Sistema de Mapeamento e Localização Simultânea para Inspeção Subaquática

Magda Alexandra de Mesquita Meireles Ribeiro

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização em Sistemas Autónomos

Laboratório de Sistemas Autónomos

Instituto Superior de Engenharia do Porto

Novembro de 2014

Tese/Dissertação, 2 º ano, Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: Magda Alexandra de Mesquita Meireles Ribeiro, Nº1120111, 1120111@isep.ipp.pt

Orientador: Alfredo Manuel Oliveira Martins, aom@lsa.isep.ipp.pt

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização em Sistemas Autónomos

Laboratório de Sistemas Autónomos

Instituto Superior de Engenharia do Porto

Novembro de 2014

Agradecimentos

No culminar desta etapa da minha vida académica, é com muita satisfação que expresso aqui o mais profundo agradecimento aos meus pais por todo apoio e confiança prestados ao longo destes dois anos.

Agradeço a todos professores que me acompanharam, em especial ao Eng. Alfredo Martins, Eng. José Almeida e Eng. André Dias pela ajuda e constante apoio na execução desta dissertação.

Aproveito também para agradecer aos meus amigos e colegas e claro ao André Pinto por toda a paciência e apoio demonstrados.

"Success is the ability to go from failure to failure with no loss of enthusiasm"

Winston Churchill

Resumo

A sequente dissertação resulta do desenvolvimento de um sistema de navegação subaquático para um *Remotely Operated Vehicle* (ROV). A abordagem proposta consiste de um algoritmo em tempo real baseado no método de Mapeamento e Localização Simultâneo (SLAM) a partir de marcadores em ambientes marinhos não estruturados. SLAM introduz dois principais desafios: (i) reconhecimento dos marcadores provenientes dos dados *raw* do sensor, (ii) associação de dados. Na deteção dos marcadores foram aplicadas técnicas de visão artificial baseadas na extração de pontos e linhas.

Para testar o uso de *features* no visual SLAM em tempo real nas operações de inspeção subaquáticas foi desenvolvida uma plataforma modificada do RT-SLAM que integra a abordagem EKF SLAM. A plataforma é integrada em ROS *framework* e permite estimar a trajetória 3D em tempo real do ROV VideoRay Pro 3E até 30 fps. O sistema de navegação subaquático foi caracterizado num tanque instalado no Laboratório de Sistemas Autónomos através de um sistema *stereo* visual de *ground truth*.

Os resultados obtidos permitem validar o sistema de navegação proposto para veículos subaquáticos. A trajetória adquirida pelo VideoRay em ambiente controlado é validada pelo sistema de *ground truth*. Dados para ambientes não estruturados, como um gasoduto, foram adquiridos e obtida respectiva trajetória realizada pelo robô. Os dados apresentados comprovam uma boa precisão e exatidão para a estimativa da posição.

Palavras-chave: Sistema de Navegação Subaquático, SLAM, EKF SLAM, ROV, *ground truth*.

Abstract

This thesis presents a subaquatic navigation system for a Remotely Operated Vehicle (ROV). The proposed approach consists in a real time algorithm based in Simultaneous Localisation and Mapping (SLAM) with landmarks in unstructured subaquatic environments. SLAM introduces two main challenges: (i) to recognize the landmarks from raw sensor data, (ii) data association . In the landmark detection process a artificial vision techniques based on point and line extraction was applied.

To test the use of environment features in real-time visual slam underwater robotic inspection operations was developed a modified implementation of the RT-SLAM approach that integrates EKF SLAM. The platform is integrated within a ROS framework and allows to estimate an underwater remote operated vehicle trajectory (ROV) in real-time (at 30fps). The subaquatic navigation system was characterized in a teste tank at Autonomous Systems Laboratory.

The obtained results allows to validate the proposed navigation system for subaquatic vehicles. The acquired trajectory by the ROV in the controlled environment it is validated by ground truth. Results for unstructured subaquatic environments were also acquired, such as a pipeline, in the sea bottom scenario were the performed robot trajectory is consistently obtained by the system. The presented data validates the accuracy and precision for position estimation.

Keywords: Subaquatic Navigation System, SLAM, EKF SLAM, ROV, ground truth.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

AUV *Autonomous underwater vehicle*

ASV *Autonomous Surface Vehicle*

GPS *Global Positioning System*

ROV *Remotely Operated Vehicle*

SLAM *Simultaneos Localization and Mapping*

EKF *Extended Kalman Filter*

ISEP Instituto Superior de Engenharia do Porto

LSA Laboratório de Sistemas Autónomos

PF *Particle Filter*

IMU *Inertial measurement unit*

SURF *Speeded-Up Robust Features*

SIFT *Scale Invariant Feature Transform*

HSV Hue Saturation Value

PTAM *Parallel Tracking and Mapping*

ROI *Region of interest*

CAPÍTULO 1

Introdução

1.1 Âmbito da Dissertação

A crescente utilização de veículos autónomos e remotamente controlados deve-se às respectivas caraterísticas que permitem a sua aplicação em tarefas subaquáticas que requerem situações arriscadas ou acesso a locais inóspitos para o Homem. A robótica subaquática tem igualmente um grande impacto no sector comercial da indústria hidrográfica [5], onde ROVs (*Remotely Operated Vehicle*) e AUVs (*Autonomous Underwater Vehicle*) são frequentemente utilizados na intervenção em estruturas imersas ou levantamento de áreas de grande variedade. Os maiores utilizadores privados [5] são as empresas de gás e petróleo, onde ROVs são destinados à realização de tarefas como inspeção submarina, construção e operações de reparação. Contudo, os veículos subaquáticos são também utilizados em situações diversas como em missões militares e estudos científicos (biologia marinha, oceanografia, exploração da plataforma oceânica monitorização ambiental).

Nos últimos anos, tém-se assistido a um significativo esforço da comunidade robótica no desenvolvimento de soluções para a navegação de veículos subaquáticos [5]. Em robótica móvel uma resposta frequente para este problema em cenários não-estruturados e desconhecidos refere-se às técnicas de *Simultaneous Localization and Mapping* (SLAM). SLAM consiste na capacidade de um robô móvel construir incrementalmente um mapa coerente do ambiente, enquanto, simultaneamente, determina a sua localização dentro

do mesmo mapa. A solução para o problema probabilístico de SLAM é um elemento essencial para tornar possível um robô móvel completamente autónomo [6].

As características do ambiente subaquático representam um conjunto de problemas no desenvolvimento de sistemas de navegação. A hostilidade do meio vai limitar a oferta sensorial, já que o *Global Positioning System* (GPS) e sensores de odometria habitualmente utilizados em abordagens SLAM são pouco utilizáveis. Os sensores visuais são inclusivamente afetados pela falta de visibilidade e desfocagem prejudicando a textura da imagem adquirida pelo robô subaquático.

Esta dissertação aborda a navegação de veículos autónomos subaquáticos em ambientes não estruturados. Em particular endereça o problema de "Visual SLAM" neste tipo de veículos, ou seja a utilização de informação visual para a resolução do problema de localização e mapeamento simultâneo. Neste projeto foi desenvolvida uma abordagem do RT-SLAM [7] em plataforma ROS [8] para caracterização da navegação de um pequeno *Remotely Operated Vehicle* (ROV).

1.2 Domínio

A formulação do problema probabilístico de SLAM ocorreu em 1986, no IEEE Robotics and Conference Automation realizada em São Francisco. A resolução do problema foi vista como o meio necessário para tornar um robô móvel completamente autónomo. Contudo, a estrutura do método probabilístico com o acrónimo "SLAM" [9] apenas foi apresentado pela primeira vez em papel em 1995 no Simpósio Internacional sobre Investigação Robótica.

O método probabilístico SLAM permite a estimação da trajetória de uma plataforma robótica assim como a determinação da localização de todos os pontos de referência, sem a necessidade de qualquer conhecimento à priori da localização. Na

resolução deste problema são utilizadas abordagens probabilísticas que lidam com a incerteza inerente ao processo e às observações [6].

Ao longo dos anos a formulação do problema teórico SLAM tem sido alvo de inúmeras abordagens. Algumas contribuições focam a eficiência computacional através de técnicas de sub mapeamento, assegurando estimativas consistentes e precisas para o mapa e posicionamento do veículo. Outros estudos abordam a aplicação em sistemas não lineares, associação de dados e caracterização de *landmarks*, que são vitais para alcançar uma implementação prática e robusta de SLAM [1].

1.2.1 EKF SLAM versus FastSLAM

A possibilidade de ter veículos verdadeiramente autónomos depende fortemente da sua capacidade de construir modelos precisos de mapas dos ambientes que atravessam assim como da posição dos marcadores de referência. As soluções disponíveis na distribuição de probabilidade SLAM incluem a implementação de abordagens baseadas no *Extended Kalman Filter* (EKF) [10] e *Particle Filter* (PF) [11].

EKF SLAM é a designação atribuída à solução que faz uso do EKF. O algoritmo de distribuição probabilística EKF tem um comportamento monoticamente convergente [6]. Ou seja, aumentando o número de marcadores diminui a quantidade de erro devido às características da matriz de covariância de EKF. Em contrapartida, a etapa de atualização da observação exige que todos os marcadores e respetiva matriz covariância sejam atualizados a cada vez que uma observação é feita. Desta forma, o processamento computacional aumenta ao quadrado com número de marcadores,

$$processamento = k(marcador^2) + C \tag{1.1}$$

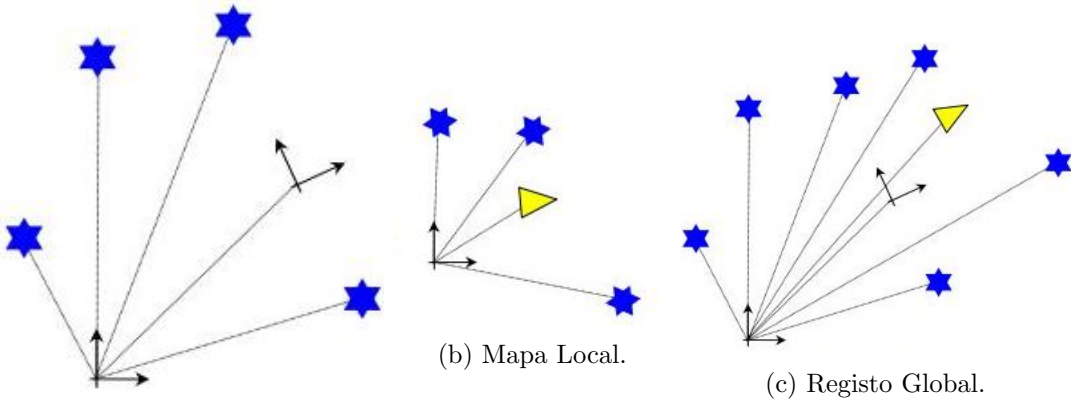
Para colmatar esse problema algumas implementações recorrem à utilização de técnicas de sub mapeamento [12] (documentado na Secção 1.2.2).

Considerando uma situação em que um robô descreve uma trajetória em ciclo ("loop") para re-observar os pontos de referência, após uma grande travessia, pode ocorrer uma incorreta associação dos marcadores. A solução referida é especialmente frágil quando os marcadores medidos pelos sensores são demasiado simples. A aplicação da solução EKF SLAM apresenta outro inconveniente, a linearização de modelos não lineares de movimento e de observação. Este processo pode originar resultados inconsistentes.

A primeira abordagem a representar diretamente o modelo de processo não linear e de distribuição não gaussiana é baseada no método de Monte Carlo recursivo, FastSLAM [10]. Neste processo a trajetória é representada por amostras ponderadas e o mapa é calculado analiticamente. Ou seja, a estimativa recursiva é baseada na filtragem de partículas. No entanto, a grande dimensão do espaço de estados da filtragem de partículas torna a computação não praticável [6]. Contudo, esse problema pode ser minimizado se for reduzido o espaço de amostragem utilizando o algoritmo Rao-Blackwellisation [13]. O filtro Rao-Blackwellisation corresponde a uma forma de melhorar a eficiência computacional através da redução da complexidade de cada amostra. Este método reduz o número de variáveis recolhidas, identificando quais as variáveis que não interessam à computação.

1.2.2 Técnicas de sub mapeamento

A abordagem direta para a redução da complexidade computacional envolve a exploração do problema SLAM, reformulando o tempo essencial das equações de observação e atualização. As soluções para melhorar o processamento consistem em reformular as equações de atualização de tempo e as equações de atualização das observações. A limitação das equações de atualização das observações poderá ser realizada usando partições (sub mapas) [1]. O método de sub mapeamento (Observar Figura 1.1) consistem na ideia de que um mapa pode ser dividido em regiões com sistemas de coordenadas locais e organizados de forma hierárquica.



(a) Mapa Global.

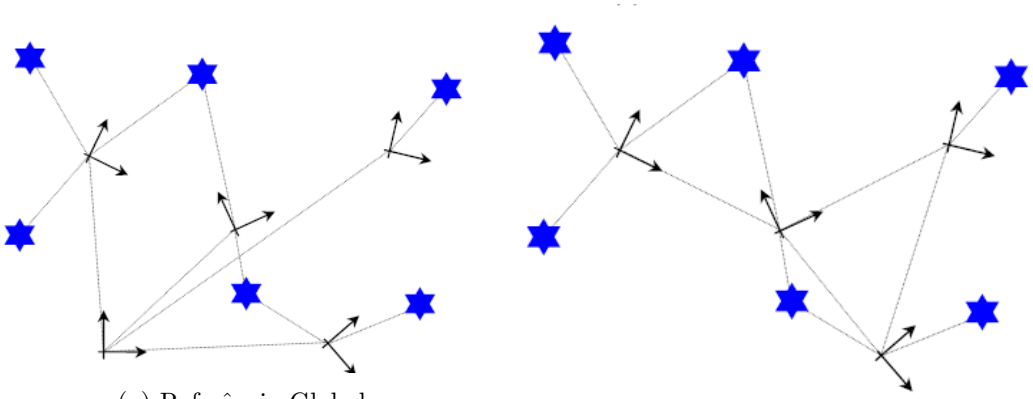
(b) Mapa Local.

(c) Registo Global.

Figura 1.1: A fronteira SLAM é construído em um mapa local (b) que regista periodicamente com um mapa global (a) para produzir uma estimativa global ideal (c) (adaptado de [1]).

Os métodos de sub mapeamento podem ser de dois tipos: globais e relativos. Os globais são referenciados a partir de um ponto de referência global. E os relativos são referenciados relativamente entre eles. Ou seja, os sub mapas globais partilham a mesma coordenada enquanto os sub mapas relativos guardam a posição do sub mapa vizinho. Na Figura 1.2 encontram-se ilustradas as técnicas de sub mapeamento global e relativo respetivamente.

A aplicação de mapas particionados apresenta inúmeras vantagens, o número de *landmarks* a ser atualizadas a qualquer momento limita-se apenas à quantidade descrita no sub mapa das coordenadas locais. Assim, a atualização completa, e propagação de estimativas locais, pode ser realizada como uma tarefa de fundo a uma taxa de atualização muito mais baixa. Uma vantagem dos mapas de referência relativa, reside no não acréscimo do valor da incerteza na obtenção da localização global do robô móvel. No entanto, os sub mapas de referência global não diminuem problemas de linearização decorrentes de grandes incertezas da posição do robô móvel. Enquanto os sub mapas relativos atenuam esses problemas, uma vez que produzem mapas localmente ótimos com complexidade computacional independente do tamanho do mapa completo.



(a) Referência Global.

(b) Referência Relativa.

Figura 1.2: Técnicas de sub mapeamento com referência global (a) e relativa (b) (adaptado de [1]).

1.3 Motivação e Enquadramento

O sistema de navegação subaquático apresentado surge da sequência de projetos desenvolvidos ao longo dos anos no Laboratório de Sistemas Autónomos (LSA) do ISEP. O LSA tem vindo a desenvolver atividades de investigação na área de sistemas autónomos, tais como navegação, controlo e atividades de coordenação múltiplas entre robôs. Na robótica aquática destaca-se a utilização à superfície do *Autonomous Surface Vehicle* (ASV) ROAZ II e a plataforma subaquática ROV VideoRay Pro 3E.

Os recursos proporcionados pelo laboratório (tanque de testes) permitem um conjunto de condições benéficas para o desenvolvimento de projetos no âmbito da robótica subaquática. A facilidade disponibilizada resultou em trabalhos recentes na área do controlo [14], e na instalação do sistema INESC TEC/ISEP *indoor ground truth* [2] no tanque de testes. Destes esforços surge a necessidade de projetar um sistema de localização em tempo real, implementável em plataformas como o ROV, que permitam a associação aos projetos previamente existentes. A implementação de um sistema de navegação e localização subaquático requerer a validação dos dados a partir do sistema

de perceção visual INESC TEC/ISEP *indoor ground truth*.

A modularidade de um sistema de navegação baseado na detecção de *features* visuais em cenários não estruturados permite a sua utilização em veículos com características não subaquáticas. Veículos terrestres também poderão usufruir das potencialidades, claro que numa versão adaptada, da abordagem proposta nesta tese.

1.3.1 Cenários de Operação

Os ambientes de navegação nem sempre oferecem as condições necessárias para a implementação de algoritmos de localização que dependam de um determinado padrão das infra-estruturas a inspecionar. Com o intuito de colmatar estes problemas, recentemente foram apresentadas algumas soluções para ambientes parcialmente estruturados [15, 16].

A abordagem proposta não requer necessariamente padrões e/ou texturas sistemáticas padronizadas na obtenção da localização do robô subaquático. No entanto, requer estruturas visíveis que permitam a obtenção de marcadores visuais. Os cenários onde é aplicável o referido sistema incluem:

- Portos Naúticos e Docas (Na verificação do estado dos cascos das embarcações, monitorização da poluição, observar Figura 1.3);
- Barragens (Inspeção do estado das infra estruturas);
- Industria petrolífera e gás (Manutenção e inspeção de oleodutos e gasodutos).

Figura 1.3: Doca, cenário de aplicação.

1.4 Objetivos

O objetivo fundamental da dissertação visa contribuir para o desenvolvimento de um sistema de navegação subaquático, baseado em informação proveniente de sensores do espectro visual, tendo em conta que o algoritmo proposto deverá basear-se no método probabilístico SLAM.

Os objetivos propostos para esta dissertação são os seguintes:

1. Desenvolver sistema de mapeamento e localização simultâneo que permita a obtenção da localização do robô móvel subaquático.

   (a) Detetar *features* dos diferentes cenários, no caso pontos e linhas, a partir da informação sensorial recebida da plataforma robótica;

2. Implementar sistema que permita o registo do sistema proposto sincronizado com o Sistema de *ground truth* [2] no tanque de testes;

3. Desenvolver ferramentas que permitam avaliar e corrigir o *performance* do sistema de navegação durante a realização de testes;

4. Avaliar a solução proposta para os vários cenários de aplicação.

1.5 Estrutura do Documento

Este documento é composto por oito capítulos organizados da seguinte forma:

- O segundo capitulo (Capítulo 2) consta de um relato e respetiva análise de técnicas aplicadas em SLAM nas diversos áreas de aplicação e em especial para aplicações subaquáticas.
- A formulação do problema elementar do desenvolvimento deste projeto e respetiva análise de requisitos do mesmo são realizadas no Capítulo 3.
- No Capítulo 4 são apresentados os conceitos base em visão computacional, estes consistem nos fundamentos base na aplicação do sistema desenvolvido.
- A técnica *Simultaneous Localization and Mapping* é descrita no Capitulo 5, assim como respectivos métodos de estimação.
- No sexto capítulo (Capítulo 6) são discutidos aspetos importantes da implementação realizada como arquitetura e aspectos ligados ao *software* da aplicação.
- O Capítulo 7 é composto pela apresentação das experiências realizadas com respetivos resultados e análise dos mesmos.
- Por fim, no o último capítulo (Capítulo 8) é apresentada a conclusão e cumprida a respetiva discussão dos resultados obtidos. Assim como, uma apreciação geral do trabalho desenvolvido e sugestão de trabalhos futuros.

CAPÍTULO 2

Estado da Arte

2.1 Sistemas de Navegação SLAM

A implementação de um sistema de navegação que permita a um robô subaquático movimentar-se autonomamente requer a garantia que a sua localização seja calculada com elevada exatidão. A localização do veículo procura responder à pergunta "onde estou?", ou seja determinar a posição cartesiana assim como a orientação do robô em relação a um referencial do Mundo. Uma das formas de resolver este problema é através da técnica denominada de *Simultaneous Localization and Mapping* (SLAM). A aplicação de SLAM exige o recurso aos sensores do robô para extrair pontos de referência do meio envolvente (*landmark*) e determinar a sua posição. A estimação da posição não requer conhecimento à priori do Mundo onde o robô subaquático é inserido.

Alguns contributos na implementação de sistemas de navegação baseados em SLAM abordam propostas para aplicações terrestres, no domínio da aplicação de modelos multi sensoriais e questões de estimação. As abordagens SLAM em meio subaquático focam outros problemas relacionados com a aplicação de visual SLAM e sistemas em tempo real. Visual SLAM implica a implementação de algoritmos de extração de *landmark* visuais para a estimação do posicionamento do veículo. Os algoritmos SLAM baseados na deteção de *features* visuais exigem um elevado processamento computacional. Alguns contributos têm sido apresentados para reduzir esses problemas e permitir aplicações em tempo real em ambiente subaquático.

2.1.1 Aplicações terrestres SLAM

As primeiras implementações em plataformas robóticas SLAM foram atribuídas a robôs móveis terrestres. Sensores ultrasónicos são utilizados isoladamente na caracterização do ambiente de teste em [17,18]. No entanto, a utilização exclusiva de um sensor tem algumas limitações na correção do erro da trajetória obtida. Alguns contributos descritos em [19] demostram a importância da utilização de sistemas multi sensoriais na caracterização do ambiente. A referida abordagem apresenta a utilização de um sistema multi sensorial composto de um laser 2D e de uma camara CCD. Experiências reais demostram contribuição na re-observação e estimação quando os veículos descrevem uma trajetória em "loop" (ciclo).

O *open source* Hector SLAM [20] refere-se a um exemplo da utilização combinada dos dados provenientes do laser scanner com um sistema de estimativa de posição 3D através da aplicação de uma unidade IMU na correção da posição do robô móvel [21]. O sensor inercial permite a correção do fator de escala durante a localização do veículo no Mundo. O algoritmo Hector SLAM tem também uma particularidade que o destaca relativamente a outras abordagens, os respetivos módulos de *software* encontram-se implementados em plataforma *Robot Operating System* (ROS) [8]. Além disso, esta abordagem SLAM apresenta um conjunto de módulos de *software* que fornecem as ferramentas necessárias para aplicações *Urban Search and Rescue* (USAR). Na competição RoboCup 2013, este algoritmo foi aplicado no veículo Hector UGV Ackermann-direccionais e Hector UGV Lightweight (Figura 2.1).

O algoritmo SLAM não consiste da única aplicação que permite a localização e mapeamento de um robô móvel em ambientes desconhecidos. Outra técnica baseada na deteção de pontos de referência a partir da aquisição de dados de um câmara é denotada de *Parallel Tracking and Mapping* (PTAM). O PTAM [29] consiste num método

Figura 2.1: Veículos Hector, à esquerda Hector UGV Ackermann-direccionais, à direita Hector UGV Lightweight.

de estimação da posição de uma câmara numa cena desconhecida. O algoritmo divide o *tracking* e mapeamento em duas tarefas distintas, processadas em segmentos paralelos de um computador *dual-core*. Portanto, uma *thread* lida de forma robusta com a tarefa de *tracking* de movimento, enquanto a outra *thread* produz um mapa 3D das *features* de pontos previamente observados das *frames* adquiridas. Isso permite o uso de técnicas de otimização com um elevado processamento computacional. Com PTAM é obtido um sistema que produz mapas detalhados com milhares de pontos de referência, em simultâneo o sistema de tracking recebe imagens da câmara e mantém uma estimativa em tempo real da posição da mesma [29].

2.1.2 Aplicações subaquáticas SLAM

Conforme já foi referido as questões de associação e estimação em SLAM atualmente são consideradas resolvidas [1]. Desta forma, muitos dos contributos documentados para sistemas de localização e mapeamento subaquáticos abordam a aplicação de técnicas de extração de *landmarks*. Algumas propostas contribuem com métodos eficientes para a deteção de *features* em cenários típicos de operação como barragens ou

portos náuticos. Outras propostas contribuem com a aplicação de soluções para reduzir o processamento computacional e permitir a aplicação em tempo real.

Um dos grandes contributos na navegação subaquática pertence a David Ribas [15,16,22], que apresenta uma abordagem projetada para ambientes parcialmente estruturados, tais como barragens, portos, marinas e plataformas marítimas. Em ambientes subaquáticos podem ser aplicadas soluções hibridas [22] a partir da combinação de informação visual e dados de um sonar. Neste algoritmo é utilizado um sonar de imagem digitalizada (*Mechanically Scanned Imaging Sonar* - MSIS) para obter informação sobre a localização de estruturas planas verticais presentes em tais ambientes. Depois, um algoritmo de referência baseado na transformada de Hough [23], utiliza a característica da linha, em conjunto com as suas incertezas, a partir do fluxo de dados do sonar e integra a informação obtida num EKF. Por forma a ilustrar esta abordagem, um modelo de extração de linha pode ser observado na Figura 2.2.

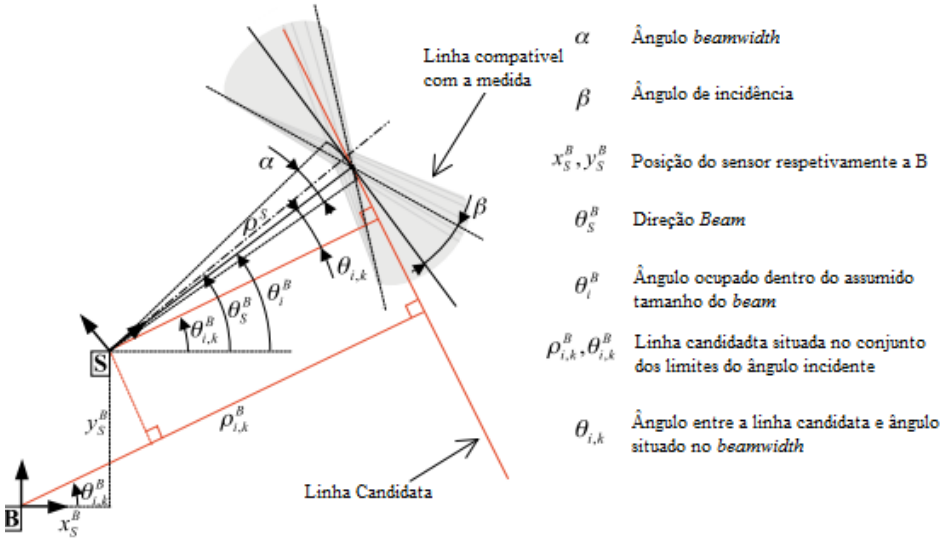
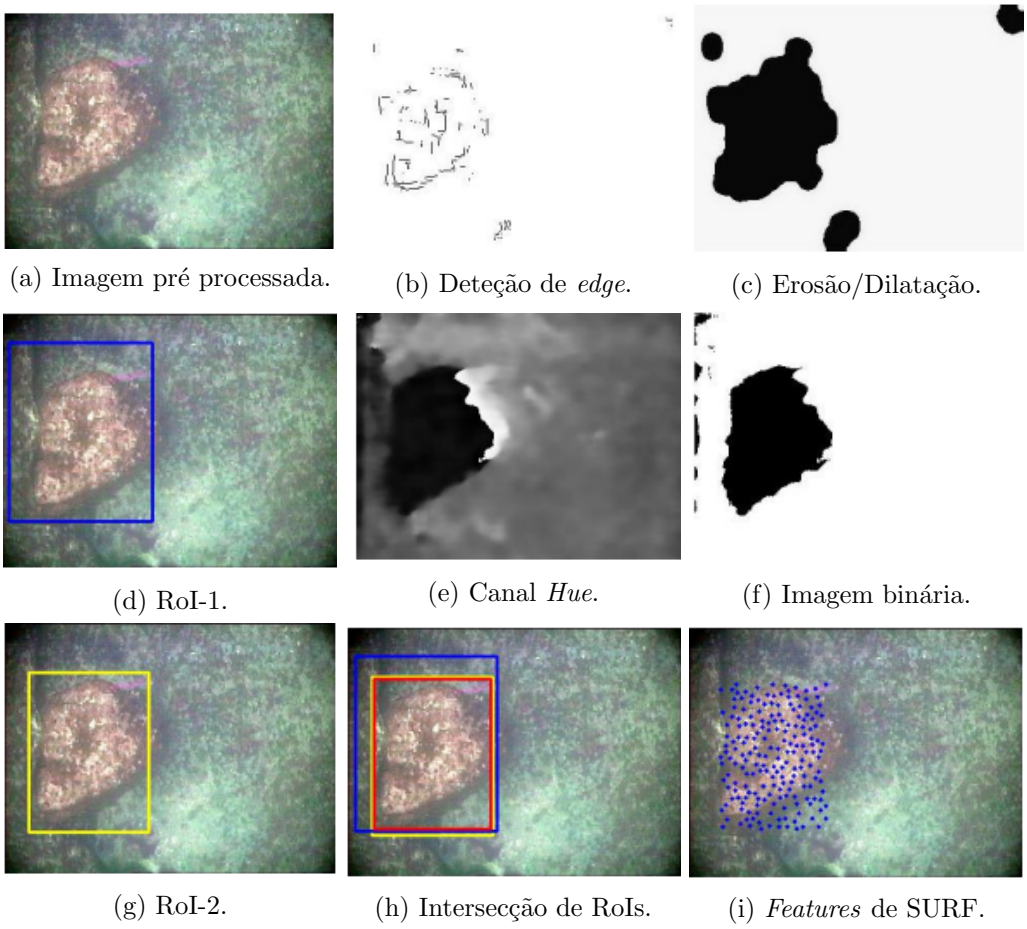


Figura 2.2: Modelo do sensor sonar às linhas de candidatas compatíveis. B é a *frame* de referência de base, e S é um referencial ligado à *head* de transdutor do sonar (adaptado de [22]).

Os ambientes não estruturados subaquáticos também têm sido alvo de estudo por parte de investigadores na área da robótica móvel. No contributo documentado no artigo "Feature extraction for underwater visual SLAM" [24] utilizam-se três fontes de informação: 1) informação topológica tridimensional de SLAM; 2) informações de contexto para caracterizar as regiões de interesse (*Region of interest* (ROI)); e 3) extração de *features* dessas ROIs (Figura 2.3). Informação topológica é feita a partir do algoritmo de SLAM, isto é, a posição tridimensional aproximadas do ponto de referência com um certo grau de incerteza.



(a) Imagem pré processada. (b) Deteção de *edge*. (c) Erosão/Dilatação.

(d) RoI-1. (e) Canal *Hue*. (f) Imagem binária.

(g) RoI-2. (h) Intersecção de RoIs. (i) *Features* de SURF.

Figura 2.3: Processo para extrair regiões de interesse (ROI). A ROI final corresponde à assinalada a vermelho em h), e suas características SURF são mostrados em azul em i) (adapatdao de [24]).

A informação contextual é obtida através da segmentação da imagem em segundo plano e aplicação de ROIs. A extração de pontos de interesse das imagens são calculadas usando métodos de deteção de *features* da imagem, como *SIFT* [25] e *SURF* [26]. Esta informação é utilizada para associar novas observações com marcadores conhecidos, obtidos a partir de observações anteriores. Os resultados foram obtidos através de experiências realizadas no veículo autónomo subaquático Sparus AUV. O processamento aplicado às imagens adquiridas do veículo subaquático inicia-se com a deteção de *edge*, produzindo a imagem binária mostrada na Figura 2.3b. Posteriormente, as operações de erosão/dilatação são realizadas, unindo regiões e eliminando os pontos insignificantes (Figura 2.3c). O passo seguinte consiste na pesquisa de uma região dentro da imagem a preto e branco, produzindo a segmentação mostrado na Figura 2.3d. Por outro lado, o segundo processo utiliza o canal *Hue* do sistema de cores HSV (Figura 2.3e). Neste canal é aplicado "blur" a fim de suavizar a imagem inteira. Em seguida, é imposto um *threshold*, obtendo-se os resultados apresentados na Figura 2.3f. Este limiar é escolhido de acordo com o valor médio da imagem Hue. Posteriormente, é aplicada novamente uma região de pesquisa, conforme ilustrado na Figura 2.3g. Por fim, ambos os processos são fundidos: a ROI final é selecionada através do cruzamento de ambas as segmentações (ver Figura 2.3h).

Na mesma linha de investigação refere-se o trabalho apresentado por Ayoung Kim, desenvolvido para a vigilância subaquática da superfície de um casco de navio em tarefas de deteção de objetos estranhos, de inspeção e manutenção automática. O algoritmo denotado de "Pose-graph Visual SLAM" [27] utiliza na percepção um sistema constituído de uma câmara monocular calibrada montada num atuador de inclinação para que a câmara mantenha aproximadamente uma visão nadir ao casco. O método de deteção de *features* visuais consiste numa combinação dos algoritmos *SIFT* e *Harris* [28] baseado num quadro de registo de pares de imagens utilizadas para a obtenção da posição relativa do robô [27].

As aplicações subaquáticas de SLAM que foram apresentadas são conduzidas de modo *offline*, contudo algumas situações exigem a aplicação de algoritmos em tempo real. O MonoSLAM [7] é um algoritmo em tempo real que permite a obtenção de uma trajetória 3D de uma câmara monocular, movendo-se rapidamente através de um ambiente previamente desconhecido. Esta é a primeira aplicação da robótica móvel, em tempo real, bem sucedida para aplicações com uso exclusivo de uma única câmara. As principais contribuições incluem uma abordagem ativa para o mapeamento e medição, o uso de um modelo geral para o movimento da câmara, e as soluções para a inicialização de *features* e estimativa de orientação da *feature*.

As aplicações em tempo real também contaram com o contributo de Ayoung Kim novamente na inspeção subaquática de um casco de navio [30]. Neste algoritmo, aplicado na inspeção subaquática de um casco de navio, a deteção de pontos de referência é feita a partir de saliências encontradas no ambiente onde o robô móvel se encontra inserido. A técnica aplica duas medidas diferentes de saliência: saliência local (ou seja , intra imagem) e saliência global (isto é, inter imagem). Ambas são calculadas através da utilização de um paradigma de categorização de objetos, "bag-of-words" para a representação da imagem. Neste algoritmo são empregues os conceitos de saliência local e global como duas medidas diferentes de imagem de registo nesta secção. Para cada tipo de saliência, é representada uma curva "bag-of-words" (BoW) que, neste caso, consiste num histograma para a categorização de objeto. Posteriormente, são examinados histogramas para a medida da saliência local (frequência intra imagem), e marcado o nível de saliência, avaliando a sua entropia. Esta medida de entropia capta a diversidade de palavras (descritores) que aparecem dentro de uma imagem. Para saliência global, frequência inter imagem, é examinada a ocorrência de todas as imagens vistas anteriormente. Na Figura 2.4 encontra-se um exemplo de aplicação da saliência local para cores e tons de cinza das imagens sob diferentes perspetivas do casco de um navio. De notar que para imagens coloridas, o canal da matriz *Hue* permite distinguir a riqueza de *features* do ambiente, no entanto para tons de cinza verifica-se que o histograma de

intensidade da imagem não deteta riqueza de features, enquanto que o histograma BoW ainda assim funciona bem. Uma das complicações associadas a este algoritmos reside na textura da imagem que pode impedir o reconhecimento de determinadas saliências. Contudo, a textura não é o único fator que define saliência, por exemplo no caso de uma imagem de um padrão xadrez ou uma parede de tijolos. Imagens deste tipo têm alta textura, mas provavelmente irá falhar o registo devido a *aliasing* espacial de *features* comuns.

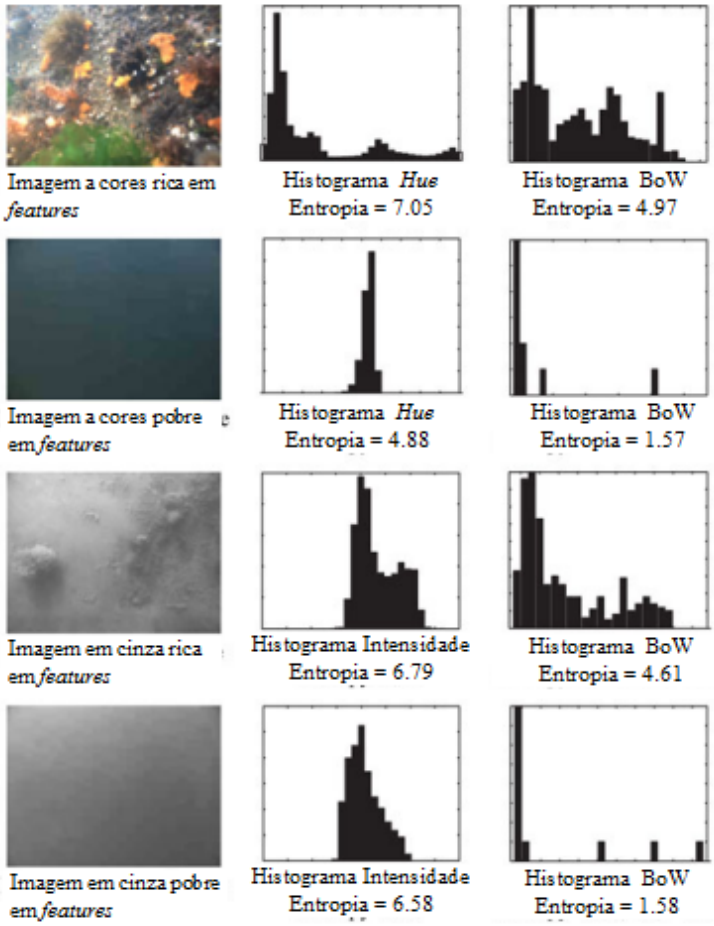


Figura 2.4: Exemplo saliência local. Em cada resultado, à esquerda mostra o histograma de intensidade da imagem, à direita mostra o histograma "bag-of-words" (adaptado de [30]).

Outros trabalhos [31] registam a construção em tempo real de um mosaico do fundo do mar dependente do *framework* SLAM. Em vez de usar um mosaico global, este trabalho utiliza a combinação de um conjunto de mosaicos locais construídos na proximidade dos pontos SLAM de referência visual. Outra das contribuições deste artigo trata-se do uso de *SURF* na detecção de *features* que permitem eliminar falsos positivos na associação de dados de marcadores visuais SLAM. A utilização de *SURF* permite um seguimento das *landmark* o que não acontece com abordagens de correlação puras [31]. De notar que nesta abordagem são utilizadas técnicas de sub mapeamento para corrigir os efeitos do "drift" no mosaico em tempo real.

2.2 Discussão

Os métodos SLAM constituem, hoje em dia, soluções viáveis para o mapeamento e localização de robôs móveis autónomos. As principais questões de representação, computação e associação de dados são, de uma forma geral, consideradas resolvidas. Assim, o desafio tem sido demonstrar soluções SLAM em ambientes mais complexos assim como em situações que requerem um algoritmo em tempo real. Os contributos apresentados de um conjunto de soluções para robôs móveis terrestres sem dúvida que tiveram um importante papel no desenvolvimento das aplicações em meio subaquático. A aplicação de sistemas multi sensoriais e em particular a utilização de um sensor inercial na correção da posição de um robô são propostas que têm um grande potencial para o desenvolvimento de sistemas de navegação subaquáticos.

As limitações sensoriais impostas em ambientes subaquáticos, determinaram o foco de pesquisa em algoritmos [15, 30, 31] de processamento de imagem. A implementação de métodos de deteção de features deve ser adequada aos ambientes subaquáticos onde são utilizados veículos como ROVs em tarefas de inspeção. Assim, a utilização de robôs móveis subaquáticos em barragens e portos impulsionou o desenvolvimento de algoritmos baseados na técnica Hough para ambientes semi estruturados. No entanto, para

ambientes não estruturados, os métodos de detecção de *features* apresentados utilizam técnicas de visão computacional tradicionais (*SIFT*, *SURF*, *Harris*) aliados a técnicas que permitem a redução de constrangimentos em tempo real, com a ROI. Novos conceitos também são introduzidos como o "bag-of-words", útil na caracterização de objetos por meio de paradigmas de categorização, neste estudo foi apresentado sob a forma de histogramas.

CAPÍTULO 3

Caracterização do Problema

3.1 Formulação do Problema

A necessidade de construir um sistema de navegação subaquático aplicável em plataformas robóticas e em tempo real formula a base de desenvolvimento da presente tese. O problema de posicionamento e localização deverá ser descrito segundo métodos probabilísticos. SLAM traduz-se na construção de um mapa coerente e simultânea localização dentro do mesmo.

O veículo submerso vai trazer um conjunto de novos desafios bastante diferentes aos encontrados em sistemas de navegação terrestre ou aéreos. A atenuação da radiação acústica e electromagnética ocorre rapidamente na água, restringindo a gama de sensores acústicos e assim como os limites na largura de banda de comunicação. Como consequência não é viável o uso de GPS e os sensores acústicos podem não oferecer uma solução eficaz de localização. A locomoção típica de robôs subaquáticos impede a utilização de sensores odométricos (*encoders*), que são habitualmente utilizados noutros cenários de aplicação SLAM. Os sistemas de navegação baseados em visão artificial são igualmente afetados, principalmente em ambientes não controlados, como em docas ou barragens. A qualidade das imagens pode ser severamente afetada pelas pela água menos límpida, e assim comprometer a fidelidade de resultados.

Apesar destas limitações, existem situações onde pela natureza da missão é ex-

pectável à disponibilidade de informação visual útil para navegação. São destas exemplos como tarefas de monitorização e inspeção junto a infra-estruturas ou operação com visibilidade do fundo marinho.

3.2 Análise de Requisitos

3.2.1 Visual SLAM

De acordo com as características do problema, a utilização de visual SLAM surge como requisito primário do sistema. A aplicação em meio subaquático exige uma solução baseada em visual SLAM devido as limitações impostas pelo ambiente. Mesmo tendo a aplicação de uma câmara como meio de detecção de *features* tenha a sua limitações no campo de visão ou na velocidade de processamento. A utilização de sensores do espetro visual ultrapassa problemas inerentes às condições propostas como a atenuação de ondas acústicas e eletromagnéticas.

3.2.2 Plataforma de testes

Para efeitos de teste foi utilizado o ROV videoRay PRO 3E, plaforma robótica presente no LSA (Figura 3.1). O veículo tem 3 DOF (surge, heave e sway), sendo projetado para profundidades até 152 metros. A plataforma robótica tem três propulsores de comando, dois para os movimentos horizontais e uma para o movimento vertical. Além disso, o veículo é equipado com duas câmaras, sem visão sobreposta, uma à frente e outra atrás, um sensor de pressão, uma bússola e altímetro.

3.2.3 Sistema de visão *ground truth*

Ao longo dos anos no Laboratório de Sistemas Autónomos (LSA) têm-se vindo a desenvolver vários projetos no domínio da robótica subaquática. As condições proporcionados pelo laboratório, permitiram o desenvolvimento de recursos úteis para o

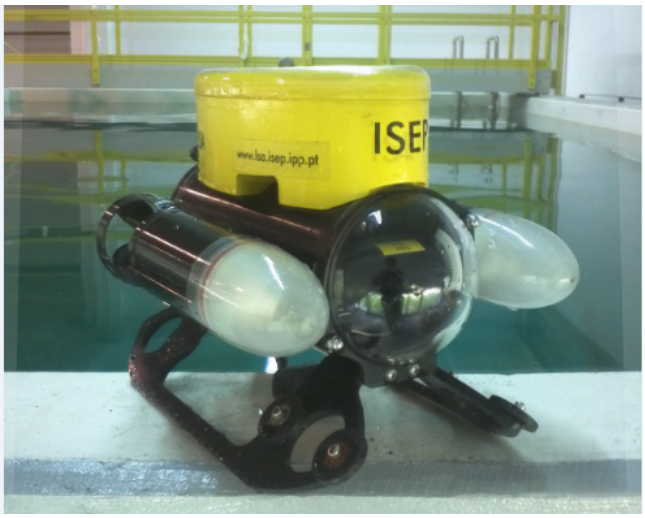


Figura 3.1: ROV videoRay PRO 3E.

desenvolvimento deste projeto. O sistema INESC TEC/ISEP *indoor ground truth* [2] surge como uma ferramenta de validação externa dos dados adquiridos durante os procedimentos experimentais.

A arquitetura do *ground truth*, ver Figura 3.2, consiste num sistema de captura de imagem realizada por um *trigger* externo de câmaras *stereo* a uma profundidade muito pequena. As câmaras foram colocadas no canto do tanque com uma inclinação descendente. Estas são duas câmaras digitais Basler acA1300-30gc cor Gigabit Ethernet (GigE) com *trigger* externo, com resolução máxima de 1278x958 e com *framerate* até 30 fps. A configuração do sistema para processamento de imagem consiste no computador de gravação e também de um conjunto de marcadores predefinidos ligados ao veículo subaquático, de modo a calcular a sua posição. A estimativa da posição 3D é realizada com base na estrutura do marcador (uma esfera para a determinação da posição e de 4 pontos marcadores para cálculo da orientação). A identificação do marcador monocular é realizada com base na segmentação de cor ou com um *template* de correspondência, neste último caso, é possível obter a localização externa do veículo sem a utilização de marcadores.

3.2.4 Modelo de previsão

Uma solução simples para a aplicação do modelo de previsão consiste no modelo cinemático de velocidade constante:

$$X = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{v} \\ \mathbf{w} \end{bmatrix} \tag{3.1}$$

Onde p and q são respectivamente a position e o quaternião de orientação, v e w correspondem as velocidades linear e angular. A utilização deste modelo não requer a instalação de *hardware* complexo, apenas é necessária a integração de uma câmara monocular. No entanto, o fator de escala não é observável para este modelo.

3.2.5 Estado de saída do Veículo

O sistema proposto deverá ler os dados *raw* de uma câmara e através do método

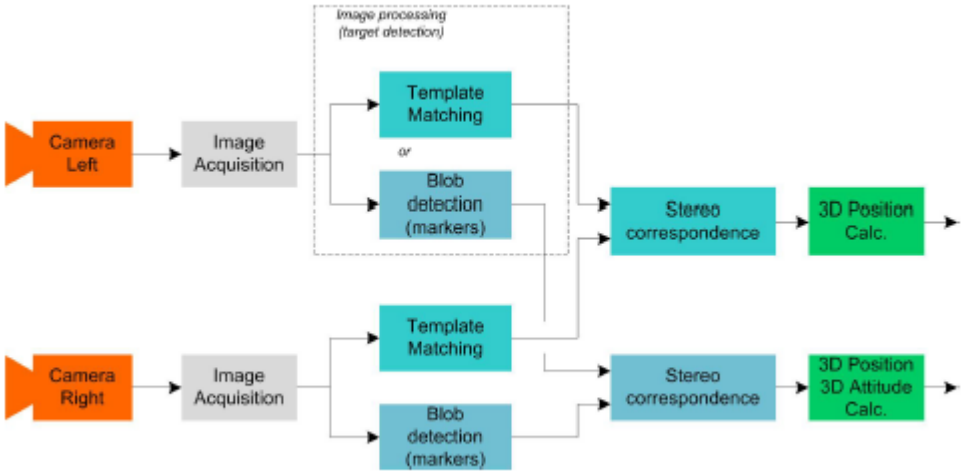


Figura 3.2: Arquitetura sistema g*round truth* (adaptado de [2]).

probabilístico SLAM obter a posição de um veículo no Mundo. Assim o estado de saída do veículo deverá ter as componentes que se seguem:

$$s_{state} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} \tag{3.2}$$

Onde x, y e z são as coordenadas da posição 3D do veículo e ϕ (roll), θ (pitch) e ψ (yaw) transmitem a orientação do veículo. O sistema de coordenadas aplicado ao veículo pode ser consultado na Secção 4.3.1.

CAPÍTULO 4

Visão Computacional

4.1 Modelo da Câmara

O modelo da câmara consiste no mapeamento entre o mundo 3D e a imagem 2D. Um modelo muito utilizado na interpretação e análise corresponde ao modelo de *pinhole*. No referido modelo da câmara, um ponto no espaço com coordenadas $X = (X, Y, Z)^T$ é mapeado para o ponto no plano da imagem, onde uma linha que une o ponto X do centro de projeção intersecta-se com o plano da imagem (Figura 4.1). Isto, considerando que a projeção central de pontos do espaço num plano com o centro de projeção e a origem de um sistema de coordenadas Euclidiana com o plano Z = f, que é denotado como plano de imagem ou plano focal [32].

Por semelhança de triângulos, conclui-se que que o ponto $(X, Y, Z)^T$ é mapeado para o ponto $(fX/Z, fY/Z, f)^T$ no plano da imagem. O mapeamento do espaço Euclidiano de 3 dimensões $\mathbb{R}^3$ em duas dimensões $\mathbb{R}^2$ é dado por:

$$(X, Y, Z)^T \rightarrow (fX/Z, fY/Z)^T. \tag{4.1}$$

O centro de projeção é conhecido como o centro óptico, a linha do centro da câmara perpendicular ao plano de imagem é denotada como eixo principal e o ponto em que o eixo principal se encontra com o plano de imagem é designado de ponto principal. O plano que passa pelo centro da câmara paralelo ao plano da imagem é referido como plano principal da câmara (Figura 4.1).

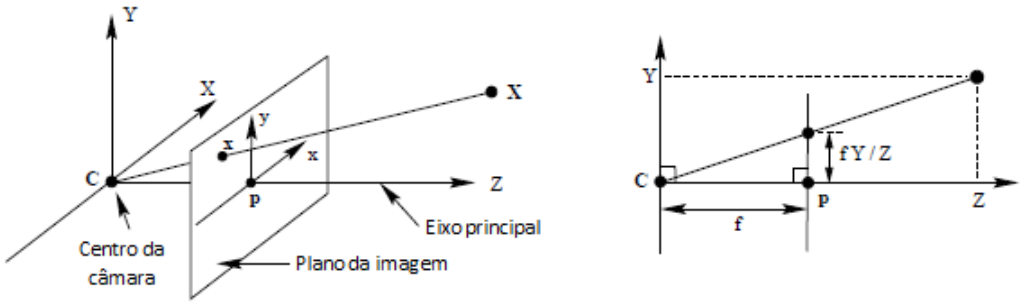


Figura 4.1: Geometria da câmara *pinhole* (adaptado de [32]).

4.1.1 Projeção através coordenadas homogenias

No caso do mundo e a imagem serem representados por vectores homgéneos, a projeção poderá ser expressa da seguinte forma:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{4.2}$$

A matriz nesta expressão pode ser escrita como $diag(f, f, 1)[\mathbf{I}|0]$ onde a primeira expressão corresponde à matriz diagonal e a segunda expressão é a matriz identidade mais um vetor coluna, no caso um vetor zero.

Considerando a notação $\mathbf{X}$ para o ponto no mundo e $\mathbf{x}$ o ponto na imagem e $\mathbf{P}$ a matriz projetiva a equação 4.2 pode ser escrita como se segue,

$$\mathbf{x} = \mathbf{PX}. \tag{4.3}$$

Onde a matriz da câmara para o modelo *pinhole* é representada por:

$$\mathbf{P} = diag(f, f1[\mathbf{I}|0]). \tag{4.4}$$

4.1.2 Deslocamento do ponto principal

A expressão apresentada em 4.1 assume que a origem do plano de imagem é o ponto principal, o que na prática não acontece, assim obtém-se;

$$(X, Y, Z)^T \rightarrow (fX/Z + p_x, fY/Z + p_y)^T, \tag{4.5}$$

onde (px, py) são as coordenadas do ponto principal. Desta forma a expressão 4.2 pode ser escrita novamente,

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & Pz & 0 \\ 0 & f & Py & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{4.6}$$

A matriz calibração de calibração da câmara é dada por,

$$K = \begin{bmatrix} fx & 0 & Pz \\ 0 & fy & Py \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.7}$$

Assim a expressão 4.5 pode ser substituída por,

$$x = \mathbf{K}[I|0]X_{cam}. \tag{4.8}$$

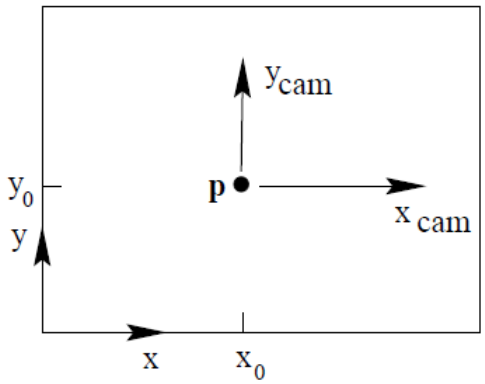


Figura 4.2: Sistema de coordenas da Imagem (x,y) e da câmara(xcam,ycam) (adaptado de [32]).

4.1.3 Rotação da câmara

O eixo de coordenadas da câmara nem sempre corresponde ao eixo de coordenadas Mundo. Os diferentes pontos no espaço podem ser expressos e relacionados por meio de uma rotação (Figura 4.3). Se $\tilde{X}$ é um vetor não homogéneo que representa as coordenadas de um ponto do mundo e $\tilde{X_{CAM}}$ representa o mesmo ponto na câmara, então podemos escrever $\tilde{X_{CAM}}$ = R($\tilde{X}$ - $\tilde{C}$), onde **C** representa as coordenadas do centro da câmara no mundo, e **R** é uma matriz de 3x3 de rotação que representa a orientação da câmara. Esta equação pode ser escrita em coordenadas homogêneas como

$$X_{cam} = \begin{bmatrix} R & -R\tilde{C} \\ 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\tilde{C} \\ 1 & 0 \end{bmatrix} X. \tag{4.9}$$

Desta forma a através da expressão 4.1.2 pode-se deduzir que:

$$x = \mathbf{KR}[\mathbf{I}| - \tilde{\mathbf{C}}]\mathbf{X}. \tag{4.10}$$

Os parâmetros contidos em **K** correspondem aos parâmetros internos da câmera, ou à orientação interna da câmara. Os parâmetros **R** e $\tilde{C}$ relacionam a orientação e posição da câmera através de um sistema de coordenadas de parâmetros externos.
Nem sempre é conveniente utilizar o câmara como centro explicito, pelo contrário pode-se representar o mundo a transformação da imagem como $\tilde{X_{CAM}}$ = R$\tilde{X}$ + t. Assim a matriz da câmara pode ser representada por:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|t], \tag{4.11}$$

onde t=-**RC** [32].

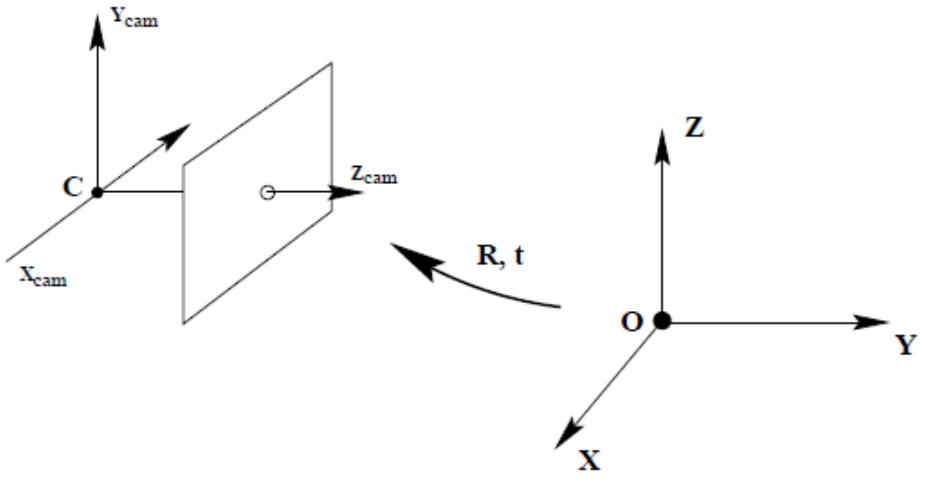


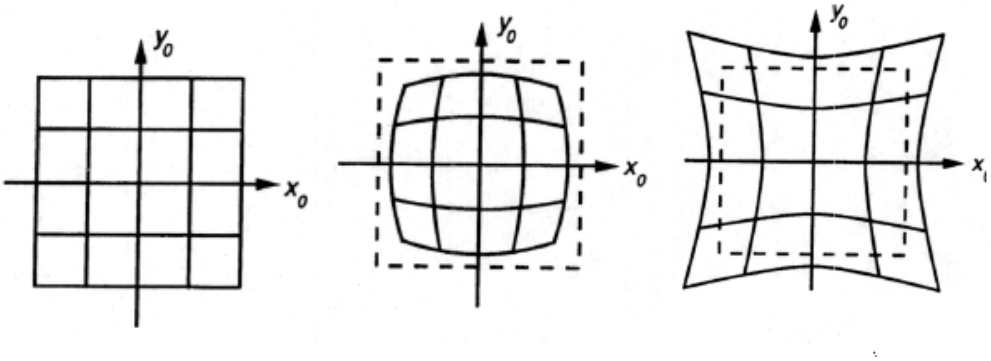
Figura 4.3: Transformação Euclediana entre o sistema de coordenadas da câmara e da imagem (adaptado de [32]).

4.2 Distorção da Lente

O modelo *pinhole* assume que tendo um ponto no Mundo, o seu ponto de projeção na imagem e centro óptico seriam colineares. No entanto, linhas rectas no mundo não coincidem com linhas rectas na imagem, acontece um desvio geralmente provocado por uma distorção entre o ponto do Munto e seu ponto de projeção na imagem. O motivo deste fenómeno deve-se à refração dos raios luminosos quando passam na lente. A lente mantém este formato pois assim permite aumentar o ângulo de visão. O efeito da lente pode ser classificado como distorção tangencial e distorção radial [32].

A distorção radial deve-se à variação do ângulo de refração à medida que aumenta a distância ao centro da lente. A imagem adquire distorção em forma de "barril"quando a refração é menor nos extremos da lente (Figura 4.5b). Em contrapartida, quando a refração é maior nos extremos, ocorre um aumento da ampliação com o aumento da distância ao centro, distorção "almofada"(Figura 4.5c).

A distorção tangencial tem como origem o desalinhamento físico dos constituintes



(a) Sem distorção.

(b) Distorção radial em forma de "barril".

(c) Distorção radial em forma de almofada.

Figura 4.4: Tipo de distorção radial.

da lente, resulta da impossibilidade dos fabricantes em alinhar os eixos óticos das lentes que compões a objetiva. Os fenómenos físicos associados a este efeito são designados por "prisma"ou "descentralização" [33].

4.2.1 Correção da distorção da lente

Através da caracterização da distorção óptica da lente é possível obter um modelo que permite corrigir a imagem. A obtenção de uma imagem sem distorção assim como o cálculo da distancia focal e do ponto principal, consiste num processo de calibração dos parâmetros intrínsecos, a metodologia aplicada deve ser consultada na Secção 6.4.0.1.

A determinação dos parâmetros intrínsecos obtém-se a partir da relação das coordenadas 3D do espaço com a sua respectiva projeção no plano da imagem. Tendo as coordenadas da câmera $XX_c = (X_c; Y_c; Z_c)$ de uma série de pontos do espaço determinam-se as coordenadas normalizadas desses mesmos pontos. As coordenadas normalizadas são obtidas dividindo as duas primeiras componentes pela terceira (seguindo a convenção do eixo Z_c ser o perpendicular à superfície da lente). Note-se que deste modo todos os pontos de um raio projetante (com origem no centro da lente) terão as mesmas coordenadas normalizadas.

$$x_n = \begin{bmatrix} \frac{X_c}{Z_c} \\ \frac{y_c}{Z_c} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.12}$$

Com os resultados da calibração é possível estimar a distorção da lente que irá afetar estes pontos normalizados. Geralmente a influência da distorção radial é bastante superior comparativamente ao da distorção tangencial. A distorção será tanto maior quando maior for a distância normalizada destes *pixels* ao ponto principal normalizado – de coordenadas $x_n = (0; 0)$.

O vector da distorção tangencial dx é dado por:

$$dx = \begin{bmatrix} 2.kc_t(1).x.y + kc_t(2).(r^2 + 2.x^2) \\ kc_t(1).(r^2 + 2.y^2) + 2.kc_t(2).x.y \end{bmatrix} \tag{4.13}$$

onde $kc_t(1)$ e $kc_t(2)$ são os coeficientes da distorção tangencial e $r^2 = x^2 + y^2$.

As coordenadas normalizadas com distorção x_dn calculam-se da seguinte forma:

$$x_dn = \begin{bmatrix} x_d \\ y_d \end{bmatrix} = (1 + kc_t(1).r^2 + kc_r(2).r^4).x_n + dx \tag{4.14}$$

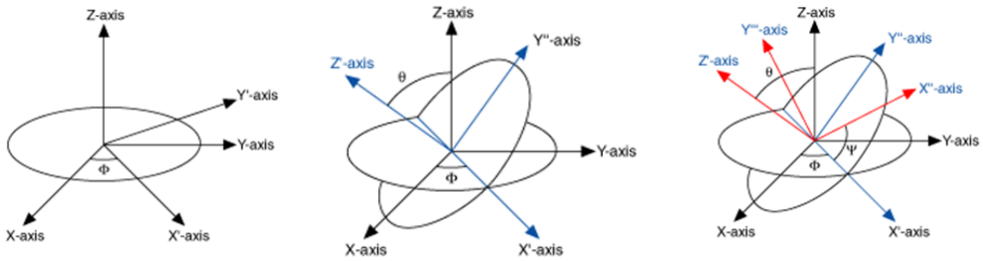
onde $kc_r(1)$ e $kc_r(2)$ correspondem aos coeficientes da distorção radial.

Obtidas as coordenadas normalizadas com distorção desses pontos obtêm-se as coordenadas pixel através da seguinte equação linear expressa pela equação 4.6. A matriz **K** (equação 4.7) é designada como a matriz de transformação da câmera. Esta matriz permite converter coordenadas normalizadas em coordenadas pixel. Depende dos valores estimados para a distância focal da câmera (fx,fy) e do ponto principal (Px,Py) (ambos expressos em pixels).

4.3 Ângulos de Euler

Existem várias formas de representação da orientação 3D de um objecto, os ângulos de Euler são um dos métodos utilizados. As abordagens apesentadas neste relatório vão exigir uma percepção do conceito descrito.

Os ângulos de Euler (formulados por Leonard Euler) [34] descrevem a orientação de um corpo rígido num espaço euclidiano tridimensional. As rotações descritas segundo os eixos X, Y e Z correspondem a *roll* (ϕ), *pitch* (θ) e *yaw* (ψ). A representação tridimensional pode ser consultada na Figura 4.5.



(a) Rotação segundo x. (b) Rotação segundo y. (c) Rotação segundo z.

Figura 4.5: Representação esquemática dos ângulos de Euler.

O ângulo roll representa a rotação em torno do eixo dos xx's, descrita matematicamente pela seguinte relação:

$$R(x,\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}. \tag{4.15}$$

A rotação segundo o eixo dos yy's é denominada pelo pitch e representada pela expressão que se segue,

$$R(y,\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta)) \end{bmatrix}. \tag{4.16}$$

Por fim, o ângulo yaw descrito como a rotação segundo o eixo dos zz's é equivalente à seguinte expressão,

$$R(z,\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.17}$$

A rotação final segundo ϕ, θ e ψ é matematicamente equivalente à expressão seguinte,

$$R = R^T(x,\phi)R^T(y,\theta)R^T(z,\psi) \tag{4.18}$$

$$= \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \cos\psi\sin\phi\sin\theta - \cos\theta\sin\psi & \sin\psi\sin\phi\sin\theta + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\psi\sin\theta\cos\phi + \sin\phi\sin\psi & -\sin\phi\cos\psi + \sin\psi\sin\theta\cos\theta & \cos\theta\cos\phi \end{bmatrix} \tag{4.19}$$

4.3.1 Sistema de coordenadas do veículo

As grandezas físicas envolvidas no modelo do veículo são representadas sob dois sistemas de coordenadas, o primeiro fixo ao veículo e o segundo fixo no mundo. Ambas as referências podem ser consultados na Figura 4.6, o sistema de coordendas do veículo é representado por x_b, y_b, z_b o sistema de coordenadas global fixo no mundo é representado por x_e, y_e, z_e.

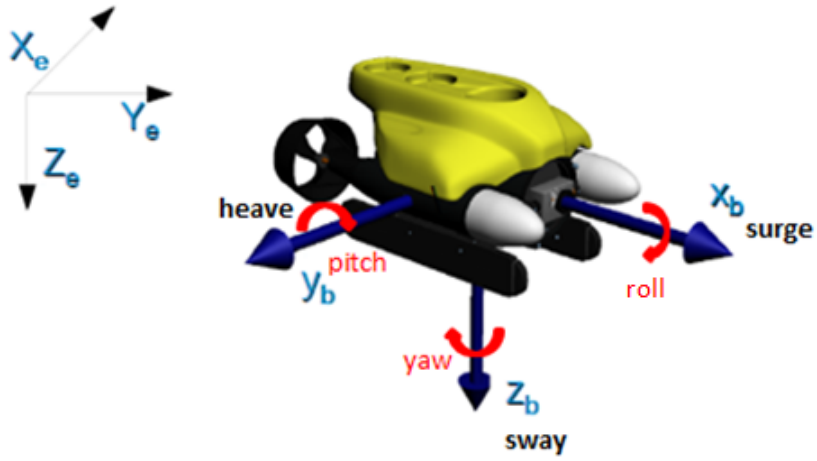


Figura 4.6: Referenciais e graus de liberdade do veículo subaquático.

Na modelação de veículos subaquáticos é conveniente utilizar notações práticas que permitam a representação gráfica dos graus de liberdade do veículo. A nomenclatura apresentada na Figura 4.6 para descrever os movimentos e a orientação do veículo é proposta em [35]. Assim, a notação utilizada para descrever o movimento do veículo em x_b, corresponde a *surge*, o movimento de y_b é denotado por *heave*, e *sway* corresponde ao movimento do veículo segundo o eixo z_b. Por sua vez, as rotações segundo x_b, y_b e z_b correspondem respectivamente a *roll*, *pitch* e *yaw*.

CAPÍTULO 5

SLAM *Simultaneous Localization and Mapping*

5.1 Conceitos Introdutórios

O problema *Simultaneous Localization and Mapping* (SLAM) questiona a possibilidade de um robô móvel, colocado numa localização e ambiente desconhecidos, construir incrementalmente um mapa consistente desse mesmo ambiente e ao mesmo tempo determinar sua localização dentro desse mapa. Atualmente SLAM já é considerada uma questão resolvida da robótica móvel [1]. No entanto, não consiste numa aplicação com um único método de resolução, podem ser implementadas várias abordagens de acordo com a aplicação pretendida.

Na resolução do problema são alguns os desafios colocados, que podem ser divididos e interpretados de diversa formas segundo diferentes metodologias. Para cada situação podem ser aplicados diferentes métodos, de acordo com as características quer do robô quer da aplicação desejada. No entanto pode-se resumir as fases essenciais SLAM de acordo com a caracterização atribuída na Tabela 5.1.

As *landmark* (elementos identificativos no cenário) são um requisito essencial para a aplicação SLAM, constituem os marcadores que possibilitam a navegação do robô. Estas deverão ser facilmente observadas e distintos do ambiente envolvente, assim como, devem ser detetáveis a partir de diferentes posições e ângulos. Desta forma, devem possuir características únicas que permitam distingui-los dos restantes marcadores anterio-

res. Outra nota importante refere-se à necessidade dos marcadores serem estacionários, caso contrário o robô não vai conseguir mapeá-los corretamente.

Fases SLAM	Descrição
Landmark extractiom	Extração de *features*, marcadores.
Data association	As novas observações terão de ser associadas com as observações já existentes no mapa.
State estimation	Estimação do estado usando o método de distribuição de probabilidade.
State Update	Atualização do estado usando o método de distribuição de probabilidade.
Landmark update	Atualização dos marcadores.
Mapping	Construção do mapa.

Tabela 5.1: Fases algoritmo SLAM.

O processo "Data Association" (associação de dados) vai ter o seu contributo no mapeamento durante a associação das novas observações com as previamente existentes no mapa. Relativamente ao "State estimation", "State update", "Landmark Update", vão depender do método de distribuição de probabilidade aplicado. Os métodos mais aplicados são dois o EKF-SLAM e Fast SLAM (consultar Secção 1.2.1). Por fim, a fase mapeamento vai consistir na construção de um mapa a partir da associação das *landmark* observadas pelo robô.

Para o caso particular de visual SLAM, as *landmark* vão ser extraídas a partir de métodos de processamento de imagem, no entanto, o restante algoritmo é aplicado de igual forma. Portanto, o robô adquire a imagem, extrai a *landmark* e tenta associar aos marcadores já observações. No caso da *landmark* ser re-observada é utilizada para atualizar a posição do robô, quando é registada uma nova observação é adicionada ao mapa (processo ilustrado na Figura 5.1).

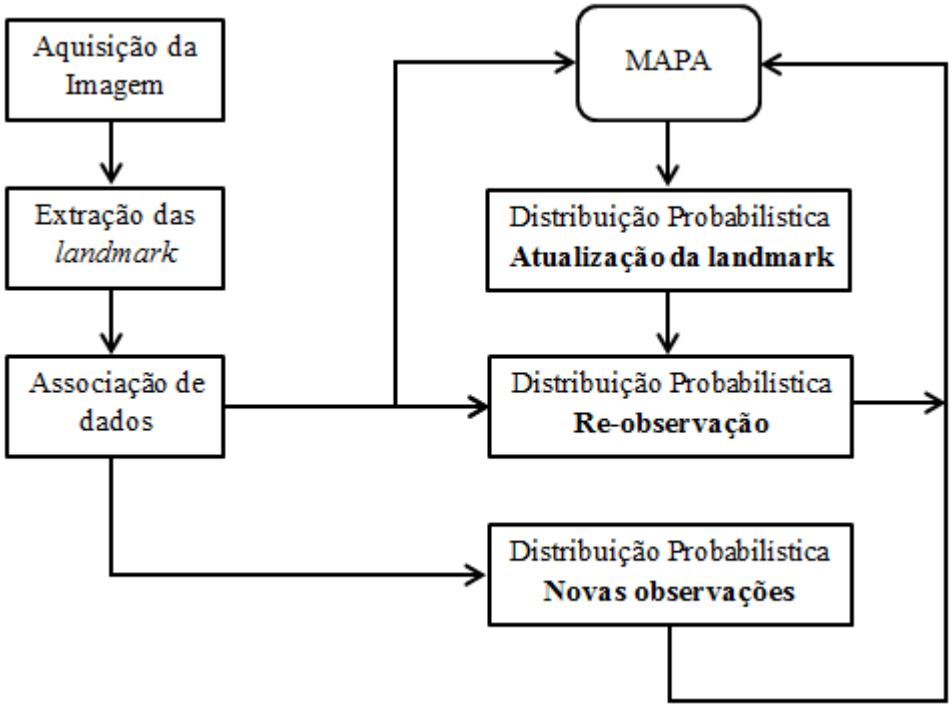


Figura 5.1: Esquema do visual SLAM.

5.2 Métodos de Estimação SLAM

SLAM estima a trajetória e a localização de todos os pontos de referência sem nenhum conhecimento *à priori*. No entanto, a verdadeira localização do robô nunca é conhecida ou medida diretamente, as observações são feitas entre o robô e a localização das *landmark*.

Considerando um robô móvel [6] em movimento através do ambiente (Figura 5.2) a fazer observações relativas a um número de *landmarks* desconhecidas com um sensor localizado no robô. No instante k, as variáveis são definidas do seguinte modo:

- x_k: O vector de estado que descreve a localização e a orientação do veículo.
- u_k: O vector de controlo, aplicado no instante $k-1$ para conduzir o veículo a um

estado x_k no tempo k.

- m_i: Vector que descreve a localização da *landmark* i^{th} cuja verdadeira localização é assumida invariante no tempo.
- z_{ik}: Observação efetuada a partir do veículo da localização da *landmark* i^{th} no instante k. Quando ocorrem múltiplas observações a qualquer momento ou quando uma específica *landmark* não é relevante, a observação será escrita como z_k.

Assim, também são definidos os seguintes conjuntos:

- $X_{0:k} = \{x_0, x_1, ..., x_k\} = \{_{0:k-1}, x_k\}$: Histórico da localização do veículo.
- $U_{0:k} = \{u_1, u_2, ..., u_k\} = \{U_{0:k-1}, u_k\}$
- $m = \{m_1, m_2, ..., m_n\}$: Conjunto de todas as *landmarks*.
- $Z0 : k = \{z_1, z_2, ..., z_k\} = \{Z_{0:k-1,z_k}\}$: O conjunto de todas as observações *landmark*.

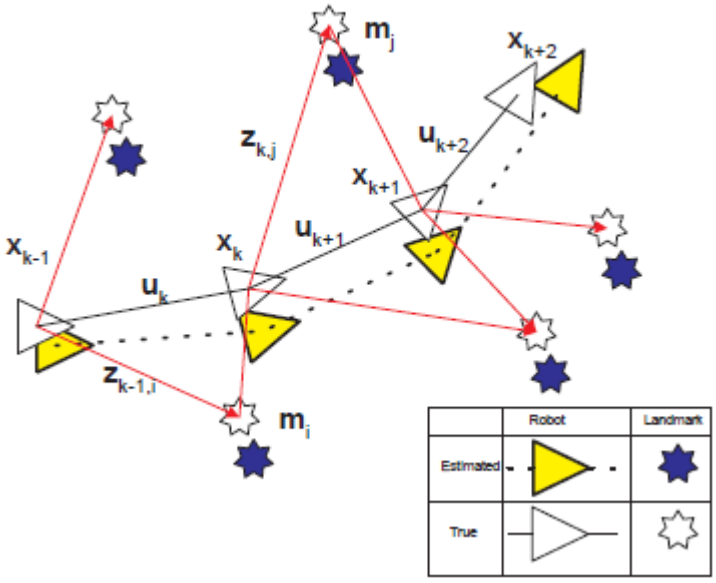


Figura 5.2: Representação do essencial de SLAM (adaptado de [6]).

5.2.1 Modelação probabilística

A resolução do problema SLAM requer uma distribuição de probabilidade definida por:

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0). \tag{5.1}$$

Esta distribuição de probabilidade descreve a densidade de probabilidade conjunta (posterior) articulação da localização das *landmark* e estado do veículo (no instante k) dadas as observações registadas e entradas de controlo até e incluindo o instante k em conjunto com o estado inicial do veículo.

O modelo de observação descreve a probabilidade de fazer uma observação z_k quando a localização do veículo e *landmarks* são conhecidos, expresso por:

$$P(z_k|x_k, m). \tag{5.2}$$

O modelo de movimento para veículo pode ser descrito de acordo com uma distribuição de probabilidade sobre transições de estado na forma:

$$P(x_k|x_{k-1}, u_k). \tag{5.3}$$

Ou seja, o estado seguinte x_k depende apenas do estado imediatamente anterior x_{k-1} e do controlo aplicado u_k, sendo independente de ambas as observações e do mapa.

O algoritmo SLAM é implementado recursivamente em dois passos, previsão (*time-update*) e correção (*measurement-update*):

Time-update

$$P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0)$$

$$= \int P(x_k|x_{k-1}, u_k) \times P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1}, x_0) dx_{k-1} \tag{5.4}$$

Measurement-update

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0)$$

$$= \frac{P(z_k|x,m)P(x_k, m|Z_{0:k-1}, U_{0_k}, x_0)}{P(z_k|Z_{0:k-1}, U_{0:k})} \tag{5.5}$$

As equações 5.4 e 5.5 fornecem um procedimento recursivo para calcular à posteriori $P(x_k; m|Z_{0:k}, U_{0:k}, x_0)$ o estado robô x_k e mapa m no instante k com base em todas as observações $Z_{0:K}$ e todo o controlo de entradas $U_{0:k}$ até e incluindo o instante k. A recursividade corresponde a uma função do modelo do veículo $P(x_k j|x_(k-1), u_k)$ e um modelo de observação $P(z_k|x_k, m)$. O problema de construção de mapa pode ser formulada através do caculo da densidade condicional $P(m|X_{0:k}, Z_{0:k}, U_{0:k})$. Isso assume que a localização do veículo x_k é conhecida em todos os momentos, sujeito ao conhecimento da localização inicial. Um mapa m é então construído por fusão de observações a partir de localizações diferentes. Por outro lado, o problema da localização pode ser formulado para calcular a distribuição de probabilidade $P(x_k|Z_{0:k}, U_{0:k}, m)$. Pressupõe-se que a localização das *landmark* é conhecida com certeza e o objetivo é calcular uma estimativa da localização do veículo em relação a essas *landmark*.

O problema SLAM envolve encontrar uma solução adequada para o modelo de observação (Equação 5.2) e modelo de movimento (Equação 5.3) que permite a computação eficiente e consistente das distribuições expressas nas Equações 5.4 e 5.5. O *Extended Kalman Filter* (EKF) consiste de uma solução comum utilizada na forma de um modelo de espaço de estado com ruído gaussiano aditivo. Neste caso, uma distribuição de probabilidade não gaussiana pode ser aplicada ao modelo de movimento do veículo (Equação 5.3). A resolução de SLAM pode ser igualmente determinada através da utilização do filtro de partículas (Rao-Blackwellised), denominado de FastSLAM. Assim EKF-SLAM e FastSLAM são dois dos métodos de solução que vão ser analisados com pormenor nas Secções 5.2.2 e 5.2.3 respetivamente.

5.2.2 EKF-SLAM

No método EKF-SLAM o movimento do veículo deverá ser descrito segundo:

$$P(x_k|x_{k-1}, u_k) \Longleftrightarrow x_k = f(x_{k-1}, u_k) + w_k, \tag{5.6}$$

onde $f()$ corresponde à cinemática do modelo de veículos e onde w_k representa ruído no modelo correspondente, perturbações de movimento Gaussinano não correlacionadas com covariância Q_k. O modelo de observação é descrito da seguinte forma:

$$P(z_k|x_k, m) \Longleftrightarrow z(k) = h(x_k, m) + v_k, \tag{5.7}$$

onde $h()$ corresponde à geometria da observação e v_k consiste em ruído na observação representado, erros de observação Gaussinanos não correlacionados com covariância R_k. Tendo em consideração o método *standard* do EKF [11] pode ser aplicado para cacular a média da seguinte forma:

$$\begin{bmatrix} \hat{x}_{k|k} \\ \hat{m}_k \end{bmatrix} = E \begin{bmatrix} x_k \\ m \end{bmatrix} |Z_{0:k} \Bigg], \tag{5.8}$$

e covariância:

$$\begin{aligned} P_{k|k} &= \begin{bmatrix} P_{xx} & P_{xm} \\ P_{xm}^T & P_{mm} \end{bmatrix}_{k|k} \\ &= E \left[\begin{pmatrix} x_k - \hat{x}_k \\ m - \hat{m}_k \end{pmatrix} \begin{pmatrix} x_k - \hat{x}_k \\ m - \hat{m}_k \end{pmatrix}^T |Z_{0:k} \right] \end{aligned} \tag{5.9}$$

Do conjunto de distribuição $P(x_k, m|Z_{0:k}, U_{0:k}, x_0)$ a partir de:

Time-update

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k) \tag{5.10}$$

$$P_{xx,k|k-1} = \nabla f P_{xx,k-1|k-1} \nabla f^T + Q_k \tag{5.11}$$

∇f corresponde ao Jacobiano de f avaliado pela estimativa de $\hat{x}_{k-1|k-1}$.

Observation-update

$$\begin{bmatrix} \hat{x}_{k_k} \\ \hat{m}_k \end{bmatrix} = \begin{bmatrix} \hat{x}_{k|k-1} \\ \hat{m}_{k-1} \end{bmatrix} + W_k[z(k) - h(\hat{x}_{k|k-1}, \hat{m}_{k-1})] \tag{5.12}$$

$$P_{k|k} = P_{k|k-1} - W_k S_k W_k^T \tag{5.13}$$

onde

$$S_k = \nabla h P_{k|k-1} \nabla h^T + R_k \tag{5.14}$$

$$W_k = P_{k|k-1} \nabla h^T S_k^{-1} \tag{5.15}$$

Nas expressões anteriores ∇h corresponde ao jacobiano de h avaliado pelo $\hat{x}_{k|k-1}$ e $\hat{m}_{k-1}$.

5.2.3 FastSLAM

O algoritmo FastSLAM [10, 36], teve um importante contributo na resolução de SLAM através de uma probabilística recursiva. FastSLAM baseia-se na amostragem recursiva de Monte Carlo, Filtro de Partículas (PF), foi o primeiro a representar diretamente o modelo de processo não linear e distribuição não Gaussiana da posição.

A elevada dimensão do espaço de estados do problema SLAM torna a aplicação direta de filtros de partículas computacionalmente inviável. Contudo, é possível reduzir o espaço da amostra por aplicação de Rao-Blackwellisation (RB). No referido filtro um estado comum é repartido de acordo com a regra do produto $P(x_1, x_2) = P(x_2|x_1)P(x_1)$ e, no caso de $P(x_2|x_1)$ poder ser representado de forma analítica por $P(x_1)$ necessita de ser amostrado $x_1^{(i)} \sim P(x_1)$. Desta forma a distribuição conjunta deverá ser representado pelo conjunto $\left\{x_1^{(i)}, P(x_2|x_1^{(i)})\right\}_i^N$ e estatísticas, tais como:

$$P(x_2) \approx \frac{1}{N} \sum_i^N P(x_2|x_i^{(i)}) \tag{5.16}$$

Estas podem ser obtidas com maior precisão do que seria possível por meio de amostragem ao longo do espaço. O conjunto do estado SLAM pode ser tido em conta num componente do veículo e num componente de mapa condicional. Esta é uma propriedade fundamental da FastSLAM, e a razão para a sua velocidade. O mapa é representado como um conjunto de Gaussianas com complexidade linear independente, ao invés de um mapa de covariância conjunta com a complexidade quadrática. A estrutura essencial do FastSLAM, corresponde um estado *RaoBlackwellised*, onde a trajetória é representada por amostras ponderadas e o mapa é calculado analiticamente (Figura 5.3). Assim, o conjunto da distribuição no instante k, é representado pelo conjunto $\left\{w_k^{(i)}, X_{0:k}^{(}i), P(m|X_{0:k}^{(i)}, Z_{0:k})\right\}_i^N$, onde o mapa que acompanha cada partícula é composto por distribuições gaussianas independentes $P(m|X_{0:k}^{(i)}; Z_{0:k}) = \amalg_j^M P(m_j|X_{0:k}^{(i)}, Z_{0:k})$. A identificação recursiva é realizada através da filtragem de partículas para os estados da posição e o EKF aplicado nos estados do mapa.

A forma geral de um filtro de partículas RB na aplicação é descrito em seguida. Considerando que, no instante $k-1$, o conjunto do estado é representado por $\left\{w_{k-1}^{(i)}, X_{k-1}^{(i)}, X_{0:k-1}^{(i)}, P(m|X_{0:k-1}^{(i)}, Z_{0:k-1})\right\}_i^N$.

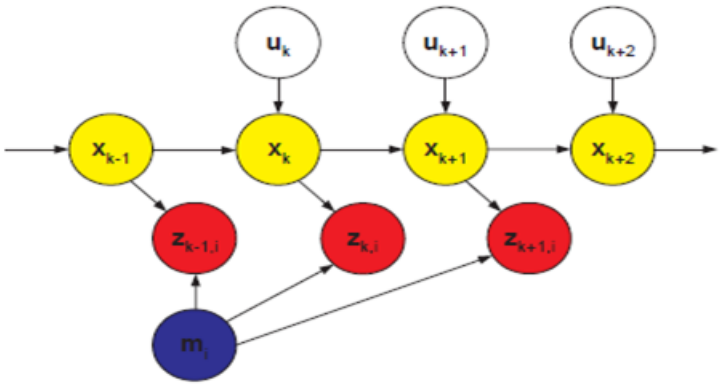


Figura 5.3: Modelo gráfico do algoritmo SLAM. Para FastSLAM, cada partícula define uma diferente hipótese da trajetória do veículo (adaptado de [6]).

1. Para cada partícula, é necessário calcular a distribuição proposta, condicionando a história de partícula específica, e aplicar uma amostragem

$$x_k^{(i)} \sim \pi(x_k | X_{0:k-1}^{(i)}, Z_{0:k}, u_k) \tag{5.17}$$

Esta nova amostra é unida ao histórico da partícula $\left\{ X_{0:k}^{(i)} \triangleq X_{0:k-1}^{(i)}, x_k^{(i)} \right\}$.

2. A pesagem de cada amostra deve estar de acordo com a importância da função

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{P(z_k | X_{0:k}^{(i)}, Z_{0:k-1}) P(x_k^{(i)} | x_{k-1}^{(i)}, u_k)}{\pi(x_{k-1}^{(i)} | X_{0:k-1}^{(i)}, Z_{0:k}, u_k)} \tag{5.18}$$

Na equação 5.18, os termos do numerador correspondem ao modelo de observação e modelo de movimento respectivamente.

$$\begin{aligned} & P(z_k | X_{0:k}, Z_{0:k-1}) \\ = & \int P(Z_k | x_k, m) P(m | X_{0:k-1}, Z_{0:k-1}) dm \end{aligned} \tag{5.19}$$

3. Realizar *Resampling* (Re-Amostragem), que é efetuado selecionando as partículas, com a substituição, a partir do conjunto $\left\{ X_{0:k}^{(i)} \right\}$, incluindo os seus mapas associados, com probabilidade de seleção proporcional ao $w_k^{(i)}$. A partir da partículas selecionadas obtém-se o peso uniforme, $w_k^{(i)} = \frac{1}{N}$.

4. Para cada partícula, executar uma atualização do EKF nas *landmark* observadas como uma operação de mapeamento simples, com a posição conhecida do veículo.

Existem diferentes versões do FastSLAM, FastSLAM 1.0 [10] e FastSLAM 2.0 [36]. Estes algoritmos diferem apenas em termos da sua forma de distribuição proposta (passo 1) e, consequentemente a sua importância de peso (passo 2). A segunda versão apresenta uma solução mais eficaz. Para FastSLAM 1.0, a distribuição proposta é o modelo de movimento

$$x_k^{(i)} \sim P(x_k | x_{k-1}^{(i)}, u_k) \tag{5.20}$$

Portanto, a partir da Equação 5.18, as amostras são ponderadas de acordo com o modelo de observação marginalizados.

$$w_k^{(i)} = w_{k-1}^{(i)} P(z_k | X_{0:k}^{(i)}, Z_{0:k-1}); \tag{5.21}$$

Para FastSLAM 2.0, a distribuição proposta inclui a observação atual

$$x_k^{(i)} \sim P(x_k | X_{0:k-1}^{(i)}, Z_{0:k}, u_k) \tag{5.22}$$

onde

$$\begin{aligned} & P(x_k | X_{0:k-1}^{(i)}, Z_{0:k}, u_k) \\ = & \frac{1}{C} P(z_k | x_K, X_{0:K-1}^{(i)}, Z_{0:k-1}) P(x_k | x_{k-1}^{(i)}, u_k) \end{aligned} \tag{5.23}$$

e C é uma constante de normalização. A vantagem de FastSLAM 2.0 consiste da sua distribuição proposta ser localmente ótima, ou seja, para cada partícula, obtêm-se a menor variância possível em peso $w_k^{(i)}$ condicionada pela informação disponível, $X_{0:k-1}^{(i)}, Z_{0:k} e U_{0:k}$.

5.3 RT-SLAM

5.3.1 Generalidades

RT-SLAM [3] consiste numa biblioteca de *software* desenvolvida em C ++ pelo laboratório LAAS-CNRS em Toulouse (França), para prepósitos SLAM em tempo real. Foi especialmente desenvolvida com o intuito de criar uma ferramenta flexível e genérica. O objectivo é possibilitar ao utilizador o desenvolvimento, a evolução e teste de diversas abordagens SLAM.

O método probabilístico implementado é baseado no *Extended Kalman Filter*, abordagem EKF SLAM. Corre até 60 Hz com imagens 640×480, além disso permite

alterar e adaptar o código a novos modelos de robôs, sensores e *landmarks*. O algoritmo suporta movimentos altamente dinâmicos, aplicável a humanoides ou robôs todo o terreno [3].

5.3.2 Arquitetura

No problema típico SLAM, um ou mais robôs navegam num determinado ambiente, descobrindo e mapeando *landmarks* no caminho através de um ou mais seus sensores a bordo. O RT-SLAM foi desenvolvido segundo uma metodologia *object-oriented* e contempla 5 objetos essenciais, mapa, robô, sensor, *landmark* e observação. Os objetos SLAM podem ser observados na Figura 5.4.

O mapa é constituído por robôs e *landmarks*. Por sua vez, este objeto contém o método de estimação probabilística, EKF SLAM. O mapeamento da aplicação foi criado a partir do *template* Boot's uBlas (componente em C++ da biblioteca Boost). Com o RT-SLAM é possível fazer o mapeamento de mais do que um robô, sendo que cada um deles pode ser constituído de um ou vários sensores. O algoritmo disponibiliza a utilização de simples modelos cinemáticos, ou modelos aplicados a sensores proprioceptivos. Sensor proprioceptivo refere-se a objeto genérico que pode conter informação proveniente de diferente hardware com a mesma funcionalidade.

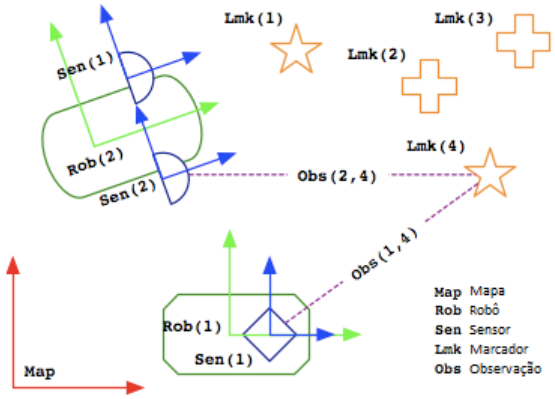


Figura 5.4: Objetos RT-SLAM (adaptado de [3]).

As observações advêm das *landmark*. A associação de dados é feita através de uma característica que estas possuem, o descritor. O descritor é representado por um estado, este permite que a mesma *landmark* seja observada por diversos sensores e reconhecida. Por sua vez, as observações são instanciadas por pares sensor/*landmark*, independentemente de o sensor ter realmente observado o marcador ou não, e tem o mesmo tempo de vida do marcador associado. Os estados do robô, sensor e *landmark* são guardados sob a forma de mapas estocásticos (não determinísticos).

A associação de dados é gerida por dois *managers*, o *data manager* e o *map manager*. Ambos comunicam entre si, por exemplo, quando o *data manager* obtém uma nova *landmark* questiona ao *map manager* se tem espaço para alocar uma nova marca.

O *data manager* explora processadores de dados em bruto (*raw data*) provenientes de sensores, para detecção de *features* e *tracking*. Decide quais as observações que devem ser corrigidas e em que ordem, de acordo com a quantidade e qualidade da informação.

O *map manager* mantém o mapa limpo, com informação relevante e com um tamanho controlável, removendo *landmarks* de acordo com a qualidade e critério pré estabelecido (sub mapas).

A arquitetura apresenta uma estrutura hierárquica dos objetos constituintes do RT-SLAM, conforme ilustrado na Figura 5.5. Onde W representa o mundo, M é o mapa, R o robô, O representa a observação, S trata-se do sensor e por fim DM e MM representam o *data manager* e *map manager* respetivamente [3].

5.3.3 *Parental links*

No módulo RT-SLAM a *family-tree* difere de uma árvore regular em que um objeto pode ter mais de um pai (ou seja. Um pai e uma mãe). A organização global dos principais objetos é representado de uma forma UML (consultar Anexo A).

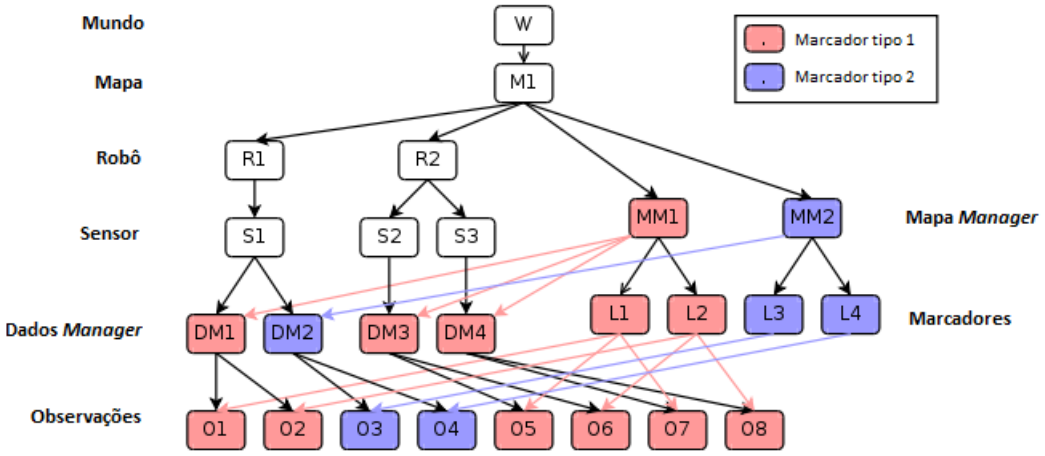


Figura 5.5: Arquitetura RT-SLAM (adaptado de [3]).

As relações *parental link* descrevem-se da seguinte forma:

- Mapa: Tem dois objetos *children*, **Robôs** e **Landmarks**;
- Robô: Objeto **Sensores** tem relação*children*. O objeto **Mapa** corresponde ao *parent*;
- Sensor: Tem como objetos *children* as **Observações** e como *parent* **Robô**.
- *Landmark*: Os objetos *children* correspondem a **Observações** e **Mapa** é o respetivo *parent*;
- Observação: Não tem relações *children*. Tem relação **parent** com **Sensor** [37].

Supondo uma linha imaginária que atravessa o **sensor** e as **observações** separa as partes do problema SLAM que correspondem ao espaço de estados (Mapa, Robô, Sensor, *Map Manager*, Landmark) e o espaço de medida (Sensor, *Data Manager*, Observação). Cada **observação** cria um laço de *links*. Todos estes circuitos são visíveis na Figura 5.6. Por exemplo, o *loop* destacado em vermelho: M1$\longrightarrow$ R2$\longrightarrow$S2 $\longrightarrow$ DM2$\longrightarrow$ O4$\longrightarrow$ L2$\longrightarrow$ MM1$\longrightarrow$ M1.

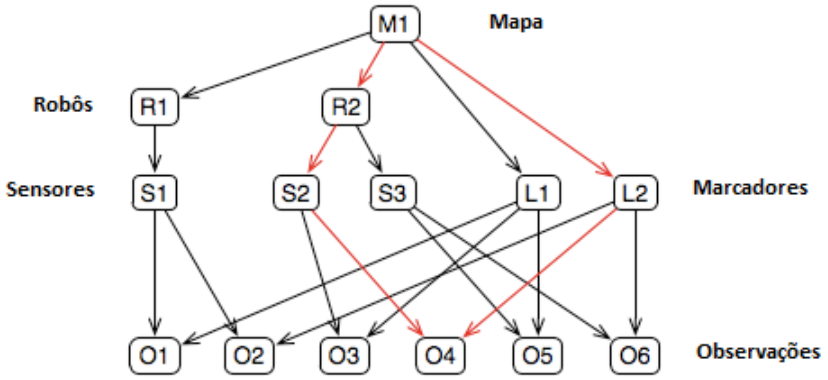


Figura 5.6: Arquitetura RT-SLAM simplificada (adaptado de [37]).

Para explicar essas relações, o RT-SLAM utiliza listas de ponteiros para um *children*, e um único ponteiro para um *parent*. Cada link do gráfico da Figura 5.7 é armazenada duas vezes. Isso permite que uma referência sistemática para a frente e para trás entre os objetos.

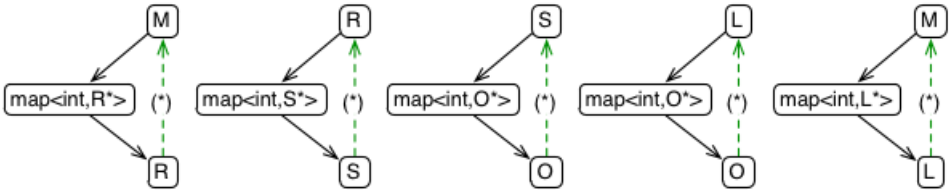


Figura 5.7: Ligações bidirecionais em RT-SLAM (adaptado de [37]).

5.3.4 Visual SLAM

5.3.4.1 Active Search

O visual SLAM com RT-SLAM aplica uma técnica baseada no *active search* [38] de Andrew j. Davidson. Em RT-SLAM a técnica é aplicada pelo *data manager*. A pesquisa é limitada ao local onde é mais provável determinar a *landmark* e são descartados dados não pertencentes a 3σ, elipse de incerteza. Isto vai permitir a diminuição da quantidade de processamento de *raw data* e desta forma reduzir constrangimentos em tempo real (Figura 5.8).

A aplicação do *ative search* é efetuada com inicialização de *landmark*. Cada Sensor procura manter as *features* no seu campo de visão com uma grelha de movimento aleatório e tamanho fixo. A deteção de *features* é limitadas ao número de células da grelha vai permitir uma melhor observabilidade do movimento realizado.

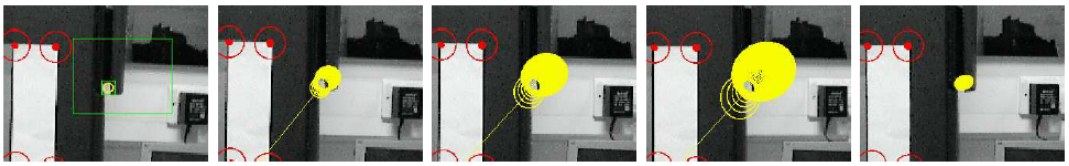


Figura 5.8: Active search.Identificação de zona de pesquisa, projeção na zona de elipses de incerteza, aplicação do Filtro Bayesian (*re-weighthing*) e identificação de *landmark* (adaptado de [38])

5.3.4.2 One Point Ransac

Para além da aplicação do *active search* para eliminar dados que não interessam processar, em RT-SLAM é aplicado o *One-point Ransac* [39] para descartar *outliers*. Esta consiste de uma abordagem hibrida do algoritmo RANSAC [40] com o filtro de Kalman. A implementação deste algoritmo permite a pesquisa de recursos numa pequena área e não em toda a elipse de incerteza. Mais uma vez, a quantidade de dados processados é reduzida melhorando a *performance em* tempo real.

5.3.4.3 Point matching

Durante a aquisição de imagens é aplicado um algoritmo de correlação que permite descartar as imagens sem interesse. A aplicação é baseada no *Zero–mean Normalized Cross Correlation* (ZNCC) [41], o mesmo é então utilizado para obter a correlação entre dois pontos de duas imagens distintas de acordo com a correlação estabelecida por ZNCC as imagens são aceites ou descartadas.

CAPÍTULO 6

Implementação

6.1 Arquitetura do Sistema

A arquitetura proposta para o sistema de navegação subaquático baseado no método probabilístico SLAM encontra-se representada na Figura 6.1. O sistema de navegação proposto é baseado no algoritmo monocular visual SLAM denotado de RT-SLAM [3] com *software* integrado em *framework* ROS [8]. A modularidade do *software* desenvolvido permite a sua aplicação não só para o ROV videoRay PRO 3E mas também para qualquer situação que seja permitida a aquisição de um tópico ROS de uma imagem. Para isso, basta o "SLAM node" receber o tópico de uma imagem que posteriormente publicará um outro tópico com a posição a três dimensões do robô móvel para cada instante e assim gerar a trajetória percorrida.

6.1.1 Componentes do sistema

O sistema de testes é constituído pelo VideoRay PRO 3E. O veículo tem 3 graus de liberdade conferidos pelos três propulsores de comando, dois para os movimentos horizontais e uma para o movimento vertical. O ROV é equipado com duas câmaras, uma à frente e outra atrás, um sensor de pressão, uma bússola e altímetro.

O veículo subaquático está ligado a uma caixa de interface *host* através de um cabo umbilical, que transporta a fonte de alimentação do ROV, o sinal de vídeo PAL e

os dados entre o veículo e a estação *host* são conduzidos através de uma interface CAN. A caixa de interface está ligada a 240 VAC, também é responsável por permitir a interação com o ROV através de comunicação serie (RS232). Esta interação é realizada por um computador Intel Pentium dual core, com um sistema operativo Linux, que também recebe o sinal de vídeo depois de ser digitalizado por um *framegrabber* (Pinnacle DVC100) a uma resolução de 640x480. O controlo do veículo por teleoperação é assegurado por um joystick (Logitech RumblePad) ligado ao PC.

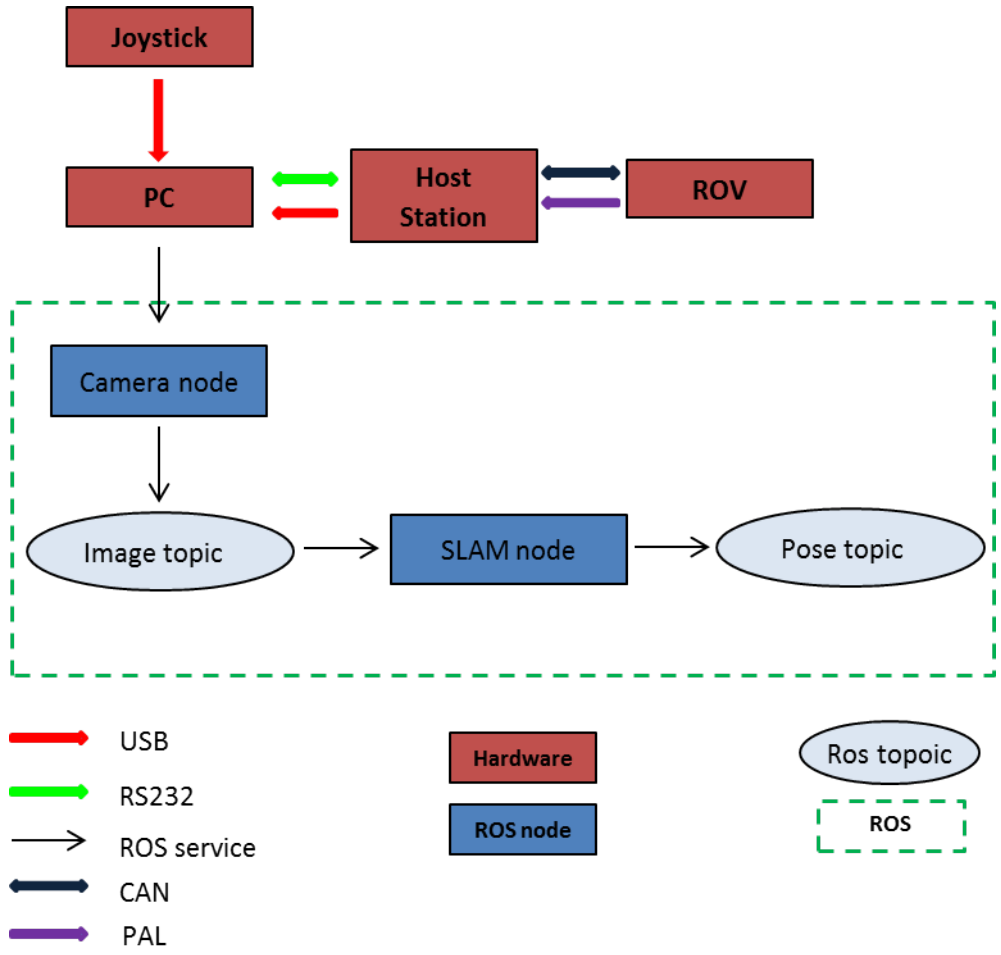


Figura 6.1: Arquitetura implementada para sistema de navegação subaquático (adaptado de [4]).

6.2 ROS

O *Robot Operating System* ou ROS [2], consiste num *middleware* orientado para desenvolvimento de aplicações robóticas. Este fornece uma camada de comunicação estruturada por cima de sistemas operativos bem como um conjunto de ferramentas de desenvolvimento e teste de código que facilitam o desenvolvimento de software robótico. ROS consiste num *framework* projetado para ser tão distribuído e modular quanto possível. A modularidade do ROS permite escolher as partes que se pretende implementar e quais as parte que interessam implementar no projeto a desenvolver.

Assim, ROS fornece serviços típicos de um sistema operativo, tais como abstração de *hardware*, controlo de dispositivos de baixo nível, comunicação através do envio de mensagens entre processos e gestão de pacotes. Conjuntos de processos em execução baseados em ROS são representados através de uma rede de "nós" que correspondem aos módulo de *software* que pertencem à aplicação em execução. Este grafo de nós é dinamicamente reconfigurável. Com efeito, a utilização de ROS oferece inúmeras vantagens que vão ser aproveitas na tarefa de desenvolvimento do *software* do algoritmo proposto para uma determinada plataforma robótica.

Para compreender a aplicação de ROS é importante conhecer os conceitos fundamentais. Esta *framework* é composta pelos nós, mensagens, tópicos e serviços. Os nós são processos que executam a computação, um sistema é tipicamente composta por muitos nós. Neste contexto, o termo "nó" é intercambiável com "módulo de software". Os nós comunicam uns com os outros pelo envio de mensagens. A mensagem consiste de uma estrutura de dados rigorosamente definida. Um nó envia uma mensagem para publicá-la através um determinado tópico. Um nó que está à espera de um determinado tipo de dados vai subscrever o tópico apropriado. Pode haver múltiplos publicadores de um tópico, e um único nó pode publicar e/ou subscrever vários tópicos. Finalmente, um serviço é definido por uma *string* de nome e um par de mensagens: um para a solicitação

e outro para a resposta.

6.2.1 Integração do projeto em ROS

O *software* em ROS está organizado em pacotes. Um pacote de ROS é simplesmente um diretório que contém um arquivo XML que descreve o pacote e integra a informação de todas as dependências. Um pacote pode conter nós ROS, uma biblioteca ROS independente, um conjunto de dados, arquivos de configuração, um pedaço de software de terceiros, ou qualquer outra coisa que constitui logicamente um módulo útil.

Para criar um novo pacote é utilizada a ferramenta *catkin* típica do *framework*. Esta desenvolvida pela comunidade ROS recorre à ferramenta CMake para o desenvolvimento do código. Para um pacote ser considerado um pacote *catkin* deve cumprir alguns requisitos: conter um arquivo *CMakeLists.txt* (Configuração tradicional do Cmake) e um *package.xml*. O primeira descreve como construir o código e onde instalá-lo. O outro define as propriedades sobre o pacote, como o nome do pacote, números de versão, os autores, mantenedores e dependências de outros pacotes *catkin* [42]. A implementação dos dois arquivos no novo pacote ROS exigiu a elaboração metódica do código segundo está definido na Tabela 6.1 para o arquivo CmakeLists.txt e na Tabela 6.2 para ficheiro XML.

Estruturas *CMakeLists.txt*	Função
cmake _ minimum _ required	Versão Cmake
project()	Nome do pacote
find _ package()	Outros pacotes necessários.
add _ message _ files()	Gerador de mensagens.
catkin _ package()	Fornecimento catkin do CMake.
add _ library()	Incluir bibliotecas.
add _ executable()	Incluir executáveis.
install()	Regras de Instalação

Tabela 6.1: Estruturas requiridas no arquivo CMake.txt.

	TAG	Descrição
Estrutura básica	<package> <//package>	*tag* raiz do documento *tag* raiz do documento
Tags requiridos	<name> <version> <description> <license> <maintainer>	Nome do pacote Versão do pacote Conteúdo do pacote Licença do *software* Entidade de manuteção
Dependências	<buildtool_ depend> <build_ depend> <run_ depend> <test_ depend>	Ferramentas de pacote Construção do pacote Corre código de pacote Teste ao pacote
Metapackages	<name>	Multiplos pacotes agrupados
Tags adicionais	<url> <author>	URL informativo Autor do pacote

Tabela 6.2: Estruturas requiridas no arquivo package.xml.

O processo de integração da infra-estrutura RT-SLAM em ROS consistiu em duas tarefas principais, por um lado a integração do código existente no processo e *framework* de desenvolvimento do ROS e por outro no desenvolvimento de código específico de interface e implementação de uma arquitectara que permita a disponibilização das funcionalidades pretendidas na rede de nós do ROS.

Na Figura 6.2 podemos observar de uma forma geral a hierarquia de software. O sistema operativo (Linux) fornece as funcionalidades básicas, nomeadamente os *device drivers* de interface com o hardware e *frameworks* como o *GStreamer* (*framework* multimédia), Qt ou GTK. É também usual a utilização de bibliotecas de *software* especializadas por forma a facilitar o desenvolvimento das aplicações. Neste caso, pode-se salientar entre outras as bibliotecas de cálculo matemático BOOST uBLAS e de visão computacional OpenCv. Como já foi referido anteriormente, o ROS para além de um ambiente de desenvolvimento fornece uma estrutura de suporte *runtime* para as aplicações robóticas. Em particular o serviço de comunicação de mensagens e gestão de parâmetros globais (implementado no *rosmaster*). A aplicação do robô irá consistir no conjunto de

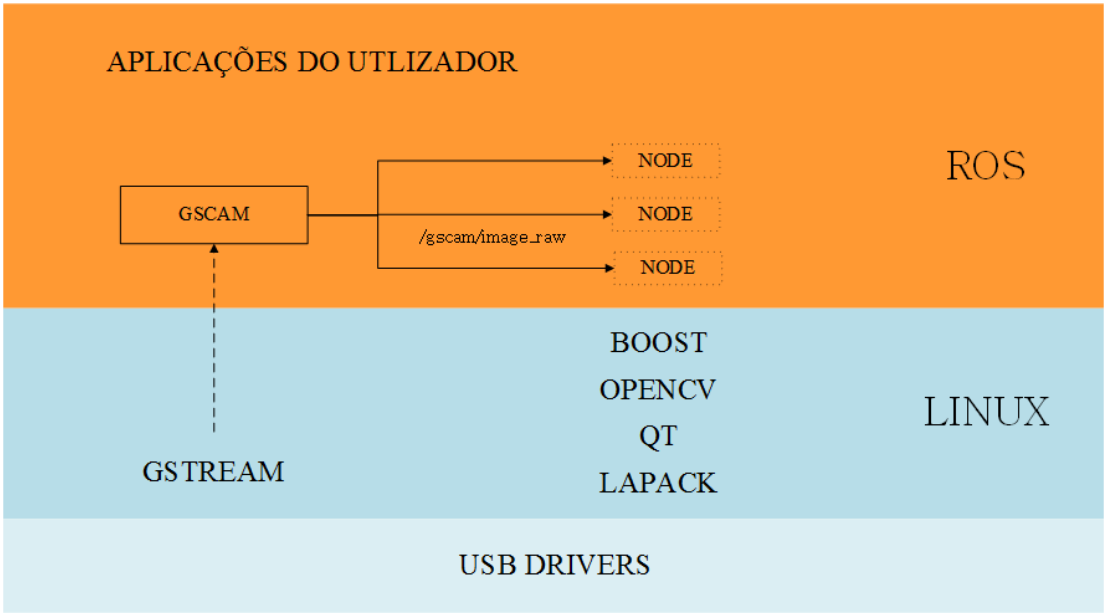


Figura 6.2: Hierarquia do Sistema.

nós em execução e na estrutura de comunicação por eles constituída. Aqui salienta-se a utilização de alguns nós como o gscam que permite disponibilizar imagens recolhidas pelo o *framegrabber* na infraestrutrura ROS (e consequentemente para os módulos SLAM e outros).

6.3 Aquisição de Imagem

A fase de aquisição de imagens consiste na obtenção de informação visual do robô móvel subaquático. Esta informação deverá ser posteriormente processada para a obtenção de *features* e posterior aplicação de visual SLAM. Na adquisição das imagens será necessário recorrer a *drivers* da câmara correspondente. ROS fornece já um conjunto de *drivers* para diferentes câmaras. A esquematização do processo de aquisição de imagem está representado na Figura 6.3.

A solução escolhida para a aquisição de imagens traduz-se na aplicação Gscam em plataforma ROS. Gscam utiliza o Gstreamer, um *framework* em Linux multimédia seme-

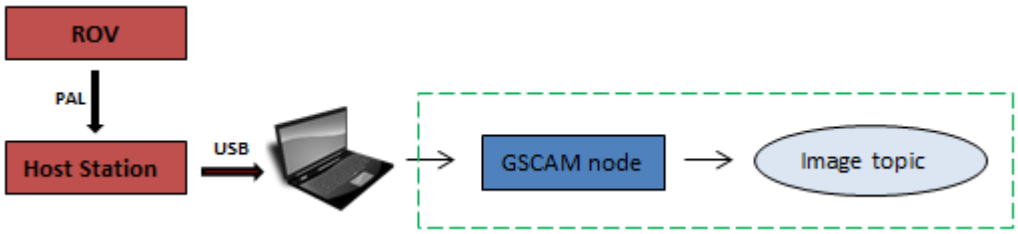


Figura 6.3: Esquema teste de aquisição de imagem.

lhante ao DirectShow em Windows. O Gscam pode ligar-se a um *pipeline* especialmente formatado. Portanto, enquanto o *pipeline* está a processar o vídeo RGB, Gscam irá transmitir o vídeo sobre um formato *standard* de transporte de imagem ROS e câmara ROS. O Gstreamer é compatível com quase todos os padrões de captura de vídeo no Linux, desta forma a aplicação do Gscam permite a utilização de um vasto número de diferentes câmaras, permitindo a utilização de qualquer câmara ROS compatível ou web-cam como sensor para a implementação do RT-SLAM. Note-se que na sua configuração inicial o RT-SLAM apenas suporta um modelo de câmara [43].

O "Roslaunch" é uma das ferramentas disponibilizadas por ROS, é utilizada para facilitar o lançamento simultâneo de nós ROS. Além disso incluí arquivos de configuração XML que especificam os parâmetros que definem os nós. Para a aquisição da imagem foi utilizada a referida ferramenta para lançar o nó do Gscam . A utilização desta aplicação já permite a introdução dos parâmetros intrínsecos e obtenção de imagens retificadas.

6.4 Calibração de Parâmetros Intrínsecos

A calibração consiste em determinar os parâmetros que caracterizam o sistema, o ambiente em que se insere e o modo como vai ser utilizado. Os parâmetros intrínsecos servem para relacionar as coordenadas *pixel* relativamente às imagem com as coordenadas de pontos do espaço medidos no sistema referencial com origem no centro da câmara. De notar que estes parâmetros dependem exclusivamente das características físicas da

câmara (da sua geometria interna, do tipo de lente) [32].

A determinação dos parâmetros intrínsecos é efetuada a partir da relação das coordenadas 3D do espaço com a sua respectiva projeção no plano da imagem. Tendo as coordenadas da câmara de uma série de pontos no espaço determinam-se as coordenadas normalizadas desses mesmos pontos. As coordenadas normalizadas são obtidas dividindo as duas primeiras componentes pela terceira (seguindo a convenção do eixo dos zz's ser o perpendicular à superfície da lente). Note-se que deste modo todos os pontos de um raio projetante (com origem no centro da lente) terão as mesmas coordenadas normalizadas. Com os resultados da calibração é possível estimar a distorção da lente que irá afetar estes pontos normalizados. Geralmente a influência da distorção radial é bastante superior comparativamente ao da distorção tangencial. A distorção será tanto maior quando maior for a distância normalizada destes *pixels* ao ponto principal normalizado.

As expressões matemáticas que caracterizam o processo de calibração dos parâmetros internos da câmara estão devidamente descritos no Capítulo 4. Onde o vector da distorção tangencial é dado pela expressão 4.13 , o vector da distorção radial é expresso segundo a Equação 4.14. A matriz **K** que permite converter coordenadas normalizadas em coordenadas pixel da forma que está expresso na Equação 4.7. Os parâmetros obtidos na calibração são os seguintes:

- Distância focal (em *pixels*);
- Ponto principal da imagem;
- Coeficientes de distorção radial;
- Coeficientes de distorção tangenciais.

6.4.0.1 Metodologia

A exatidão deste processo poderá comprometer o funcionamento do RT-SLAM. A calibração deverá ser feita com um elevado nível de confiança para garantir bons resultados na estimativa da posição do robô. Além disso convém utilizar um método relativamente simples e rápido. A solução encontrada pertence à plataforma ROS, "Camera Calibration".

O *package* "Camera calibration" fornece ferramentas para a calibração de câmaras monoculares e stereo, neste caso apenas foi preciso a primeira aplicação. Este pacote utiliza um *software* incluído na OpenCV (Open Source Computer Vision Library) para determinar os parâmetros intrínsecos da câmara [44]. A utilização deste *software* requer a utilização dos Gscam para aquisição de dados da câmara e imagem através da subscrição de tópicos ROS. O gráfico representativo do processo ROS e respectivo hardware pode ser observado na Figura 6.4.

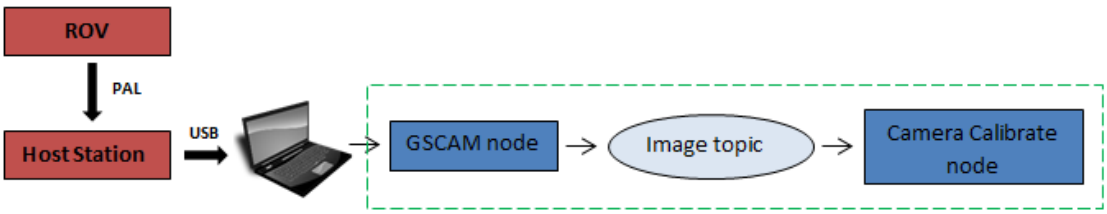


Figura 6.4: Esquematização de processo de calibração dos parâmetros intrínsecos.

A calibração é realizada a partir de múltiplas imagens e posições diferentes de uma grelha quadriculada. O padrão utilizada é constituída por quadrículas de 0.08 metros que podem ser observados na Figura 6.5. A partir da dimensão conhecida das quadrículas, o programa determina qual a posição exata no espaço tridimensional que cada ponto de grelha ocupa.

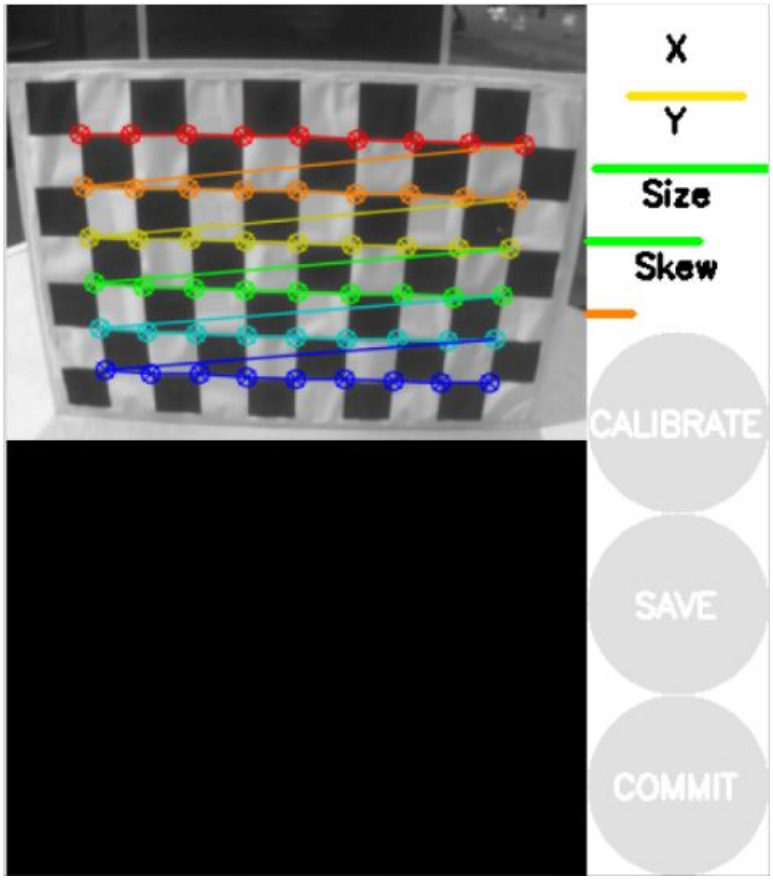


Figura 6.5: *Software* de calibração de parâmetros intrínsecos.

O *software* de calibração mostra as imagens das câmaras atuais, com 3 botões disponíveis para o utilizador (Figura 6.5). O botão CALIBRAR, calcula os parâmetros de calibração de câmara, o botão SAVE guarda os parâmetros num ficheiro, o botão COMMIT carrega esses novos parâmetros de calibração para o *driver* em utilização.

Finalizado o processo de calibração dos parâmetros intrínsecos, os respectivos valores da calibração são devolvidos pela aplicação sob a forma de um *script*. Os dados obtidos para a calibração da câmara monocular do VideoRay ROV podem ser consultados na Tabela 6.3.

Parâmetro	Valor
Dimensões (largura,altura)	592, 576 (pixel)
Distância Focal (fx,fy)	397.135190, 395.132498 (pixel)
Ponto Principal (px,py)	299.522697, 285.144424 (mm)
Parâmetros de distorção	-0.295967, 0.070487, 0.002813, 0.001537

Tabela 6.3: Valores dos parâmetros intrínsecos.

6.5 Algoritmo SLAM

A sequência apresentada na Secção 6.2.1 expõe o procedimento necessários para integrar um projeto no *framework* da plataforma ROS. A execução adequada de RT-SLAM exige a correta interpretação do formato dos dados, no caso, no formato da imagem. Assim como um conhecimento exaustivo dos principais ficheiros de configuração. RT-SLAM recebe imagens de formato "png" e os ficheiros de configuração podem ser consultados na Tabela 6.4. Os ficheiros "mocap.log" e "MTI.log" não foram utilizados, visto que o objectivo da projeto em desenvolvimento consiste reporta uma aplicação visual SLAM.

Ficheiro	Descrição
setup.cfg	Configuração de Hardware.
estimation.cfg	Configuração da estimativa. SLAM
sdate.log	Tempo de iniciação.
mocap.log	Log do GPS.
MTI.log	Log do inercial.

Tabela 6.4: Ficheiros de configuração RT-SLAM.

6.5.1 Aquisição de imagem SLAM

Com o projeto integrado em pacote ROS, um nó será responsável pela aquisição da imagem. O nó Gscam vai ser responsável pela publicação de tópicos que contêm informação da imagem e câmara ROS. Os nós são os responsáveis pelo transporte de mensagens, são destinados para uma comunicação *streaming* unidirecional. As mensagens definem quais os tópicos informativos transferidos através do preenchimento de

campos, a mensagem ROS responsável pela aquisição da imagem encontra-se descrita na Tabela 6.5.

Campo	Descrição
headear	Composto por campos do tipo Header, *seq*, *stamp* e $frame\_id$.
height	Largura da imagem.
width	Altura da imagem.
encoding	Codificação de pixeis.
step	Comprimento de linhas em bytes.
data	Matriz da imagem.

Tabela 6.5: Mensagem imagem ROS.

O ficheiro "setup.cfg" contém os dados de configuração da câmara, possui um campo que nos permitir selecionar o *driver* desejado (para aplicações online apenas são suportadas a câmaras USB Firewire e Ueye USB câmaras). Um novo *driver* baseado no *gstreamer* em ROS foi instalado, este permite a utilização de qualquer câmara USB e deve ser definido adequadamente no ficheiro "setup.cfg". O *device driver* foi desenvolvido em C++ e já converte a imagem para o formato pretendido e lê igualmente valores do *time stamp* a partir dos campo recebidos no tópico imagem recebido.

A aplicação do *driver* em RT-SLAM já integrado em plataforma ROS consistiu no desenvolvimento de dois ficheiros ficheiros, o "hardwareSensorCameraRosTopic.cpp" e "hardwareSensorCameraRosTopic.hpp". No primeiro ficheiro foram criadas as funções responsáveis pela leitura e escrita dos dados da imagem e *timestamp* (6.6), no segundo encontram-se instanciadas as estruturas que contêm todas as variáveis. A função "callback" do ficheiro lê os dados adquiridos da imagem em bgr e converte para cvImage especificando a codificação desejada, rgb. Os drivers e respectiva opção podem ser consultados na Tabela 6.7.

O controlador instalado em *framework* ROS permite correr a aplicação de SLAM

online e *offline*. Desta forma, é aplicável quando o sensor está a receber e publicar diretamente informação ou então a partir da utilização de dados previamente guardados em ferramentas ROS (*rosbag*) que permitem publicar os respetivos tópicos sempre que necessário. Isto permite correr SLAM com dados adquiridos as vezes necessárias até corrigir uma estimação adequadamente para SLAM.

Função	Descrição
callback()	Sempre que adquirida uma *frame* converte e guarda a imagem no *buffer* do rtslam.
init()	Inicializa inicializa o *buffer* das imagens.
HardwareSensorCameraRosTopic	Cria objeto objeto.
start()	Inicia a leitura do tópico ROS.
stop()	Pára a leitura do tópico ROS.
join()	Função aguarda fim da thread de leitura.

Tabela 6.6: Lista de funções "hardwareSensorCameraRosTopic.cpp".

6.5.2 Publicação da posição SLAM

O objetivo da aplicação SLAM corresponde à localização e mapeamento de um veículo subaquático. Assim, a obtenção da posição do veículo é igualmente executada através de mensagens ROS, sempre que processada uma *frame* um tópico da posição do robô móvel é lançado. A comunicação entre os nós ROS é feita através de mensagens, que consistem em estruturas de dados simples, composto por campos digitados. O arquivo de texto simples responsável pela publicação da posição do robô móvel é constituído por 4 campos *header*, id, pose e *convariance*, conforme descrito na Tabela 6.8.

Driver	Tipo de câmara
Câmaras USB Firewire	"1"
Ueye USB câmaras	"2"
Offline	"3"
ROS	"4"

Tabela 6.7: Tabela *drivers* câmara.

Campo	Descrição
headear	Composto por campos do tipo Header, *seq*, *stamp* e $frame\_id$.
position	Posição segundo x, y, z.
orientation	Quaternion da posição.
covariance	Covariância da posição

Tabela 6.8: Mensagem *pose* ROS.

A visualização gráfica é uma das características típicas da plataforma ROS, que disponibiliza ferramentas especificas para o efeito. O comando "rqt_graph" fornece um *plugin* GUI para visualizar o gráfico de computação ROS. O gráfico ROS da implementação SLAM pode ser consultado na Figura 6.6.

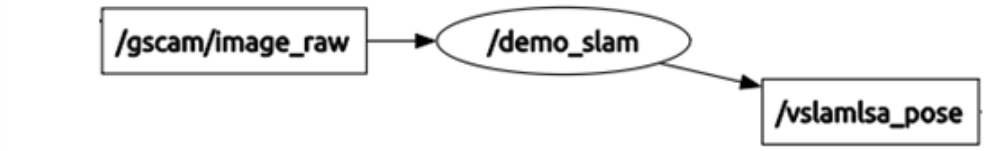


Figura 6.6: Gráfico ROS da implementação SLAM.

6.6 Integração de *Features*

A visão computacional tem sido um grande contributo no desenvolvimento de algoritmos SLAM. No entanto, problemas típicos como a associação e altas taxas de deteção de dados, assim como manter a correspondência entre uma medição e um marcador. Outros incluem o atraso do subsistema de entradas e saídas para fornecer ao processador de dados da imagem, uma velocidade suficientemente elevada.

Técnicas de deteção de *features* consistem em operações de processamento de imagem de baixo nível, representando o cálculo das características locais da imagem ou

conteúdo de informação local na imagem. É o ponto de partida para a extração do marcador e associação de dados. As propriedades distintas do resto da imagem fazem do ponto de interesse uma escolha primária na visão SLAM.

6.6.1 Deteção de Ponto

A aplicação da técnica de deteção de pontos de interesse tem sido bastante utilizada em sistemas de localização e mapeamento (Conforme descrito no Capítulo 2). Os pontos caracterizam-se por uma definição matemática clara com uma posição bem definida na área de imagem. Além disso, não é suscetível a interferências, como a deformação (isto é, de orientação ou alterações de escala). Estes atributos são compatíveis com a técnica de deteção de *Harris*.

A técnica de *Harris* consiste numa um detector de canto e *edge* combinados com base na função de auto-correlação local [33]. Esta é utilizada para a deteção de pontos de interesse para a estimação da posição do robô móvel em ambientes não estrturados. A matriz correlação A é descrita como,

$$A(x) = \sum_{x,y} \begin{vmatrix} I_x^2(x) & I_x I_y(y) \\ I_x I_y(x) & I_y^2(y) \end{vmatrix} \tag{6.1}$$

onde I_x e I_y são as respectivas derivadas na direção de x e y, o $w(x, y)$ corresponde à função ponderação.

A extração de pontos é baseada no método *Harris* com várias optimizações. Uma delas consiste na utilização de uma máscara derivativa, $[-1,\ 0,\ 1]$, assim como uma máscara de convulsão quadrada e constante, a fim de minimizar as operações [30] .

6.6.2 Deteção de Linha

Alguns autores [22] já abordaram a temática da utilização de linhas como re-

ferências SLAM em ambientes subaquáticos com resultados positivos. No entanto, a implementação realizada é um pouco divergente relativamente ao algoritmo proposto por esse mesmo autor.

O algoritmo implementado para a deteção de segmentos de linha em uma imagem é baseado na abordagem descrita em [45]. A abordagem detecta segmentos de forma incremental com base no gradiente da imagem, usando um filtro de Kalman linear, que estima os parâmetros de os segmentos de suporte e os desvios associados.

O modelo aplicado para o segmento corresponde ao descrito segundo a expressão que se segue:

$$x(t) = a.t + x_0 \tag{6.2}$$
$$x(t) = a.t + y_0 \tag{6.3}$$

onde (a, b) é a direção do vector diretor da linha e origem(x_0, y_0).

O princípio do algoritmo é adaptar os parâmetros do modelo para o segmento da imagem. Como seria computacionalmente muito pesado detectar um segmento em todos os pixels: o processo é iniciado apenas em alguns pixels ("sementes"), que correspondem a um gradiente máximo local. Considerando $G_{i,j}$ e $\phi_{i,j}$ a norma do gradiente e a fase calculado para o pixel (i,j), pelo algoritmo de Canny [46]. Um pixel (i,j) é considerado uma "semente" se seguir as seguintes condições:

$$G_{i,j} - G_i + \cos(\phi_{i,j}), j + \sin(\phi_{i,j}) > \tau_{gradmax} \tag{6.4}$$

$$G_{i,j} - G_i - \cos(\phi_{i,j}), j - \sin(\phi_{i,j}) > \tau_{gradmax} \tag{6.5}$$

$$G_{i,j} > -G_i - \sin(\phi_{i,j}), j + \cos(\phi_{i,j}) \tag{6.6}$$

$$G_{i,j} > -G_i - \sin(\phi_{i,j}), j - \cos(\phi_{i,j}) \tag{6.7}$$

As duas primeiras condições indicam que o pixel em causa é provavelmente localizado num segmento. As outras duas condições asseguram que o pixel corresponde a um

gradiente máximo ao longo do segmento assumido: são utilizados para selecionar as "sementes" mais robustas. Dado um pixel considerado como uma semente, os parâmetros do segmento associado que inicializam o estado do filtro de Kalman são:

$$x_0 = i, y_0 = j \tag{6.8}$$

$$a = -\sin(\phi_{i,j}), b = \cos(\phi_{i,j})) \tag{6.9}$$

Uma vez que um pixel é considerado uma semente é utilizado um processo iterativo de expansão. Portanto, procura pontos adicionais ao longo da direção estimada do segmento de acordo com o procedimento que se segue:

- $t = 1$
- δ_t é a distância até as extremidades do segmento atual, para o qual são pesquisados novos pontos de apoio.
- Previsão: baseada nas estimação dos parâmetros do segmento atual e respetivas variâncias, uma estimativa das coordenadas (x_t, y_t) do próximo ponto de suporte é determinada, assim como o erros escalar associado σ_n para a distância do segmento atual.
- Observação: um conjunto de medições são feitas sobre o segmento normal que passa através de (x_t, y_t), e um processo de seleção que determina que irá ser utilizado quando a atualização dos parâmetros do segmento.

O processo de seleção das observações é o seguinte, seja $a' = a\sqrt[2]{a^2 + b^2}$ e $b' = b\sqrt[2]{a^2 + b^2}.s = (-b' \times \sigma_n, a' \times \sigma_n n_0)^T$ é o vetor diretor para o qual são procuradas duas observações, com uma norma que depende do erro $sigma_n$ e de n_0, uma constante que define o número de medidas que serão avaliadas. As medidas são:

$$M = \{(x_t, y_t) + i \times s\}, \in [-n_0, n_0] \tag{6.10}$$

A medição $m_i \epsilon M$ é selecionada como a observação atual do segmento sendo aquela que satisfaz as condições seguintes:

- Este é um gradiente máximo local:

$$G_{xt,yt} + i \times s > G_{xt,yt} + (i+1) \times s \tag{6.11}$$

$$G_{xt,yt} + i \times s > G_{xt,yt} + (i-1) \times s \tag{6.12}$$

- A direção do seu gradiente é consistente com o segmento atual:

$$\cos(\phi_{(xt,yt)} + i \times s - \arctan 2(a, -b)) > \tau_{angle} \tag{6.13}$$

No caso de ambos os pontos m_{i1} e m_{i2} satisfazerem estas condições, a sua distância do segmento atual é avaliada em primeiro lugar, sendo o mais próximo selecionado. Se as distâncias forem iguais, o ponto na fase do gradiente é o mais compatível com o segmento actual é selecionada como uma observação. Portanto, m_i1 é selecionado se $|i_1| < |i_2|$, no caso de $|i_1| = |i_2|$, m_{i1} é selecionado se:

$$\cos(\phi_{(xt,yt)} + i \times s - \arctan 2(a, -b)) < \cos(\phi_{(xt,yt)} + i \times s - \arctan 2(a, -b)) \tag{6.14}$$

Quando nenhum ponto está selecionado como a observação do processo de expansão, a pesquisa é estendida a um passo adicional $(t \pm \delta_t)$ para impedir a ruptura causada por segmentos de ruído na imagem. O processo de expansão, em seguida, para se não houver pontos adicionais selecionados como observação.

Os parâmetros da linha são estimados por um filtro Kalman, o estado estacionário do filtro é descrito por:

$$x_k = x_{k-1} \tag{6.15}$$

Os parâmetros do vetor de estado são inicializado como indicado nas expressões 6.8 e 6.8. A matriz covariância associada a esta inicialização é a seguinte,

$$p_{0|0} = \begin{bmatrix} \sigma_a^2 & 0 & 0 & 0 \\ 0 & \sigma_{x0}^2 & 0 & 0 \\ 0 & 0 & \sigma_b^2 & 0 \\ 0 & 0 & 0 & \sigma_{x0}^2 \end{bmatrix} \tag{6.16}$$

A equação para o modelo de observação é dada por:

$$z_k = H_k \times x_k + v_k \tag{6.17}$$

Este é um modelo de observação linear, onde v_k é o ruído de observação de covariância R, e H_k é a matriz de observação:

$$H_k = \begin{bmatrix} t & 1 & 0 & 0 \\ 0 & 0 & t & 1 \end{bmatrix} \tag{6.18}$$

O erro da observação é definido por duas variâncias, a variância ao longo da linha é igual σ_t^2 e a variância na direção normal à linha é igual a $0,5^2$. Os valores utilizados são pessimistas de modo que a condição do filtro não se torne inconsistente. A matriz do erro é definida por:

$$\begin{bmatrix} \sigma_t^2 & 0 \\ 0 & 0.5^2 \end{bmatrix} \tag{6.19}$$

Para mudança de referência, expressa na referência da imagem é:

$$R_k = \mathfrak{R} \begin{bmatrix} max(\sigma_t^2) & 0 \\ 0 & 0.5^2 \end{bmatrix} \times \mathfrak{R}^T \tag{6.20}$$

onde $\mathfrak{R}$ corresponde à matriz de rotação definida por:

$$\mathfrak{R} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{6.21}$$

Após o processo de deteção, pode ocorrer a sobreposição de segmentos. Neste caso, um processo de fusão é aplicado para verificar se as extremidades do segmento Sp podem ser considerados como observações do segmento Sn. Se sim, então Sn e Sp são fundidos, e os parâmetros do segmento Sn são atualizados no filtro de Kalman.

6.7 Ferramentas MATLAB

A avaliação do desempenho da aplicação foi uma das fases no desenvolvimento de um método adequado de navegação subaquática. A obtenção de informação sobre o *performance* de SLAM iniciou com a publicação de tópicos ROS. Os dados publicados foram convertidos para um *log* e transformados em gráficos de análise com o auxílio do MATLAB (conforme esquematizado na Figura 6.7).

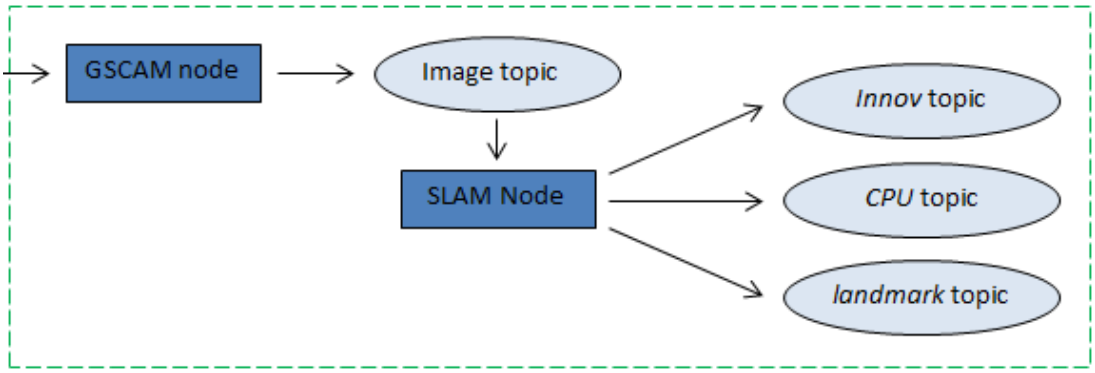


Figura 6.7: Gráfico ROS estatísticas.

6.7.1 Análise de *landmarks*

As *landmarks* representam os marcadores que permitem o reconhecimento SLAM durante a navegação de uma determinada plataforma robótica. Para uma correta interpretação do seu desempenho num dado cenário de aplicação foi necessário a avaliação de acordo com a evolução temporal ao longo do tempo das *landmarks*. A avaliação quanti-

tativa das *landmarks* permite reconhecer os marcadores que constam no campo de visão do robô móvel. Ou seja, avaliar a quantidade de marcadores detetados por cada *frame* processada pelo detector quer de pontos de interesse quer de linhas. Por sua vez, o tempo que uma das *landmark* permanece no campo de visão alcançado pelo robô denuncia uma análise qualitativa das mesmas. O tempo que a *landmark* permanece visível dá-nos uma indicação da qualidade da medida. Na Tabela 6.9 encontram-se descritos os campos da mensagem ROS responsável pela publicação de dados *landmark*.

Campo	Descrição
headear	Composto por campos do tipo Header, *seq*, *stamp* e $frame\_id$.
lmk_ id	id da *landmark*.
lmktype	tipo *landmark*, linha ou ponto.
converg	Estado convergente.
init	Estado inicializado
covariance	Valores de covariância

Tabela 6.9: Mensagem inovação ROS.

6.7.2 Análise do estimador

A inovação [11] é utilizada como um método de avaliar a qualidade das medidas obtidas ao longo das experiências realizadas. Como os estados "verdadeiros" nunca estão disponíveis a inovação é a única medida que permite avaliar o *performance* do estimador. A inovação é aplicada para "afinar" o estimador e construir modelos das fontes de ruído de observação e de processo. A inovação consiste no desvio das medidas entre a sequência de observação $z(k)$ e a previsão da observação $H(k)\hat{x}(k|k-1)$,

$$v(k) = z(k) - H(k)\hat{x}(k|k-1) \tag{6.22}$$

A aquisição de dados estatísticos a partir do tópico ROS pode ser observado na Tabela 6.10.

Campo	Descrição
header	Composto por campos do tipo Header, *seq*, *stamp* e $frame\_id$.
avg_ innov	Média da Inovação.
max_ innov	Máximo da Inovação.
nor_ innov	Inovação Normalizada.
avg_innov_pos	Média da inovação da posição.
avg_innov_ori	Média da inovação da orientação.

Tabela 6.10: Mensagem *landmark* ROS.

6.7.3 Processamento de dados

O processamento de dados consiste de uma série de atividades executadas ordenadamente, que resultará numa espécie de arranjo de informações. Inicialmente são coletadas informações, ou dados, que passam por uma organização que deverá ter o arranjo adequado à aplicação final do utilizador. O tempo de processamento dos dados é especialmente exigente em sistemas que requerem aplicações em tempo real. O tempo de aquisição de leitura de dados e respectivo processamento é avaliado no sentido de analisar qual o impacto das atividades executadas no tempo de processamento. A mensagem responsável pela aquisição de dados sobre o processamento está descrita na Tabela 6.11.

Campo	Descrição
headear	Composto por campos do tipo Header, *seq*, *stamp* e $frame\_id$.
update_ ti	Tempo inicial.
update_ tf	Tempo final.
wait_ time	Tempo de espera de dados.
convergcount	Número de *landmarks* convergidas.
initcount	Número de *landmarks* convergidas..
linecount	Número de Linhas.
n_ obs	Número de observações.

Tabela 6.11: Mensagem CPU ROS.

CAPÍTULO 7

Resultados

7.1 Deteção de *Landmarks*

A trajetória do ROV VideoRay foi realizada por controlo remoto para obter dados experimentais da câmara monocular. A deteção de *features* do ambiente em condições operacionais são dadas por estruturas subaquáticas quer num ambiente de campo controlado, quer noutros cenários de operação. Durante a descrição da trajetória, o veículo foi recolhendo as medidas a partir da câmara frontal. Estes dados são utilizados para extrair as *features* visuais que permitem a obtenção da localização do veículo e construção do respectivo mapa. As *landmarks* a detetar constam de dois tipos destintos, pontos e linhas, é possível selecionar a utilização de apenas um tipo ou os dois em simultâneo. O primeiro método é baseado no detector de canto de Harris [28].

7.1.1 Resultados em ambiente controlado

O tanque de testes apresentado na Figura 7.3 encontra-se nas instalações do Laboratório de Sistemas Autónomos (LSA). Este foi utilizado na realização de experiências e teste para a validação da abordagem proposta nesta dissertação. As dimensões do tanque correspondem aproximadamente a uma largura de 6 metros, comprimento de 11 metros e uma profundidade próxima dos 5 metros.

Figura 7.1: Tanque de Testes.

As *landmarks* do tipo ponto detetadas em ambiente controlado encontra-se ilustradas na Figura 7.2. Os resultados na deteção de linhas para o mesmo ambiente controlado são apresentados na Figura 7.3. As estruturas subaquáticas presentes no tanque de testes tiveram um importante papel no mapeamento da trajetória do ROV. Objetos como janelas, câmaras, vigas e redes foram utilizados na determinação da posição do ROV. Como se pode observar nas imagens encontram-se marcos assinalados que correspondem às *landmarks* observadas ou não pelo robô subaquático. As cores são atribuídas segundo o código que se segue:

- **magenta:** *Landmarks* com parametrização inicial que foram observados na *frame*;
- **vermelho escuro**: *Landmarks* com parametrização inicial que não foram observados na *frame*;
- **ciano**: *Landmarks* com parametrização convergente que foram observados na *frame*;
- **azul escuro**: *Landmarks* com parametrização convergente que não foram observados na *frame*;

Por sua vez, a representação das *landmark* em cada frame adquirida segue uma determinada topologia:

- **cor e cruz +**: a posição prevista da *landmark*;
- **uma elipse de cor**: a incerteza previsão de 3 sigma;
- **uma cruz x laranja**: a posição medida da *landmark*;
- **etiqueta colorida**: o id do marcador, seguido da pontuação (qualidade da medida).

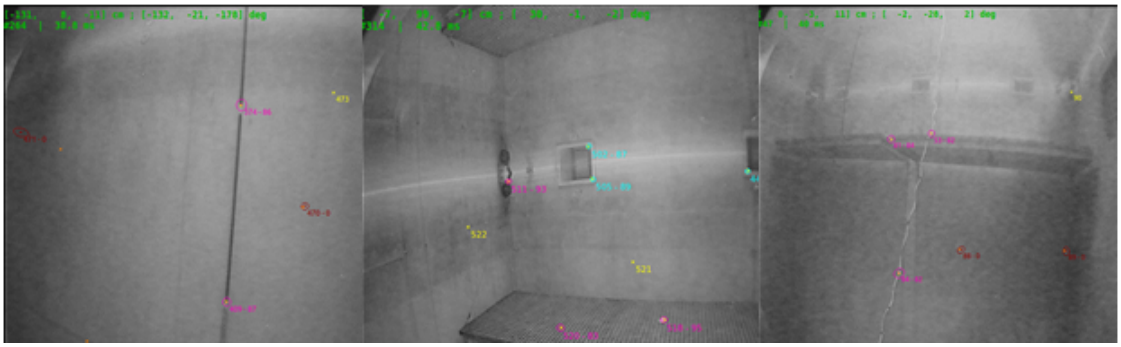


Figura 7.2: Deteção de pontos no tanque de testes.

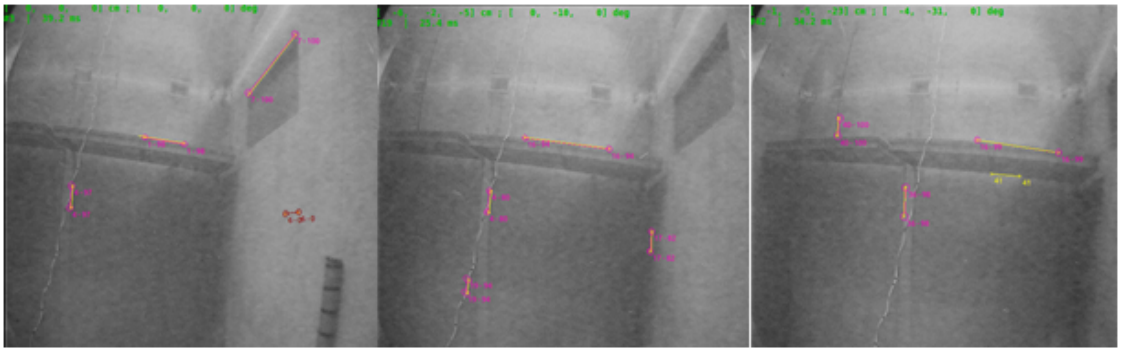


Figura 7.3: Deteção de linhas no tanque de testes.

7.1.2 Resultados em cenários de aplicação

A flexibilidade da abordagem proposta para sistemas de navegação subaquática permite correr aplicações *offline*. Os dados foram adquiridos a partir de uma câmara, guardados e posteriormente processados. Os testes foram realizados em cenários como um gasoduto e leito de um rio para avaliar o desempenho da deteção das *landmarks* em ambientes não estruturados. A obtenção da trajetória do veículo subaquático foi realizada através da deteção de pontos e linhas nas *frames* processadas. Para efeitos de teste foram adquiridas dados de um gasoduto a mais de 1000 metros de profundidades, assim como do leito de um rio (Rio Douro). Os resultados obtidos da deteção de *landmarks* nos cenários de aplicação do algoritmo proposto encontram-se evidenciados nas Figuras 7.4 e 7.5.

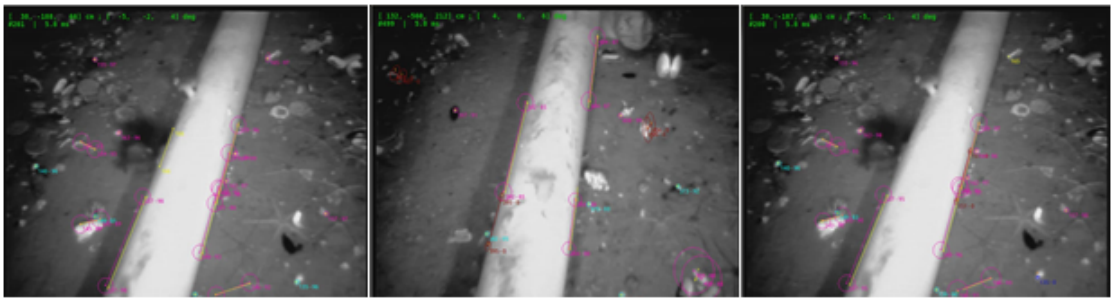


Figura 7.4: Deteção de linhas e pontos num gasoduto.

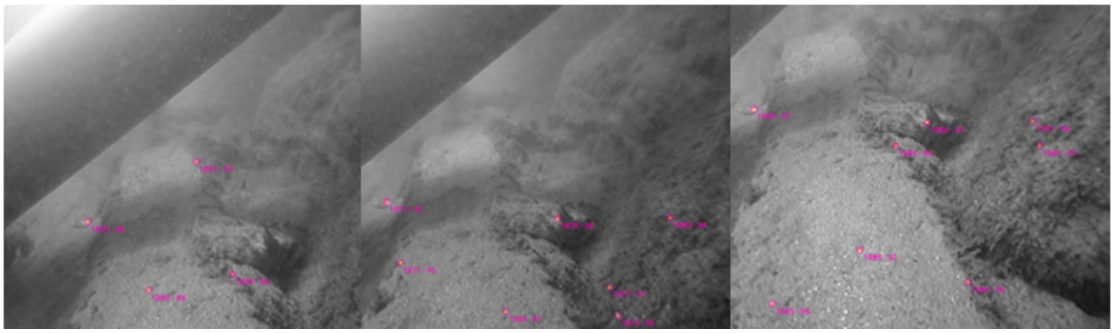


Figura 7.5: Deteção de linhas e pontos no leito do Rio Douro.

7.2 Trajetória simples do ROV

Os resultados experimentais obtidos para a trajetória do ROV (Figura 7.6) utilizando a abordagem foram obtidos através de um *script* em MATLAB. Como os dados encontram-se no formato de quaterniões por esse motivo obteve-se o *heading* do veículo subaquático com os ângulos *roll*, *pitch* e *yaw* (Figura 7.7).

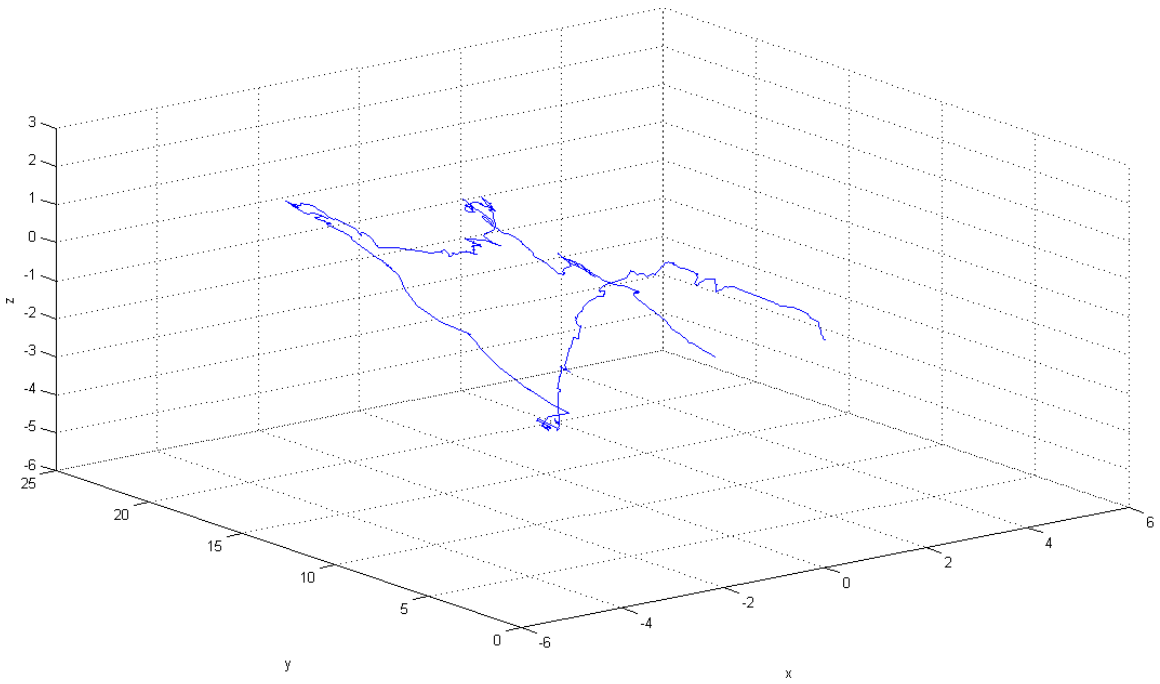


Figura 7.6: Trajetória realizada pelo Video-Ray.

7.3 Trajetória com validação *Ground truth*

A fim de obter uma validação externa da aboradagem RT-SLAM, utilizou-se o sistema INESC TEC/ISEP *indoor ground truth* [2]. O *ground truth* consiste num sistema de percepção visual baseado em visão stereo utilizado para a validação externa da localização do robô em ensaios subaquáticos. Na Figura 7.8 encontra-se ilustrado o cenário da aquisição de dados realizado em ambiente controlado, com o VideoRay e uma das câmaras *ground truth* (assinalada a vermelho).

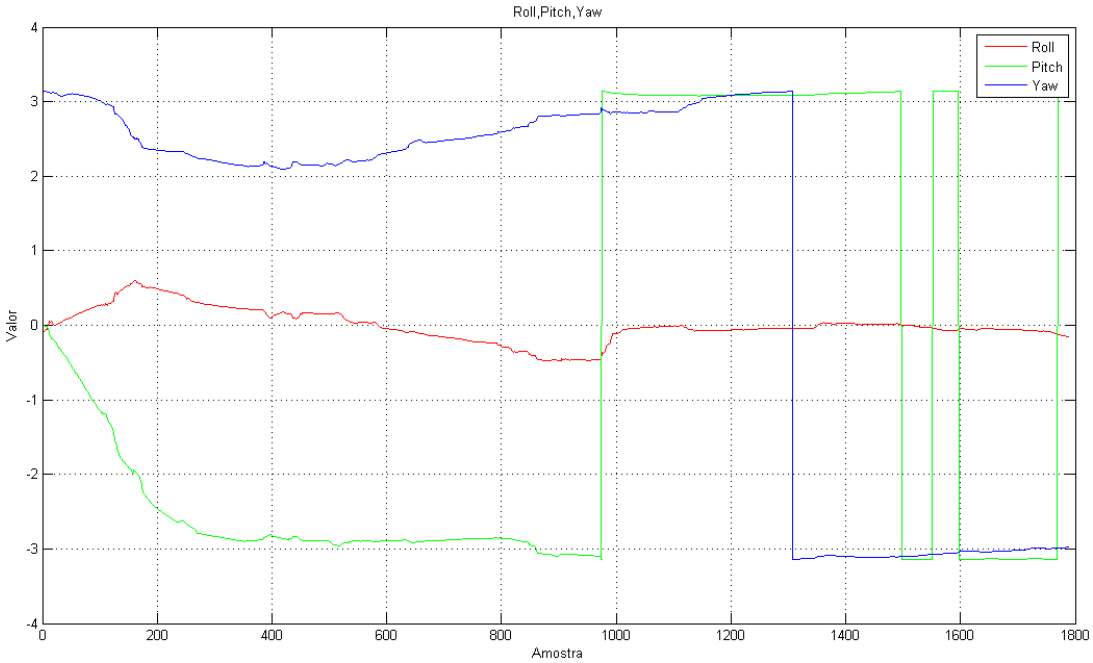


Figura 7.7: Roll, pitch e yaw do Video-Ray

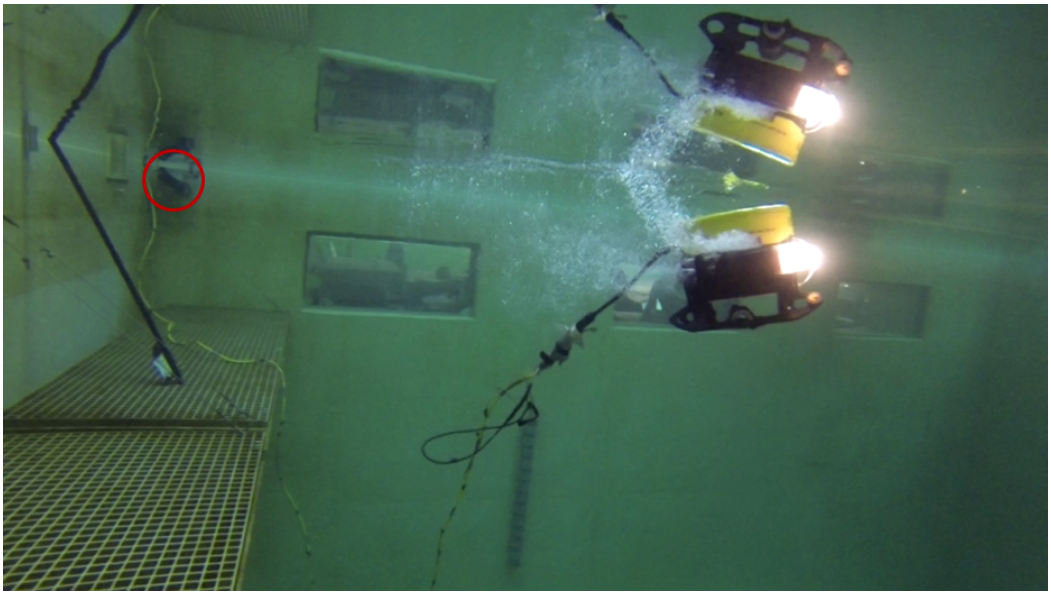


Figura 7.8: Cenário *ground truth*.

7.3.1 *Setup* experimental

A obtenção de dados em simultâneo das duas câmeras Basler acA1300-30gc a cores GigabitEthernet (GigE) e respectiva câmara do ROV, exigiu um esforço adicional para garantir a sincronização dos dados recebidos das três câmaras (observar Figura 7.9). As câmaras do *ground truth* estão conetadas num switch Gigabit que fornece interface de rede executado num CPU Intel i5 em sistema operativo Linux. Os dados da câmara do ROV são adquiridos pelo mesmo portátil através de um CABO USB ligado a uma estação *Host* recebe o sinal de vídeo através de um *framegrabber* onde é convertido e digitalizado. Os dados adquiridos das três câmaras são guardados através de uma ferramenta ROS, o *rosbag*.

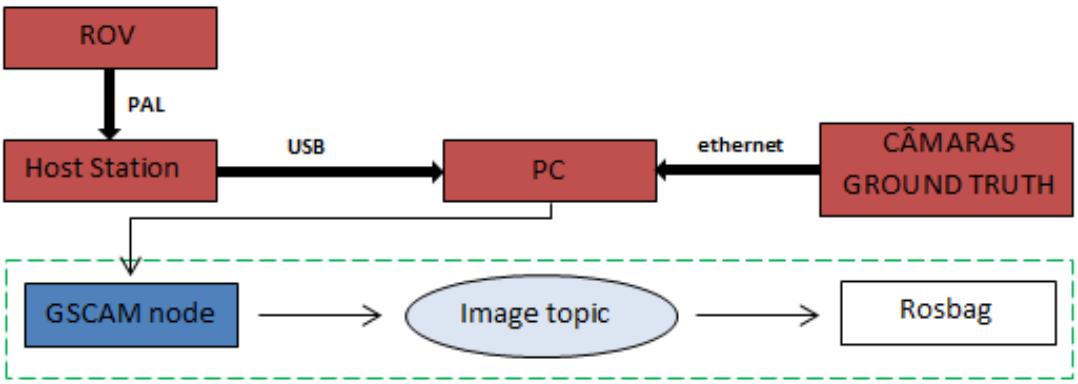


Figura 7.9: *Setup* experimental para validação ground truth.

7.3.2 Trajetória realizada

A trajetória obtida das manobras realizadas pelo veículo subaquático para a abordagem proposta e respectivo sistema de validação *ground truth*, podem ser consultadas na Figura 7.10.

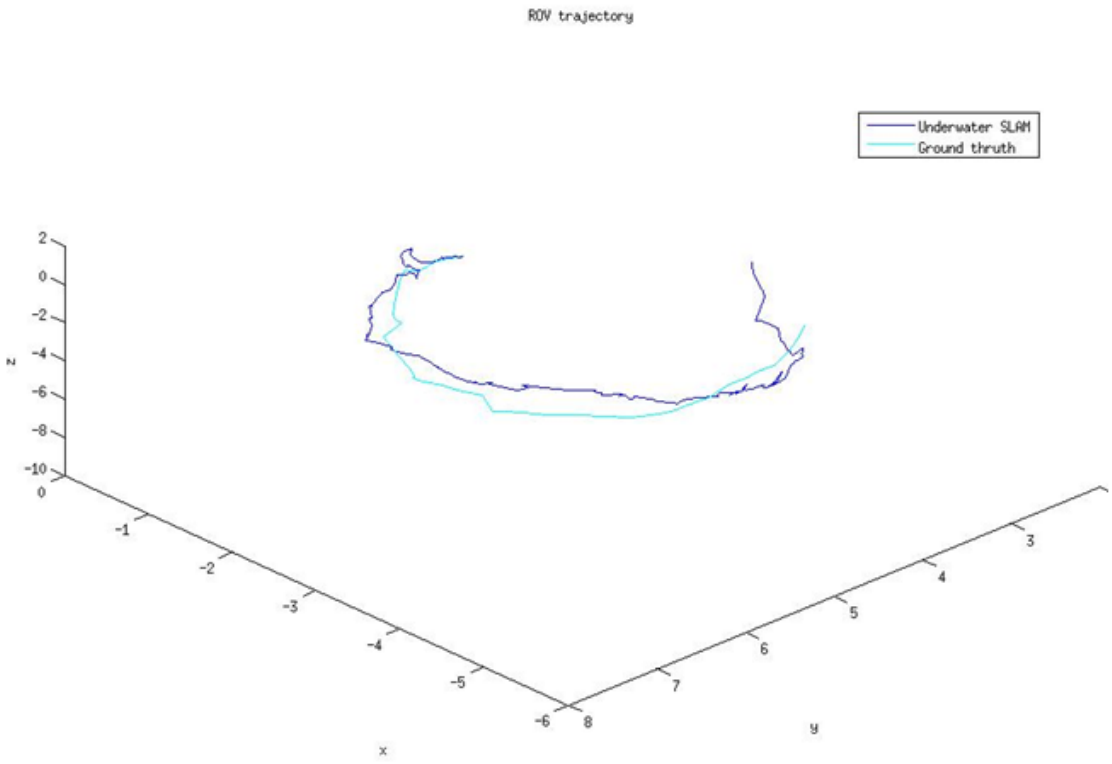


Figura 7.10: Trajetória do VideoRay ROV validada pelo *ground truth*.

7.4 Outros Cenários de Aplicação

O algoritmo proposto aborda a aplicação SLAM em ambientes não estruturados, com o intuito de validar a aplicabilidade para cenários reais foram realizados um conjunto de testes. Os gasodutos e oleodutos de empresas petrolíferas, instalados a grandes profundidades, são um dos cenários onde veículos subaquáticos como ROVs ou AUVs são muito utilizados nomeadamente em tarefas de inspeção [5]. A solução proposta foi testada e obtidos resultados positivos na deteção de *landmarks*, conforme ilustado na Figura 7.4. Posteriormente, através do MATLAB foi obtida a trajetória realizada pelo veículo na inspeção de gasodutos, observar Figura 7.11.

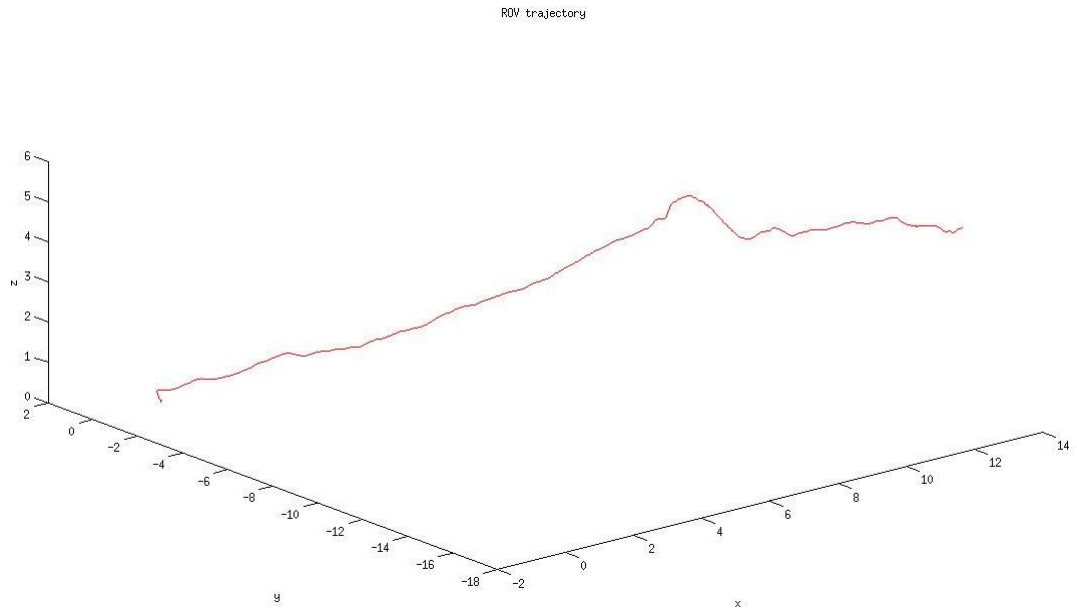


Figura 7.11: Trajetória obtida num gasoduto.

7.4.0.1 Análise

A inovação permite avaliar a qualidade das medidas adquiridas ao longo do processo experimental. Os resultados obtidos para a inovação das medidas do mapa estão representados no gráfico da Figura 7.12, a inovação da posição e respetiva orientação do robô subaquático são apresentados na Figura 7.13. Na inovação média do primeiro gráfico e nas inovações do segundo gráfico verifica-se uma elevada incerteza da medida inicial que tende para valores cada vez mais baixos. Este facto ocorre uma vez que o filtro é inicializado com um valor de incerteza considerável.

Para a avaliação do desempenho das *landmark* foi realizada uma análise baseada no estudo do tempo de permanência do marcador. Na Figura 7.14 pode ser observada uma parcela das *landmarks* obtidas de um *log* adquirido. O gráfico apresenta resultados bastantes satisfatórios para o tempo de permanência das *landmarks*, linhas e pontos, no processo de mapeamento.

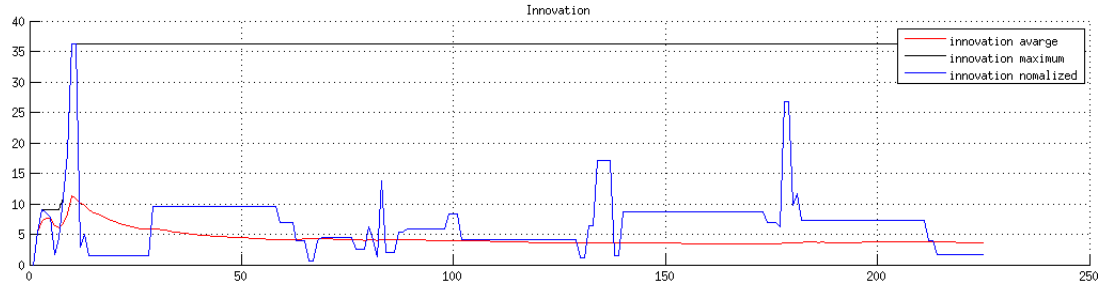


Figura 7.12: Inovação das medidas do mapa.

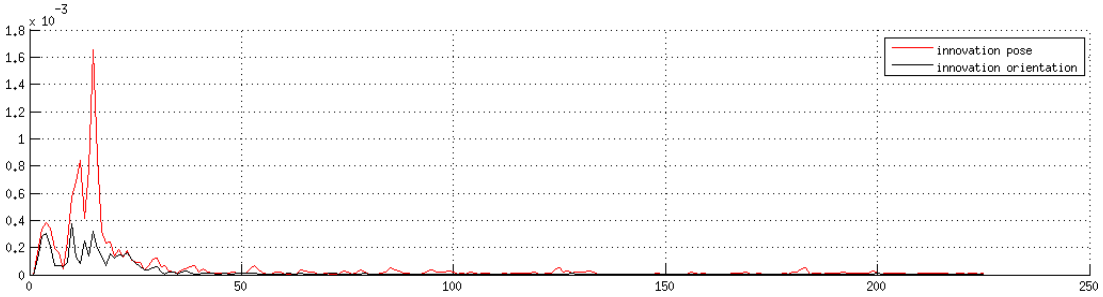


Figura 7.13: Inovação das medidas da posição e orientação do ROV.

A aplicação de algoritmos complexos exige elevados custos computacionais, assim uma avaliação do impacto de cada método de deteção de *features* foi realizada. A análise foi efetuada num para as três modalidades possíveis, para utilização de linhas, pontos e para a aplicação de linhas e pontos em simultâneo. Através da observação dos resultados obtidos (Figura 7.15) conclui-se que existe um relação entre o número de observações e o tempo de processamento (em ms), o aumento do número de observações provoca um aumento do tempo de processamento. Outro dado relevante corresponde ao contributo no peso computacional das linhas no tempo de processamento por cada *frame*. A observação dos gráficos da Figura 7.15 também permite concluir que para uma quantidade próxima de 20 observações o tempo de processamento não excede os 40ms num Intel Penium dual core.

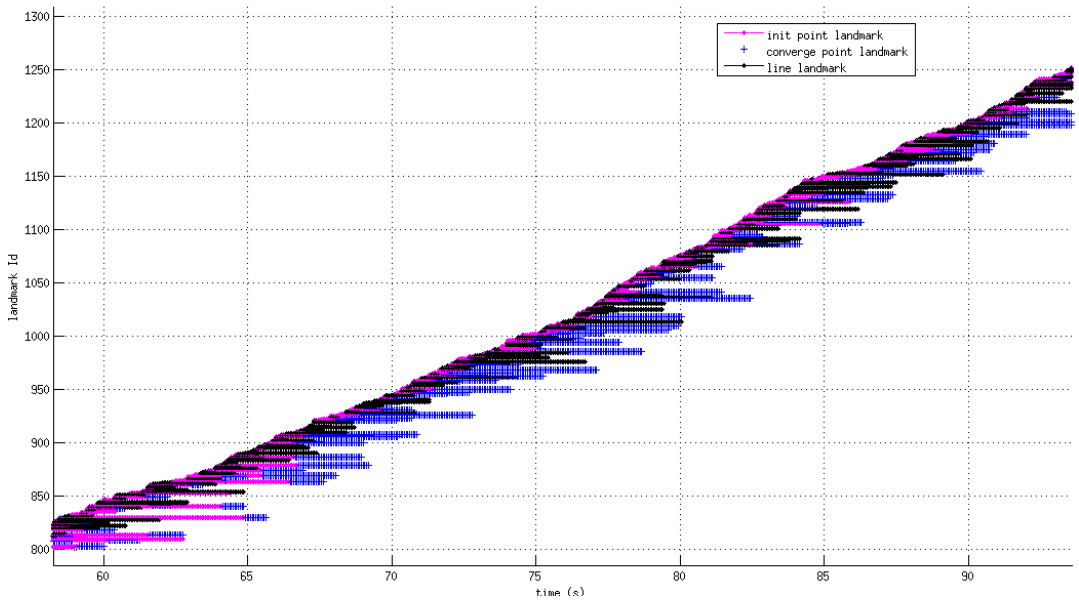


Figura 7.14: Análise das *landmark* (evolução temporal).

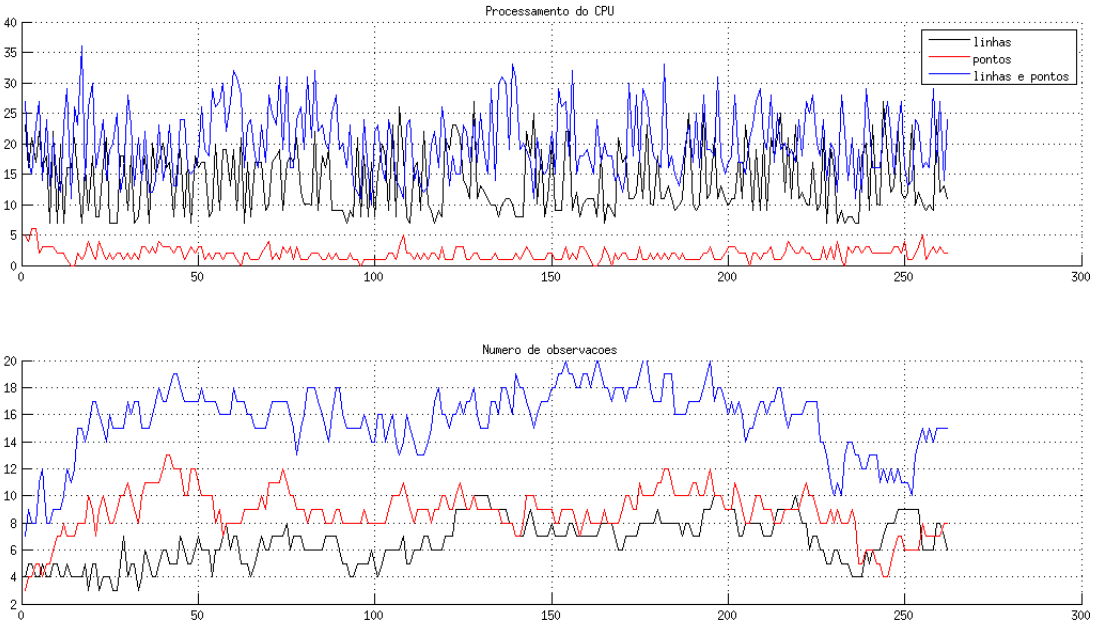


Figura 7.15: Tempo de processamento versus quantidade de observações.

CAPÍTULO 8

Conclusões e Trabalho Futuro

8.1 Discussão de Resultados

Na presente dissertação foi apresentado um algoritmo para a aplicação de SLAM em tempo real através da extração de *features* visuais em ambientes subaquáticos. A deteção de *features* foi baseada em algoritmos de visão artificial na extração quer de pontos de interesse, quer de linhas. O sistema apresentado foi objeto de uma recente publicação [4], e apresentado na conferência internacional OCEANS MTS/IEEE 2014.

Este sistema é caracterizado num ambiente de campo controlado e validado através de um sistema de perceção visual externo, o INESC TEC/ISEP *indoor ground truth*. A trajetória descrita pelo ROV é obtida utilizando *features* visuais do ambiente, como linhas e pontos extraídas de estruturas subaquáticas. O sistema de navegação robótico produz bons resultados com o VideoRay Pro 3E ROV até 30 fps, devidamente validados pelo sistema de validação externo referido anteriormente.

A proposta SLAM foi desenvolvida para a utilização em cenários de aplicações correspondentes a ambientes não estruturados. O positivo desempenho do algoritmo é demonstrado através dos resultados em ambientes subaquáticos típicos na aplicação de tarefas de inspeção robóticas. As *landmarks* são detetadas e uma trajetória coerente vai sendo realizada através de uma estimação da posição relativa às respectivas observações adquiridas.

Um conjunto de ferramentas MATLAB foram desenvolvidas para análise do *performance* do algoritmo proposto. Os dados adquiridos para a inovação permitiram avaliar o bom desempenho do filtro EKF integrado no processo de mapeamento. De forma a avaliar o desempenho das *landmark* uma análise foi realizada e obtidos resultados satisfatórios. O processamento computacional da abordagem proposta foi igualmente analisado e apresentados resultados do desempenho em tempo real.

8.2 Trabalho Futuro

Em aplicações SLAM são várias as expectativas para o trabalho futuro. Este algoritmo pode ser adaptado para a fusão de dados obtidos de uma câmara com os dados de um sensor inercial, o que permite estimar o parâmetro de escala desconhecida no *framework* monocular SLAM. Desta forma, é possível melhorar o performance da aplicação já que em vez de um modelo de velocidade constante seria utilizado um modelo onde as velocidades são adquiridas dos dados do sensor inercial.

A abordagem SLAM apresentada também poderá vir a ser beneficiada com a integração de outros métodos de deteção de *features*. Este algoritmo pode ser adaptado para integrar tipos adicionais de *features*, tais como pontos de interesse SIFT.

Outros contributos podem ser realizados em aplicações SLAM multi robóticas, ou seja o mapeamento e localização de dois robôs no mesmo ambiente. Assim dois robôs podem realizar tarefas de inspeção subaquática em simultâneo sem que sejam sobrepostas as respectivas trajetórias. Finalmente, podem ser desenvolvida aplicações de sub mapeamento para resolver problemas de constrangimentos em tempo real. Portanto, utilizar métodos que permitam a diminuição do processamento computacional.

Bibliografia

[1] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[2] Alfredo Martins, André Dias, Hugo Silva, José Almeida, Pedro Gonçalves, Flávio Lopes, André Faria, Joao Ribeiro, and Eduardo Silva. Groundtruth system for underwater benchmarking.

[3] Cyril Roussillon, Aurélien Gonzalez, Joan Solà, Jean-Marie Codol, Nicolas Mansard, Simon Lacroix, and Michel Devy. Rt-slam: a generic and real-time visual slam implementation. In *Computer Vision Systems*, pages 31–40. Springer, 2011.

[4] Magda Meireles et al. Real time visual slam for underwater robottic inspection. In *Oceans, 2014*. IEEE, St. Jonhs,Canda, 2014.

[5] Aleksandar Lazinica. *Mobile robots: towards new applications*. Pro-Literatur-Verlag, 2006.

[6] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.

[7] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[8] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[9] Hugh Durrant-Whyte, David Rye, and Eduardo Nebot. Localization of autonomous guided vehicles. In *Robotics Research*, pages 613–625. Springer, 1996.

[10] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI*, pages 593–598, 2002.

[11] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[12] Jose E Guivant and Eduardo Mario Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *Robotics and Automation, IEEE Transactions on*, 17(3):242–257, 2001.

[13] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2432–2437. IEEE, 2005.

[14] Maria J Costa, Pedro Goncalves, Alfredo Martins, and Eduardo Silva. Vision-based assisted teleoperation for inspection tasks with a small rov. In *Oceans, 2012*, pages 1–8. IEEE, 2012.

[15] David Ribas, Pere Ridao, Jose Neira, and Juan D Tardos. Slam using an imaging sonar for partially structured underwater environments. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5040–5045. IEEE, 2006.

[16] David Ribas, Pere Ridao, Juan D Tardos, and José Neira. Underwater slam in a marina environment. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1455–1460. IEEE, 2007.

[17] John J Leonard and Hugh F Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems' 91.'In-*

telligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on, pages 1442–1447. Ieee, 1991.

[18] John J Leonard and Hugh F Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, 7(3):376–382, 1991.

[19] Jose A Castellanos, JM Martinez, Jose Neira, and Juan D Tardos. Simultaneous map building and localization for mobile robots: A multisensor fusion approach. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1244–1249. IEEE, 1998.

[20] Stefan Kohlbrecher, Johannes Meyer, Thorsten Graber, Karen Petersen, Oskar von Stryk, Uwe Klingauf, K Petersen, A Kleiner, O von Stryk, S Kohlbrecher, et al. Hector open source modules for autonomous mapping and navigation with rescue robots. In *Proceedings of 17th RoboCup international symposium*, 2013.

[21] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pages 155–160. IEEE, 2011.

[22] David Ribas, Pere Ridao, Juan Domingo Tardós, and José Neira. Underwater slam in man-made structured environments. *Journal of Field Robotics*, 25(11-12):898–921, 2008.

[23] John Illingworth and Josef Kittler. A survey of the hough transform. *Computer vision, graphics, and image processing*, 44(1):87–116, 1988.

[24] Josep Aulinas, Marc Carreras, Xavier Llado, Joaquim Salvi, Rafael Garcia, Ricard Prados, and Yvan R Petillot. Feature extraction for underwater visual slam. In *OCEANS, 2011 IEEE-Spain*, pages 1–7. IEEE, 2011.

[25] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[26] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[27] Ayoung Kim and Ryan Eustice. Pose-graph visual slam with geometric model selection for autonomous underwater ship hull inspection. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1559–1565. IEEE, 2009.

[28] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[29] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[30] Ayoung Kim and Ryan M Eustice. Real-time visual slam for autonomous underwater hull inspection using visual saliency. In *Conference on Intelligent Robots and Systems*, volume 1, page 2, 2013.

[31] Fausto Ferreira, Gianmarco Veruggio, Massimo Caccia, and Gabriele Bruzzone. Real-time optical slam-based mosaicking for unmanned underwater vehicles. *Intelligent Service Robotics*, 5(1):55–71, 2012.

[32] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[33] D. C. Brown. Decentering distortion of lenses. pages 1052–1067, 1996.

[34] Richard L Pio. Euler angle transformations. *Automatic Control, IEEE Transactions on*, 11(4):707–715, 1966.

[35] Thor I Fossen. *Guidance and control of ocean vehicles*, volume 199. Wiley New York, 1994.

[36] Michael Montemerlo and Sebastian Thrun. Fastslam 2.0. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*, pages 63–90, 2007.

[37] Cyril Roussillon. Module rtslam. http://homepages.laas.fr/croussil/doc/jafar/group__rtslam.html. [consultado em junho de 2014].

[38] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[39] Javier Civera, Oscar G Grasa, Andrew J Davison, and JMM Montiel. 1-point ransac for ekf-based structure from motion. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3498–3504. IEEE, 2009.

[40] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[41] Luigi Di Stefano, Stefano Mattoccia, and Federico Tombari. Zncc-based template matching using bounded partial correlation. *Pattern recognition letters*, 26(14):2129–2134, 2005.

[42] ROS. Tutorials - creating package. http://homepages.laas.fr/croussil/doc/jafar/group__rtslam.html. [consultado em Abril de 2014].

[43] ROS. Gscam. http://wiki.ros.org/gscam. [consultado em Março de 2014].

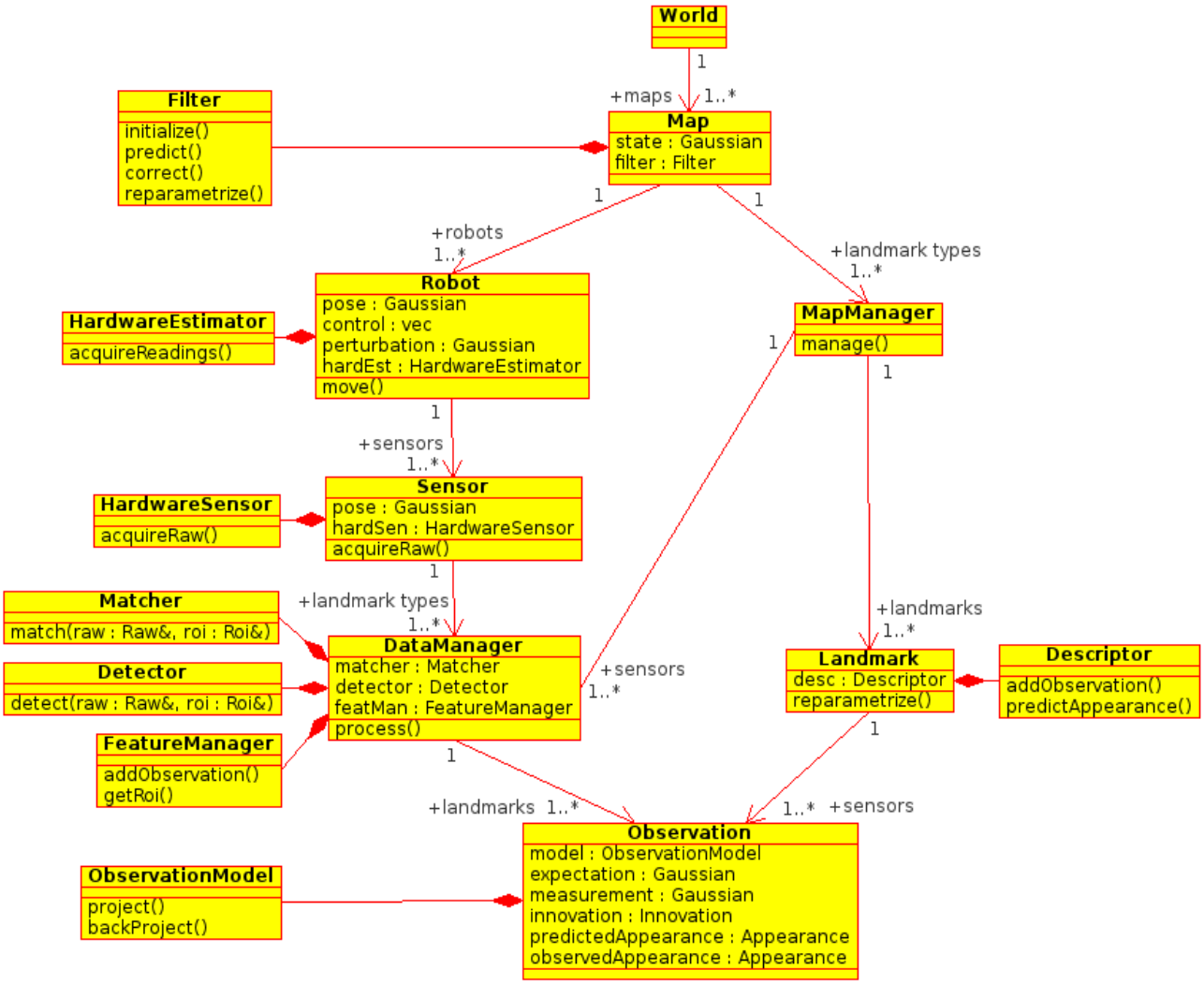
[44] ROS. Camera calibration. http://wiki.ros.org/gscam. [consultado em Abril de 2014].

[45] Cyrille Berger and Simon Lacroix. Dseg: Détection directe de segments dans une image.

[46] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

APÊNDICE A

Family tree


World
1
+maps
1..*
Map
state : Gaussian
filter : Filter
Filter
initialize()
predict()
correct()
reparametrize()
1
1
+robots
1..*
+landmark types
1..*
Robot
pose : Gaussian
control : vec
perturbation : Gaussian
hardEst : HardwareEstimator
move()
HardwareEstimator
acquireReadings()
MapManager
manage()
1
1
1
+sensors
1..*
Sensor
pose : Gaussian
hardSen : HardwareSensor
acquireRaw()
HardwareSensor
acquireRaw()
1
Matcher
match(raw : Raw&, roi : Roi&)
+landmark types
1..*
DataManager
matcher : Matcher
detector : Detector
featMan : FeatureManager
process()
+sensors
1..*
+landmarks
1..*
Landmark
desc : Descriptor
reparametrize()
Descriptor
addObservation()
predictAppearance()
Detector
detect(raw : Raw&, roi : Roi&)
FeatureManager
addObservation()
getRoi()
1
1
+landmarks 1..*
1..* +sensors
Observation
model : ObservationModel
expectation : Gaussian
measurement : Gaussian
innovation : Innovation
predictedAppearance : Appearance
observedAppearance : Appearance
ObservationModel
project()
backProject()