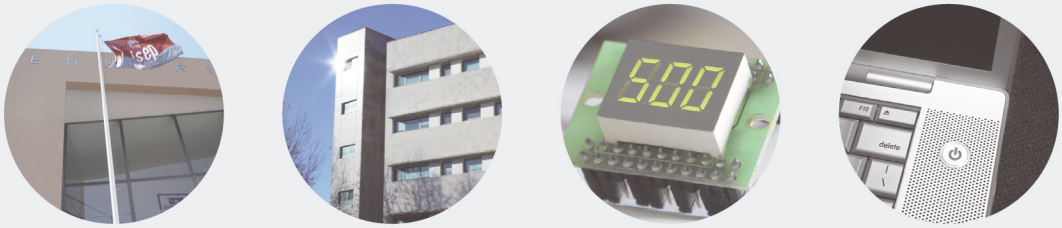
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Sistema de reconstrução 3D com LiDAR e câmara de espetro visível para veículo autónomo aéreo

PEDRO ANDRÉ PEIXOTO FONSECA DE CASTRO E SILVA
março de 2020

POLITÉCNICO
DO PORTO

Sistema de reconstrução 3D com LiDAR e câmara de espetro visível para veículo autónomo aéreo

Pedro André Peixoto Fonseca de Castro e Silva
Nº 1140607

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Sistemas Autónomos
Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

2020

Dissertação, para satisfação parcial dos requisitos do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: Pedro André Peixoto Fonseca de Castro e Silva

Nº 1140607

Orientador: André Miguel Pinheiro Dias

Co-Orientador: Alfredo Manuel Oliveira Martins

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização de Sistemas Autónomos

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

5 de Março de 2020

Agradecimentos

Em primeiro lugar, queria agradecer ao meu orientador, Eng. André Dias, pela oportunidade e desafio proporcionado, bem como pela ajuda disponibilizada ao longo do desenvolvimento desta dissertação.

Gostaria de agradecer ao Eng. Alfredo Martins pelo apoio e acompanhamento disponibilizado ao longo do mestrado.

À família do Laboratório de Sistemas Autónomos (LSA) com quem tenho aprendido bastante nos últimos tempos.

Aos meus colegas de trabalho Paulo Rodrigues, Tiago Miranda e Alexandre Amaral e Ana Pires pelo suporte no desenvolvimento deste projeto.

Um agradecimento especial à Ana Pires e Sara Freitas pelo apoio, incentivo, motivação e ajuda durante a realização da dissertação.

A todos os meus amigos de longa data.

Por fim, um agradecimento à minha família, em especial à minha mãe e avós, pela educação que me proporcionaram, pelo esforço, apoio, incentivo e condições para que pudesse chegar até aqui.

Muito obrigado.

Pedro André Silva

Esta página foi intencionalmente deixada em branco.

This thesis was developed within the scope of MineHeritage Project (EIT/RAW MATERIALS/SGA2019/1) funded by EIT Raw Materials [Nr. 18111; Main Theme 1 Exploration and raw materials resources assessment; Area D3 RM Academy; Segment D3.1 Wider Society Learning].

Esta página foi intencionalmente deixada em branco.

Resumo

O património cultural é uma das pontes mais importantes entre o passado e o futuro, sendo indispensável para o desenvolvimento individual e social do ser humano. A sua preservação é atualmente uma das questões mais importantes para a sociedade, levando a que fossem utilizadas novas tecnologias para recolher dados do património cultural. Hoje em dia, uma das novas tecnologias usadas são os *Unmanned Aerial Vehicle*s (UAVs) devido ao seu baixo custo, flexibilidade, elevada precisão e capacidade de aquisição de dados, através de diferentes sensores.

O projeto MineHeritage surge da necessidade da preservação do património cultural. Nesta dissertação propõe-se endereçar a área da reconstrução *Three-Dimensional* (3D) com UAVs, combinando a informação proveniente de uma câmara de espetro visível com um *Light Detection and Ranging* (LiDAR).

Como em qualquer sistema, o processo de calibração é uma das componentes mais importantes para o bom funcionamento do mesmo. Deste modo, foi implementado um algoritmo de calibração dos parâmetros extrínsecos entre o LiDAR e a câmara, permitindo combinar a informação de ambos os sensores. Além disso, foi também desenvolvido um método de reconstrução 3D, na qual é realizada a associação da cor da nuvem de pontos proveniente do LiDAR, através da informação de cor adquirida pela câmara de espetro visível.

Este sistema foi testado em dois *datasets* realizados no Mosteiro de Tibães, Braga, Portugal, no âmbito do projeto MineHeritage. Os resultados experimentais de calibração e de reconstrução 3D demonstram que o sistema desenvolvido apresenta resultados consistentes, é eficaz e capaz de ser implementado em tempo real.

Palavras-Chave: Reconstrução 3D, LiDAR, câmara, calibração LiDAR-câmara, UAV

Esta página foi intencionalmente deixada em branco.

Abstract

Cultural heritage is one of the most important bridges between the past and the future, being indispensable for the individual and social development of human beings. Preservation is currently one of the most important issues for society, leading to the use of new technologies to collect cultural heritage data. Nowadays, one of the new technologies used are UAVs due to their low-cost, flexibility, high precision and data acquisition capacity.

MineHeritage project arises from the need to preserve cultural heritage. This thesis addresses the 3D reconstruction area using UAVs, combining the information from a visible spectrum camera with a LiDAR.

As in every system, the calibration process is one of the most important components for its proper functioning. A calibration algorithm for the extrinsic parameters between LiDAR and camera was implemented, allowing to combine the information from both sensors. In addition, a 3D reconstruction algorithm was also developed, in which the color association of the point cloud from LiDAR is carried out through the color information acquired by the visible spectrum camera.

This system was tested in two datasets held at the Monastery of Tibães, Braga, Portugal, within the scope of MineHeritage project. The experimental results of calibration and 3D reconstruction demonstrate that the developed system presents consistent results, is effective and capable of being implemented in real-time.

Keywords: 3D reconstruction, LiDAR, camera, LiDAR-camera calibration, UAV

Esta página foi intencionalmente deixada em branco.

Conteúdo

Esta página foi intencionalmente deixada em branco.

Lista de Figuras

Esta página foi intencionalmente deixada em branco.

Lista de Tabelas

Esta página foi intencionalmente deixada em branco.

Lista de Algoritmos

Esta página foi intencionalmente deixada em branco.

Lista de Siglas e Acrónimos

2D *Two-Dimensional*

3D *Three-Dimensional*

CCD *Charge-Coupled Device*

CMOS *Complementary Metal-Oxide Semiconductor*

CPU *Central Processing Unit*

ENU *East-North-Up*

FoV *Field of View*

GigE *Gigabit Ethernet*

GNSS *Global Navigation Satellite System*

GPS *Global Positioning System*

ICARSC *International Conference on Autonomous Robot Systems and Competitions*

IMU *Inertial Measurement Unit*

INESC TEC Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência

IR Infrared Radiation

ISEP Instituto Superior de Engenharia do Porto

LAN *Local Area Network*

LiDAR *Light Detection and Ranging*

LM *Levenberg-Marquardt*

LRF *Laser Range Finder*

LSA Laboratório de Sistemas Autónomos

LTP *Local Tangent Plane*

MATLAB *MATrix LABoratory*

MEMS *MicroElectroMechanical System*

NMEA *National Marine Electronics Association*

OpenCV *Open Source Computer Vision Library*

P2P *Peer-To-Peer*

PCL *Point Cloud Library*

PnP *Perspective-n-Point*

PPS *Pulse Per Second*

RANSAC *Random Sample Consensus*

RGB *Red Green Blue*

ROS *Robot Operating System*

RTK *Real-Time Kinematic*

SLAM *Simultaneous Localization And Mapping*

SVD *Singular Value Decomposition*

TCP/IP *Transmission Control Protocol/Internet Protocol*

ToF *Time-of-Fligh*

UAV *Unmanned Aerial Vehicle*

UNEXMIN *Underwater Explorer for Flooded Mines*

UTC *Coordinated Universal Time*

VAMOS *Viable Alternative Mine Operating System*

Capítulo 1

Introdução

Ao longo da última década, os UAVs têm merecido uma atenção elevada por parte da comunidade científica, levando a um crescimento significativo do ponto de vista tecnológico e cientifico [1]. Os UAVs foram inicialmente concebidos para aplicações militares [2]. No entanto, estão a tornar-se cada vez mais importantes para os humanos, de modo a ajudarem na realização de algumas tarefas mais complexas, repetitivas e que colocam a vida humana em perigo. Estas tarefas vão desde operações de busca e salvamento, vigilância e monitorização do ambiente, inspeção de infraestruturas, agricultura, reconstrução 3D, entre outras [3].

Esta dissertação procura endereçar a área da reconstrução 3D com UAVs, através da combinação da informação de uma câmara de espetro visível com um LiDAR. Esta enquadra-se no âmbito do projeto europeu MineHeritage, onde se pretende efetuar a reconstrução 3D de locais históricos.

O LiDAR fornece a informação da estrutura 3D do ambiente, na forma de uma nuvem de pontos com base nas distâncias medidas pelos feixes de luz. As principais vantagens do LiDAR são a elevada precisão, mesmo a distâncias elevadas, e a invariância às condições de iluminação, sendo usado em qualquer condição luminosa [4]. Contudo, apesar das grandes vantagens deste sensor, este não fornece informação visual (cor e textura) do ambiente [5].

Por outro lado, as câmaras de espectro visível capturam a cor, a textura e a aparência do ambiente [6]. Combinando as funcionalidades do LiDAR com o sistema de visão permitirá a produção de uma nuvem de pontos 3D, onde cada feixe *laser* do LiDAR poderá ser preenchido com o valor da cor *Red Green Blue* (RGB) adquirida pela câmara de espetro visível.

O desafio principal na combinação destes dois sensores é a calibração precisa dos

parâmetros extrínsecos, rotação e translação, entre os dois sensores. Dados os parâmetros extrínsecos, é possível transformar a nuvem de pontos do LiDAR no referencial da câmara e projetar estes pontos na imagem.

Nesta dissertação propõe-se o desenvolvimento de uma solução de estimação automática dos parâmetros extrínsecos entre o LiDAR e uma câmara de espetro visível e, com base nestes parâmetros, efetuar a reconstrução 3D combinando LiDAR com imagem, utilizando um UAV.

1.1 Enquadramento e motivação

O LSA e o Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência (INESC TEC) têm estado involvidos em vários projetos ao longo dos últimos anos. Estes projetos estão enquadrados em diferentes áreas do campo da robótica, como por exemplo: robótica aérea e robótica marinha (veículos subaquáticos e de superfície), como demonstrado na figura 1.1.

Um dos projetos mais recentes é o projeto *Underwater Explorer for Flooded Mines* (UNEXMIN) que tem como objetivo desenvolver um novo sistema robótico para a exploração autônoma e mapeamento das minas inundadas na Europa. *Viable Alternative Mine Operating System* (VAMOS) foi um projeto europeu financiado pelo programa de pesquisa e inovação Horizonte 2020 da União Europeia, criado para desenvolver novas tecnologias para a mineração automática e mineração de pequenos depósitos. Foram desenvolvidos dois robôs, na qual o veículo de menor dimensão realizava o mapeamento 3D (figura 1.1(c)) e o de maior cortava a rocha e bombeava a rocha moída para a superfície.

Esta dissertação está inserida no projeto MineHeritage (EIT/RAW MATERIALS/ SGA2019/1). Este projeto tem como objetivo, através do património cultural, da mineração e das matérias-primas, a criação de ferramentas educacionais com a finalidade de divulgar a importância das matérias-primas para a sociedade.

O INESC TEC é responsável pela aquisição e processamento dos dados. Com este propósito, é utilizado um UAV que será responsável pela obtenção de imagens de alta resolução e de mapear a área em estudo no exterior, um laser 3D estático de alta resolução e, de modo a ser possível mapear o interior de uma mina, foi desenvolvido um sistema robótico que contém vários sensores, mais concretamente, câmaras, LiDAR *Two-Dimensional* (2D), *Inertial Measurement Unit* (IMU), entre outros.

No final do projeto será desenvolvido um jogo e uma aplicação educacional, na qual os vários locais poderão ser visitados e explorados de forma virtual.

(a) STORK

(b) Roaz II

(c) EVA

Figura 1.1: Alguns dos robôs desenvolvidos no LSA.

1.2 Cenários de aplicação

A principal aplicação do algoritmo proposto é a reconstrução 3D de terrenos e áreas inacessíveis, utilizando um UAV. O algoritmo proposto opera em tempo real e pode ser usado em diferentes aplicações. Algumas possibilidades são:

- Inspeção de infraestruturas;
- Construção - monitorização do estado da obra;
- Agricultura - ajuda o planeamento e gestão de fazendas agrícolas;
- Mineração - morfologia da área onde se pretende efetuar a extração do minério;
- Arqueologia - ajuda o arqueólogo a entender a superfície.

1.3 Objetivos

Nesta dissertação é abordada a combinação sensorial entre um LiDAR e uma câmara, com a finalidade de obter uma nuvem de pontos 3D colorida. Para realizar esta tarefa, existem vários objetivos que precisam de ser cumpridos, nomeadamente:

- Estudo dos trabalhos existentes relacionados com a calibração dos parâmetros extrínsecos e reconstrução 3D combinando LiDAR e câmara, de forma a facilitar o desenvolvimento e implementação de um algoritmo de calibração e de reconstrução 3D;
- Implementação de um método de calibração dos parâmetros extrínsecos, para que recorrendo a estes parâmetros seja possível transformar os pontos do LiDAR no referencial da câmara;
- Desenvolvimento de um algoritmo de combinação entre LiDAR e câmara, para ser executado em tempo real, resultando na reconstrução 3D do ambiente que o UAV mapeou;
- Caracterização do erro de calibração da posição e orientação entre LiDAR e câmara;
- Validação do método desenvolvido em diferentes cenários de aplicação, sendo um deles os locais históricos a mapear no âmbito do projeto MineHeritage.

1.4 Estrutura

Esta dissertação encontra-se organizada em oito capítulos. No próximo capítulo é apresentado um estudo preliminar dos trabalhos relacionados com o tópico da dissertação.

No capítulo três é efetuada uma exposição de alguns conceitos fundamentais para a compreensão dos temas desenvolvidos ao longo da dissertação. O capítulo quatro contém uma visão geral da arquitetura de *hardware* e de *software*.

A descrição do algoritmo proposto é feita ao longo do capítulo cinco. Este está dividido em três partes, nomeadamente, calibração dos parâmetros extrínsecos, colorização da nuvem de pontos e, por último, reconstrução 3D.

O capítulo seis lista as especificações do UAV usado e dos sensores utilizados, câmara e LiDAR. Neste capítulo também é realizada uma descrição do *software* desenvolvido de modo a implementar o algoritmo descrito no capítulo anterior.

Os resultados são expostos e analisados no capítulo sete. De modo a testar o algoritmo desenvolvido foram realizados dois *datasets*.

Por último, no capítulo oito são feitas as considerações finais e uma análise do trabalho desenvolvido. Este capítulo termina com algumas sugestões para trabalhos futuros.

Capítulo 2

Estado da Arte

Neste capítulo é exposto o estado da arte que tem como objetivo a apresentação de estudos relevantes para a temática desta dissertação. Este capítulo encontra-se dividido em três partes. Na primeira parte foi recolhida informação sobre métodos de calibração dos parâmetros extrínsecos entre câmara e LiDAR. A segunda parte expõe vários trabalhos relacionados com a combinação entre câmara e LiDAR. Por último, são apresentadas várias metodologias para calibrar a cor de uma ou várias câmaras.

2.1 Métodos de calibração de extrínsecos entre câmara e LiDAR

Zhang e Pless [7] desenvolveram um método para a calibração de extrínsecos entre uma câmara e um *Laser Range Finder* (LRF) 2D, isto é, a rotação e a translação entre os dois sensores, assumindo que os parâmetros intrínsecos da câmara eram conhecidos. Este método faz uso de um padrão de calibração plano (*checkerboard*) que é visualizado simultaneamente pela câmara e pelo LRF. Para cada *pose*, são guardados os *scans* do laser, assim como as imagens obtidas pela câmara. É de realçar que os pontos do laser são invisíveis para a câmara. O esquema de calibração pode ser visualizado na figura 2.1.

O procedimento de calibração segue os seguintes passos:

- Colocação do *checkerboard* numa posição visível a ambos os sensores em diferentes orientações;
- Para cada *pose*, realizar a extração dos pontos laser e a deteção dos *corners* do alvo na imagem;

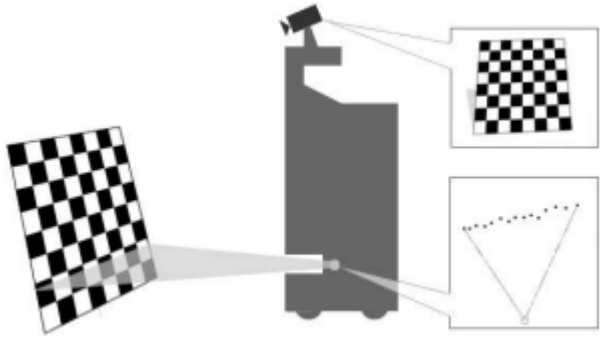


Figura 2.1: Esquema de calibração utilizando uma câmara e um LRF [7].

- Estimar a rotação e a translação da câmara em relação ao padrão de calibração;
- Estimação da posição e orientação relativa da câmara relativamente ao LRF;
- Otimização dos parâmetros extrínsecos obtidos.

Além de estimar a *pose* entre a câmara e o laser, este algoritmo também é capaz de estimar os parâmetros intrínsecos da câmara. Para tal, realiza uma otimização global, tendo como parâmetros iniciais a rotação e a translação estimada entre os dois sensores, bem como os parâmetros intrínsecos estimados.

Os autores apresentam resultados aceitáveis, obtendo melhorias na ordem dos 30 % para os parâmetros extrínsecos. Por outro lado, quando se realiza a otimização global, os parâmetros intrínsecos da câmara são melhorados em cerca de 30 %.

Unnikrishnan et al. [8] usam um *checkerboard*, utilizado frequentemente para estimar os parâmetros intrínsecos de uma câmara, de modo a calibrar um LiDAR 3D e uma câmara. O alvo é colocado em várias posições com diferentes orientações e em cada *pose* são estimados os parâmetros do plano no referencial de ambos os sensores. Os parâmetros no referencial do LiDAR são obtidos a partir de um estimador de *least squares* que ajusta os pontos 3D que formam o plano, enquanto que os parâmetros no referencial da câmara são obtidos através da estimação da *pose* em relação ao alvo.

De modo a obter os parâmetros extrínsecos entre o LiDAR e a câmara, é estimada a transformação que minimiza a diferença nas observações de cada plano. Para tal, este processo é dividido em duas etapas. Na primeira etapa, é obtida uma estimação inicial da translação e da rotação. A translação é obtida minimizando a distância entre os planos, enquanto que a rotação é obtida minimizando a diferença entre as normais em

cada referencial. Na segunda fase, a rotação e a translação são otimizadas, minimizando a distância entre os pontos 3D em relação ao plano correspondente observado na imagem.

O método proposto por Pandey et at. [9] é semelhante ao [7]. Os autores utilizaram uma câmara omnidirecional, sendo esta composta por 6 câmaras. A transformação de todas as câmaras e o referencial comum, denominado de *camera head*, é conhecida. De modo a restringir completamente o problema de otimização, o número mínimo de *poses* são três.

Os autores concluíram que o erro de estimação dos parâmetros extrínsecos decresce com o aumento do número de *poses* do alvo, dado que aumenta o número de restrições na otimização. Além disso, o aumento da área do alvo diminuí o erro, já que existem mais pontos na superfície do alvo.

Pusztai et al. [4] utilizaram caixas comuns, visto que estas apresentam margens perpendiculares entre si que podem ser facilmente detetadas por um LiDAR de baixa resolução. O *setup* experimental é ilustrado na figura 2.2.

Figura 2.2: Setup experimental utilizando caixas comuns [4].

O método proposto não utiliza a informação de intensidade fornecida pelo LiDAR e apresenta dois requisitos. O primeiro requisito é que três lados da caixa de calibração precisam de ser visíveis tanto na nuvem de pontos como na imagem e o segundo é que o tamanho da caixa deverá ser conhecido.

A ideia principal deste algoritmo é que se sete cantos (ao longo das interseções de três planos) da caixa de calibração são detetados no referencial do LiDAR e as projeções des-

ses cantos também são conhecidas na imagem, a *pose* é obtida pelo problema *Perspective-n-Point* (PnP). Este método, além de permitir a calibração dos parâmetros extrínsecos entre uma câmara e um LiDAR, proporciona a obtenção da *pose* entre dois LiDARs.

Dhall et al. [10] desenvolveram um novo método para descobrir a transformação rígida (rotação e translação) entre um LiDAR e uma câmara. Para tal, realizaram correspondências de pontos 3D-3D no referencial do LiDAR e da câmara. Estes autores utilizaram dois alvos de calibração planos com uma forma retangular, na qual colaram marcadores de ArUco, conforme evidenciado na figura 2.3.

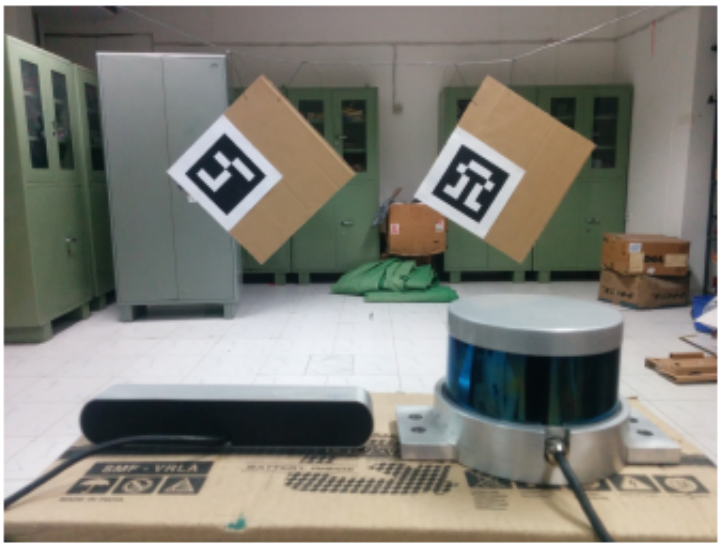


Figura 2.3: Setup experimental com marcadores ArUco e cartões rectangulares [10].

Os marcadores de ArUco permitem obter a rotação e a translação entre a câmara e o alvo. Esta transformação permite obter os cantos (pontos 3D) do alvo no referencial da câmara. Estes pontos 3D no referencial do LiDAR são obtidos pela interseção das retas que formam o alvo rectangular. Os parâmetros das retas são obtidos através dos pontos que se encontram na borda do alvo, sendo aplicado o *Random Sample Consensus* (RANSAC) com o modelo de uma linha. Assim que as correspondências 3D-3D são atingidas, é utilizado o algoritmo de Kabsch, de modo a obter a transformação entre os dois sensores.

Zhou et al. [6] desenvolveram um algoritmo de calibração dos parâmetros extrínsecos, usando correspondências de linhas e planos. O alvo utilizado é um *checkerboard* e os 4 contornos permitem calcular os parâmetros das linhas em ambos os referenciais. Por outro lado, a superfície plana do alvo permite obter os parâmetros do plano no referencial

da câmara e do LiDAR. Este método reduz o número de *poses* necessárias para uma. O algoritmo pode ser evidenciado nos seguintes passos:

- Deteção do *checkerboard* e das linhas na imagem;
- Cálculo dos parâmetros do plano e das linhas no referencial da câmara;
- Especificar uma posição aproximada da localização do alvo no referencial do LiDAR;
- Deteção do *target* através de RANSAC;
- Obtenção dos parâmetros do plano e das linhas no LiDAR;
- Estimar a rotação e a translação inicial;
- Otimização dos parâmetros obtidos.

Os autores compararam o seu método com o desenvolvido em [8], obtendo resultados mais precisos. O resultado deste método utilizando apenas uma *pose* é semelhante ao [8] quando este usa 6 *poses*. As restrições das linhas 3D aumentam significativamente a diversidade das medições, levando a que este algoritmo obtenha melhores resultados com um número inferior de *poses*.

2.2 Combinação LiDAR-câmara

Zhou [11] desenvolveu um algoritmo para combinar um LiDAR 3D e uma câmara, sendo este dividido em três etapas. Na primeira etapa, os pontos 3D do laser são convertidos no referêncial da câmara através dos parâmetros extrínsecos entre o LiDAR e a câmara. Os pontos que são potencialmente visíveis em relação à câmara são mantidos, enquanto os restantes são descartados.

No segundo passo, a nuvem de pontos é organizada usando uma *mesh* quadrilátera, economizando tempo de processamento e armazenamento de memória em comparação com a *mesh* triangular. Os cantos falsos são eliminados, quadriláteros que tenham uma normal inconsistente são suprimidos e os pontos não retornados pelo laser são manipulados de modo a evitar grandes buracos na *mesh* reconstruída.

Por fim, na última etapa é decidido quais os pontos 3D que podem ser observados pela câmara, fornecendo a correspondência entre um ponto 3D e um píxel. De modo a remover os pontos ocultos, é utilizado o algoritmo *z-buffer* [12].

Zeng et at. [13] propuseram um método para gerar uma *point cloud* colorida, utilizando uma câmara panorâmica e um *laser scanner*. Este método faz uso do princípio

colinear, na qual o centro do sistema omnidirecional de múltiplas câmaras, o ponto da imagem na esfera e o ponto do objeto estão alinhados. Assim sendo, é possível obter as coordenadas dos píxeis com base nas coordenadas 3D dos pontos do *laser* e atribuir a estes pontos o valor RGB dos píxeis.

Por outro lado, alguns objetos na imagem panorâmica são bloqueados por outros, e a sua visibilidades é diferente em várias imagens adjacentes. Por isso mesmo, estes autores propõem selecionar a imagem mais próxima para cada ponto através do tempo do *Global Positioning System* (GPS) ou na distância geométrica. Após isto, analisam a visibilidade dos pontos na imagem e selecionam as imagens adjacentes de modo a colorir os pontos caso os objetos se encontrem bloqueados na imagem atual.

Vechersky et al. [14] desenvolveram um algoritmo para colorir nuvens de pontos obtidas por um LiDAR 3D, adicionando uma câmara ao sistema. Estes autores identificaram aspetos importantes, sendo estes a sincronização do relógio entre o LiDAR e a câmara, a determinação da visibilidade dos pontos e a atribuição de cores para cada ponto 3D.

A sincronização entre os sensores, LiDAR e câmara, é relativamente simples se ambos partilharem o mesmo relógio e/ou *trigger*. Caso contrário, correlacionando o *yaw-rate* obtido da câmara com o extraído pela navegação do robô, é possível sincronizar os dois sensores.

A câmara só pode observar um subconjunto da nuvem de pontos e, para identificar os pontos visíveis, os autores utilizaram o algoritmo proposto por Katz et al. [15, 16]. A atribuição da cor para cada ponto 3D é obtida estimando a média e a covariância de uma distribuição gaussiana de um conjunto de cores. A cor final atribuída ao ponto é a média desta distribuição estimada.

Moussa et al. [17] apresentam um método automático para combinar os dados de um *laser scanner* e imagens de modo a obter uma representação completa de um cenário. Este método é baseado em *bundle adjustment* que estima a orientação das imagens geradas, a partir de dados do *laser* e imagens da câmara. Para tal, é implementado um método otimizado de reconstrução baseado em *Structure and Motion* [18].

Neubauer et al. [19] utilizaram um *laser* 3D terrestre estático e uma câmara montada no topo deste, de modo a obter informações geométricas e a aparência detalhada das pirâmides de Giza. A nuvem de pontos é convertida numa *mesh*, permitindo que as informações da textura preencham o espaço entre os pontos 3D da *point cloud*. O processamento de dados foi realizado pelo software RiSCAN PRO [20].

Abdelhafiz et al. [21] divulgam uma abordagem para combinar a nuvem de pontos 3D do LiDAR com as imagens da câmara, produzindo uma nuvem de pontos 3D colorida. A ideia principal é referenciar as imagens com a nuvem de pontos num sistema de

coordenadas e, em seguida, a cor de cada ponto da nuvem será dada pela imagem. De modo a obter uma *point cloud* mais preenchida, é extraída a textura fornecida pela imagem e esta é incorporada na *cloud*, usando as coordenadas 3D dos píxeis adjacentes.

Point clouds coloridas são geralmente o *output* dos algoritmos de *Simultaneous Localization And Mapping* (SLAM). Em [22, 23] é utilizado um LiDAR e uma câmara, enquanto que em [24, 25] usam um LiDAR e um par *stereo* de câmaras. O foco principal destes artigos é a estimação da *pose* e do mapa em vez da obtenção da nuvem de pontos colorida. A cor de cada ponto é tipicamente determinada a partir de uma única observação.

O FARO *Laser Scanner Focus* 3D [26] é um LiDAR 3D estático com uma elevada precisão, resolução e velocidade, sendo fabricado pela empresa FARO. Este sensor tem um *Field of View* (FoV) horizontal de 360 graus, 300 graus na vertical e contém uma câmara a cores com cerca de 70 megapixeis (dependendo do modelo). Este LiDAR permite obter uma nuvem de pontos colorida do ambiente que o rodeia, realizando a correspondência entre o ponto 3D obtido e o respetivo pixel.

mdLiDAR3000 [27] é um sistema desenvolvido pela empresa *Microdrones*, composto por um LiDAR e uma câmara de 42.4 megapixeis (figura 2.4). Este sistema é usado para produzir nuvens de pontos 3D coloridas, tendo uma gama ampla de aplicações, desde mapeamento, inspeção de áreas, construção, mineração e agricultura.

Figura 2.4: mdLiDAR3000 integrado num UAV da Microdrones [27].

2.3 Calibração da cor da câmara

Taki et al. [28] propuseram um método de calibração de cores para um sistema composto por várias câmaras. Uma câmara é calibrada inicialmente para ser a referência, enquanto que as restantes são calibradas automaticamente usando cores de objetos que se encontram na visão comum da câmara de referência. O método proposto consiste em

três etapas principais: calibração da câmara de referência, estimação da característica das câmaras não calibradas e correspondência de cores em todas as câmaras.

Na primeira etapa, a câmara de referência é calibrada através da medição de várias amostras de cores geradas por um projetor de imagens. Na segunda etapa, as características da cor das câmaras não calibradas são obtidas, usando as cores de objetos na visão comum da câmara de referência, minimizando os erros de estimação. Na última etapa, a calibração das cores de todas as câmaras é realizada usando as características estimadas.

A limitação deste trabalho é que as câmaras (excepto a de referência) devem ser calibradas novamente quando as condições de iluminação mudarem drasticamente.

Joshi et al. [29] apresentam uma *pipeline* de calibração para um grupo elevado de câmaras, assumindo que as condições de iluminação são estáticas para cada câmara. Estes autores também assumem que é possível colocar um alvo de calibração (*Macbeth*), na qual este fica visível para todas as câmaras. O objetivo deste método é obter respostas uniformes entre todas as câmaras e não precisão absoluta das cores.

Esta *pipeline* é divida em duas etapas: configuração e caracterização. A configuração consiste em ajustar automaticamente os ganhos e *offsets* da câmara antes da aquisição de imagens. A caracterização consiste em três partes: correção da não linearidade do sensor, correção da queda radiométrica e a minimização global do erro da cor.

Capítulo 3

Fundamentos Teóricos

Neste capítulo serão introduzidos alguns conceitos que são considerados importantes para ajudar na compreensão do trabalho desenvolvido. Estes conteúdos vão desde conceitos matemáticos, características mecânicas dos sensores e arquitetura de *middleware*.

3.1 Visão computacional

A perceção sensorial tem sido sujeita a elevados esforços de desenvolvimento pela comunidade científica, visto que é uma temática indispensável para que os robôs possam executar as suas tarefas.

Como tal, nos dias que correm, existem vários sensores capazes de captar características do ambiente que rodeiam o sistema, como por exemplo, sensores óticos, sonares e sensores de distância. Todavia, os sensores óticos possuem um papel importante, já que são capazes de fornecer uma elevada quantidade de informação (textura, aparência, entre outras) semelhante à informação dada pelo olho humano.

Por exemplo, no olho humano, a córnea é o revestimento transparente que rodeia o olho, ao passo que a pupila define a abertura da íris de maneira a regular a quantidade de luz que reflete na retina, realizando a função do sensor ótico. O cristalino atua como uma lente, permitindo focar a imagem na retina. A figura 3.1 ilustra alguns elementos do olho humano de forma a facilitar a sua identificação.

No caso das câmaras, estas contêm um sensor ótico do tipo *Charge-Coupled Device* (CCD) ou *Complementary Metal-Oxide Semiconductor* (CMOS), desempenhando o papel da retina comparativamente com o olho humano. Além disso, estes sistemas possuem um mecanismo, denominado de obturador, que controla a quantidade de luz

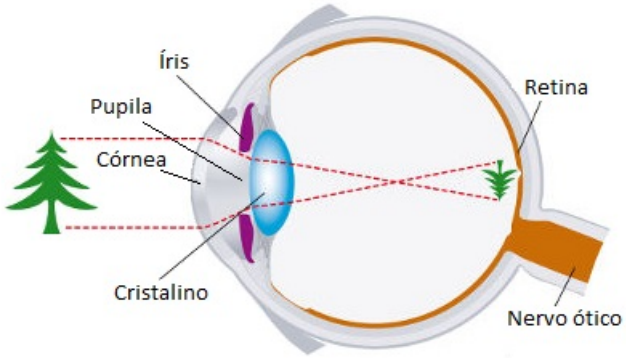


Figura 3.1: Anatomia do olho humano (adaptado de [30]).

que incide na retina, atuando de forma semelhante à pupila do olho humano [31].

No entanto, existem algumas diferenças entre o olho humano e as câmaras. Uma dessas diferenças é o facto da retina humana ser côncava, enquanto que as dos sensores óticos são planas. Em termos de foco, o olho humano consegue mudar a forma do cristalino de modo a focar objetos a diferentes distâncias. No caso das câmaras, as lentes comuns são focadas para observar ao perto ou ao longe, nunca ambas as situações ao mesmo tempo.

3.1.1 Modelo Pinhole

A modelização do sensor de imagem, nas câmaras visíveis, é realizada normalmente através do modelo *pinhole* (figura 3.2). Este modelo descreve a relação matemática entre as coordenadas de um ponto 3D e a projeção desse mesmo ponto no plano da imagem. Contudo, não tem em conta as distorções geométricas ou os objetos desfocados através da lente. Os erros do método dependem da qualidade do sensor, na qual este propaga-se do centro da imagem até às margens.

Este modelo consiste na colocação de todos os raios de luz emitidos e refletidos a atravessarem um pequeno orifício, designado de *pinhole* da câmara. Isto formará uma imagem invertida do objeto observado no plano da imagem.

A coordenada de cada ponto, num sistema visual monocular regido pelo modelo *pinhole*, pode ser deduzida através da geometria dos triângulos semelhantes, conforme definido na equação 3.1.

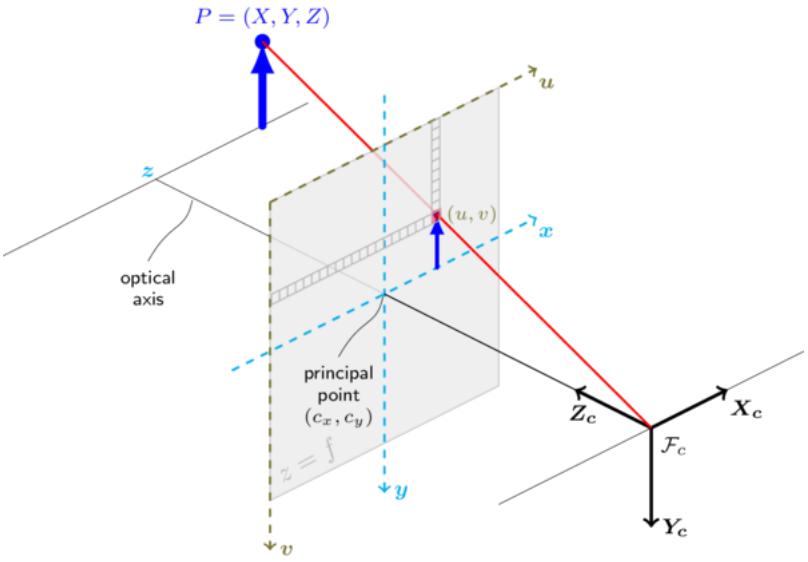


Figura 3.2: Representação ideal do modelo *pinhole* [32].

$$\frac{-y^I}{f} = \frac{y^C}{z^C} \Leftrightarrow -y^I = f \cdot \frac{y^C}{z^C} \tag{3.1}$$

onde f representa a distância focal, y^I é o comprimento do objeto no plano da imagem, y^C é o comprimento do objecto e z^C é a distância entre a câmara e o objeto. Na figura 3.3 pode ser observado cada um destes elementos.

A projeção perspetiva pode ser representada pela equação linear 3.2, devido à existência de uma relação geométrica entre os dois planos, p^I e p^C.

$$\begin{bmatrix} x^I \\ y^I \end{bmatrix} = \frac{f}{z^C} \begin{bmatrix} x^C \\ y^C \end{bmatrix} \tag{3.2}$$

A relação entre os pontos no mundo (x^W, y^W, z^W) e os pontos no referencial da câmara (x^C,y^C,z^C) é obtida através da uma transformação, em coordenadas homogéneas (equação 3.3).

$$\begin{bmatrix} x^C \\ y^C \\ z^C \end{bmatrix} = \begin{bmatrix} R|t \end{bmatrix} \begin{bmatrix} x^W \\ y^W \\ z^W \\ 1 \end{bmatrix} \tag{3.3}$$

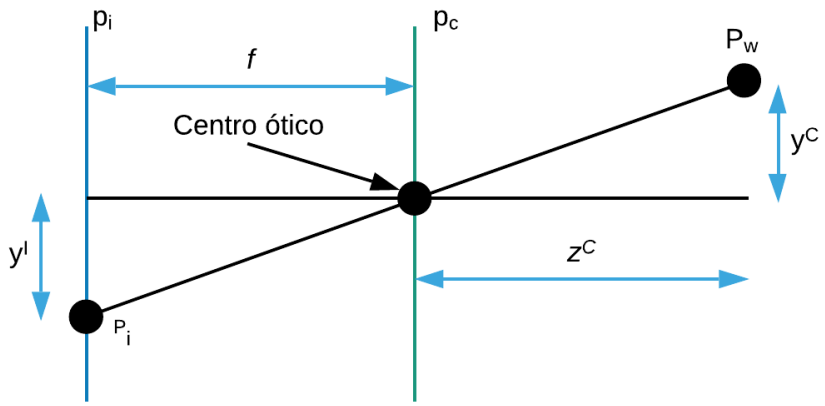


Figura 3.3: Modelo *pinhole*.

onde a matriz R e o vetor t representam a rotação e a translação, respetivamente, que transforma os pontos do mundo no referencial da câmara. Combinando as equações 3.2 e 3.3, pode-se mapear pontos do mundo no referencial 2D da imagem, através da equação 3.4:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} R|t \end{bmatrix} \begin{bmatrix} x^W \\ y^W \\ z^W \\ 1 \end{bmatrix} \tag{3.4}$$

onde s representa um fator de escala arbitrário, u e v representam as coordenadas do ponto projetado em píxeis e A é a matriz dos parâmetros intrínsecos (veja a secção 3.1.2). A matriz $A[R|t]$ é designada de matriz de projeção de perspectiva.

3.1.2 Parâmetros intrínsecos

Os parâmetros intrínsecos de uma câmara são caracterizados exclusivamente pelas suas características físicas, tais como o tipo de lente e a geometria interna. Estes parâmetros são normalmente definidos pela matriz A (equação 3.5), na qual as componentes f_x e f_y representam as distâncias focais e as componentes c_x e c_y as coordenadas do centro ótico.

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

A matriz de parâmetros intrínsecos de uma câmara representa as características físicas do sensor ótico, permitindo relacionar as coordenadas do plano da câmara com o plano da imagem, ou seja, associa um ponto no referencial da câmara com o respetivo píxel da imagem.

Tal como referido na secção anterior, o modelo *pinhole* não tem em conta a distorção da lente, a nitidez dos objetos imposta pela distância focal, entre outras. A distorção da lente é composta por duas componentes: radial e tangencial. A distorção radial consiste na deformação de um objeto no plano da imagem, devido às características físicas da lente. Esta distorção é nula no centro ótico da imagem e aumenta desde o centro até às extremidades [33]. Este efeito é intensificado quando a lente apresenta uma grande abertura angular. Na figura 3.4 são ilustrados os tipos de distorções radiais.

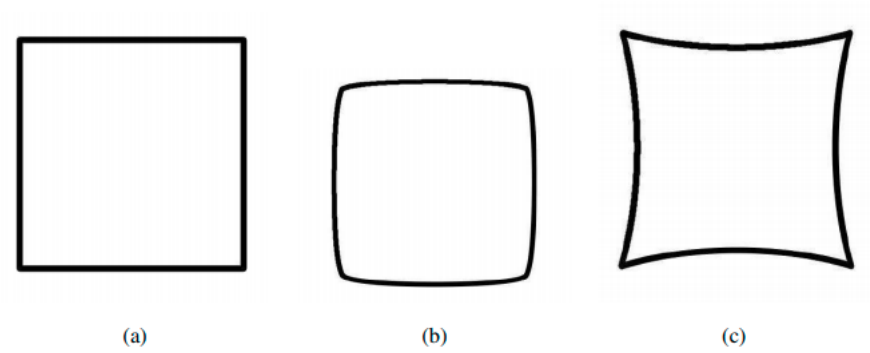


Figura 3.4: Distorção radial: sem distorção (a), *Barrel distortion* (b), *Pincushion distortion* (c) [34].

A distorção tangencial é causada pelo desalinhamento físico, em orientação e posição, entre o centro do semicondutor e o centro da lente. Este erro é mínimo, sobretudo nos sensores modernos com distância focal fixa, podendo ser desprezado nestes casos [35].

Com o recurso a modelos de estimação, é possível obter os coeficientes de distorção, permitindo corrigir a imagem através das equações 3.7 e 3.8 para o caso da distorção radial, e 3.9 e 3.10 para a distorção tangencial. Estes parâmetros são normalmente representados por $\boldsymbol{k}$ (equação 3.6), onde k_1, k_2 e k_3 são relativos à distorção radial e k_4 e k_5 à distorção tangencial.

$$k = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 & k_5 \end{bmatrix} \tag{3.6}$$

$$x_{u(radial)} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.7}$$

$$y_{u(radial)} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.8}$$

$$x_{u(tangencial)} = x + (2k_4 y + k_5(r^2 + 2x)) \tag{3.9}$$

$$y_{u(tangencial)} = y + (k_4(r^2 + 2y) + 2k_5 x) \tag{3.10}$$

onde r representa a distância ao centro ótico, x e y a localização do ponto em píxeis com distorção e x_u e y_u a posição do ponto sem distorção.

3.1.3 Parâmetros extrínsecos

Os parâmetros extrínsecos convertem coordenadas 3D do referencial do mundo para o referencial da câmara. Com o objetivo de obter as coordenadas de um ponto 3D de um referencial para o outro, é essencial obter a relação entre os dois referencias. Esta relação pode ser dividida em duas componentes: rotação e translação. A rotação é tipicamente definida por uma matriz e mapeia a relação angular entre os dois referenciais e a translação representa o deslocamente entre a origem dos dois sistemas de coordenadas. Na figura 3.5 é ilustrado a rotação e a translação de um ponto p no referencial do mundo para o referencial da câmara.

A matriz de rotação R pode ser obtida a partir dos ângulos de *Euler* (*roll* (ϕ), *pitch* (θ) e *yaw* (ψ)) [36] e resulta da sequência ordenada de rotações em torno dos três eixos (equação 3.11).

$$R = R_z(\psi) R_y(\theta) R_x(\phi) \tag{3.11}$$

A rotação em torno do eixo x (ϕ) é expressa matematicamente através da sequinte matriz:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{3.12}$$

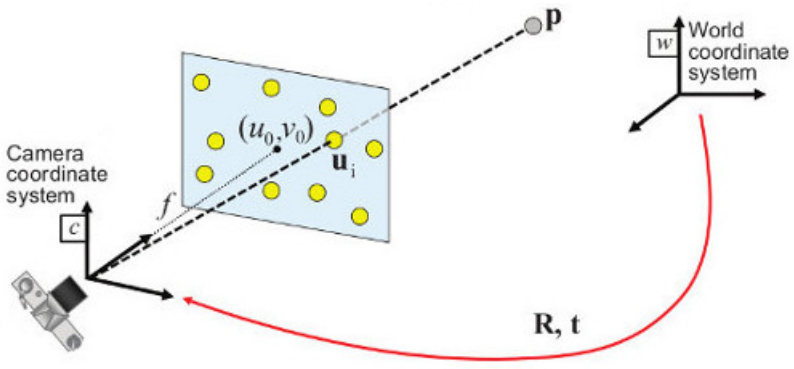


Figura 3.5: Projeção das coordenadas do objeto para as coordenadas da câmara (adaptado de [32]).

A rotação de um ângulo θ em torno do eixo y é definida por:

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{3.13}$$

E, por fim, na equação 3.14 é definida a rotação de um ângulo ψ em torno do eixo z.

$$R_z(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

De modo a complementar a explicação acima, na figura 3.6 são ilustrados os ângulos de *Euler* no referencial de um UAV.

A translação é representada pelo vetor coluna t (equação 3.15).

$$t = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T \tag{3.15}$$

3.1.4 Cor

A perceção da cor é muito importante para os humanos. Esta depende da física da luz e do processamento complexo do olho-cérebro que integra propriedades do estímulo com a experiência. Os seres humanos utilizam as informações das cores para distinguir objetos, materiais, locais, comida e até a hora do dia [38].

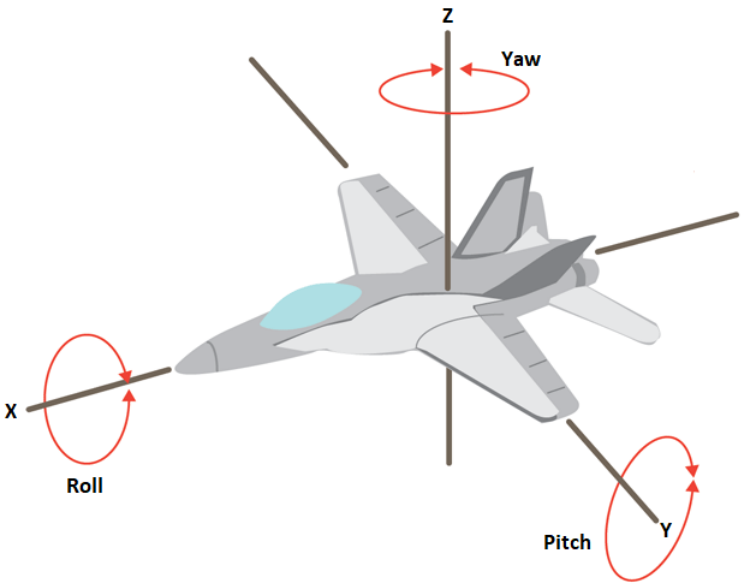


Figura 3.6: Representação dos ângulos de *Euler* num UAV (adaptado de [37]).

3.1.4.1 Física da Cor

A radiação eletromagnética com comprimento de onda λ na faixa dos 400 e 700 nanômetros estimula os neurosensores humanos e produz a sensação de cor [38]. Na figura 3.7 é ilustrado o espectro visível.

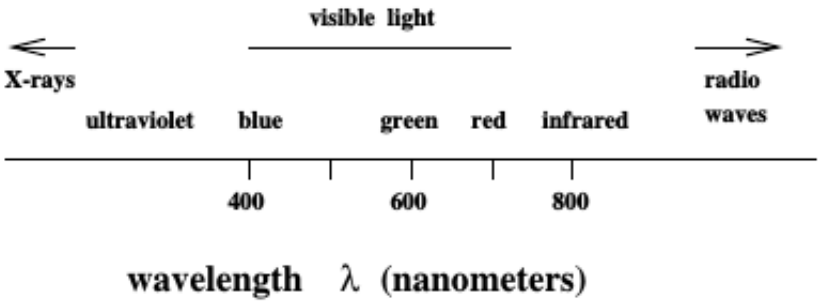


Figura 3.7: Espectro electromagnético da luz visível [38].

As imagens não existem sem luz e, para as produzir, o cenário deve ser iluminado com uma ou mais fontes de luz [39]. A fonte de luz natural mais importante é o sol. O sol é normalmente modelado como um ponto brilhante e distante [40].

A energia luminosa ou radiação é emitida a partir da superfície, devido à energia que interage com as moléculas da superfície do objeto. Algumas superfícies irradiam ou estimulam um elemento do sensor da câmara [38] (figura 3.8). A sensação da cor de um objeto depende de três fatores gerais:

- O espectro de energia em vários comprimentos de onda que ilumina a superfície do objeto;
- A refletância espectral da superfície do objeto, que determina como a superfície altera o espectro recebido no espectro irradiado;
- A sensibilidade espectral do sensor irradiado pela energia da superfície do objeto.

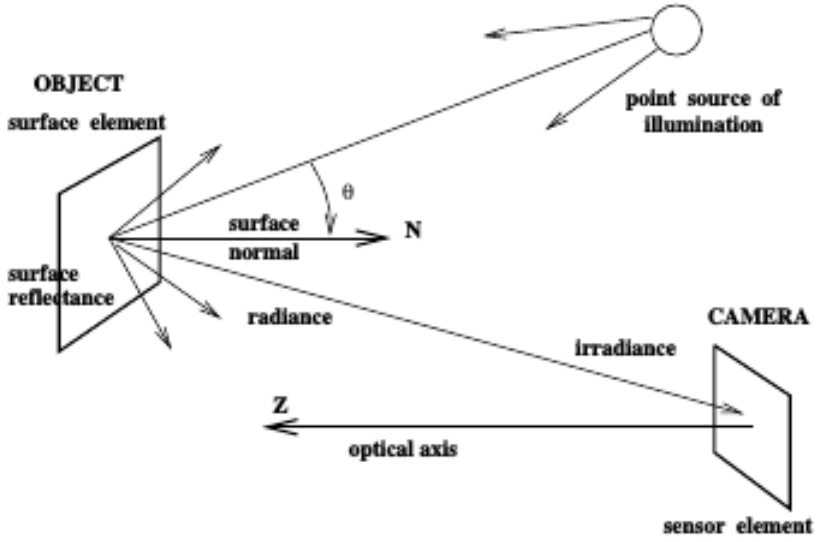


Figura 3.8: A energia luminosa de uma fonte reflete na superfície de um objeto e irradia um elemento do sensor da câmara [38].

3.1.4.2 Codificação RGB

Utilizando apenas três tipos de recetores, os seres humanos são capazes de distinguir milhares de cores. A codificação RGB em sistemas gráficos geralmente utiliza três *bytes*, um para cada componente, permitindo obter cerca de 16 milhões de códigos de cores distintos. As máquinas podem distinguir qualquer combinação de *bits* diferentes, mas estas podem ou não representar diferenças significativas no mundo real [38].

A codificação de uma cor arbitrária no espectro visível pode ser feita através da combinação das três cores primárias (vermelho, verde e azul). A quantidade de cada componente dá a intensidade. O sistema RGB é um sistema de cores aditivo, uma vez que as cores são criadas adicionando as componentes ao preto (0,0,0). Se todas as componentes tiverem a intensidade máxima, a cor obtida é a branca. Na figura 3.9 encontra-se representado o cubo RGB normalizado de cores.

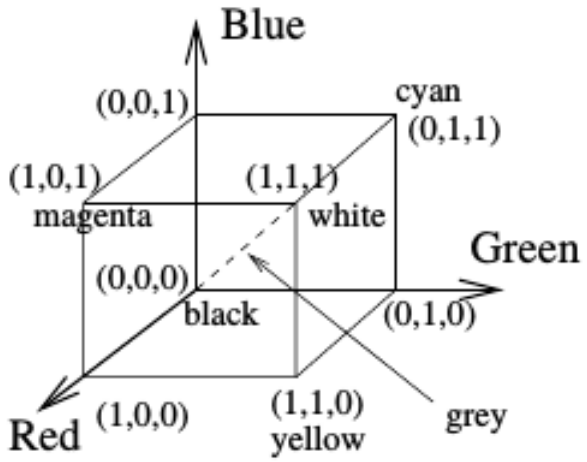


Figura 3.9: Cubo de cores para as coordenadas RGB normalizadas [38].

3.2 LiDAR

O LiDAR é um sensor que mede distâncias em relação a objetos que se encontram ao seu redor, sendo o seu funcionamento baseado no *Time-of-Fligh* (ToF), na qual um pulso de luz é emitido. Durante muitos anos, estes sensores eram *spinning* ou *flash*, mas, ultimamente, devido ao constante desenvolvimento têm aparecido novos LiDARs, como o *phased-array* e o *MicroElectroMechanical System* (MEMS) [41, 42].

3.2.1 Spinning LiDAR

Um *spinning* LiDAR faz medições periodicamente através da emissão de um feixe de luz. Ao adicionar um espelho móvel na direção do raio laser emitido e rodando a estrutura móvel do sensor (ilustrado na figura 3.10), é possível obter um mapa 3D do ambiente que rodeia o sensor.

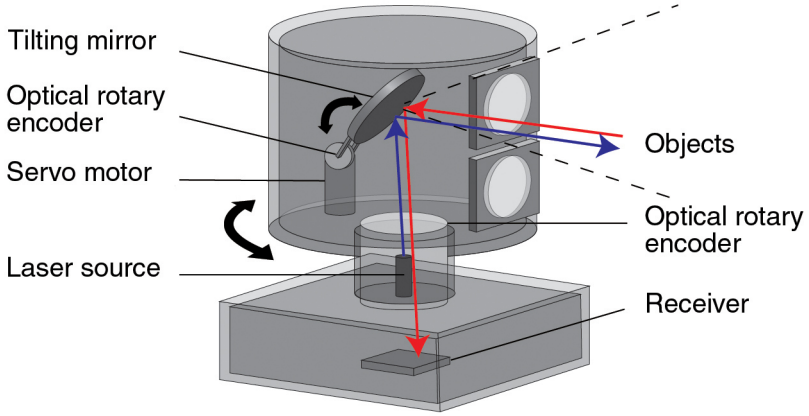


Figura 3.10: Exemplo de um *spinning* LiDAR [43].

Esta configuração permite obter um FoV horizontal de 360 graus e é capaz de emitir o raio laser com uma potência superior comparativamente a um sensor estático, atingindo alcances maiores [44].

Apesar destas vantagens, estes sensores são geralmente volumosos e mecanicamente frágeis a choques e a vibrações. Além disso, a vida útil dos componentes mecânicos é baixa, habitualmente até 2000 horas [41].

3.2.1.1 Conversão de coordenadas polares para coordenadas cartesianas

Um *spinning* LiDAR 3D retorna o ponto P correspondente do feixe em coordenadas polares $P = (r, \omega, \alpha)$, onde r representa a distância medida, ω o ângulo de elevação e α o ângulo de *azimuth* (figura 3.11). De modo a converter em coordenadas cartesianas 3D centradas no referencial do sensor, as equações 3.16, 3.17 e 3.18 devem ser aplicadas.

$$x = r \cdot cos(\omega) \cdot sin(\alpha) \tag{3.16}$$

$$y = r \cdot cos(\omega) \cdot cos(\alpha) \tag{3.17}$$

$$z = r \cdot sin(\omega) \tag{3.18}$$

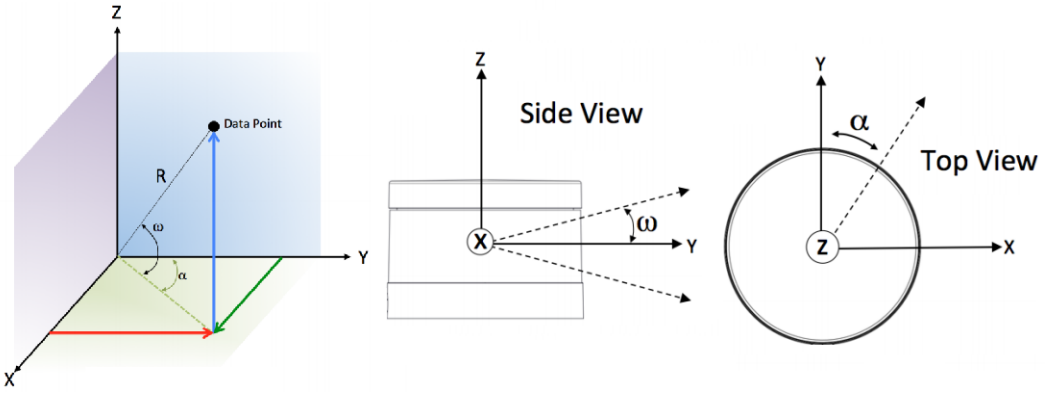


Figura 3.11: Exemplo de um sistema de coordenadas de um *spinning* LiDAR (adaptado de [45]).

3.2.2 Solid-state LiDAR

Solid-state LiDARs são sensores que não possuem peças móveis e estão a ser desenvolvidos para resolver os problemas de tamanho, confiabilidade, complexidade e custo dos LiDARs que utilizam peças mecânicas [42].

3.2.2.1 Phased-array LiDAR

Num *phased-array*, existe um vetor com um número elevado de antenas óticas sincronizadas de uma maneira específica. Ao controlar a sua fase, é possível formar um padrão de radiação que tem um certo tamanho e é apontado numa determinada direção [42] (figura 3.12).

Estes sistemas são mais interessantes economicamente que os restantes, mas têm a tendência de produzir feixes que divergem mais. Isto dificulta a combinação de alta resolução, FoV amplo e alcance longo [44].

3.2.2.2 Flash LiDAR

Este tipo de LiDAR dispara periodicamente um feixe de luz poderoso em direção a uma lente que o propagará para o meio ambiente. A luz refletida será capturada por um vetor de foto-sensores e, como apenas uma pequena parte da luz retorna, há a necessidade da utilização de recetores muito sensíveis, o que faz com que o custo deste sensor seja elevado [41].

A principal vantagem é a aquisição completa do FoV num momento, mas a cobertura

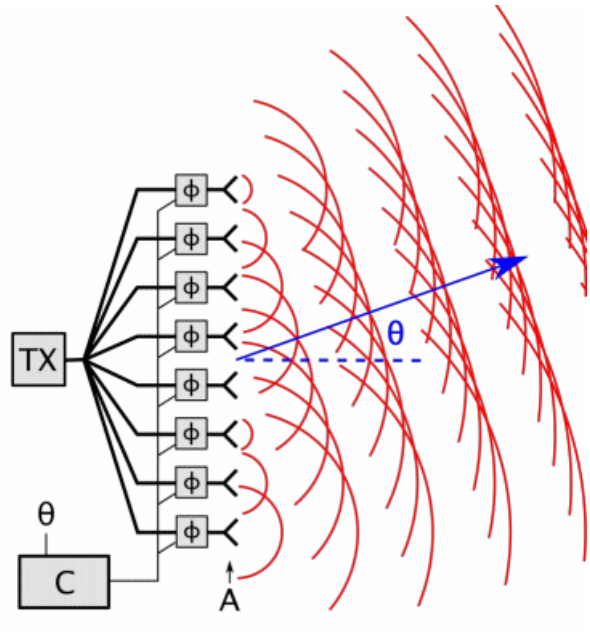


Figura 3.12: Conceito de um *phased-array* LiDAR (adaptado de [41].

do FoV é frequentemente pequena.

3.2.3 MEMS LiDAR

No MEMS LiDAR, os espelhos móveis de alta precisão dos *spinning* LiDARs são substituídos por micro-espelhos MEMS rotatórios de modo a realizar a transmissão direcionando os feixes laser [46].

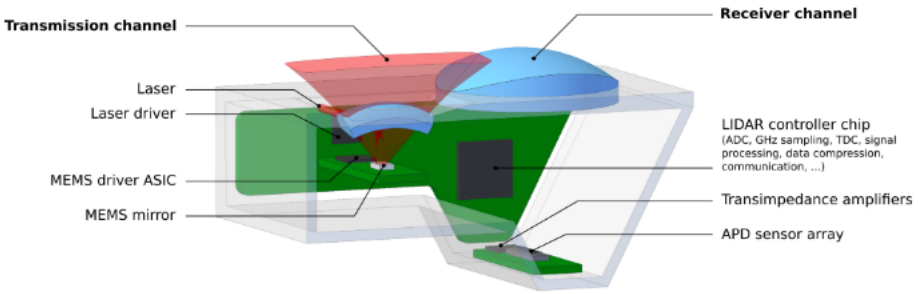


Figura 3.13: Exemplo de um MEMS LiDAR [46].

A redução das partes mecânicas permite uma diminuição do tamanho do LiDAR,

levando a um corte significativo no custo e na sensibilidade às vibrações [46]. Apesar disto, ainda pode estar sujeito a vibrações, choques e a desalinhamento do espelho, necessitando de ser recalibrado. Além disso, alterações elevadas na temperatura podem levar a um sistema não calibrado [41].

3.3 Equação geral do Plano

Um plano no espaço de coordenadas 3D é definido por um vetor que é perpendicular ao plano e um ponto, como ilustrado na figura 3.14. Seja $P_0 = (x_0, y_0, z_0)$ um ponto do plano, $\vec{n} = (a, b, c)$ o vetor normal do plano e $P = (x, y, z)$ qualquer ponto do plano.

Como $\vec{n}$ é ortogonal ao plano, é também ortogonal a qualquer vector que pertença ao plano, mais concretamente ao vetor $\vec{P_0P}$. O produto escalar entre dois vetores ortogonais é igual a 0, como representado na equação 3.19.

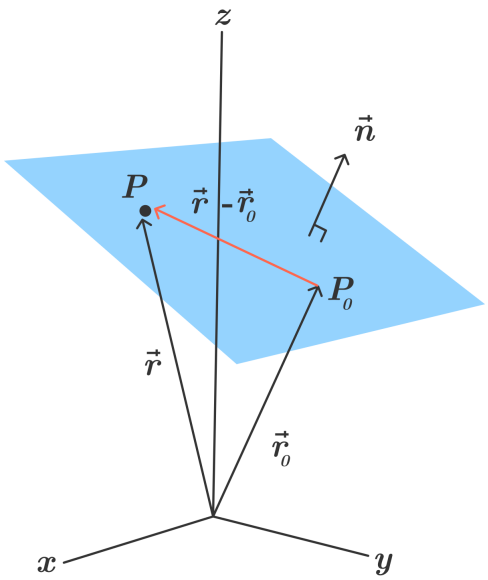


Figura 3.14: Representação de um plano no sistema de coordenadas cartesiano 3D [47].

$$\vec{P_0P} \cdot \vec{n} = 0 \tag{3.19}$$

Desenvolvendo a equação 3.19, obtêm-se a equação cartesiana do plano (equação 3.20).

$$a(x - x_1) + b(y - y_1) + c(z - z_1) = 0 \tag{3.20}$$

A equação 3.21 representa a equação geral do plano, obtida a partir de 3.20.

$$ax + by + cz + d = 0 \tag{3.21}$$

onde a, b e c representam a normal do plano e d a distância do plano à origem do referencial. Para calcular o valor d, resolve-se a equação 3.21 em ordem a d, resultando na equação 3.22.

$$d = -ax - by - cz \tag{3.22}$$

3.4 Equação vetorial de uma linha 3D

Uma linha em $\mathbb{R}^3$ é definida por um ponto e por um vetor direção que é paralelo à reta, como representado na figura 3.15. Seja $P_0 = (x_0, y_0, z_0)$ um ponto da linha, $\vec{v} = (a, b, c)$ um vetor paralelo à linha e $P = (x, y, z)$ qualquer ponto pertencente à linha. Além disso, representando os pontos P_0 e P como vetores posição, temos $\vec{r_0}$ e $\vec{r}$ os vetores de P_0 e P, respetivamente, e seja $\vec{a}$ o vetor $\vec{P_0P}$.

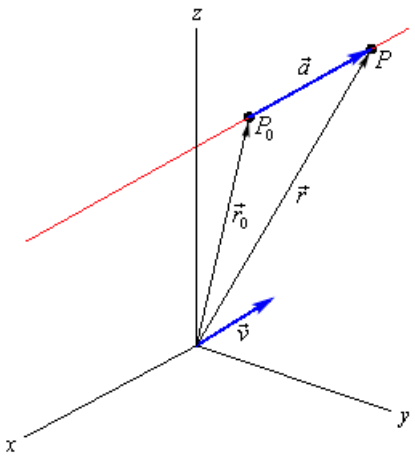


Figura 3.15: Representação de uma linha no sistema de coordenadas cartesiano 3D (adaptado de [47]).

O vetor $\vec{r}$ pode ser definido como:

$$\vec{r} = \vec{r_0} + \vec{a} \tag{3.23}$$

Como os vetores $\vec{a}$ e $\vec{v}$ são paralelos, existe um número t que:

$$\vec{a} = t\vec{v} \tag{3.24}$$

Substituindo a equação 3.24 na equação 3.23, obtêm-se a equação vetorial da linha.

$$\vec{r} = \vec{r_0} + t\vec{v} = (x_0, y_0, z_0) + t(a, b, c) \tag{3.25}$$

onde (x_0, y_0, z_0) representa o ponto da linha, (a, b, c) o vetor direção e t é o parâmetro variável da equação.

3.5 RANSAC

O RANSAC [48] é um método iterativo usado para estimar parâmetros de modelos matemáticos (planos, linhas, cones, cilindros, etc), a partir de um conjunto de dados. Este método seleciona aleatoriamente um subconjunto de amostras de dados e utiliza-o para estimar os parâmetros do modelo. Em seguida, determina os pontos que estão dentro de uma determinada tolerância de erro do modelo gerado. Este processo é repetido para um número de iterações ou caso o número de *inliers* seja superior a um *thershold*, é retornado o modelo que apresenta o maior número de *inliers*.

As amostras que estão de acordo com o modelo são consideradas como *inliers*, enquanto que as restantes como *outliers*. O algoritmo do RANSAC é apresentado em algoritmo 1.

Algorithm 1 RANSAC

Seja:
S: todo o conjunto de dados.
T: threshold.
N: número máximo de iterações.

1: Escolhe uma amostra aleatória de pontos do conjunto S e estima o modelo com os pontos escolhidos.
2: Determina o conjunto de pontos S_i que estão dentro de uma determinada tolerância para o modelo.
3: Se o tamanho de S_i (número de *inliers*) é maior do que um determinado *threshold* T, o modelo é reestimado usando todos os pontos de S_i e termina.
4: Caso contrário, repete os passos 1 até 3 (até N).
5: Depois de N iterações, é escolhido o conjunto S_i com maior número de *inliers* e o modelo é reestimado usando todos os pontos do subconjunto S_i.

Uma desvantagem do RANSAC é que não há um limite para o tempo necessário de modo a estimar os parâmetros do modelo. Quando o número de iterações é limitado, a

solução obtida pode não ser ótima e a melhor que se adapta ao conjunto de dados. Por outro lado, ao aumentar o número de iterações, a probabilidade de obter um modelo razoável aumenta. Porém, o tempo computacional também aumentará.

3.6 Transformada de Hough

A transformada de *Hough* é um técnica de extração de *features*, tais como linhas, círculos ou outras curvas paramétricas, usada em visão computacional, análise de imagem e processamento de imagem digital. A deteção de linhas em imagens, utilizando a transformada de *Hough*, foi usada primeiro por [49].

A equação geral de uma linha é dada pela equação 3.26, onde m representa o declive e b a ordenada na origem. Porém, esta fórmula não funciona para linhas verticais. Portanto, a transformada de *Hough* utilizada a equação 3.27, onde ρ é a distância perpendicular da linha até à origem e θ é o ângulo que a perpendicular faz com o eixo x, como ilustrado na figura 3.16.

$$y = mx + b \tag{3.26}$$

$$\rho = x\cos\theta + y\sin\theta \tag{3.27}$$

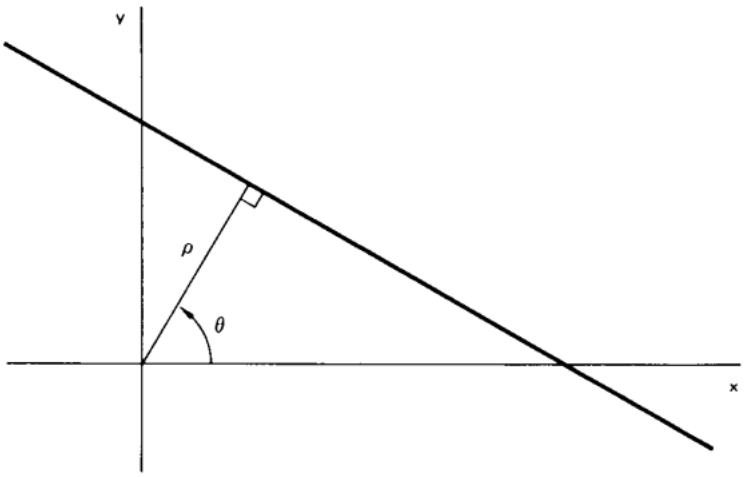


Figura 3.16: Representação da equação de uma linha na forma normal [49].

O algoritmo da transformada de *Hough* necessita de um acumulador A, cuja dimensão corresponde ao número de parâmetros desconhecidos da equação. Por exemplo, o número

de parâmetros na equação de uma linha é igual a dois (ρ e θ). É possível definir a resolução de ambos os parâmetros, ρ e θ, no acumulador.

Cada pixel é examinado, assim como os seus vizinhos na imagem, determinando se existem evidências suficientes de um *edge* (alteração significativa na intensidade da imagem) naquele pixel e, se sim, calcula os parâmetros da linha naquele pixel. Assim que os parâmetros num determinado pixel são estimados, $A[\rho, \theta]$ é incrementado [38].

Depois de todos os pixeis estarem processados, o acumulador é analisado. Para tal, é definido um *threshold* e todos os valores superiores a esse *threshold* são interpretados como linhas.

O algoritmo 2 descreve em detalhe a transformada de *Hough* para a deteção de linhas na imagem. De modo a aplicar este método, é necessário primeiro encontrar os *edges* na imagem, através do algoritmo de *Canny* [50], *Sobel* [51], entre outros.

Algorithm 2 Transformada de Hough

1: **Input:** Imagem binária obtida a partir de um *edge detector*, *threshold*, resolução dos parâmetros ρ e θ
2: **Output:** Valores de ρ e θ de cada linha
3: Inicializar A$[\rho, \theta] = 0$
4: **for each** pixel(x,y) **do**
5: **for each** θ $\theta_j = -90$ até $\theta_j = 90$ **do**
6: $\rho = x\cos(\theta_j) + y\sin(\theta_j)$
7: Aproximar ρ para o valor mais próximo da célula p_q
8: $A[q, j] = A[q, j] + 1$ se θ_j resulta em p_q
9: **end for**
10: **end for**
11: Descobrir os elementos de $A[q, j]$ que são superiores ao *threshold* definido
12: Estes elementos formam uma linha

3.7 Robot Operating System

O *Robot Operating System* (ROS) [52] é uma *framework open-source* de *software* usada em aplicações robóticas, estando a tornar-se cada vez mais utilizada pela comunidade de investigação robótica. A sua implementação compreende uma infraestrutura de comunicações distribuídas, *drivers* de baixo nível para uma grande variedade de sensores e atuadores, um conjunto elevado de algoritmos de robótica e de ferramentas de desenvolvimento e visualização. A comunicação entre processos é baseada numa topologia *Peer-To-Peer* (P2P) [52].

Um sistema ROS geralmente é constituído por vários processos independentes, designados de *nodes*, que se comunicam através de um mecanismo de *publish-subscribe* [53]. A comunicação é feita passando mensagens através de tópicos. No geral, os publicadores e os subscritores não estão cientes da existência um do outro [52].

A comunicação P2P entre os *nodes* é estabelecida por um agente intermediário, o *roscore*: quando um novo nó é criado, ele coneta-se ao *roscore* e lista os tópicos que publicará e aqueles que se deseja subscrever; o *roscore* retorna as informações dos *nodes* que publicam os tópicos desejados, permitindo que os nós novos e os existentes façam a conexão. O *roscore* também é capaz de trabalhar com *nodes* distribuídos ao longo de uma *Local Area Network* (LAN) (figura 3.17).

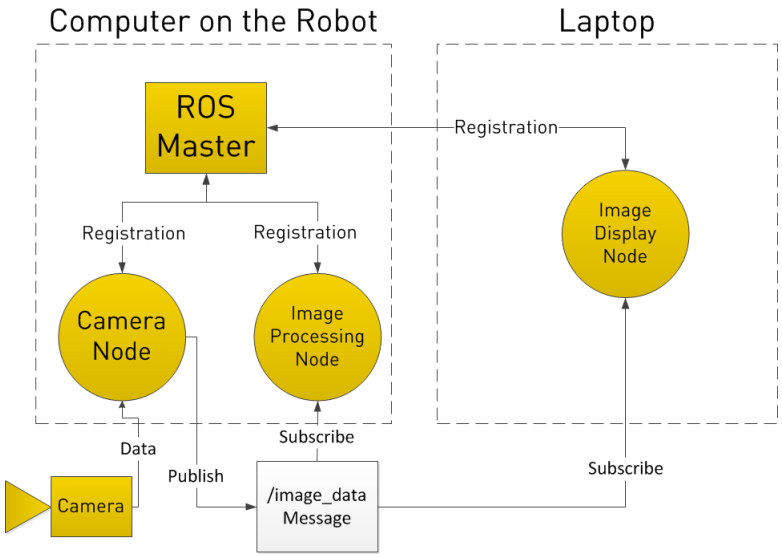


Figura 3.17: Arquitetura de alto nível ROS usando uma LAN [54]. Os círculos amarelos representam os *nodes*, o quadrado branco o tópico e o *roscore* é representado pelo quadrado amarelo.

Além dos *nodes*, o ROS oferece o uso de *nodelets*. Estes comportam-se de maneira semelhante aos *nodes*, mas têm um custo de cópia zero entre *nodelets* do mesmo *handler* [55]. Quando um *nodelet* se subscreve ou publica num determinado tópico do mesmo *nodelet manager*, em vez de a mensagem ser transmitida por *Transmission Control Protocol/Internet Protocol* (TCP/IP), apenas é passado um apontador C++. Isto elimina a ineficiência da transmissão das mensagens, mantendo a natureza modular dos nós ROS [56].

Esta página foi intencionalmente deixada em branco.

Capítulo 4

Arquitetura do Sistema

A utilização de um UAV para realizar reconstrução 3D apresenta várias exigências relativamente à utilização de alguns sensores de perceção. Estes podem ser usados para adquirir dados para reconstruir o ambiente que o robô mapeou, mas também para ajudar alguns algoritmos de deteção de obstáculos e de linhas elétricas, bem como para executar algumas manobras autónomas predefinidas.

Este capítulo contém uma visão geral da arquitetura de *hardware* e de *software* com a finalidade de aplicar o algoritmo desenvolvido.

4.1 Arquitetura de *Hardware*

O algoritmo desenvolvido destina-se a ser aplicado num UAV, sendo este constituído por vários sensores, que vão desde IMU, LiDAR, câmara e recetor *Global Navigation Satellite System* (GNSS). Além disso, o robô tem um computador de bordo responsável por adquirir todos os dados enviados pelos sensores. A arquitetura de *hardware* de alto nível encontra-se ilustrada na figura 4.1.

Hoje em dia, as câmaras e os LiDARs podem facilmente fornecer muitas imagens e centenas de milhares de pontos a cada segundo, respetivamente. Dada esta grande quantidade de dados, estes sensores necessitam de estabelecer uma comunicação com o computador de bordo através de uma conexão *ethernet* (figura 4.1, a roxo) ou outra que permita uma taxa elevada de transmissão de dados. A interface USB 3.0, apesar de ser rápida, pode causar ruído no magnetómetro do UAV, sendo preferível utilizar a conexão *ethernet*. Apesar do IMU ter uma frequência elevada, a quantidade de dados transmitida por trama é reduzida e, portanto, utiliza uma interface RS-422 (figura 4.1, a laranja).

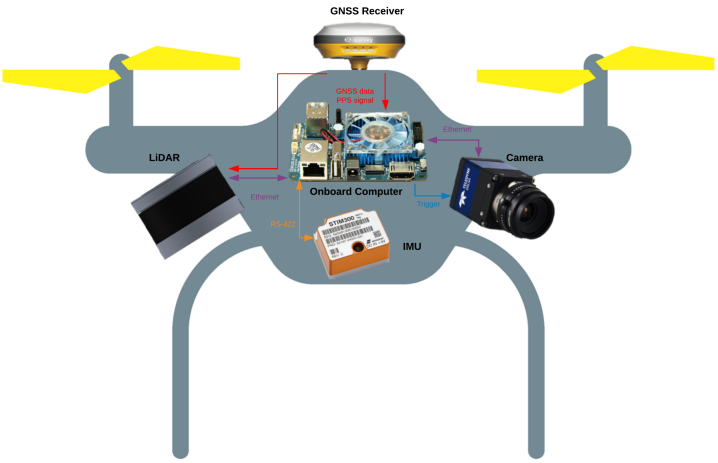


Figura 4.1: Arquitetura de *hardware* de alto nível.

O recetor GNSS permite sincronizar todos os *timestamps* de dados dos subsistemas, usando como referência o *Coordinated Universal Time* (UTC). O recetor envia os dados GNSS e um sinal *Pulse Per Second* (PPS) (figura 4.1, a vermelho) que são usados para sincronizar os dados do LiDAR, assim como o *clock* do *Central Processing Unit* (CPU). Como o CPU está sincronizado, todos os dados do IMU e da câmara também estarão sincronizados com os restantes subsistemas.

A câmara é disparada por *hardware* pelo computador de bordo (figura 4.1, a azul), na qual o instante de tempo do *trigger* é armazenado para, em pós-processamento, ser associado o *timestamp* à respetiva imagem.

4.2 Arquitetura de *Software*

A *pipeline* de software para reconstruir o ambiente que o UAV mapeou, combinando a informação dada por um LiDAR e uma câmara, encontra-se dividida em duas camadas. Na figura 4.2 é exposto a *pipeline* de *software* de alto nível, na qual a maior parte foi desenvolvida utilizando a *framework* ROS.

A primeira camada é realizada em *offline*, enquanto que a segunda pode ser realizada em *real-time*. A camada *offline* compreende o processo de calibração dos parâmetros extrínsecos entre o LiDAR e a câmara, isto é, a rotação e a translação entre ambos os sensores. Esta camada permitirá mapear os pontos do LiDAR no referencial da câmara. O método de calibração é explicado na secção 5.1.

Por outro lado, a segunda camada é composta por várias etapas, tal como demons-

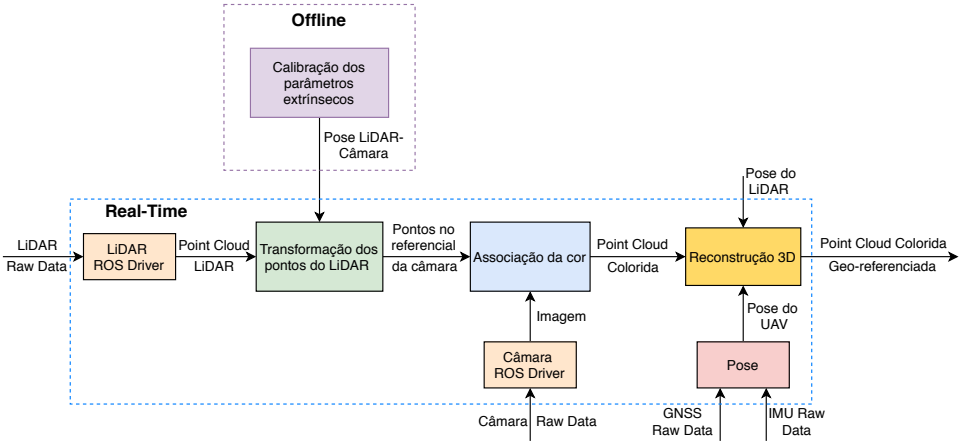


Figura 4.2: *Pipeline* de *software* de alto nível.

trado na figura 4.2. Os dados *raw* do LiDAR são convertidos numa nuvem de pontos no referencial do sensor. A partir dos parâmetros extrínsecos, a *point cloud* do LiDAR é transformada no referencial da câmara.

A imagem é obtida a partir de um *driver*, que interpreta as tramas de dados enviadas pela câmara. Os pontos do LiDAR no referencial da câmara são convertidos para o plano da imagem, onde a maior parte são removidos, uma vez que o FoV da câmara é bastante inferior ao do LiDAR. Após isto, é realizada a associação da cor de cada ponto, através da componente RGB da imagem, obtendo-se uma *point cloud* colorida.

A *pose* do LiDAR pode ter uma relação fixa em relação ao UAV, se estiver fixa à plataforma, ou uma relação dinâmica, se estiver conetado a um *gimbal*, por exemplo. A *pose* do UAV é estimada a partir dos dados *raw* do recetor GNSS e do IMU. A localização do robô permitirá, juntamente com a nuvem de pontos colorida e a *pose* do LiDAR, realizar a reconstrução 3D do meio ambiente que o robô mapeou.

Esta página foi intencionalmente deixada em branco.

Capítulo 5

Algoritmo

Este capítulo detalha o algoritmo desenvolvido para cumprir os objetivos e requisitos abordados na dissertação. Na primeira secção é apresentado o método de calibração dos parâmetros extrínsecos entre uma câmara e um LiDAR 3D, na seguinte o algoritmo desenvolvido para colorir uma *point cloud* e, por último, o procedimento da reconstrução 3D, utilizando as *point clouds* coloridas.

5.1 Calibração dos parâmetros extrínsecos

O método implementando baseia-se no trabalho desenvolvido em [6], na qual os autores usam correspondências entre planos e linhas 3D de modo a obter os parâmetros extrínsecos entre uma câmara e um LiDAR.

5.1.1 Definição do problema e notações

O problema da calibração extrínseca de uma câmara e um LiDAR é estimar a rotação e a translação relativa entre os dois sensores. Escalares, vetores e matrizes são representados em itálico, minúsculas a negrito e maiúsculas a negrito, respetivamente. A matriz de rotação e a translação do LiDAR para a câmara é simbolizada por $\mathbf{R}_L^C$ e $\mathbf{t}_L^C$, respetivamente.

Como já referido, é utilizado um *checkerboard* como alvo de calibração, estando visível para ambos os sensores. O plano e as extremidades do *target* são explorados para estimar $\mathbf{R}_L^C$ e $\mathbf{t}_L^C$, como ilustrado na figura 5.1.

Dada a *pose* número i do *checkerboard*, é possível estimar o plano π_i^C e os quatro limites l_{ij}^C (j = 1,2,3,4) no plano da imagem. O plano π_i^C é parametrizado por $[\mathbf{n}_i^C; \mathbf{d}_i^C]$,

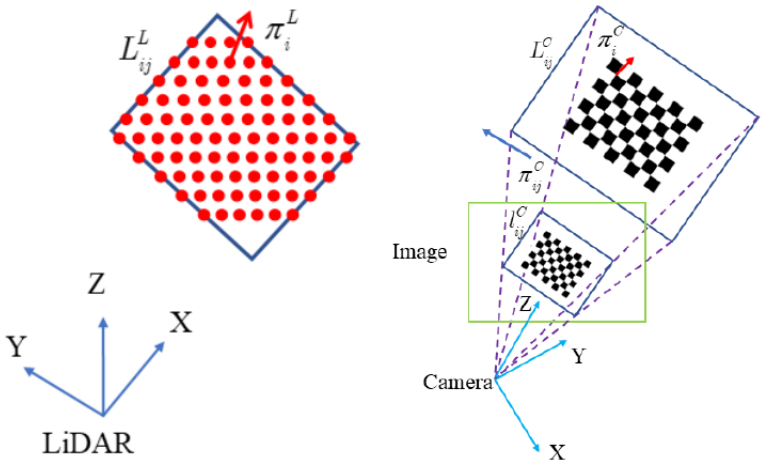


Figura 5.1: Restrições geométricas do alvo de calibração [6].

onde $\mathbf{n}_i^C$ é a normal e $\mathbf{d}_i^C$ é a distância do π_i^C à origem do referencial, enquanto que L_{ij}^C é parametrizado por $[\mathbf{d}_{ij}^C; \mathbf{p}_{ij}^C]$, na qual $\mathbf{d}_{ij}^C$ e $\mathbf{p}_{ij}^C$ são a direção e um ponto de L_{ij}^C, respetivamente. $\mathbf{n}_i^C$ e $\mathbf{d}_{ij}^C$ são vetores unitários.

Por outro lado, também é possível estimar o plano π_i^L do *checkerboard* e os seus quatro limites L_{ij}^L no referencial do LiDAR. Os pontos do *laser* no π_i^L são representados por $\{\mathbf{P}_{im}^L\}$ $(m = 1, \cdots, N_i)$, e os pontos no L_{ij}^L por $\{\mathbf{G}_{ijk}^L\}$ $(k = 1, \cdots, K_{ij})$. Dado $\{\mathbf{P}_{im}^L\}$ e $\{\mathbf{G}_{ijk}^L\}$, é possível calcular a normal $\mathbf{n}_i^L$ e o centróide $\bar{\mathbf{P}}_i^L$ do π_i^L e também as direções $\mathbf{d}_{ij}^L$ e os centróides $\bar{\mathbf{G}}_{ij}^L$ do L_{ij}^L no referencial do LiDAR.

Este método explora as correspondências $\pi_i^C \leftrightarrow \pi_i^L$ e $L_{ij}^C \leftrightarrow L_{ij}^L$ para estabelecer restrições de modo a obter $\mathbf{R}_L^C$ e $\mathbf{t}_L^C$.

5.1.2 Extração automática de features

Os parâmetros do plano no sistema de coordenadas da câmara podem ser facilmente calculados, através dos parâmetros extrínsecos entre o *checkerboard* e a câmara. Para tal, é necessário detetar os cantos do padrão e, posteriormente, estimar a *pose* do alvo em relação à câmara. Esta estimação é um problema PnP, na qual são conhecidos os pontos da imagem e as coordenadas 3D correspondentes no referencial do mundo (figura figura 5.2).

A matriz de rotação $\mathbf{R}_W^C$ e o vetor de translação $\mathbf{t}_W^C$ relacionam coordenadas do mundo no referencial da câmara. $\mathbf{R}_W^C$ é uma matriz ortogonal e as colunas representam

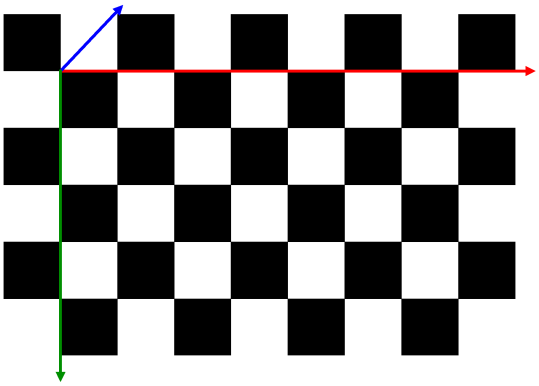


Figura 5.2: Referencial do mundo definido no *checkerboard*. O eixo x está representado a vermelho, y a verde e z a azul.

os três vetores de direção unitários de cada eixo do sistema cartesiano (x,y,z). Assim sendo, a normal $\mathbf{n}_i^C$ do plano π_i^C do *checkerboard* no referencial da câmara pode ser obtida através da equação 5.1.

$$\mathbf{n}_i^C = \mathbf{R}_W^C \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \tag{5.1}$$

A distância $\mathbf{d}_i^C$ pode ser obtida através da equação 5.2.

$$\mathbf{d}_i^C = -a_i x_c - b_i y_c - c_i z_c \tag{5.2}$$

onde $\mathbf{n}_i^C = [a_i, b_i, c_i]$ é a normal do plano e $\mathbf{P}_W^C = [x_c, y_c, z_c]$ é um ponto do mundo no referencial da câmara.

As linhas na imagem são detetadas usando a transformada de *Hough* (explicado na secção 3.6) [49]. Dado o alvo detetado, são escolhidas as quatro linhas que formam os quatro limites do alvo e, posteriormente, é calculada a interseção destas linhas de modo a obter os quatro cantos do *checkerboard* no plano da imagem. O ponto na imagem da interseção de duas linhas é obtido resolvendo o sistema linear definido na equação 5.3.

$$\begin{aligned} \rho_1 &= x \cos\theta_1 + y \sin\theta_1 \\ \rho_2 &= x \cos\theta_2 + y \sin\theta_2 \end{aligned} \tag{5.3}$$

Estes pontos são convertidos para o referencial do mundo e, em seguida, para o referencial da câmara, utilizando $\mathbf{R}_W^C$ e $\mathbf{t}_W^C$. De forma a obter os cantos do alvo no

referencial do mundo, é utilizada a equação 3.4. Assumindo que o plano está em $Z = 0$ no sistema de coordenadas do mundo (como ilustrado na figura 5.2) e que $\mathbf{R} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{bmatrix}$, a equação 3.4 pode ser simplificada, obtendo-se a equação 5.4.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} x^W \\ y^W \\ 1 \end{bmatrix} \tag{5.4}$$

Os pontos do mundo são obtidos resolvendo a equação 5.4 em ordem a estes, obtendo-se a equação 5.5.

$$\begin{bmatrix} x^W \\ y^W \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{5.5}$$

Os pontos do mundo no referencial da câmara são obtidos a partir da equação 3.3. A partir destes, é possível calcular o vetor direção e o ponto das quatro linhas que formam os limites do alvo. Sejam $\mathbf{A} = (x_0, y_0, z_0)$ e $\mathbf{B} = (x_1, y_1, z_1)$ os dois pontos de uma linha, o vetor direção unitário poderá ser dado pela equação 5.6. As *features* detetadas na imagem podem ser visualizadas na figura 5.3.

$$\vec{d}_{\mathbf{AB}} = \frac{\mathbf{B} - \mathbf{A}}{\|\mathbf{B} - \mathbf{A}\|} \tag{5.6}$$

onde $\|.\|$ representa a norma.

Para extrair as *features* no LiDAR, é necessário definir aproximadamente a área do *checkerboard*, facilitando a obtenção dos pontos que formam o plano π_i^L. Os pontos do laser do plano são detetados usando o algoritmo RANSAC (explicado na secção 3.5) com o modelo de um plano, obtendo-se os parâmetros do plano (normal e distância à origem).

Após este passo estar concluído, é possível obter os pontos que formam os limites do alvo. Para tal, deteta-se cada linha do *laser*, utilizando o RANSAC com o modelo de uma linha. Estas linhas fornecem os limites esquerdo e direito do alvo que são posteriormente divididos em duas partes, através da maior mudança de direção entre dois vetores consecutivos. O ângulo entre dois vetores, $\vec{\mathbf{u}}$ e $\vec{\mathbf{v}}$, é dado pela equação 5.7.

$$\theta = \arccos\left(\frac{\vec{\mathbf{u}} \cdot \vec{\mathbf{v}}}{\|\vec{\mathbf{u}}\| \, \|\vec{\mathbf{v}}\|}\right) \tag{5.7}$$

Como estes pontos apresentam, por vezes, algum ruído (como ilustrado na figura 5.4),

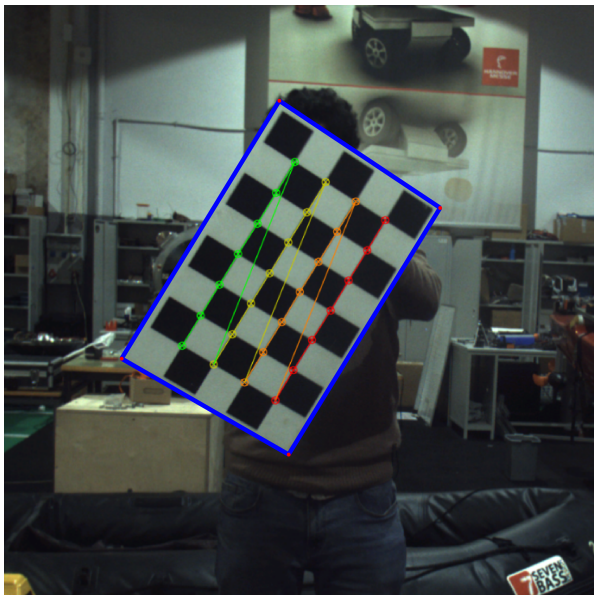


Figura 5.3: *Features* extraídas no plano da imagem.

é necessário reduzir esta distorção.

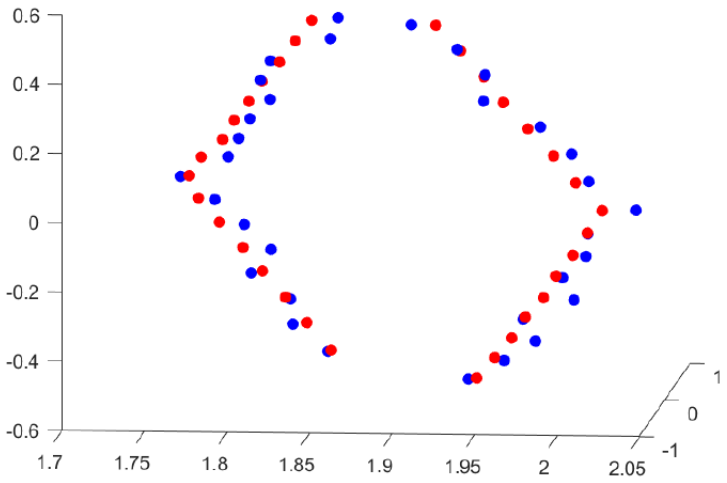


Figura 5.4: Remoção do ruído dos pontos da fronteira do alvo [6]. Os pontos azuis são os pontos originais e os pontos vermelhos são os obtidos após eliminar o ruído.

Dessa forma, primeiro projeta-se todos os pontos do laser no plano e, em seguida,

realiza-se um *fit* de uma linha (com os parâmetros obtidos pelo RANSAC) para cada linha do *scan*. Os pontos da fronteira são projetados na respetiva linha e, finalmente, é utilizado o RANSAC para remover os *outliers* e estimar os parâmetros de cada linha da fronteira do alvo. Na figura 5.5 encontram-se ilustradas as *features* no referencial do LiDAR.

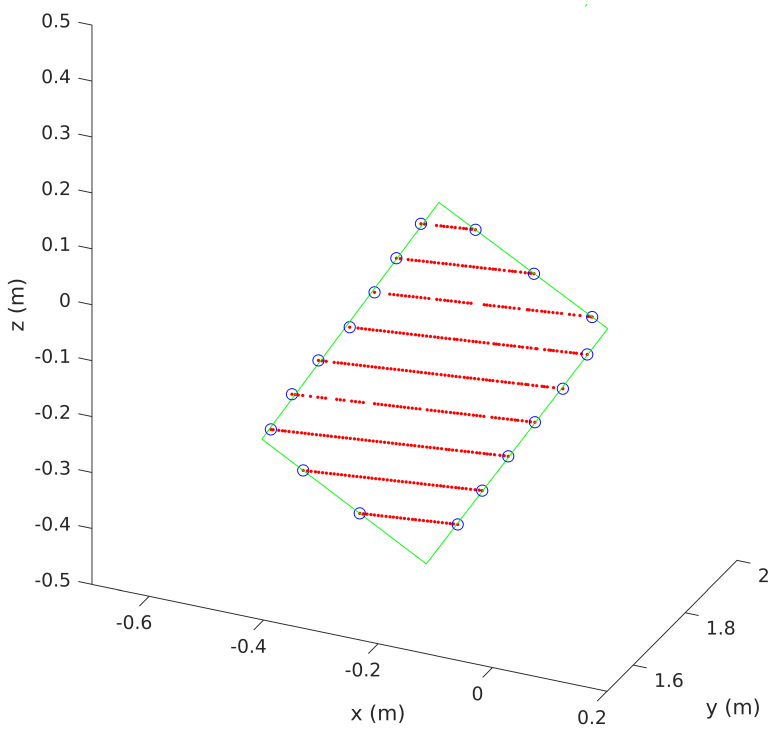


Figura 5.5: *Features* extraídas no referencial do LiDAR.

5.1.3 Calibração de extrínsecos

5.1.3.1 Restrições geométricas

Considerando a correspondência de uma linha $L_{ij}^{C} \leftrightarrow L_{ij}^{L}$, é possível obter as restrições de $\mathbf{R}_{L}^{C}$ e $\mathbf{t}_{L}^{C}$, representadas nas equações 5.8 e 5.9.

$$\mathbf{R}_{L}^{C}\mathbf{d}_{ij}^{L} = \mathbf{d}_{ij}^{C} \tag{5.8}$$

$$\left(\mathbf{I} - \mathbf{d}_{ij}^{C}\left(\mathbf{d}_{ij}^{C}\right)^{T}\right)\left(\mathbf{R}_{L}^{C}\mathbf{G}_{ijk}^{L} - \mathbf{P}_{ij}^{C} + \mathbf{t}_{L}^{C}\right) = \mathbf{0}_{3\times 1} \tag{5.9}$$

onde $\mathbf{I}$ é a matriz identidade, $\mathbf{P}_{ij}^{C}$ é um ponto da linha no referencial da câmara. Uma correspondência $L_{ij}^{C} \leftrightarrow L_{ij}^{L}$ dá quatro restrições independentes em $\mathbf{R}_{L}^{C}$ e $\mathbf{t}_{L}^{C}$. De modo a simplificar a notação, será definida a matriz $\mathbf{D}_{ij} = \mathbf{I} - \mathbf{d}_{ij}^{C}\left(\mathbf{d}_{ij}^{C}\right)^{T}$ nas seguintes descrições.

Dada a correspondência de um plano $\pi_{i}^{C} \leftrightarrow \pi_{i}^{L}$, podemos obter as equações 5.10 e 5.11 para $\mathbf{R}_{L}^{C}$ e $\mathbf{t}_{L}^{C}$.

$$\mathbf{R}_{L}^{C}\mathbf{n}_{i}^{L} = \mathbf{n}_{i}^{C} \tag{5.10}$$

$$\mathbf{n}_{i}^{C} \cdot \left(\mathbf{R}_{L}^{C}\mathbf{P}_{im}^{L} + \mathbf{t}_{L}^{C}\right) + d_{i}^{C} = 0 \tag{5.11}$$

onde $\cdot$ representa o produto escalar. Uma correspondência $\pi_{i}^{C} \leftrightarrow \pi_{i}^{L}$ fornece três restrições independentes em $\mathbf{R}_{L}^{C}$ e $\mathbf{t}_{L}^{C}$.

5.1.3.2 Estimação da pose

Dadas N *poses* do *checkerboard*, podemos usar as equações 5.8 e 5.10 para obter a rotação entre o LiDAR e a câmara. Como $\mathbf{n}_{i}^{L}$ é perpendicular a $\mathbf{d}_{ij}^{L}$, a introdução da correspondência de linhas aumenta significativamente a diversidade das medições. Uma vez que os dados apresentam ruído, a matriz $\mathbf{R}_{L}^{C}$ é obtida minimizando a função de custo, dada pela equação 5.12.

$$\tilde{\mathbf{R}}_{L}^{C} = \arg\min_{\mathbf{R}_{L}^{C}} \sum_{i=1}^{N}\sum_{j=1}^{4} \left\|\mathbf{R}_{L}^{C}\mathbf{d}_{ij}^{L} - \mathbf{d}_{ij}^{C}\right\|^{2} + \left\|\mathbf{R}_{L}^{C}\mathbf{n}_{i}^{L} - \mathbf{n}_{i}^{C}\right\|^{2} \tag{5.12}$$

Este problema pode ser resolvido utilizando o método *Singular Value Decomposition* (SVD) [57]. A decomposição em valores singulares de uma matriz $\mathbf{A}$ é a fatorização de $\mathbf{A}$ no produto de três matrizes (equação 5.13).

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{T} \tag{5.13}$$

onde $\mathbf{A}$ é uma matriz $m \times n$, $\mathbf{U}$ é uma matriz ortogonal $m \times n$, $\mathbf{S}$ é uma matriz diagonal $n \times n$ e $\mathbf{V}$ é uma matriz ortogonal $n \times n$. A matriz A é obtida através da equação 5.14.

$$\mathbf{A} = \mathbf{F}^{L}\left(\mathbf{F}^{C}\right)^{T} \tag{5.14}$$

onde $\mathbf{F}^L = \left[\mathbf{n}_1^L, \mathbf{d}_{11}^L, \cdots, \mathbf{d}_{14}^L, \cdots, \mathbf{n}_N^L, \mathbf{d}_{N1}^L, \cdots, \mathbf{d}_{N4}^L\right]$ e
$\mathbf{F}^C = \left[\mathbf{n}_1^C, \mathbf{d}_{11}^C, \cdots, \mathbf{d}_{14}^C, \cdots, \mathbf{n}_N^C, \mathbf{d}_{N1}^C, \cdots, \mathbf{d}_{N4}^C\right]$.

Após aplicar o método SVD, $\tilde{\mathbf{R}}_L^C$ é obtida pela equação 5.15, de acordo com [57].

$$\tilde{\mathbf{R}}_L^C = \mathbf{V}\mathbf{U}^T \tag{5.15}$$

Dada a estimação de $\mathbf{R}_L^C$, a translação $\mathbf{t}_L^C$ pode ser determinada pelas restrições 5.9 e 5.11. Como os pontos da borda do *checkerboard* são menos do que os pontos do plano, estes apresentam um ruído superior. Para evitar o *bias*, é utilizado o centróide do plano $\bar{\mathbf{P}}_i^L$ e os centróides das linhas $\bar{\mathbf{G}}_{ij}^L$ para estimar $\mathbf{t}_L^C$. Usando as equações 5.9 e 5.11, e a definição de $\mathbf{D}_{ij}$ mencionada anteriormente, obtemos o sistema linear dado pela equação 5.16.

$$\begin{aligned} \mathbf{n}_i^C \cdot \mathbf{t}_L^C &= -\mathbf{n}_i^C \cdot \mathbf{R}_L^C \bar{\mathbf{P}}_i^L - d_i^C \\ \mathbf{D}_{ij}\mathbf{t}_L^C &= -\mathbf{D}_{ij}\left(\mathbf{R}_L^C \bar{\mathbf{G}}_{ij}^L - \bar{\mathbf{P}}_{ij}^C\right) \end{aligned} \tag{5.16}$$

Depois de obter as estimações iniciais de $\tilde{\mathbf{R}}_L^C$ e $\tilde{\mathbf{t}}_L^C$, estas são otimizadas minimizando a função de custo dada por 5.17. Para resolver este problema de otimização não linear é utilizado o método *Levenberg-Marquardt* (LM) [58].

$$\begin{aligned} \left(\hat{\mathbf{R}}_L^C, \hat{\mathbf{t}}_L^C\right) = \arg\min_{\mathbf{R}_L^C, \mathbf{t}_L^C} &\sum_{i=1}^{N} \frac{1}{N_i} \sum_{m=1}^{N_i} \left\| \mathbf{n}_i^C \cdot \left(\mathbf{R}_L^C \mathbf{P}_{im}^L + \mathbf{t}_L^C\right) + d_i^C \right\|^2 \\ &+ \sum_{i=1}^{N} \sum_{j=1}^{4} \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} \left\| \mathbf{D}_{ij}\left(\mathbf{R}_L^C \mathbf{G}_{ijk}^L - \mathbf{P}_{ij}^C + \mathbf{t}_L^C\right) \right\|^2 \end{aligned} \tag{5.17}$$

Em aplicações onde apenas a correspondência entre o ponto do *laser* e o pixel é precisa, a distância euclidiana entre os dois sensores não é necessária. Sob esta condição, a transformação de similaridade pode substituir a transformação rígida, simplificando o processo de calibração, pois não é necessário medir o tamanho físico do alvo [6]. Além disso, existe um fator de escala s que transforma os dados do LiDAR em medidas físicas [59].

O cálculo de $\mathbf{R}_L^C$ para a transformação de similaridade é o mesmo para a transformação rígida. Para obter s e $\mathbf{t}_L^C$, as restrições 5.9 e 5.11 foram alteradas, obtendo-se o sistema linear representado na equação 5.18.

$$\begin{aligned} \mathbf{n}_i^C \cdot \mathbf{t}_L^C + \mathbf{n}_i^C \mathbf{R}_L^C \bar{\mathbf{P}}_i^L s &= -d_i^C \\ \mathbf{D}_{ij}\mathbf{t}_L^C + \mathbf{D}_{ij}\mathbf{R}_L^C \bar{\mathbf{G}}_{ij}^L s &= \mathbf{D}_{ij}\mathbf{P}_{ij}^C \end{aligned} \tag{5.18}$$

Após as estimações iniciais de $\tilde{s}$ $\tilde{\mathbf{R}}_L^C$ e $\tilde{\mathbf{t}}_L^C$ terem sido obtidas, estas são otimizadas

minimizando a função de custo dada pela equação 5.19.

$$\begin{aligned}\left(\hat{s}, \hat{\mathbf{R}}_L^C, \hat{\mathbf{t}}_L^C\right) = \arg \min_{s, \mathbf{R}_L^C, \mathbf{t}_L^C} & \sum_{i=1}^{N} \frac{1}{N_i} \sum_{m=1}^{N_i} \left\| \mathbf{n}_i^C \cdot \left(s\mathbf{R}_L^C \mathbf{P}_{im}^L + \mathbf{t}_L^C\right) + d_i^C \right\|^2 \\ & + \sum_{i=1}^{N} \sum_{j=1}^{4} \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} \left\| \mathbf{D}_{ij} \left(s\mathbf{R}_L^C \mathbf{G}_{ijk}^L - \mathbf{P}_{ij}^C + \mathbf{t}_L^C\right) \right\|^2\end{aligned} \tag{5.19}$$

O algoritmo da calibração dos parâmetros extrínsecos entre um LiDAR 3D e uma câmara encontra-se resumido no algoritmo 3.

Algorithm 3 Calibração de extrínsecos utilizando correspondências entre planos e linhas de um *checkerboard*

Input: N *poses* do alvo

Output: Transformação rígida ou transformação de similaridade do LiDAR para a câmara

1: Deteção do *checkerboard* e das suas linhas no plano da imagem. Cálculo do plano 3D π_i^C e das linhas 3D L_{ij}^C $(j = 1, 2, 3, 4)$ no referencial da câmara.

2: Especificar uma posição aproximada do alvo. Deteção do *checkerboard* através do RANSAC. Obtenção dos parâmetros do plano pi_i^L e das linhas L_{ij}^L $(j = 1, 2, 3, 4)$.

3: Utilizar a equação 5.15 e 5.16 para obter $\left(\tilde{\mathbf{R}}_L^C, \tilde{\mathbf{t}}_L^C\right)$ ou usar a equação 5.15 e 5.18 para obter $\left(\tilde{s}, \tilde{\mathbf{R}}_L^C, \tilde{\mathbf{t}}_L^C\right)$.

4: Aperfeiçoar a solução minimizando 5.17 para obter $\left(\hat{\mathbf{R}}_L^C, \hat{\mathbf{t}}_L^C\right)$ para a transformação rígida ou minimizar 5.19 para obter $\left(\hat{s}, \hat{\mathbf{R}}_L^C, \hat{\mathbf{t}}_L^C\right)$ para a transformação de similaridade.

5.2 Colorização da Point Cloud

Após a calibração dos parâmetros extrínsecos entre um LiDAR e uma câmara, é possível transformar os pontos 3D do *laser* no sistema de coordenadas da câmara.

Assumindo que o LiDAR e a câmara partilham o mesmo eixo principal, os objetos medidos pelo *laser* podem ser visíveis pela câmara [14]. Assim sendo, a cor de cada ponto pode ser obtida a partir dos *scans* mais próximos do LiDAR e da imagem obtida pela câmara. Para tal, é importante que o LiDAR e a câmara estejam sincronizados. Isto é relativamente simples se ambos os sensores partilharem o mesmo relógio e/ou *trigger*.

A conversão dos pontos do LiDAR no referencial da câmara pode ser obtida pela equação 5.20.

$$\mathbf{P}^C = \mathbf{R}_L^C \mathbf{P}^L + \mathbf{t}_L^C \tag{5.20}$$

Geralmente, um LiDAR 3D tem um campo de visão horizontal bastante superior em relação a uma câmara de projeção em perspetiva. Como tal, os pontos do laser que se encontram fora do FoV devem ser descartados. A condição dada pela equação 5.21 elimina todos os pontos provenientes do LiDAR que se encontram atrás da câmara. Esta etapa reduz amplamente o número de pontos oriundos do LiDAR.

$$Z^C > 0 \tag{5.21}$$

Os pontos no referencial da câmara podem ser convertidos para o plano da imagem (equação 5.22), utilizando a matriz de intrínsecos $\mathbf{A}$. Possivelmente, alguns pontos serão descartados pela condição 5.23, uma vez que estes se encontram fora do plano da imagem. O tamanho da imagem é igual a $[w, h]$.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x^C \\ y^C \\ z^C \end{bmatrix} \tag{5.22}$$

$$\begin{aligned} 0 \leq u < w \\ 0 \leq v < h \end{aligned} \tag{5.23}$$

Assim que as coordenadas pixeis do laser sejam obtidas no plano da imagem, é necessário remover a distorção da imagem. Em seguida, cor dos pontos 3D do LiDAR é dada pelos valores RGB de cada pixel. O algoritmo encontra-se sintetizado no algoritmo 4.

Algorithm 4 Colorização da *point cloud*

Input: *Point cloud*, imagem, transformação entre os dois sensores, parâmetros intrínsecos

Output: *Color point cloud*

1: Transformação dos pontos medidos pelo LiDAR no sistema de coordenadas da câmara, utilizando os parâmetros extrínsecos entre os dois sensores.
2: Descartar todos os pontos que não respeitem a condição 5.21.
3: Conversão dos pontos no referencial da câmara para o plano da imagem, usando a equação 5.22.
4: Eliminar os pixeis que se encontram fora do plano da imagem, através da condição 5.23.
5: Remover a distorção da imagem.
6: Atribuição da cor de cada ponto, sendo esta dada pelos valores RGB de cada pixel no plano da imagem.

5.3 Reconstrução 3D

Com o objetivo de realizar a reconstrução 3D, é necessário determinar a *pose* do UAV. A localização do robô no mundo permitirá juntar as várias *point clouds* coloridas, obtendo-se uma nuvem de pontos colorida geo-referenciada do ambiente que o UAV mapeou. Para tal, é preciso analisar todos os sistemas de coordenadas envolvidos nos diferentes componentes do sistema do robô. Na figura 5.6, encontra-se representado o sistema de coordenadas do LiDAR, bem como o sistema de coordenadas da câmera, do mundo e do *body* (robô).

As coordenadas do mundo são representadas num referencial local, também designado por *Local Tangent Plane* (LTP). Este referencial consiste numa superfície plana tangencial num ponto da Terra, onde a navegação do robô decorre. O referencial local utilizado é *East-North-Up* (ENU), na qual o eixo x aponta para Este, y para Norte e z para cima. A origem do referencial pode ser estabelecida em qualquer ponto do globo terrestre.

Após conhecer todos os referenciais envolvidos, é possível converter os pontos do LiDAR no referencial do *body*. A equação 5.24 transforma os pontos do laser no *body frame*.

$$\mathbf{P}^b = \mathbf{R}_L^b \mathbf{P}^L + \mathbf{t}_L^b \tag{5.24}$$

onde $\mathbf{P}^b$ é o ponto do LiDAR no referencial do UAV, $\mathbf{R}_L^b$ é a matriz rotação do

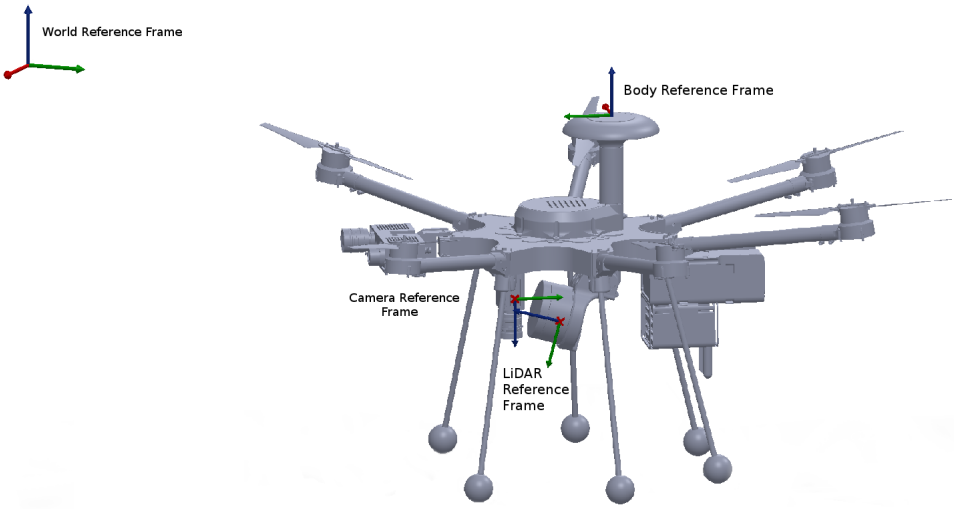


Figura 5.6: STORK I UAV. Sistema de coordenadas do *body*, LiDAR, câmara e mundo. O eixo x está representado a vermelho, y a verde e z a azul.

referencial do LiDAR para o *body*, $\mathbf{P}^L$ é o ponto do LiDAR e $\mathbf{t}_L^b$ é a translação entre o *laser* e o *body*.

A conversão dos pontos do referencial do robô para o referencial do mundo é dada pela posição e orientação do UAV em relação à origem definida do referencial local. A matriz $\mathbf{R}_b^w$ é calculada a partir dos ângulos de *Euler* (ϕ, θ, ψ), como representado na equação 3.11. A transformação dos pontos do *body* no mundo é dada pela equação 5.25.

$$\mathbf{P}^w = \mathbf{R}_b^w \mathbf{P}^b + \mathbf{t}_b^w \tag{5.25}$$

As equações 5.24 e 5.25 podem ser simplificadas, obtendo-se a equação 5.27. Para tal, são utilizadas transformações homogéneas entre todos os sistemas de coordenadas de referência. A matriz de transformação homogénea $\mathbf{T}$ é dada pela equação 5.26 [60].

$$\mathbf{T}_{4\times4} = \begin{bmatrix} & \mathbf{R}_{3\times3} & & \mathbf{t}_{3\times1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.26}$$

Onde $\mathbf{R}$ e $\mathbf{t}$ é a matriz rotação e o vetor de translação entre dois sistemas de coordenadas, respetivamente.

$$\begin{bmatrix} \mathbf{P}^W \\ 1 \end{bmatrix} = \mathbf{T}_b^W \mathbf{T}_L^b \begin{bmatrix} \mathbf{P}^L \\ 1 \end{bmatrix} \tag{5.27}$$

Capítulo 6

Implementação

Este capítulo encontra-se dividido em várias partes, nomeadamente: descrição das características do UAV usado e dos respetivos sensores, *softwares* utilizados na implementação do algoritmo descrito na secção anterior, calibração dos parâmetros intrínsecos da câmara e, por último, uma explicação do *software* desenvolvido. O *software* foi desenvolvido em C++, utilizando a *framework* ROS e as bibliotecas *Open Source Computer Vision Library* (OpenCV) e *Point Cloud Library* (PCL).

6.1 STORK UAV

O STORK (figura 6.1) é um UAV *hexacopter* desenvolvido pelo LSA e INESC TEC. A sua aplicação principal é a inspeção de linhas elétricas. Contudo, tem sido usado em outras aplicações, tais como operações de busca e salvamento, inspeções do meio ambiente, mapeamento 3D, vigilância e patrulha. Esta diversidade de aplicações é possível devido à capacidade adaptativa da configuração de *payload*, na qual utilizando a mesma *frame*, vários sensores podem ser substituídos por outros.

Este UAV pode navegar autonomamente ou manualmente, sendo também capaz de realizar algumas manobras de forma autónoma, desde descolagem, aterragem ou inspeção de uma estrutura. Este é controlado em termos de *low-level* por um *autopilot* desenvolvido pelo INESC TEC e em termos de *high-level* é controlado por um computador de bordo.

O UAV tem vários tipos de sensores acoplados, nomeadamente sensores de navegação e perceção. Para a navegação, o robô tem dois recetores GNSS e dois IMUs. O *autopilot* por *default* tem sensores *low-cost* e, portanto, para realizar mapeamento preciso, é necessário possuir recetores GNSS e IMUs com uma *performance* elevada. Para este fim,

Figura 6.1: Stork UAV.

foi integrado no veículo o STIM300, um IMU compacto de alta precisão, e K501G, um recetor de frequência dupla que suporta posicionamento *Real-Time Kinematic* (RTK).

Em termos de perceção do ambiente, o STORK tem um LiDAR 3D (Velodyne VLP-16), duas câmaras de espectro vísivel (Teledyne Dalsa G3-GC10-C2450 que está fixa, a apontar para baixo, e FLIR Point-Grey Chameleon3 que está conectada a um *gimbal*) e, em algumas aplicações, também pode ser utilizada uma câmara termográfica. Na figura 6.2 está ilustrado o LiDAR e a câmara integrada no Stork UAV.

Figura 6.2: LiDAR e câmara acoplada no Stork UAV.

6.1.1 Dalsa Genie Nano

A câmara utilizada é a Teledyne Dalsa G3-GC10-C2450 (figura 6.3), dado que cumpre os requisitos, tais como: vários modos de configuração, capacidade de receção de *trigger* externo, utilização do protocolo *Gigabit Ethernet* (GigE), *frame rate* elevado e *global shutter*. As principais características da câmara encontram-se apresentadas na tabela 6.1.

Figura 6.3: Teledyne Dalsa Nano C2450 Color [61].

Tabela 6.1: Características da Teledyne Dalsa Nano C2450 Color.

Resolução	2464 x 2056 (5.1 MP)
Taxa de Aquisição	23.8 fps (TurboDrive 52 fps)
Lentes Suportadas	C-Mount, CS - Mount
Tipo de Shutter	Global Shutter
Conectividade	GigE
Tensão de Alimentação	PoE ou 10-36 VDC
Peso	46 g
Dimensões	21 x 29 x 44 mm

6.1.2 Velodyne VLP-16

O Velodyne VLP-16 (figura 6.4) é um *spinning* LiDAR (explicado na secção 3.2.1) que produz *point clouds* 3D, utilizando 16 feixes *laser* montados numa carcaça que roda entre 5 a 20 vezes por segundo. Além disso, este sensor pode fornecer até 300,000 pontos por segundo em *single mode* ou o dobro em *dual mode* [62].

As características relevantes do Velodyne VLP-16 encontram-se expostas na tabela 6.2.

Figura 6.4: Velodyne VLP-16 [62].

Tabela 6.2: Características do Velodyne VLP-16.

Alcance Máximo	100 m
Field of View (Vertical)	-15.0°a 15.0°(30°)
Resolução Angular (Vertical)	2.0°
Field of View (Horizontal)	360°
Resolução Angular (Horizontal)	0.1°- 0.4°
Precisão de Alcance	± 3cm
Frequência	5 Hz - 20 Hz

O VLP-16 pode sincronizar o seu *timestamp* com a hora UTC, sendo útil para sincronizar os seus dados com o restante sistema. Para sincronizar, é necessário o uso de um recetor GNSS externo que gere um sinal PPS e uma mensagem *National Marine Electronics Association* (NMEA) GPRMC.

6.1.2.1 Software driver

O *package* ROS *velodyne* está dividido em vários sub-*packages* que permitem estabelecer uma conexão com vários sensores da Velodyne (entre eles, o modelo VLP-16). Estes pacotes recebem os dados do sensor e criam uma *point cloud* com eles. Os três sub-pacotes principais são o *velodyne_msgs*, *velodyne_driver* e *velodyne_pointcloud*.

No *velodyne_msgs* são colocadas as definições do tipo de mensagem ROS para os LiDARs Velodyne. O pacote *velodyne_driver* é responsável por estabelecer a comunicação com o sensor e publica os pacotes recebidos num tópico ROS (*/velodyne_packets*). Este tópico é subscrito pelo *package velodyne_pointcloud* e os pacotes são convertidos numa nuvem de pontos 3D, sendo publicados no tópico */velodyne_points*.

O pacote *velodyne_pointcloud* publica uma nuvem de pontos esparsa contendo apenas

os pontos válidos, isto é, pontos que estão dentro de um alcance e de um intervalo angular definido.

Contudo, o referencial da nuvem de pontos usado pelo *nodelet* não corresponde ao fornecido pelo fabricante no manual do sensor [45]. No referencial *default*, y_{def} aponta para a frente, x_{def} para a direita e z_{def} para cima, mas no usado pelo *nodelet*, x_{nod} aponta para a frente, y_{nod} para a esquerda e z_{nod} para cima, representando uma rotação de $\pi/2$ em torno do eixo z.

Desta forma, este pacote foi alterado de modo a que o referencial da *point cloud* esteja de acordo com o definido pelo fabricante.

6.2 Bibliotecas utilizadas

6.2.1 PCL

A biblioteca PCL [63] é utilizada para o processamento da nuvem de pontos do LiDAR, já que esta contém funções para a segmentação, processamento e estimação de parâmetros de modelos matemáticos, tais como planos, linhas, entre outros. Além disso, a implementação é em linguagem C++ e é compatível com o ROS.

O PCL tem o algoritmo RANSAC implementado, bem como os modelos matemáticos do plano e da linha integrados. A utilização do RANSAC requer uma configuração inicial de alguns parâmetros, desde o tipo de modelo, o *threshold* de distância e o número de iterações.

O utilizador pode criar um novo modelo matemático ou utilizar um já implementado, existindo várias variantes do RANSAC. O valor do *threshold* de distância é a distância máxima para que um ponto pertença ao modelo matemático definido. Por vezes, algumas superfícies observadas pelo LiDAR apresentam alguma irregularidade e, portanto, este parâmetro deve ser definido com um valor próximo dessa irregularidade. O número de iterações se for demasiado elevado pode aumentar o tempo de processamento do algoritmo, mas, ao mesmo tempo, poderão ser obtidos melhores resultados.

Além dos algoritmos referidos, esta biblioteca contém vários algoritmos de reconstrução de superfícies, deteção de *features*, *point cloud registration*, entre outros.

6.2.2 OpenCV

O OpenCV [64] é uma biblioteca de *software open source* de visão computacional e de *machine learning*. Esta biblioteca tem milhares de algoritmos otimizados que podem ser usados para detetar e reconhecer rostos, identificar objetos, extrair modelos 3D de

objetos, produzir nuvens de pontos 3D a partir de um par stereo, etc. Por outro lado, está implementada em C++ e pode ser usada em ROS.

A transformada de Hough (secção 3.6) está implementada no OpenCV, assim como a deteção e obtenção da *pose* de um objeto (neste caso o *checkerboard*). A implementação da transformada de Hough é composta pela transformada clássica e probabilística, sendo que ambas têm como *input* uma imagem binária, bem como os parâmetros apresentadas na secção 3.6. Para detetar o *checkerboard*, é necessário definir o número de cantos internos do alvo. A *pose* entre o alvo e a câmara é obtida através da correspondência de pontos 3D-2D.

Estas funções foram utilizadas para a obtenção das *features* descritas na secção 5.1.2, de modo a calibrar os parâmetros extrínsecos entre o LiDAR e a câmara. Além disso, esta biblioteca foi também utilizada na implementação do método de colorização de uma *point cloud* (secção 5.2).

6.3 Calibração dos parâmetros intrínsecos da câmara

Para a calibração dos parâmetros intrínsecos da câmara (enunciados na secção 3.1.2), foi utilizada a aplicação *Camera Calibrator* do *MATrix LABoratory* (MATLAB) [65]. Para tal, é necessário a utilização de um *checkerboard* (figura 5.2), com dimensões conhecidas. Estes parâmetros são fundamentais para qualquer método que utilize processamento de imagem.

A *toolbox* do MATLAB (figura 6.5) recebe um conjunto de imagens com um alvo e o comprimento da aresta dos quadrados. Após isto, os cantos do alvo são detetados e os parâmetros intrínsecos da câmara são estimados. Estes parâmetros são obtidos através de processos de otimização, na qual o erro da reprojeção dos pontos 3D do alvo nas imagens e os cantos detetados pela *toolbox* são minimizados.

Caso o erro de reprojeção dos pontos numa imagem seja elevado, é possível remover a imagem e os parâmetros intrínsecos são recalculados, com o objetivo de obter o melhor resultado possível. Após esta fase, a matriz de intrínsecos (equação 3.5) e os parâmetros de distorção (equação 3.6) são obtidos.

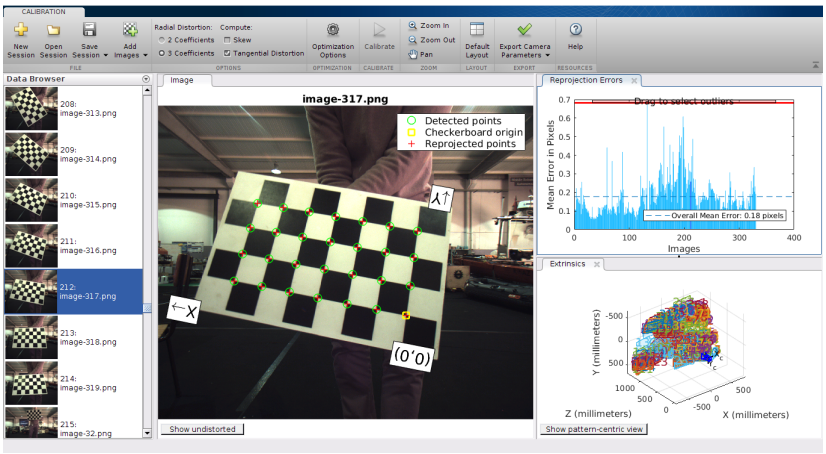


Figura 6.5: *Toolbox* do MATLAB - *Camera Calibrator*.

6.4 Software desenvolvido

6.4.1 Calibração dos parâmetros extrínsecos LiDAR-câmara

Após a obtenção dos parâmetros intrínsecos da câmara, é realizada a calibração dos parâmetros extrínsecos do sistema composto pelo LiDAR e pela câmara. Este processo consiste na obtenção das *features* detalhadas no sub-capítulo 5.1.2 e na estimação da *pose* (secção 5.1.3.2), posição e orientação, entre o LiDAR e a câmara.

Para tal, foi desenvolvido um *package* ROS, *lidar_camera_extrinsics*, em C++, utilizando a biblioteca OpenCV e PCL. Este *package* é constituído por três *nodes*, na qual o primeiro está incumbido de obter as *features* da *point cloud* e da imagem. O segundo é responsável por estimar os parâmetros extrínsecos. Por fim, o último *node* projeta os pontos do LiDAR no plano da imagem, utilizando os parâmetros extrínsecos estimados. Na figura 6.6 encontra-se representado o diagrama do processo de calibração.

O nó */feature_extraction* subscreve-se ao tópico da imagem e da *point cloud*. O driver da câmara publica num tópico uma mensagem do tipo *sensor_msgs/Image* que será convertida numa imagem, utilizando o OpenCV. Da mesma forma, o *nodelet* do *package* */velodyne_pointcloud* publica a nuvem de pontos numa mensagem do tipo *sensor_msgs/PointCloud2* e, para usar a biblioteca PCL, é necessário convertê-la numa *point cloud* PCL.

Assim que as features sejam obtidas em ambos os referencias, o *node* publica-as no tópico */feature_information*.

O tópico */feature_information* é subscrito pelo *node* */compute_extrinsics*. Este nó

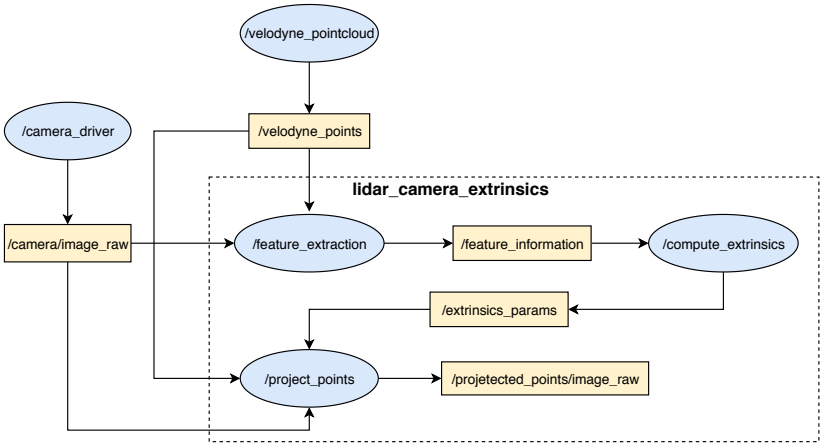


Figura 6.6: Diagrama do processo de calibração dos parâmetros extrínsecos. As elipses azuis representam os nós e os retângulos amarelos os tópicos.

estima os parâmetros extrínsecos, usando o algoritmo explicado na secção 5.1.3.2. A biblioteca Eigen [66] é utilizada para o cálculo numérico. As *features* são armazenadas em memória, dado que o número de *poses* pretendidas para a estimação pode ser superior a uma. Assim que o número de *poses* armazenadas seja igual ao pretendido, os parâmetros extrínsecos são estimados e publicados no tópico */extrinsics_params*. É possível escolher o tipo de transformação pretendida a ser estimada (rígida ou de similaridade), através de um parâmetro do *node*.

Por último, os parâmetros extrínsecos publicados no tópico */extrinsics_params* são utilizados pelo nó */project_points* para projetar os pontos do *laser* no plano da imagem.

6.4.2 Pintura da nuvem de pontos

O processo de colorização da *point cloud* é realizado em *real-time*, usando a *framework* ROS e as bibliotecas OpenCV e PCL. Uma vez obtida a transformação entre os dois sensores, LiDAR e câmara, bem como os parâmetros intrínsecos da câmara, é possível converter os pontos 3D do LiDAR no referencial da câmara e, posteriormente, projetá-los no plano da imagem.

O *package* desenvolvido é constituído por apenas um *node*, sendo este responsável pela associação da cor da nuvem de pontos proveniente do LiDAR. Para tal, subscreve-se ao tópico da nuvem de pontos e da imagem. Os pontos do *laser* são convertidos para o referencial da câmara, através dos parâmetros extrínsecos.

A condição dada pela equação 5.21 é testada para estes pontos e apenas os pontos

que passam esta condição são projetados na imagem. Os pixeis que se encontram fora do plano da imagem são eliminados, dado que não cumprem a condição 5.23.

A associação da cor dos pontos do LiDAR que são visíveis para a câmara é realizada e, posteriormente, apenas a *point cloud* colorida é publicada no tópico */pointcloud_colorized*. O diagrama do processo de colorização da *point cloud* é ilustrado na figura 6.7.

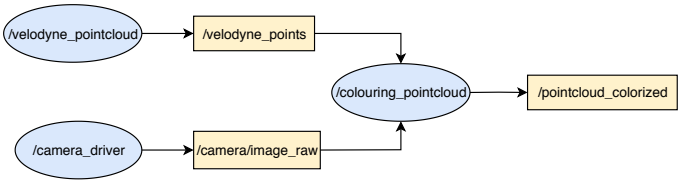


Figura 6.7: Diagrama do processo de colorização da *point cloud*. As elipses azuis representam os nós e os retângulos amarelos os tópicos.

6.4.3 Reconstrução 3D

O processo de reconstrução 3D foi efetuado em *offline*, já que a qualidade da navegação estimada pelo *autopilot* do UAV não apresenta a exatidão pretendida (devido ao referido na secção 6.1). Desta forma, a *pose* do UAV foi estimada em pós-processamento de modo a obter-se os melhores resultados possíveis.

Para realizar o processo descrito na secção 5.3, foi desenvolvido um programa em C++ que tem como *input* a *pose* do UAV, todos os *scans* do LiDAR e a rotação e translação entre o LiDAR e o referencial *body*. A posição e orientação do robô são armazenadas em memória, assim como o instante de tempo de cada *pose*. Este *timestamp* será usado para sincronizar os dados da *pose* e da *point cloud*.

Esta página foi intencionalmente deixada em branco.

Capítulo 7

Resultados

Neste capítulo são apresentados os resultados obtidos, sendo primeiro apresentados os resultados relativos da calibração dos parâmetros extrínsecos entre LiDAR-câmara e, posteriormente, os resultados referentes à reconstrução 3D, através do algoritmo descrito nas secções 5.2 e 5.3.

7.1 Calibração

Para a validação do algoritmo de calibração, foi utilizado o *setup* experimental ilustrado na figura 7.1, sendo este constituído pelo Velodyne VLP-16 e duas câmaras.

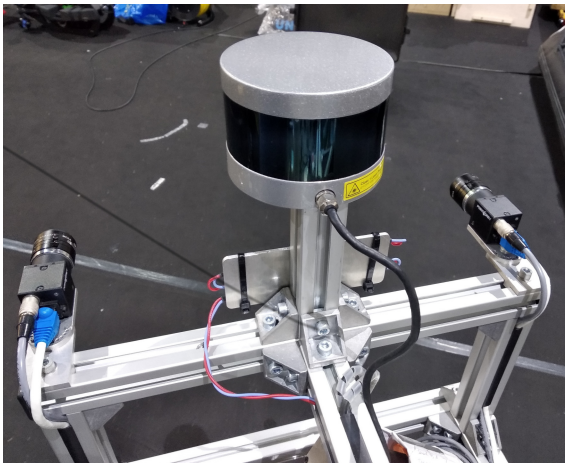


Figura 7.1: *Setup* experimental de calibração.

Os parâmetros extrínsecos entre as duas câmaras ($\mathbf{R}_{C_l}^{C_r}, \mathbf{t}_{C_l}^{C_r}$) foram utilizados como

ground truth. Estes parâmetros foram comparados com os obtidos entre o LiDAR e cada câmara, ou seja, foram estimados os parâmetros extrínsecos entre o LiDAR e a câmara esquerda ($\hat{\mathbf{R}}_L^{C_l}, \hat{\mathbf{t}}_L^{C_l}$) e entre o LiDAR e a câmara direita ($\hat{\mathbf{R}}_L^{C_r}, \hat{\mathbf{t}}_L^{C_r}$).

Os parâmetros intrínsecos de cada câmara e os extrínsecos foram obtidos pela aplicação *Stereo Camera Calibrator* do MATLAB. Nas figuras 7.2 e 7.3 são apresentados o erro de reprojeção e a *pose* do alvo em relação a ambas câmaras, respetivamente.

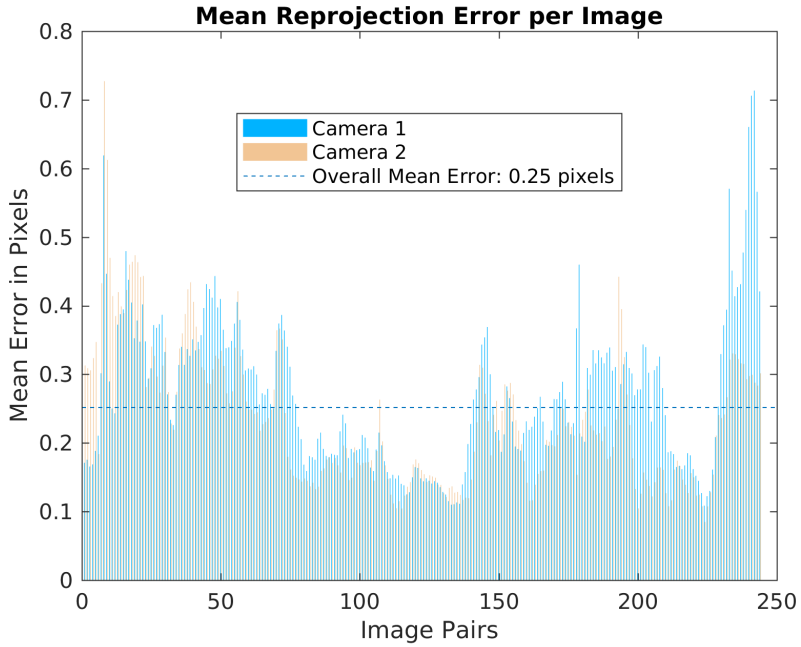


Figura 7.2: Erro médio de reprojeção em cada imagem.

Após a obtenção dos parâmetros intrínsecos, é possível dar início ao processo de calibração apresentado na secção 5.1. Para tal, foi utilizado um *checkerboard* com um número de cantos internos de 7×4, na qual a aresta de cada quadrado apresenta um comprimento de 60 mm. Este alvo foi colocado em várias *poses*, excepto com ângulos próximos de 0 e 90 graus, para se obter uma grande variedade de medições.

As *features* foram extraídas de cada imagem em ambas as câmaras, bem como no referencial do LiDAR. Os parâmetros extrínsecos foram estimados entre cada câmara e o LiDAR, a partir das *features* extraídas.

A *pose* ($\hat{\mathbf{R}}_{C_l}^{C_r}, \hat{\mathbf{t}}_{C_l}^{C_r}$) estimada entre a câmara esquerda e a direita é obtida a partir de ($\hat{\mathbf{R}}_L^{C_l}, \hat{\mathbf{t}}_L^{C_l}$) e ($\hat{\mathbf{R}}_L^{C_r}, \hat{\mathbf{t}}_L^{C_r}$). Esta *pose* ($\hat{\mathbf{R}}_{C_l}^{C_r}, \hat{\mathbf{t}}_{C_l}^{C_r}$) é comparada com a obtida pela aplicação do MATLAB ($\mathbf{R}_{C_l}^{C_r}, \mathbf{t}_{C_l}^{C_r}$). A partir da equação 7.1, é possível obter ($\hat{\mathbf{R}}_{C_l}^{C_r}, \hat{\mathbf{t}}_{C_l}^{C_r}$), utilizando

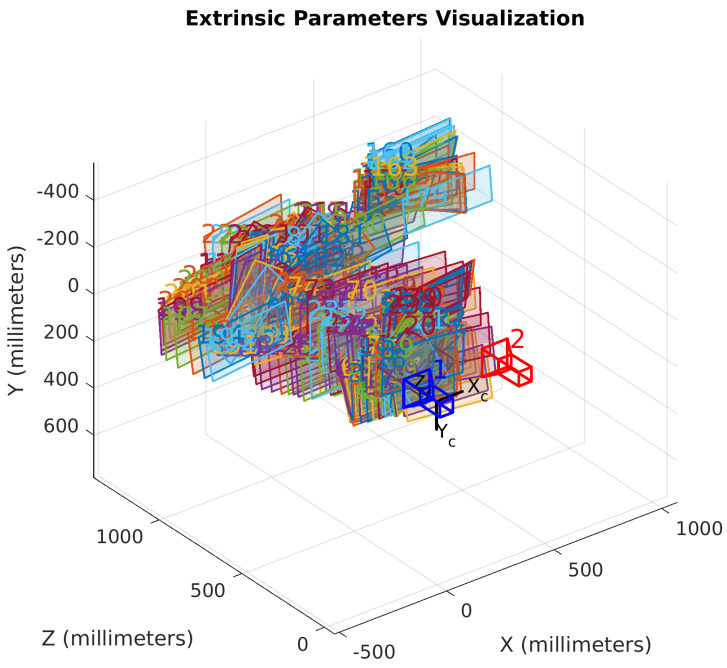


Figura 7.3: Parâmetros extrínsecos de cada imagem no referencial de ambas câmaras.

as transformações homogéneas (equação 5.27).

$$\hat{\mathbf{T}}_{C_l}^{C_r} = \hat{\mathbf{T}}_{L}^{C_r} \left(\hat{\mathbf{T}}_{L}^{C_l} \right)^{-1} \tag{7.1}$$

Os parâmetros extrínsecos entre as duas câmaras obtidos pela *toolbox* do MATLAB e estimados, encontram-se representados na tabela 7.1. O erro de rotação e de translação é calculado a partir da equação 7.2 e 7.3, respetivamente, [67].

$$e_{rotation} = \left\| \mathbf{I} - (\mathbf{R})^{-1} \hat{\mathbf{R}} \right\|_F \tag{7.2}$$

$$e_{translation} = \left\| \mathbf{t} - \hat{\mathbf{t}} \right\| \tag{7.3}$$

onde $\|.\|$ representa a norma, $\|.\|_F$ a norma Frobenius, $(\mathbf{R}, \mathbf{t})$ o *ground truth* e $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ o estimado.

A projeção dos pontos do LiDAR na imagem, usando os parâmetros obtidos da transformação rígida e de similaridade, é ilustrada na figura 7.4. A transformação de simila-

Tabela 7.1: Parâmetros extrínsecos obtidos na calibração (estimados e *ground truth*) e erros de calibração.

Parâmetros	*Ground truth*	Estimado
Roll (deg)	-0.4492	-0.0527
Pitch (deg)	-0.7390	0.3333
Yaw (deg)	0.1243	0.2543
X (mm)	-369.2	-348.4
Y (mm)	0.8	-8.0
Z (mm)	1.6	4.3
Erro de rotação (deg)	-	1.1860
Erro de translação (mm)	-	22.9

ridade apresenta melhores resultados, dado que os pontos *laser* da fronteira estão mais próximos do limite do alvo na imagem. Como o algoritmo proposto só necessita da correspondência do ponto *laser* e do pixel, será utilizada a transformação de similaridade, visto que apresenta melhores resultados.

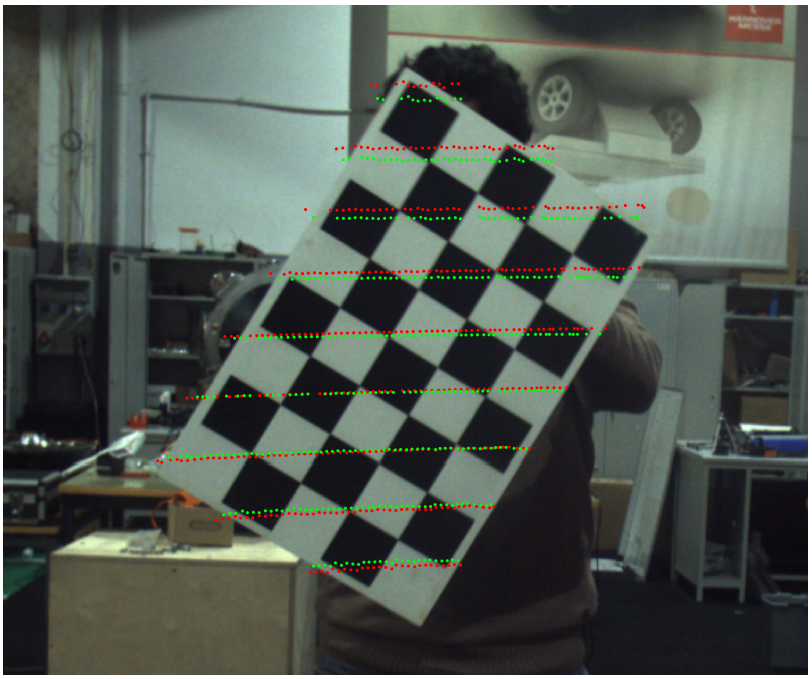


Figura 7.4: Projeção dos pontos *laser* na imagem, usando a transformação rígida (pontos vermelhos) e de similaridade (pontos verdes).

7.2 Reconstrução 3D

De modo a testar o algoritmo desenvolvido nesta dissertação, foram realizados dois *datasets*. Estes foram efetuados no Mosteiro de Tibães, Braga, Portugal, no âmbito do projeto MineHeritage.

7.2.1 *Dataset* I

Este *dataset* foi realizado no escadório do Mosteiro de Tibães (figura 7.5). O escadório é constituído por várias fontes trabalhadas (cada uma apresenta uma forma diferente) que estão intercaladas por escadas e patamares. Esta zona apresenta inúmeras árvores que dificultam a deslocação e a localização do UAV, uma vez que estas podem obstruir os sinais GNSS e a visibilidade do piloto. Apesar destas limitações, o UAV consegui mapear o escadório com os sensores *onboard* descritos anteriormente, tornando este *dataset* adequado para testar o algoritmo desenvolvido.

Figura 7.5: Escadório do Mosteiro de Tibães.

O UAV começou por mapear um campo de vegetação próximo do escadório, realizando uma linha reta para obter uma inicialização da sua localização. Após isto, este deslocou-se para o primeiro patamar do escadório e foi subindo entre as árvores até ao último patamar localizado no topo.

A trajetória do UAV encontra-se ilustrada na figura 7.6. Esta está representada num referencial ENU, na qual a origem do referencial encontra-se localizada numa estação

base em Braga. Na figura 7.7 encontra-se representada a orientação e velocidade do UAV durante o voo realizado no escadório.

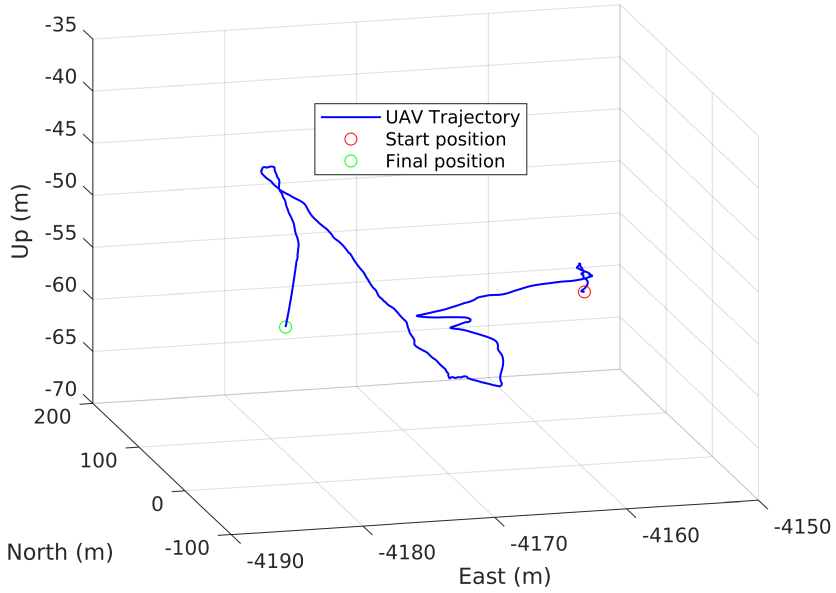


Figura 7.6: Trajetória do UAV do dataset no escadório do Mosteiro de Tibães.

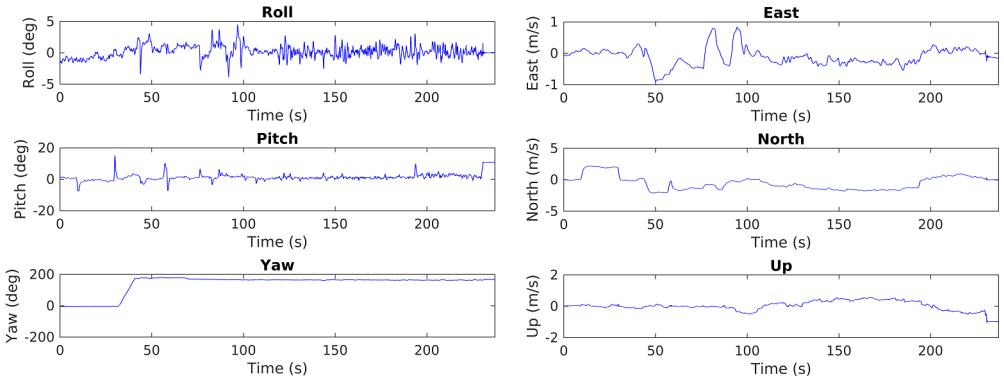


Figura 7.7: Orientação (à esquerda) e velocidade (à direita) do UAV do dataset no escadório do Mosteiro de Tibães.

A primeira fonte do escadório obtida pela câmara *onboard* do UAV encontra-se ilustrada na figura 7.8, enquanto que na figura 7.9 está representada a *point cloud raw* do LiDAR. A figura da *point cloud* do LiDAR foi capturada a partir da aplicação Rviz do

ROS.

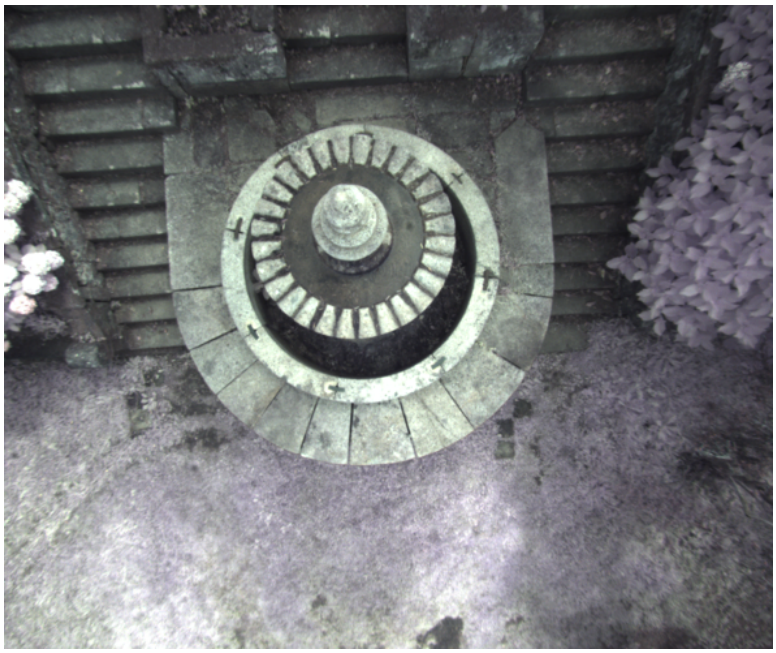


Figura 7.8: Primeira fonte do escadório de Tibães capturada pela câmara *onboard* do UAV.

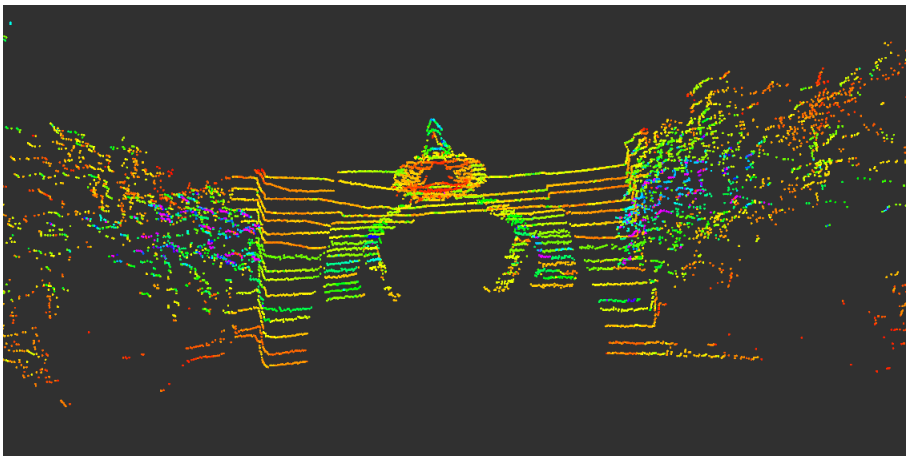


Figura 7.9: *Point cloud raw* da primeira fonte do escadório do Mosteiro de Tibães obtida pelo LiDAR. A cor de cada ponto é dada pela intensidade obtida pelo LiDAR.

Os pontos do LiDAR projetados na imagem encontram-se ilustrados na figura 7.10.

Como referido na secção 6.1.2, o LiDAR utilizado apresenta 16 feixes *laser* que podem ser visualizados na figura 7.10.

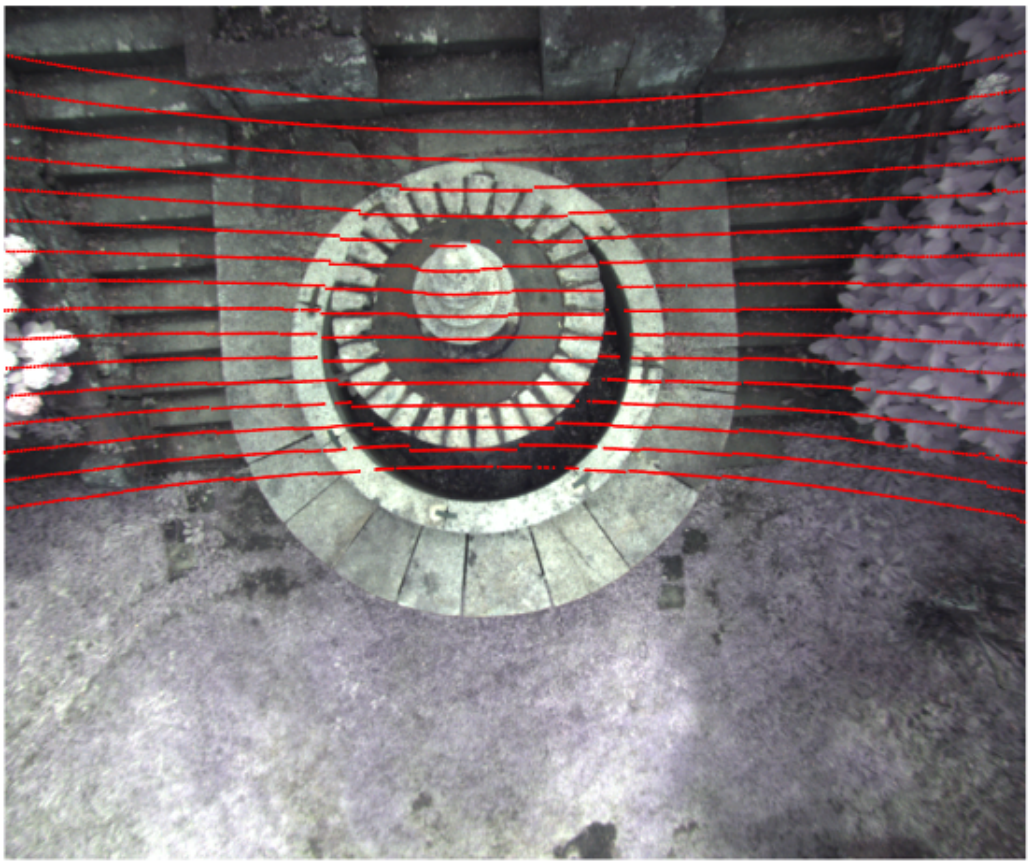


Figura 7.10: Projeção dos pontos *laser* no plano da imagem da primeira fonte do escadório.

O FoV da câmara (horizontal) é bastante inferior ao do LiDAR e, portanto, a nuvem de pontos colorida apresenta um número de pontos inferior comparativamente com a *point cloud raw*. A nuvem de pontos colorida, apenas de um *scan* do LiDAR, da primeira fonte do escadório é apresentada nas figura 7.11.

Após se efetuar o processo de colorização da nuvem de pontos (secção 5.2), procedeu-se à reconstrução 3D do escadório do Mosteiro de Tibães. Nas figuras 7.12 e 7.13 está ilustrado a primeira fonte com os vários *scans* do LiDAR, com diferentes perspectivas.

A partir destas figuras, é possível verificar que a reconstrução 3D está coerente com a morfologia real da fonte ilustrada na figura 7.8.

A reconstrução 3D do escadório com uma perspectiva lateral pode ser visualizada

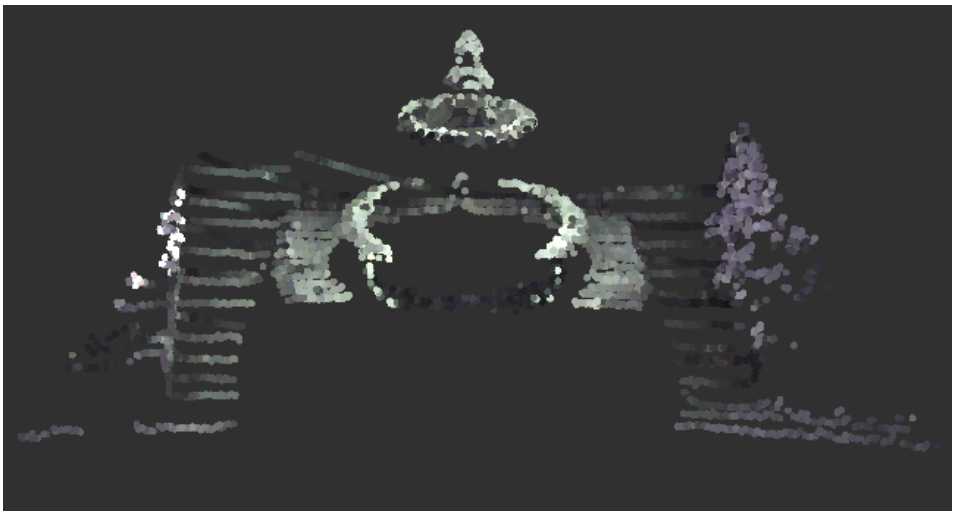


Figura 7.11: Nuvem de pontos colorida da primeira fonte do escadório do Mosteiro de Tibães com vista lateral.

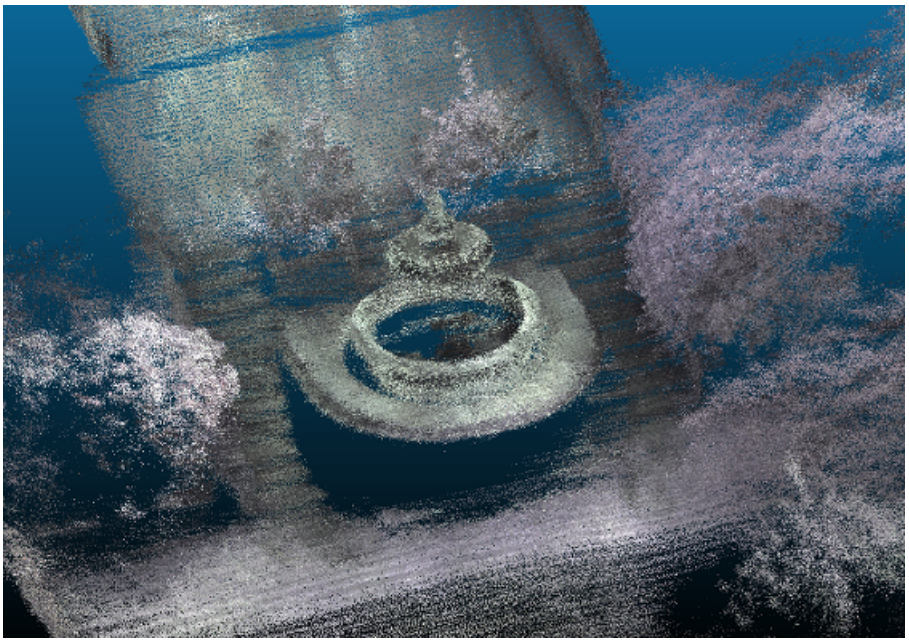


Figura 7.12: Reconstrução 3D da fonte do escadório com vista lateral.

na figura 7.14, enquanto que na figura 7.15 encontra-se ilustrado o início do escadório, na qual é possível visualizar alguns patamares e árvores. Esta *point cloud* não contém as árvores que estão presentes no escadório (figura 7.5), já que o UAV voou entre es-

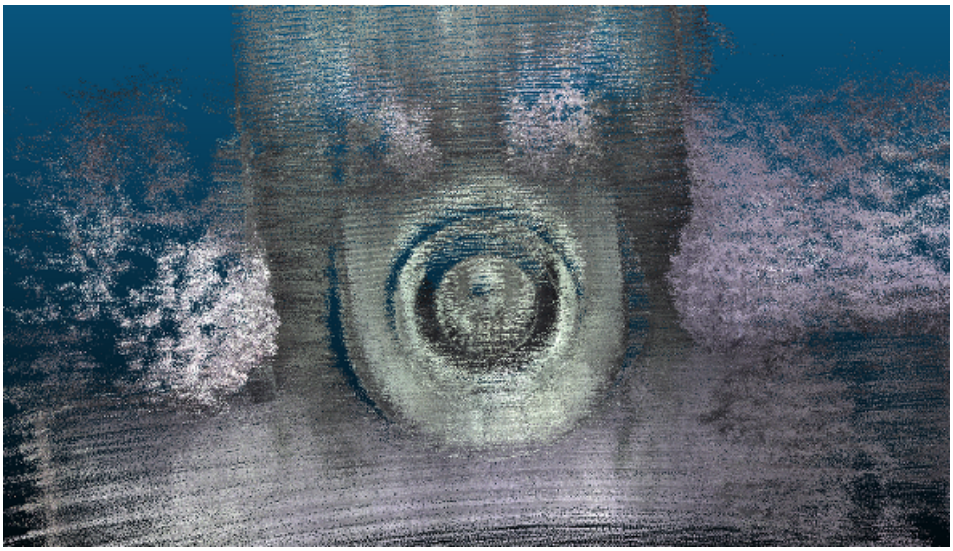


Figura 7.13: Reconstrução 3D da fonte do escadório com vista de topo.

tas. Apesar disto, comparando a figura 7.5 com a nuvem de pontos obtida, é possível constatar que a reconstrução 3D está coerente com a realidade.

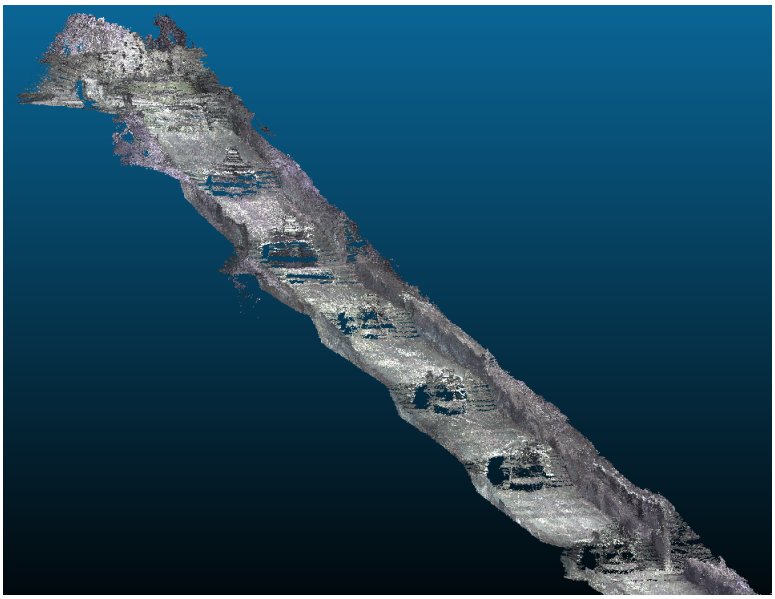


Figura 7.14: Reconstrução 3D do escadório com vista lateral.

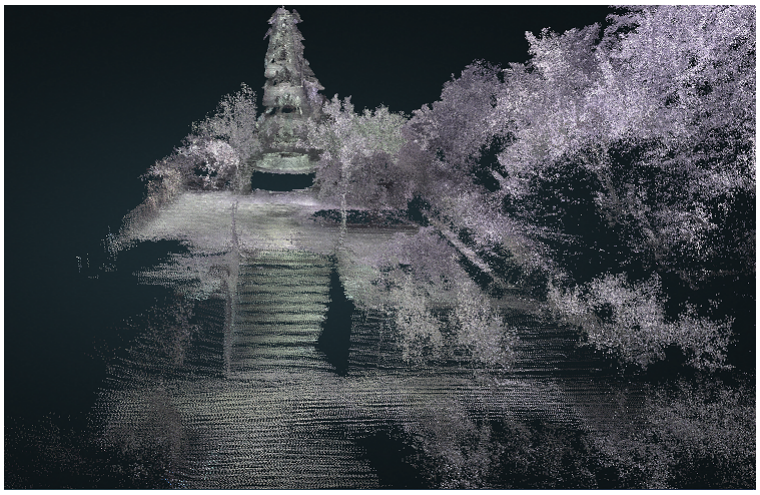


Figura 7.15: Reconstrução 3D do início do escadório.

7.2.2 *Dataset* II

O segundo *dataset* foi efetuado na cerca do Mosteiro de Tibães (figura 7.16). Esta zona é constituída maioritariamente por edifícios e vegetação. Ao contrário do escadório, esta área é ampla e, por sua vez, a localização do robô é mais precisa (os sinais GNSS são mais intensos). O procedimento neste *dataset* foi semelhante ao realizado no escadório.

Figura 7.16: Cerca do Mosteiro de Tibães.

Na figura 7.17 está ilustrada a trajetória do UAV, na qual foi utilizado a mesma origem do referencial. A orientação e a velocidade estão representadas na figura 7.18.

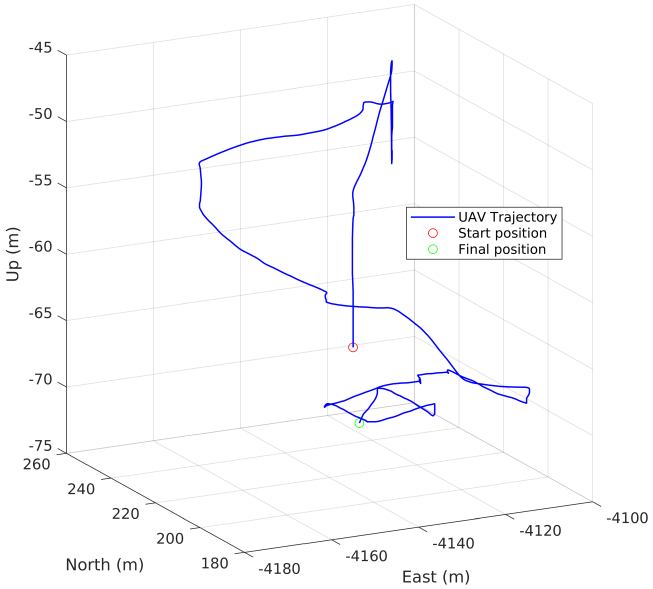


Figura 7.17: Trajetória do UAV do dataset na cerca do Mosteiro de Tibães.

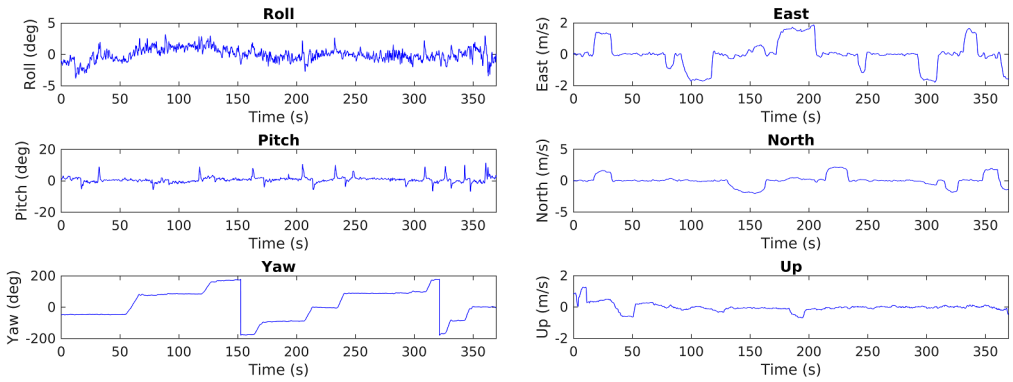


Figura 7.18: Orientação (à esquerda) e velocidade (à direita) do UAV do dataset na cerca do Mosteiro de Tibães.

Na figura 7.19 é possível observar uma parte do telhado de um edifício do Mosteiro de Tibães e alguma vegetação. A *point cloud raw* do LiDAR está ilustrada na figura

7.20.

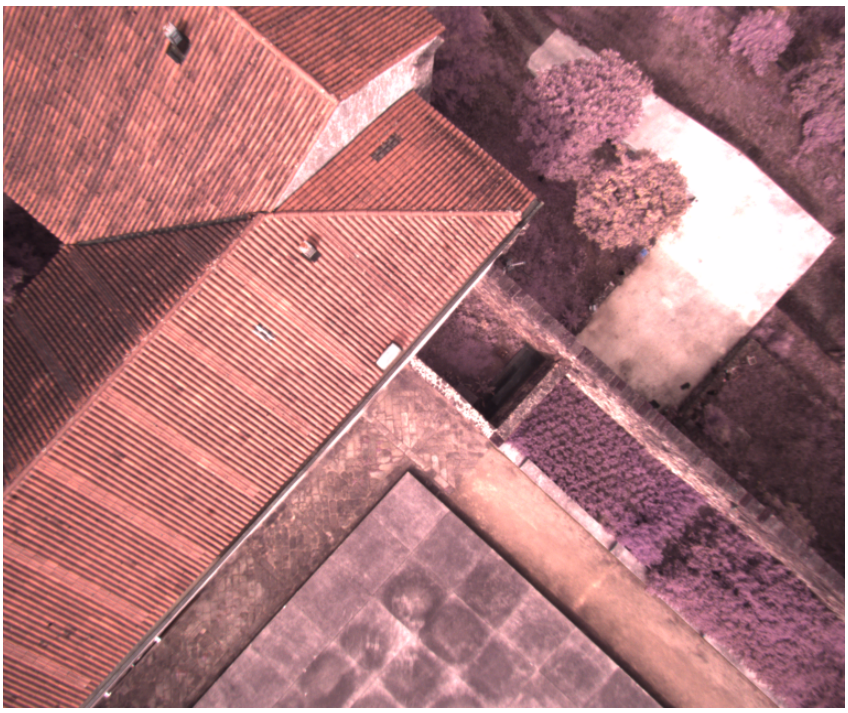


Figura 7.19: Telhado de um edifício do Mosteiro de Tibães capturado pela câmara *onboard* do UAV.

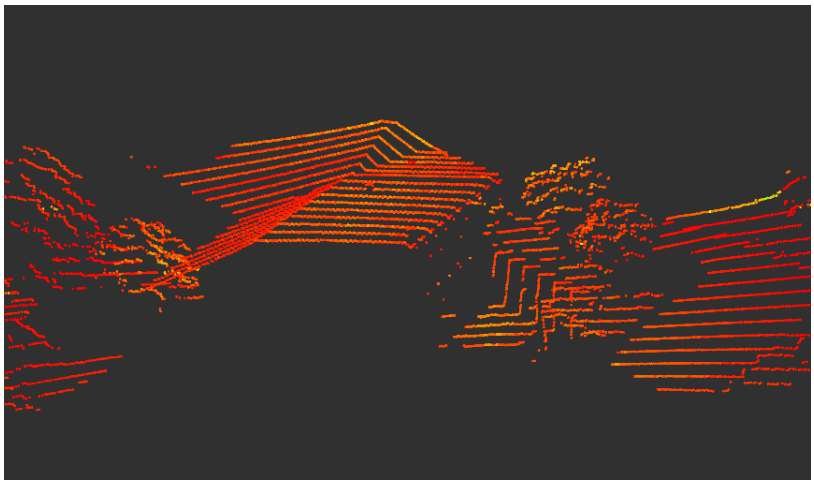


Figura 7.20: *Point cloud raw* do telhado de um edifício do Mosteiro de Tibães obtida pelo LiDAR. A cor de cada ponto é dada pela intensidade obtida pelo LiDAR.

Os pontos do *laser* foram projetados na imagem (figura 7.21), utilizando os parâmetros extrínsecos entre o LiDAR e a câmara.

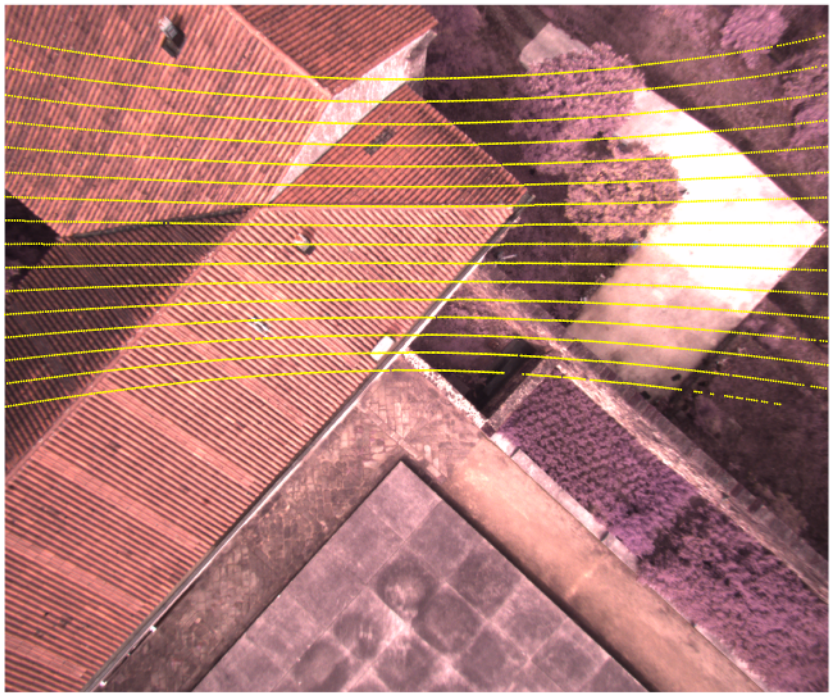


Figura 7.21: Projeção dos pontos *laser* na imagem no telhado de um edifício na cerca.

Estes pixeis na imagem são utilizados para obter a componente RGB de cada ponto do LiDAR. O FoV vertical da câmara é maior comparativamente ao FoV vertical do LiDAR e, portanto, todos os 16 feixes laser estão projetados na imagem. Contudo, o campo de visão horizontal do LiDAR é bastante superior, fazendo que a maior parte dos pontos sejam eliminados.

A figura 7.22 mostra a nuvem de pontos colorida do *scan* do LiDAR representado na figura 7.20, com uma vista lateral.

A reconstrução 3D desta zona está ilustrada na figura 7.23, enquanto que na figura 7.24 é apresentada a reconstrução 3D total da cerca, com vista de topo.l.

Como a *point cloud* do escadório e da cerca estão geo-referenciadas, estas podem ser combinadas numa única *point cloud*. Esta nuvem de pontos encontra-se ilustrada na figura 7.25.

A partir desta figura, é possível verificar a desigualdade de cor na zona de vegetação (zona intermédia da figura 7.25), que se deve à diferença de luminosidade entre os dois

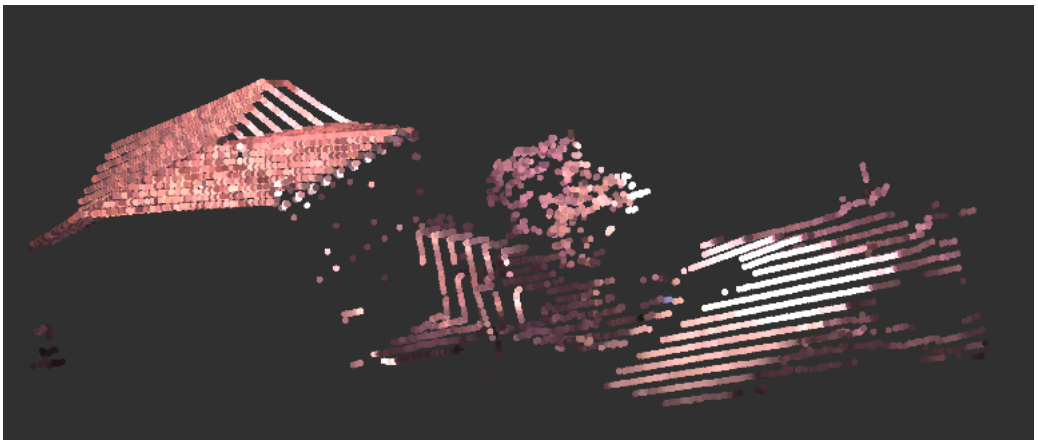


Figura 7.22: Nuvem de pontos colorida do telhado de um edifício do Mosteiro de Tibães com vista lateral.

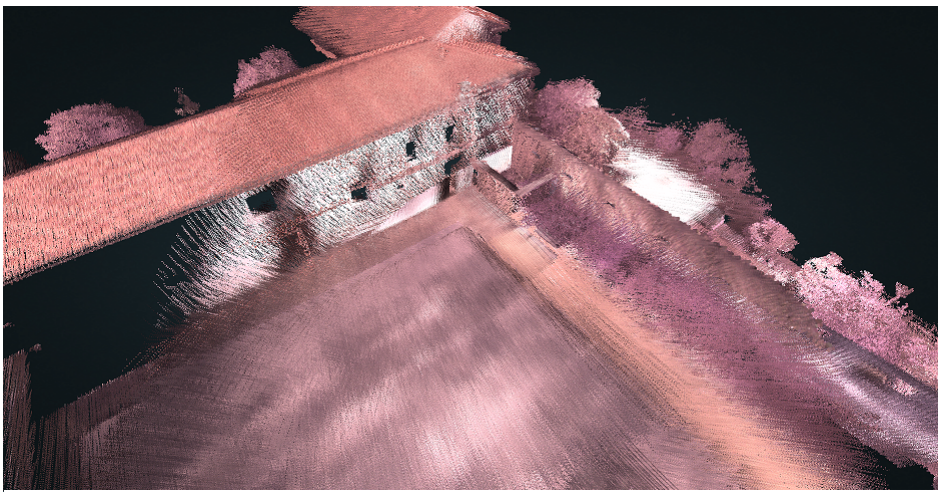


Figura 7.23: Reconstrução 3D do telhado com vista lateral.

datasets. Esta diferença é obtida principalmente devido à câmara utilizada ser sensível ao infravermelho.

Para resolver este problema, é necessário adicionar um filtro de corte de Infrared Radiation (IR), onde existem duas formas de aplicar o filtro: internamente, entre o sensor e a lente, e exteriormente, aplicado na lente. Estes filtros deixam passar a luz visível e bloqueiam a luz fora do intervalo visível (figura 3.7).

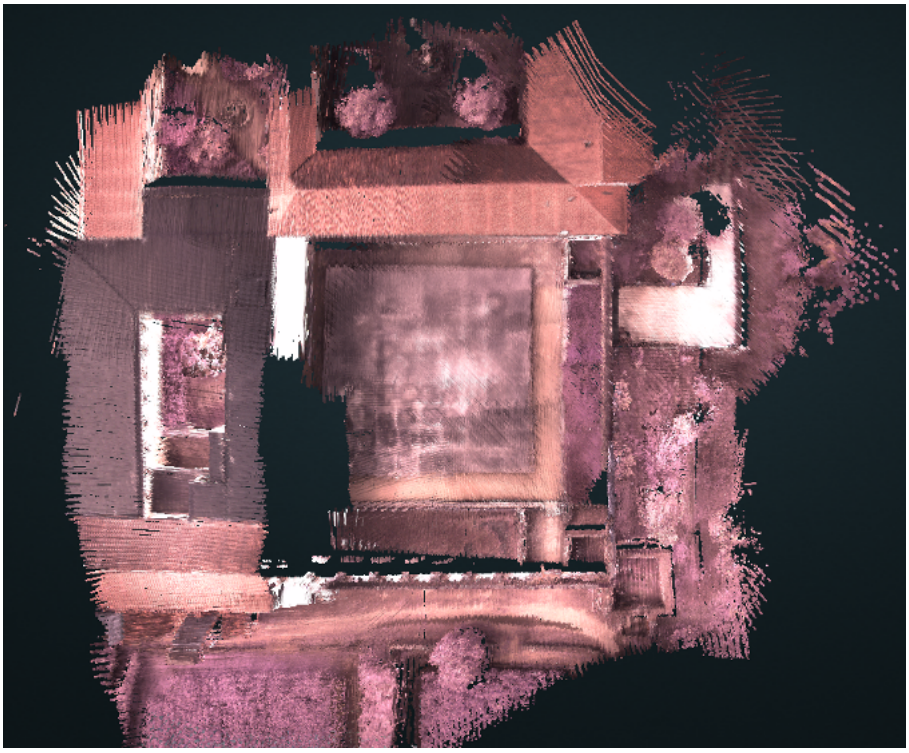


Figura 7.24: Reconstrução 3D da cerca com vista de topo.

Figura 7.25: Reconstrução 3D do escadório e da cerca do Mosteiro de Tibães, com vista de topo.

Esta página foi intencionalmente deixada em branco.

Capítulo 8

Conclusão e Trabalho Futuro

Esta dissertação aborda o desenvolvimento de um sistema de reconstrução 3D para veículos aéreos não tripulados, combinando um LiDAR 3D e uma câmara de espetro visível.

Numa fase inicial, realizou-se um estudo dos métodos existentes de calibração dos parâmetros extrínsecos entre um *laser* e uma câmara, bem como as abordagens para combinar estes dois sensores. Esta análise permitiu compreender os vários métodos já desenvolvidos, sendo estes importantes para definir a melhor abordagem ao problema e a arquitetura do sistema.

O método de calibração implementado possibilita a calibração de um sistema de reconstrução 3D composto por um LiDAR e uma câmara de espetro visível. Este método usa correspondências entre planos e linhas 3D de um alvo de calibração (*checkerboard*) para estimar os parâmetros extrínsecos entre os dois sensores. Nesse sentido, foi desenvolvido um *software* de calibração, utilizando a *framework* ROS.

Com o objetivo de validar e caracterizar o método de calibração, foi utilizado um *setup* composto por um par *stereo* de câmaras e um LiDAR. Os parâmetros extrínsecos entre as duas câmaras foram comparados com os obtidos pelo método de calibração implementado, na qual o *ground truth* foram os parâmetros extrínsecos obtidos na calibração das duas câmaras. Os resultados obtidos comprovam que o processo de calibração apresenta resultados consistentes.

Posteriormente, este sistema foi testado em dois *datasets* que contêm vários edifícios, vegetação e algumas zonas históricas do Mosteiro de Tibães. Nestes *datasets* procedeu-se à reconstrução 3D do escadório e da cerca do Mosteiro, na qual os pontos do *laser* foram coloridos pela componente RGB da câmara. Comparando a nuvem de pontos colorida com as imagens originais, é possível verificar que estas coincidem, mostrando

que o algoritmo está correto.

O trabalho desenvolvido nesta dissertação resultou na submissão do artigo "UAV Approach for 3D Reconstruction of Historical Sites"na conferência *International Conference on Autonomous Robot Systems and Competitions* (ICARSC).

No que diz respeito ao trabalho futuro, existem várias abordagens que podem melhorar a performance do sistema. Em termos de calibração, pretende-se melhorar a otimização final do algoritmo de calibração dos parâmetros extrínsecos entre LiDAR e câmara. Uma melhoria nestes parâmetros levará a uma melhor combinação entre os dois sensores. Além disso, a utilização de um alvo maior poderia ajudar na extração de *features* no LiDAR, visto que se utilizaria mais pontos e, assim, melhorar os parâmetros do plano e das linhas.

A introdução de mais câmaras com diferentes orientações seria uma mais valia, já que aumentava o campo de visão e, consequentemente, o número de pontos coloridos. Além disso, a utilização de uma câmara com maior resolução aumentaria a precisão do processo de colorização.

Bibliografia

[1] Chun Fui Liew, Danielle DeLatte, Naoya Takeishi, and Takehisa Yairi. Recent developments in aerial robotics: A survey and prototypes overview. *arXiv preprint arXiv:1711.10085*, 2017.

[2] Claudia Stöcker, Rohan Bennett, Francesco Nex, Markus Gerke, and Jaap Zevenbergen. Review of the current state of uav regulations. *Remote sensing*, 9(5):459, 2017.

[3] Hazim Shakhatreh, Ahmad H Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *IEEE Access*, 7:48572–48634, 2019.

[4] Zoltan Pusztai and Levente Hajder. Accurate calibration of LiDAR-camera systems using ordinary boxes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 394–402, 2017.

[5] Surabhi Verma, Julie Stephany Berrio, Stewart Worrall, and Eduardo Nebot. Automatic extrinsic calibration between a camera and a 3d lidar using 3d point and plane correspondences. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3906–3912. IEEE, 2019.

[6] Lipu Zhou, Zimo Li, and Michael Kaess. Automatic extrinsic calibration of a camera and a 3D LiDAR using line and plane correspondences. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5562–5569. IEEE, 2018.

[7] Qilong Zhang and Robert Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *2004 IEEE/RSJ International Conference*

on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), volume 3, pages 2301–2306. IEEE, 2004.

[8] Ranjith Unnikrishnan and Martial Hebert. Fast extrinsic calibration of a laser rangefinder to a camera. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-09*, 2005.

[9] Gaurav Pandey, James McBride, Silvio Savarese, and Ryan Eustice. Extrinsic calibration of a 3D laser scanner and an omnidirectional camera. *IFAC Proceedings Volumes*, 43(16):336–341, 2010.

[10] Ankit Dhall, Kunal Chelani, Vishnu Radhakrishnan, and K Madhava Krishna. LiDAR-camera calibration using 3D-3D point correspondences. *arXiv preprint arXiv:1705.09785*, 2017.

[11] Lipu Zhou. Fusing laser point cloud and visual image at data level using a new reconstruction algorithm. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 1356–1361. IEEE, 2013.

[12] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, 3rd ed. edition, 2008.

[13] Fanyang Zeng and Ruofei Zhong. The algorithm to generate color point-cloud with the registration between panoramic image and laser point-cloud. *IOP Conference Series: Earth and Environmental Science*, 17(1), 2014.

[14] Pavel Vechersky, Mark Cox, Paulo Borges, and Thomas Lowe. Colourising point clouds using independent cameras. *IEEE Robotics and Automation Letters*, 3(4):3575–3582, 2018.

[15] Sagi Katz, Ayellet Tal, and Ronen Basri. Direct visibility of point sets. *ACM Trans. Graph.*, 26(3):24, 2007.

[16] Sagi Katz and Ayellet Tal. On the visibility of point clouds. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[17] Wassim Moussa, Mohammed Abdel-Wahab, and Dieter Fritsch. Automatic fusion of digital images and laser scanner data for heritage preservation. In *Euro-Mediterranean Conference*, pages 76–85. Springer, 2012.

[18] Mohammed Abdel-Wahab, Konrad Wenzel, and Dieter Fritsch. Reconstruction of orientation and geometry from large unordered image datasets for low cost applications. In *Proceedings Workshop LC3D, Berlin*, pages 679–690, 2011.

[19] Wolfang Neubauer, Michael Doneus, Nikolaus Studnicka, and Jeffrey Riegl. Combined high resolution laser scanning and photogrammetrical documentation of the pyramids at giza. In *CIPA XX International Symposium*, pages 470–475. Citeseer, 2005.

[20] RiSCAN PRO 2.0. http://www.riegl.com/products/software-packages/riscan-pro/. Acedido em 26/12/2019.

[21] Ahmed Abdelhafiz, Björn Riedel, and Wolfgang Niemeier. Towards a 3d true colored space by the fusion of laser scanner point cloud and digital photos. In *Proceedings of the ISPRS Working Group V/4 Workshop (3D-ARCH*. Citeseer, 2005.

[22] Ji Zhang and Sanjiv Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181. IEEE, 2015.

[23] Thomas Lowe, Soohwan Kim, and Mark Cox. Complementary perception for handheld slam. *IEEE Robotics and Automation Letters*, 3(2):1104–1111, 2018.

[24] Will Maddern and Paul Newman. Real-time probabilistic fusion of sparse 3d lidar and dense stereo. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2181–2188. IEEE, 2016.

[25] Hatem Alismail, L Douglas Baker, and Brett Browning. Continuous trajectory estimation for 3d slam from actuated lidar. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6096–6101. IEEE, 2014.

[26] FARO FOCUS. https://www.faro.com/en-gb/products/construction-bim-cim/faro-focus. Acedido em 1/1/2020.

[27] mdLiDAR. https://www.microdrones.com/en/integrated-systems/mdlidar/mdlidar3000. Acedido em 1/1/2020.

[28] Keisuke Taki, Shoji Yamamotoa, Norimichi Tsumuraa, Toshiya Nakaguchia, and Yoichi Miyakeb. Automatic Color Calibration Method for Multi-Camera System Connected on Network.

[29] Neel Joshi, Bennett Wilburn, Vaibhav Vaish, Marc Levoy Levoy, and Mark Alan Horowitz. *Automatic color calibration for large camera arrays*. [Department of Computer Science and Engineering], University of California . . . , 2005.

[30] Instrumentos Óticos e visão humana. http://www.webquestfacil.com.br/webquest.php?wq=21314. Acedido em 2/12/2019.

[31] Flávio Lopes. Sistema de visão e laser para perceção em ambientes subaquáticos, 2014.

[32] Camera calibration and 3D reconstruction. https://docs.opencv.org/4.1.2/d9/d0c/group__calib3d.html. Acedido em 10/12/2019.

[33] Duane C Brown. Decentering distortion of lenses. *Photogrammetric Engineering and Remote Sensing*, 1966.

[34] Mario Sabbatelli. *3D Vision-based Perception and Modelling techniques for Intelligent Ground Vehicles*. PhD thesis, Università di Parma. Dipartimento di Ingegneria dell'Informazione, 2015.

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

[36] Gregory G Slabaugh. Computing Euler angles from a rotation matrix. *Retrieved on August*, 6(2000):39–63, 1999.

[37] Mark Vallis. Euler angles. https://mveeprojects.files.wordpress.com/2016/10/pitchrollyawimage.gif?fbclid=IwAR0JZ3pSr7bEbUG4S1zgYnaHekARMsxNtn40EFGWLvxbOV_pKhVbaSI28zs. Acedido em 11/12/2019.

[38] L.G. Shapiro and G.C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[39] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[40] David A Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[41] Nick Mokey. Solid-state lidar: The key to cheap self-driving cars - Digital Trends. https://www.digitaltrends.com/cars/solid-state-lidar-for-self-driving-cars. Acedido em 17/12/2019.

[42] LeddarTech. Leddar Solid-State LiDAR Technology Fundamentals. https://leddartech.com/leddar-technology-overview. Acedido em 17/12/2019.

[43] Renishaw. Optical encoders and LiDAR scanning. https://www.renishaw.com/en/optical-encoders-and-lidar-scanning--39244. Acedido em 17/12/2019.

[44] Timothy B. Lee. Why spinning lidar sensors might be around for another decade - Ars Technica. https://arstechnica.com/cars/2018/05/why-bulky-spinning-lidar-sensors-might-be-around-for-another-decade. Acedido em 17/12/2019.

[45] Velodyne LiDAR. VLP-16 User Manual and Programming Guide.

[46] Andy Mohd. Does Infineon's MEMS lidar portend a quantum leap for ADAS? https://medium.com/futuremobile2025/does-infineons-mems-lidar-portend-a-quantum-leap-for-adas-c14eb939545a. Acedido em 18/12/2019.

[47] Hemang Agarwal, Sandeep Bhardwaj, and Hobart Pao. 3D coordinate geometry - equation of a plane. https://brilliant.org/wiki/3d-coordinate-geometry-equation-of-a-plane. Acedido em 15/12/2019.

[48] 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.

[49] Richard O Duda and Peter E Hart. Use of the Hough transformation to detect lines and curves in pictures. Technical report, Sri International Menlo Park Ca Artificial Intelligence Center, 1971.

[50] John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on pattern analysis and machine intelligence*, 8(6):679–698, 1986.

[51] Ramesh Jain, Rangachar Kasturi, and Brian G Schunck. *Machine vision*, volume 5. McGraw-Hill New York, 1995.

[52] 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. Kobe, Japan, 2009.

[53] Matteo Matteucci. Publish/subscribe middleware for robotics: requirements and state of the art. *Tech. Report N 2003.3*, 2003.

[54] INTRO TO ROS - General concepts. https://www.clearpathrobotics.com/assets/guides/ros/_images/ros101three.png. Acedido em 29/12/2019.

[55] Wiki ROS - nodelet. http://wiki.ros.org/nodelet. Acedido em 29/12/2019.

[56] Waleed Mansoor. Understanding ROS nodelets. https://medium.com/@waleedmansoor/understanding-ros-nodelets-c43a11c8169e. Acedido em 29/12/2019.

[57] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Transactions on pattern analysis and machine intelligence*, 9(5):698–700, 1987.

[58] Jorge J Moré. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[59] Faraz M Mirzaei, Dimitrios G Kottas, and Stergios I Roumeliotis. 3d lidar–camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization. *The International Journal of Robotics Research*, 31(4):452–467, 2012.

[60] Taku Komura. Transformations, 2013.

[61] Genie Nano. https://www.teledynedalsa.com/en/products/imaging/cameras/genie-nano-1gige. Acedido em 10/1/2020.

[62] Velodyne LiDAR. https://velodynelidar.com/vlp-16.html. Acedido em 10/1/2020.

[63] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[64] OpenCV. https://opencv.org. Acedido em 10/1/2020.

[65] Single Camera Calibrator App. https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html. Acedido em 11/1/2020.

[66] Eigen library. http://eigen.tuxfamily.org/. Acedido em 12/1/2020.

[67] Du Q Huynh. Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.