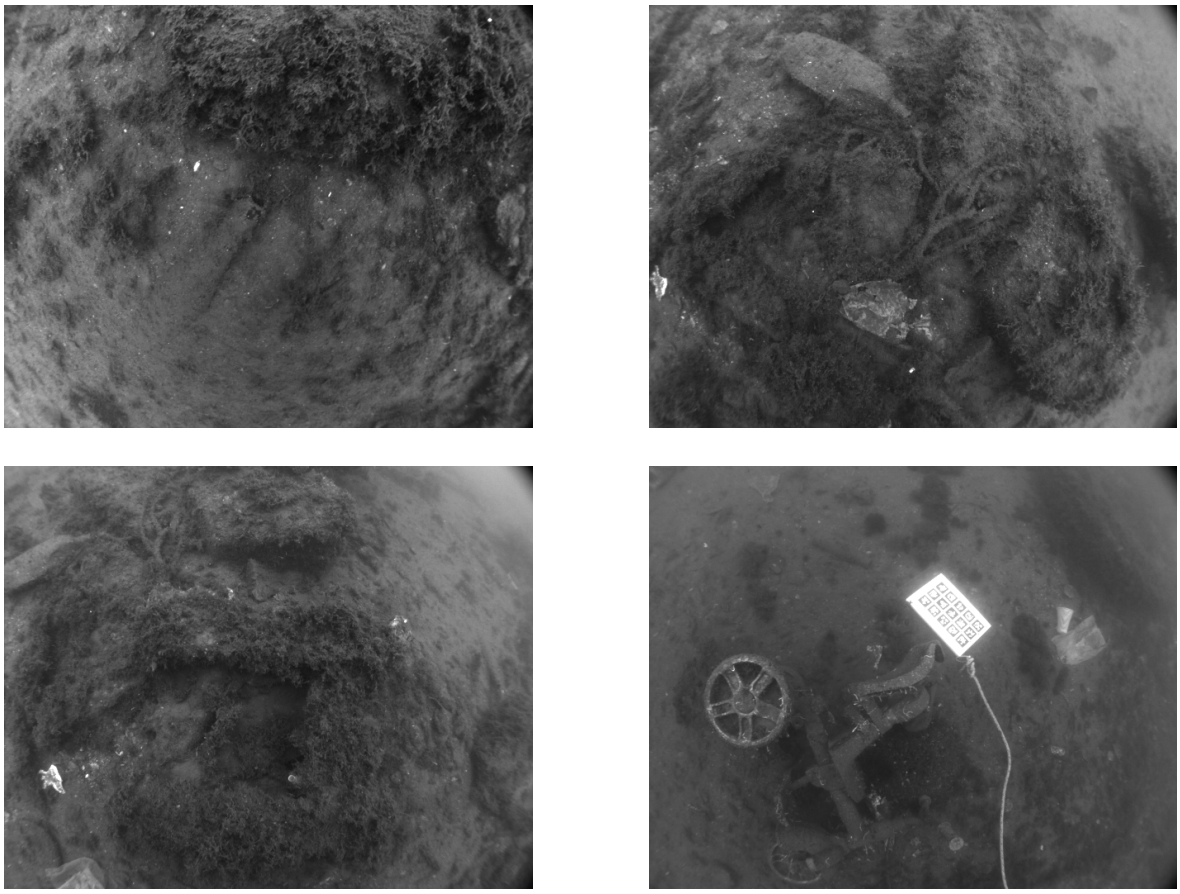
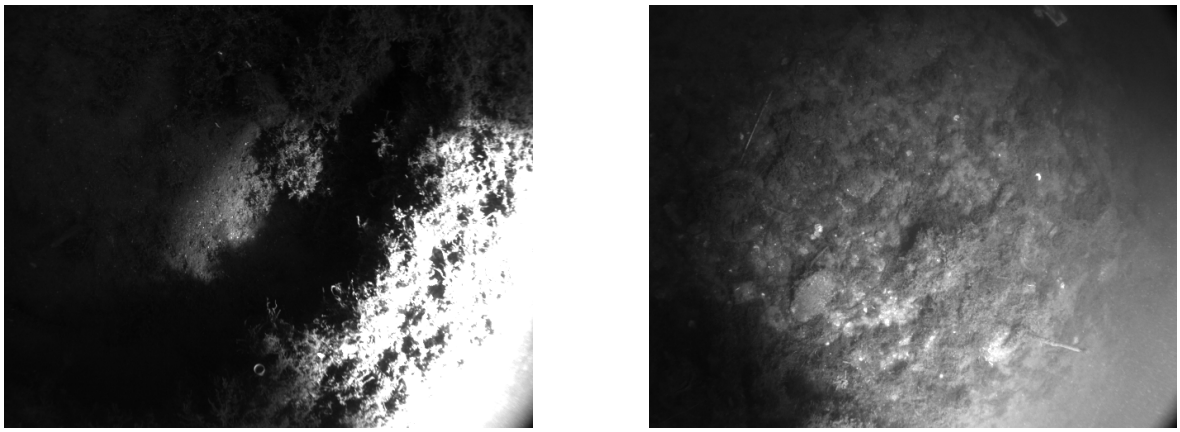


Figura A.10: Imagens da sequência 3.

Sequência 4



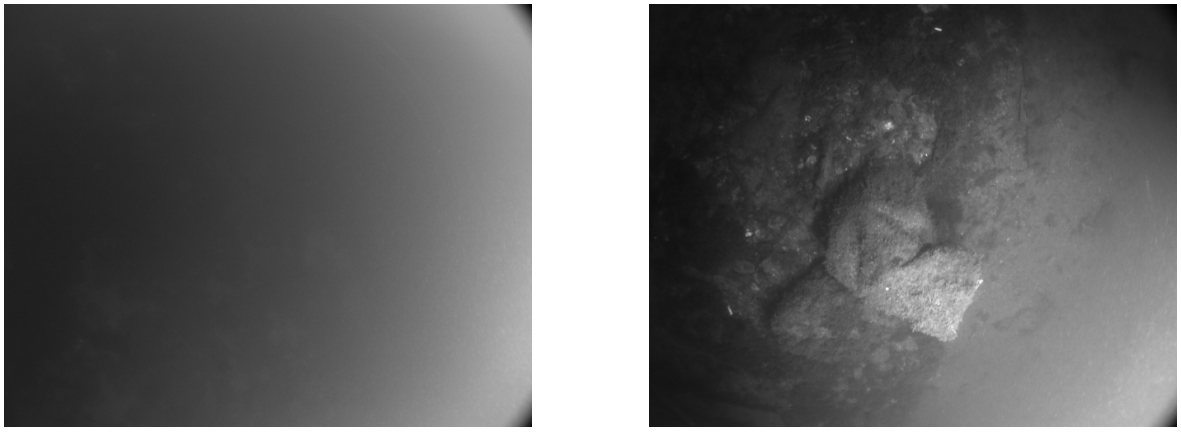


Figura A.11: Imagens da sequência 4.

Sequência 5

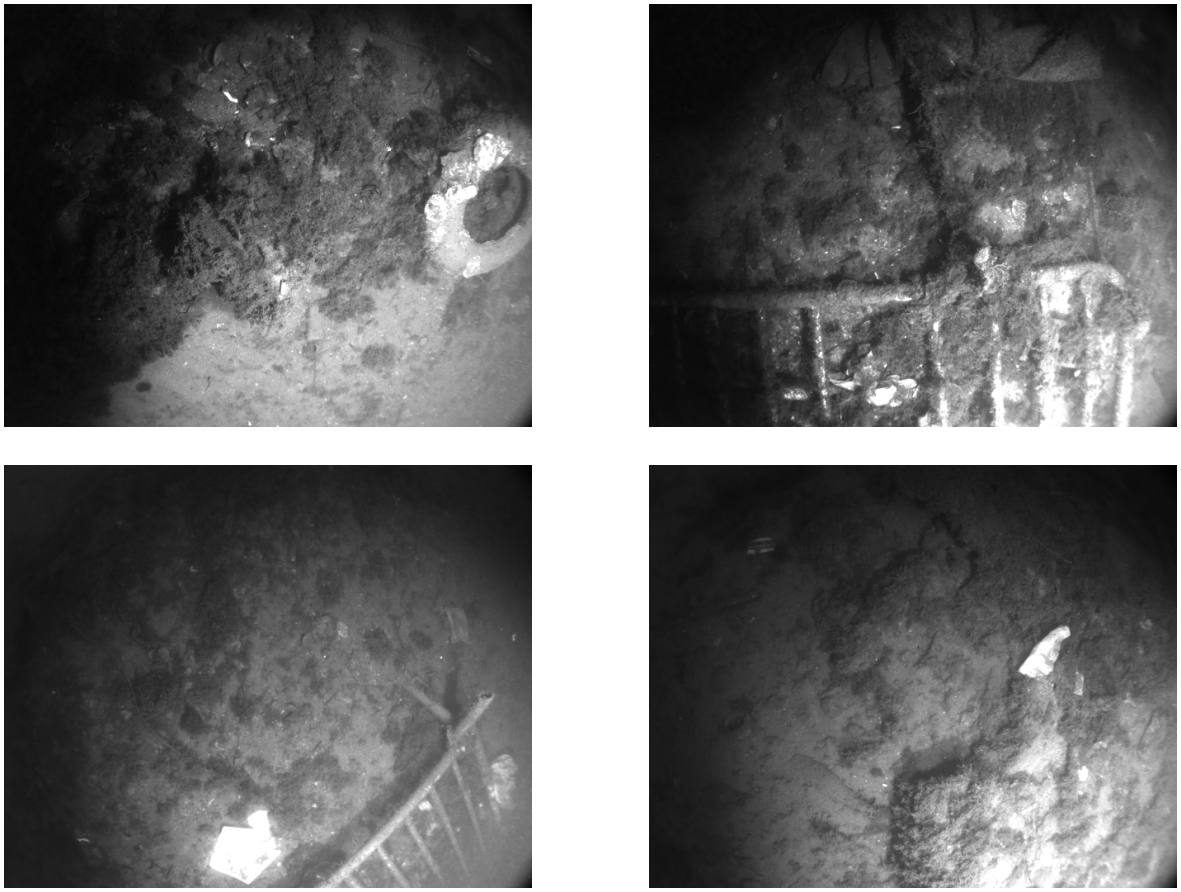


Figura A.12: Imagens da sequência 5.

Sequência 6

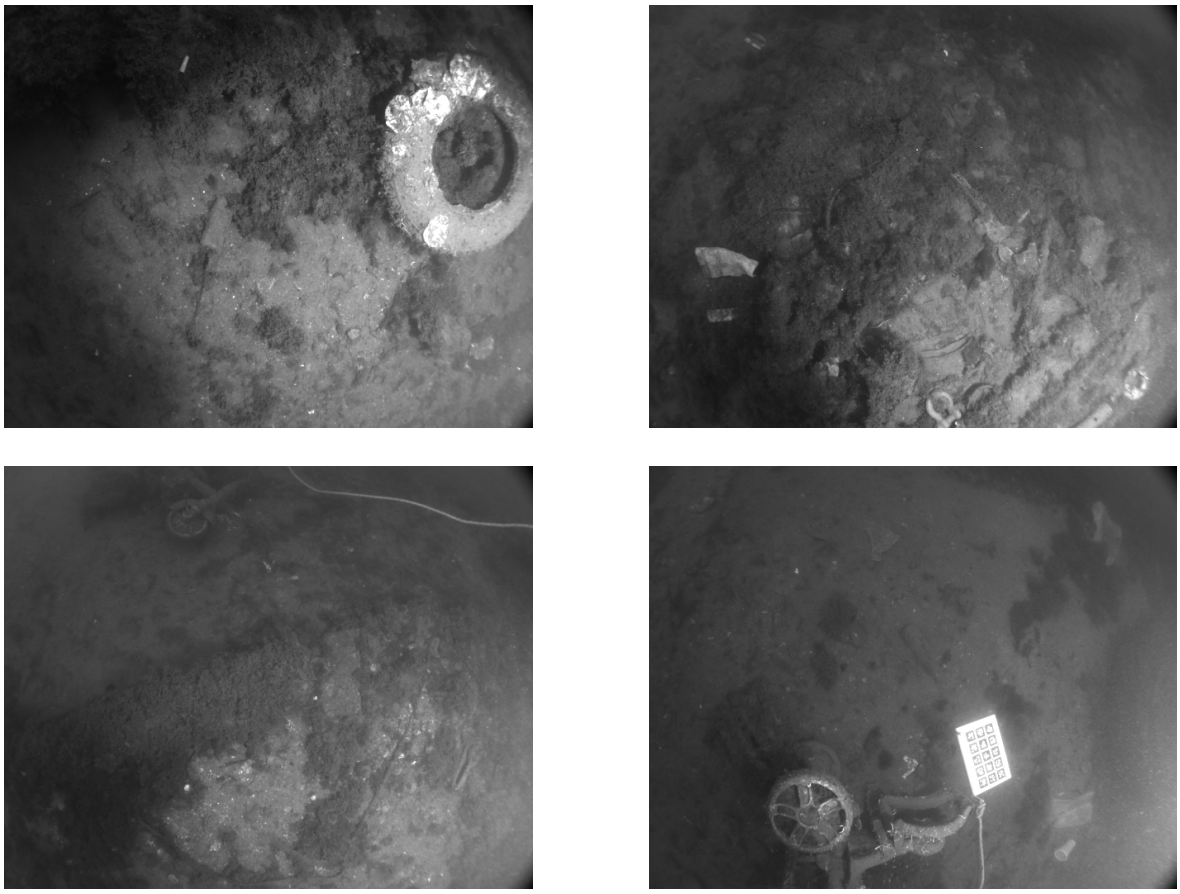
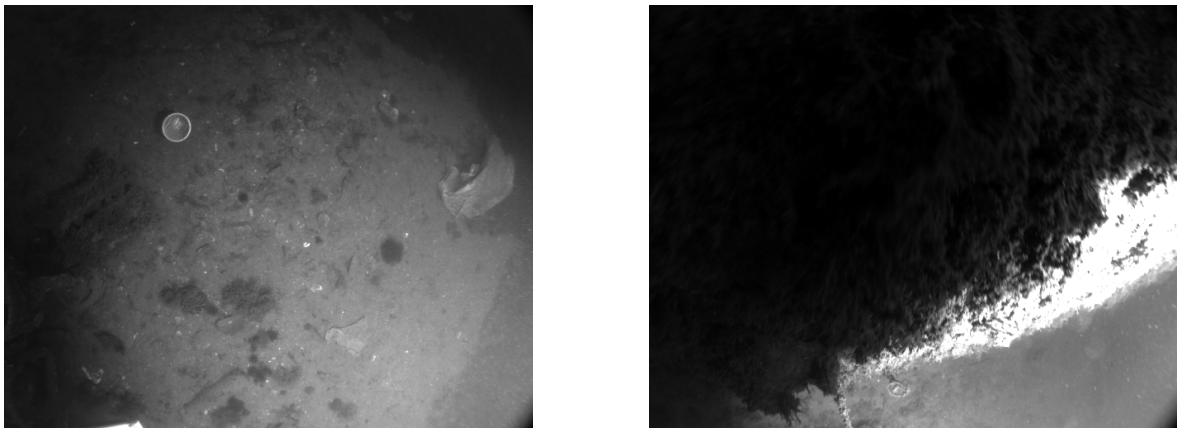


Figura A.13: Imagens da sequência 6.

Sequência 7



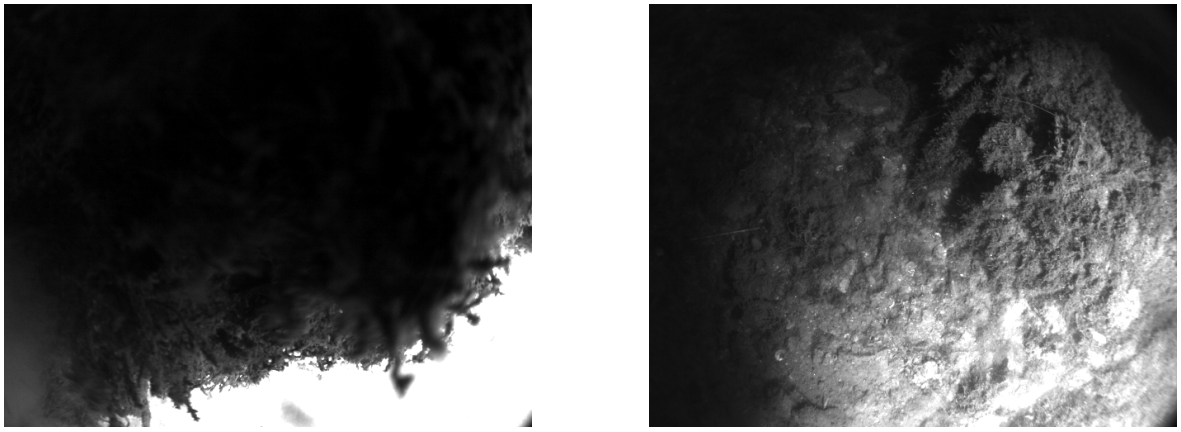


Figura A.14: Imagens da sequência 7.