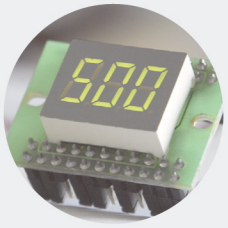
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






GeoTec: Sistema de reconstrução 3D baseado em imagem para cenários GPS-denied

PAULO MIGUEL DA CUNHA RODRIGUES
março de 2020

POLITÉCNICO
DO PORTO

GeoTec: Sistema de reconstrução 3D baseado em imagem para cenários GPS-denied

Paulo Miguel da Cunha Rodrigues
Nº 1140582

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: Paulo Miguel da Cunha Rodrigues
Nº 1140582

Orientador: André Miguel Pinheiro Dias

Co-Orientador: José Miguel Soares de Almeida

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

Esta dissertação não existiria se não fosse pela ajuda preciosa fornecida por várias pessoas, as quais gostaria de dar um agradecimento pessoal.

Em primeiro lugar gostaria de agradecer ao meu orientador, Eng. André Dias, pelo seu trabalho e ajuda na realização desta dissertação, bem como pela oportunidade que me apresentou.

Ao Eng. José Miguel Almeida pelo apoio prestado e por se mostrar sempre pronto a ajudar.

Um agradecimento a todos os colegas do Laboratório de Sistemas autónomos, por toda a ajuda e apoio prestado e por se mostrarem sempre prontos a ensinar-me coisas novas. Em especial ao André Silva, Lino Sousa, Alexandre Amaral e Tiago Miranda. O meu muito obrigado a todos.

Gostaria de agradecer em particular a ajuda de duas colegas também do laboratório sem as quais esta dissertação não seria a mesma, Sara Freitas e Ana Pires, pelo apoio, compreensão e motivação prestados.

Um agradecimento também a toda a minha família, país, irmã, tios e avós, cujos cuidados e dedicação permitiram que eu me tornasse a pessoa que sou hoje.

Por fim queria agradecer a uma pessoa muito especial, sem a qual esta dissertação não existiria, Cláudia Teixeira. Obrigado pela motivação e por nunca desistires de mim nem me deixares desistir, mesmo nos momentos mais difíceis.

A todos o meu sincero muito obrigado.

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

Existem inúmeras minas desativadas que apresentam um elevado valor patrimonial e histórico. Contudo a maioria destas minas acaba por não ser documentadas com o devido cuidado, existindo o risco de deterioração do património com o passar do tempo, pelo que se torna necessária a existência de técnicas que efetuem a reconstrução e o mapeamento 3D de modo a preservar estes locais no mundo virtual.

O projeto *MineHeritage* surge em resposta a essa necessidade, com o objetivo de documentar o património geo-mineiro e disponibilizar todos os dados para consultas futuras, utilizando para isso sistemas robóticos capazes de realizar tarefas de mapeamento e reconstrução dos ambientes subterrâneos das minas e da sua área envolvente.

A dissertação endereça o desenvolvimento de uma plataforma móvel, composta por um conjunto de sensores e equipamentos necessários para a recolha dos dados (par câmaras *stereo*, *Light Detection And Ranging* (LiDAR), sensor inercial), de forma a construir, em pós-processamento, um mapa e respectiva reconstrução 3D das áreas subterrâneas exploradas. A solução desenvolvida é capaz de obter dados que permitam calcular a sua localização relativa num cenário *GPS-denied* (ambientes subterrâneos onde não é possível obter localização GPS) recorrendo ao uso de odometria visual-inercial.

O desenvolvimento do sistema passou pelo levantamento de todos os requisitos de hardware e software necessários para o funcionamento pretendido, pelo desenho de todos os componentes e peças, e pela implementação do sistema. Numa primeira fase, foram realizados vários testes em ambiente controlado. Após a validação de todos as funcionalidades do GeoTec, procedeu-se à realização dos primeiros testes em ambiente real na mina das Aveleiras situada no recinto do mosteiro de Tibães, que se encontra localizado em Braga.

Com os dados obtidos nestes testes, foram realizados vários testes com dois programas que realizam reconstrução 3D: o programa Meshroom, que usa a *framework* de fotogra-

metria da Alice Vision, e em *Robot Operating System* (ROS), o *package* RTAB-Map, sendo que os resultados obtidos mostraram-se promissores. Foi também comparada a eficácia de cada um dos algoritmos na estimação da posição através da comparação da trajétória de um *ground truth*, obtida usando os dados do recetor GNSS RTK fundidos com o sensor inercial e as trajetórias obtidas usando ambos os programas.

Palavras-Chave: Reconstrução 3D, Cenários *GPS-denied*, Visão *stereo*, odometria visual-inercial, extração de *features*, Meshroom, RTAB-Map

Abstract

There are countless deactivated mines that have a high heritage and historical value. However, most of these mines end up not being documented with due care, with the risk of loss of that heritage over time, making it necessary to have techniques that carry out reconstruction and mapping in order to preserve these sites in the virtual world.

The MineHeritage project arises from this need, with the aim of documenting the geo-mining heritage and making all data available for future consultations, using robotic systems capable of mapping and reconstructing the underground environments of the mines and their surrounding areas.

As such, it is essential to develop a mobile platform, consisting of a set of sensors and equipment necessary for data collection (stereo pair, LiDAR, inertial sensor), in order to build, in post-processing, a map and it's respective 3D reconstruction of the underground areas to explore. It should be able to obtain its relative location in a GPS-denied scenario (underground environments where it's not possible to obtain GPS signal) using visual-inertial odometry.

The development of the system involved the survey of all the hardware and software requirements necessary for the intended operation, the design and implementation of the same. In a first phase, several tests were carried out in a controlled environment. After the validation of all GeoTec functionalities, the first missions were carried out at the Aveleiras mine located in the Tibães monastery, which is located in Braga.

With the data obtained in this field mission, several tests were carried out with two programs that perform 3D reconstruction: Meshroom program, that uses the Alice Vision photogrametry framework, and in ROS the RTAB-Map package. The results obtained revealed themselves promising. The effectiveness of each of the algorithms in estimating the position was also compared by comparing the trajectory of a ground truth, obtained using the data from the GNSS RTK receiver merged with the inertial

sensor and the trajectory estimated by each program.

Keywords: 3D reconstruction, GPS-denied scenes, stereo vision, visual-inertial odometry, feature extraction, Meshroom, RTAB-Map

Conteúdo

Agradecimentos **i**

Resumo **vi**

Abstract **viii**

Lista de Figuras **xiii**

Lista de Tabelas **xvii**

Lista de Acrónimos **xxi**

1 Introdução **1**

1.1 Motivação 3

1.2 Objetivos 4

1.3 Estrutura 5

2 Estado da Arte **7**

2.1 Odometria Visual-Inercial 7

2.1.1 Vantagens 8

2.1.2 Vulnerabilidades 9

2.1.3 Calibração 10

2.1.4 Metodologias 11

2.2 Sistemas similares existentes 12

3 Fundamentos Teóricos **15**

3.1 Visão Computacional 15

3.1.1 Modelo *pinhole* . . . 15
3.1.2 Ópticas e distorção . . . 16
3.1.3 Parâmetros intrínsecos . . . 19
3.1.4 Parâmetros extrínsecos . . . 19
3.2 Projeção de objetos . . . 21
3.3 Stereo . . . 21
3.3.1 Geometria epipolar . . . 22
3.4 Processamento de imagem . . . 23
3.4.1 Deteção de *features* . . . 23
3.4.2 *Scale-Invariant Feature Transform* (SIFT) . . . 24
3.4.3 *Speeded-Up Robust Features* (SURF) . . . 26
3.4.4 *Oriented FAST and Rotated BRIEF* (ORB) . . . 27

4 RTAB-Map & Meshroom **29**
4.1 RTAB-Map . . . 29
4.1.1 Mecanismo de gestão de memória e *loop-closures* . . . 29
4.1.2 Extensão ao SLAM e suporte ROS . . . 32
4.2 Meshroom . . . 33
4.2.1 *Framework* AliseVison . . . 34
4.2.2 *Pipeline* da fotogrametria do *AliceVision* . . . 34
4.2.2.1 Extração de *features* (*Feature extraction*) . . . 34
4.2.2.2 Associação de imagens (*Image Matching*) . . . 35
4.2.2.3 Associação de *features* (*Feature Matching*) . . . 35
4.2.2.4 SFM (*Structure-from-Motion*) . . . 36
4.2.2.5 Estimação da Profundidade (*Depth Map Estimation*) . . . 36
4.2.2.6 Reconstrução (*Meshing*) . . . 37
4.2.2.7 Obtenção de textura (*Texturing*) . . . 37

5 GeoTec System **39**
5.1 Requisitos do sistema . . . 39
5.2 Arquitetura de Hardware . . . 40
5.3 Opções de Projeto . . . 41
5.3.1 Setup de sensores . . . 41
5.3.1.1 IMU . . . 42

5.3.1.2 Unidade de processamento 43
5.3.1.3 Sensor óptico e lentes 43
5.3.1.4 LiDAR 46
5.3.1.5 GNSS 47
5.3.2 *Trigger* das câmaras por *Hardware* 48
5.3.3 *Framework* ROS 49
5.4 Arquitetura de Software 52

6 Implementação **55**
6.1 Descrição Hardware 55
6.1.1 Estrutura Mecânica - Corpo principal 55
6.1.2 Estrutura Mecânica - Suporte para movimentação 60
6.1.3 PCB de distribuição de energia 62
6.1.4 Placa de trigger 63
6.1.5 PCB de interface com o recetor GNSS 65
6.2 Descrição de Software 66
6.2.1 Sincronização do relógio do CPU 67
6.3 Implementação GeoTec 69

7 Resultados **75**
7.1 Testes de validação do sistema e sensores 75
7.2 Calibração dos parâmetros intrínsecos e extrínsecos 78
7.3 Testes *MineHeritage* na Mina das Aveleiras, Tibães 80
7.4 Ground Truth 82
7.5 Reconstruções 3D usando o Meshroom 87
7.6 RTAB MAP 91

8 Conclusão e Trabalho Futuro **95**

Bibliografia **97**

Esta página foi intencionalmente deixada em branco.

Lista de Figuras

1.1 Exemplos de rôbos desenvolvidos pelo laboratório. 4

2.1 Transformação das medições do IMU para o *frame* de navegação 10
2.2 Veículo robótico usado por Ferguson et al. 13

3.1 Componentes de uma câmara 16
3.2 Modelo *pinhole* 16
3.3 Efeitos da variação do tamanho da abertura numa imagem. 17
3.4 Tipos de distorção 18
3.5 Rotação e translação entre referenciais 20
3.6 Projeção de um ponto do mundo para a imagem 21
3.7 Geometria epipolar 22
3.8 Deteção de extremos SIFT 25
3.9 Comparação com *pixels* vizinhos 25
3.10 Comparação dos tempos de execução de vários métodos na análise de uma imagem 26

4.1 Mecanismo de gestão de memória do RTAB-Map. 31
4.2 Diagrama de blocos do nó ROS *rtabmap*. 33
4.3 Interface gráfica do *Meshroom*. 34
4.4 *Pipeline* da fotogrametria do *AliceVision* 35

5.1 Arquitetura de hardware do sistema. 40
5.2 IMU STIM 300 da Sensonor 42
5.3 Unidade de processamento CAPA881 44
5.4 Câmara FLIR Blackfly GigE - BFLY-PGE-31S4C-C 45

5.5 Hokuyo UTM-30LX 46
5.6 Recetor GNSS UB482 47
5.7 ROS Computation Graph level. 51
5.8 Arquitetura de software do sistema. 53

6.1 Perfil Alumínio Bosch. 56
6.2 Estrutura mecânica completa. 57
6.3 Pormenor das duas partes da estrutura mecânica. 57
6.4 Patamar intermédio do frame base e posicionamento dos componentes na placa de acrílico. 58
6.5 Peça para fixação das câmaras. 58
6.6 Conjunto câmara e peças de fixação. 59
6.7 Fixação do LiDAR e placa de iluminação 59
6.8 Estrutura mecânica completa com todos os componentes. 60
6.9 Base do suporte mecânico para movimentação. 61
6.10 Montagem das rodas posteriores do suporte de locomoção. 61
6.11 Desenho da PCB de distribuição de *power* e vista 3D da mesma. 63
6.12 Sistema de Luz Estruturada. 64
6.13 3D da placa de *trigger* das câmaras. 65
6.14 Placa de interface com o recetor GNSS. 66
6.15 Sub-rede de sincronização do *Network TIme Protocol* (NTP). 68
6.16 Peças de encaixe das câmaras. 70
6.17 Caixa que contêm a placa de power e a PCB de *trigger*. 71
6.18 Estrutura de movimentação completa. 71
6.19 Sistema *GeoTec* completo. 72
6.20 Sistema *GeoTec* preparado para a realização do *ground truth*. 73

7.1 Exemplo de aquisição de imagens no exterior do laboratório. 76
7.2 Receção dos dados do recetor GNSS. 76
7.3 Sincronização do *clock* do CPU usando o programa *chrony*. 77
7.4 Diferença de tempos entre aquisição e receção das imagens. 78
7.5 *Toolbox* de calibração stereo MATLAB. 79
7.6 Pormenor dos erros de reprojeção da calibração. 79
7.7 Parâmetros extrínsecos e intrínsecos obtidos através da calibração. 80

7.8 Traçado geral e entrada da mina das Aveleiras. 81
7.9 Exemplo de algumas imagens obtidas nos testes em Tibães. 81
7.10 Resultados preliminares apresentados na conferência 4GEO. 82
7.11 Exemplo de alguns pares de imagens do *ground truth*. 83
7.12 Trajetória pós-processada realizada no *ground truth*. 84
7.13 Posições e orientações do *ground truth* 84
7.14 Qualidade do sinal GPS durante o percurso realizado no *ground truth*. . . 85
7.15 Comparação da estimação da trajetória, usando o RTAB-Map e Meshroom, com o *ground truth*. 86
7.16 Estimação da trajetória nos primeiros 40m do *ground truth*. 86
7.17 Interface gráfica Meshroom com dados processados. 87
7.18 Imagem com os descritores SIFT extraídos pelo Meshroom. 88
7.19 Exemplos da reconstrução 3D usando as imagens do *ground truth*. 88
7.20 Exemplo de uma imagem obtida na mina, com e sem a extração de *features*. 89
7.21 Nuvem de pontos obtida através do Meshroom. 89
7.22 Exemplos da reconstrução 3D usando as imagens adquiridas na mina. . . 90
7.23 Par de imagens retificadas usando o *node stereo_image_proc*. 91
7.24 Trajetória e nuvem de pontos obtida usando o *software* RTAB-Map. . . . 92
7.25 Imagens com *features* extraídas usando o método SURF. 92
7.26 Reconstrução 3D obtida usando o *software* RTAB-Map. 93

Esta página foi intencionalmente deixada em branco.

Lista de Tabelas

5.1 Características STIM 300 [1] 43
5.2 Características CAPA881 [2] 44
5.3 Características BFLY-PGE-31S4C-C [3] 45
5.4 Características ópticas GMTHR23514MCN [4] 46
5.5 Características do UTM-30LX [5] 47
5.6 Especificações do recetor UB482 [6] 48
5.7 Comparação entre trigger por Hardware e Software [7]. 49

6.1 Lista de tópicos publicados 67

Esta página foi intencionalmente deixada em branco.

Lista de Siglas e Acrónimos

ASV *Autonomous Surface Vehicle*

AUV *Autonomous Underwater Vehicle*

BIOS *Basic Input/Output System*

BRIEF *Binary Robust Independent Elementary Features*

CMOS *Complementary Metal-Oxide-Semiconductor*

CPU *Central Processing Unit*

CRAS Centro de Robótica e Sistemas Autónomos

FIFO *First-In First-Out*

FPS *Frames per Second*

GigE *Gigabit Ethernet*

GNSS *Global Navigation Satellite System*

GPS *Global Position System*

HDMI *High-Definition Multimedia Interface*

IMU *Inercial Motion Unit*

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

LAN *Local Area Network*

LDO *Low-Dropout*

LED *Light Emiting Diode*

LiDAR *Light Detection And Ranging*

LoG *Laplacian of Gaussian*

LSA Laboratório de Sistemas Autónomos

LTM *Long-Term Memory*

MEMS *Micro-Eletric-Mechanical Systems*

MOD *Minimum Object Distance*

NTP *Network TIme Protocol*

ORB *Oriented FAST and Rotated BRIEF*

OS *Operating System*

PCB *Printed Circuit Board*

PPS *Pulse Per Second*

PTP *Precision Time Protocol*

RAM *Random Access Memory*

ROS *Robot Operating System*

RTAB-Map *Real-Time Appearance-Based Mapping*

RTK *Real Time Kinematic*

SAIL *Stanford Artificial Intelligence Laboratory*

SBC *Single-Board Computer*

SFM *Structure from Motion*

SIFT *Scale-Invariant Feature Transform*

SLAM *Simultaneous Location and Mapping*

SLS *Scructured Light System*

SM *Sensor Memory*

SSD *Solid State Drive*

STM *Short-Term Memory*

SURF *Speeded-Up Robust Features*

TCP/IP *Transmission Control Protocol / Internet Protocol*

UART *Universal Asynchronous Receiver-Transmitter*

UAV *Unmanned Aerial Vehicle*

UNEXMIN *Underwater Explorer for flooded Mines*

UDP *User Datagram Protocol*

USB *Universal Serial Bus*

UTC *Coordinated Universal Time*

VAMOS *Viable Alternative Mine Operating System*

WM *Working Memory*

Esta página foi intencionalmente deixada em branco.

Capítulo 1

Introdução

Nos últimos anos verificou-se um aumento do interesse em documentar locais com interesse histórico, cultural ou arqueológico. Um exemplo deste interesse é o projeto ***CyArk***[1], que originou um repositório de dados único, ***OpenHeritage3D***[2], e tem como objetivo preservar e disponibilizar (de acesso *open source*) uma base de dados extensiva e mundial com inúmeros *datasets*, que incluem informação sobre diversos tipos de património. Estes projetos baseiam-se em três linhas de ação distintas: conservação, recuperação e descoberta. Em Historic England [8] são abrangidos métodos de fotogrametria para o registo do património cultural com particular ênfase nas técnicas de *structure from motion* (SFM). Contudo, apesar do *OpenHeritage3D* disponibilizar dados de vários pontos históricos, mostrando-se uma base de dados relativamente completa, este repositório evidência uma lacuna quanto a dados sobre minas subterrâneas. Assim sendo, torna-se interessante a recolha de dados deste tipo de património histórico, permitindo um acesso rápido e fácil para pessoas que, caso contrário, não seriam capazes de obter essas informações.

É desta necessidade de mapeamento e reconstrução 3D que surge o projeto *MineHeritage*[3], cujo objetivo passa por analisar antigas minas históricas e as suas áreas circundantes. Este projecto sensibiliza para a importância de preservar e reconhecer o interesse deste tipo de património geo-mineiro, acabando também por permitir visitas virtuais destes locais.

Este crescente interesse em termos de preservação, também permite obter modelos detalhados dos complexos mineiros, que permitam a sua consulta e utilização. Isto

[1] https://www.cyark.org/
[2] https://openheritage3d.org/
[3] https://sites.fct.unl.pt/mineheritage

torna-se útil para aplicações mais práticas ou para permitir visitas virtuais ao património histórico e geológico do qual fazem parte algumas destas minas. Estas visitas tornam-se consideravelmente relevantes devido ao difícil acesso que muitas vezes estas minas apresentam, além da existência de um conjunto de fatores imprevisíveis (como guerra, catástrofes naturais, alterações climáticas, deterioração temporal, etc) que podem levar à destruição e perda deste património histórico. Muitas vezes é também de interesse integrar nessa análise espacial dados já obtidos através da geologia, hidrogeologia, alinhamentos tectónicos e zonamento geotécnico, de forma a obter um modelo mais completo possível das áreas de interesse. Muitos autores [9, 10, 11, 12, 13] já mostram o seu interesse em realizar as suas investigações nestas áreas, aplicando diversas técnicas e demonstrando, assim, a importância da documentação histórica e cultural de diversos tipos de património ou de estudos de carácter geo-científico e tecnológico.

Para se obter uma reconstrução fiel à realidade é crucial obter informações sobre a localização. Seja localização absoluta, de modo a permitir georreferenciar todos os dados obtidos, ou localização relativa permitindo a fusão e alinhamento das várias nuvens de pontos e/ou modelos 3D obtidos.

O sistema *Global Navigation Satellite System* (GNSS), método mais fiável e convencional para obter a localização, não pode ser considerado uma alternativa viável para ambientes subterrâneos uma vez que o sinal GNSS não penetra a superfície da terra. Existe, portanto, a necessidade de estimar essa localização através de outros métodos que utilizem localização relativa. A maioria desses métodos utiliza algoritmos de odometria visual, isto é, permitir a obtenção da localização através da análise e processamento de imagens. Estes algoritmos podem ou não incluir a fusão destes dados com os dados obtidos através de sistemas inerciais.

Ao longo desta dissertação serão abordados os métodos e algoritmos necessários para a reconstrução 3D de túneis e galerias de minas, utilizando odometria visual e a integração dos dados do sistema inercial, bem como o desenvolvimento de um sistema robótico que permita a recolha de dados nos ambientes subterrâneos. O trabalho aqui apresentado tem também como objetivo estudar a eficácia na estimação da trajetória efetuada pela plataforma robótica desenvolvida (*GeoTec*) em dois cenários comparativos, variando entre eles o conjunto de sensores utilizados para a estimação. Nesta plataforma existem dois tipos de sensores que fornecem a informação necessária para calcular o trajeto percorrido pela mesma ao longo do tempo: visual e inercial. Pretende-se, portanto, averiguar o comportamento da estimação em duas circunstâncias distintas: considerando apenas a informação visual, e fundindo a informação visual com a inercial. Com os resultados obtidos poderão ser avaliadas as vantagens da fusão dos dois sensores para a

modelização/reconstrução 3D.

1.1 Motivação

O Centro de Robótica e Sistemas Autónomos (CRAS) é um dos vários centros de investigação do Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência (INESC TEC). Este centro está envolvido em vários projetos de investigação, nacionais e europeus, nas áreas da robótica aérea, aquática e subaquática. Existem inúmeros robôs criados pelo laboratório, sejam *Unmanned Aerial Vehicle*s (UAVs), *Autonomous Surface Vehicle*s (ASVs) e *Autonomous Underwater Vehicle*s (AUVs).

Entre estes projetos contam-se: *Underwater Explorer for flooded Mines* (UNEXMIN), projeto europeu para a criação de uma equipa de robôs para a exploração de minas abandonadas inundadas. Outro exemplo de projeto europeu é o *Viable Alternative Mine Operating System* (VAMOS), um projeto para a exploração e extração de materiais num ambiente subaquático no qual o INESC TEC foi encarregue da criação de um robô responsável pela monitorização e mapeamento das áreas a serem exploradas. Na figura 1.1 são apresentados alguns dos robôs criados pelo laboratório, nomeadamente o STORK VII, um veiculo aéreo não tripulado autónomo, desenvolvido para realizar inspeções em torres e linhas de alta tensão, a EVA, AUV desenvolvido para o projeto VAMOS e o UX-1, um dos 3 robôs desenvolvidos para o projeto UNEXMIN.

A presente dissertação, pretende ser um importante contributo para o projeto *MineHeritage* (EIT/RAW MATERIALS/SGA2019/1). Este é um projeto educacional (segmento "*Wider Society Learning*") cujo objetivo principal é promover a importância das matérias-primas através do património histórico mineiro, criando uma base de dados geológica e incluindo também a informação sobre a história da mineração na europa. No âmbito deste projecto, o papel do desenvolvimento tecnológico tem sido significativo, na medida em que os outputs fornecidos foram obtidos através da plataforma robótica desenvolvida ao longo desta investigação.

O INESC TEC é o parceiro tecnológico responsável pela aquisição e processamento de dados, de modo a fornecer nuvens de pontos e modelos 3D para serem integrados num jogo educativo e um app no final do projeto. Assim sendo, torna-se necessário realizar a aquisição de imagens aéreas de alta resolução, através da utilização de UAVs, tal como obter um mapeamento preciso das áreas circundantes (usando um *scanner* laser 3D de alta performance), para além de um sistema que permita mapear o interior das minas subterrâneas. É neste contexto que surge o sistema *GeoTec*, que será um dos pontos abordados nesta dissertação.

Figura 1.1: Exemplos de rôbos desenvolvidos pelo laboratório.

1.2 Objetivos

Esta tese endereça a temática de mapeamento indoor e outdoor de localizações de património histórico, tal como minas, visando o desenvolvimento de um sistema de *surveing* para proceder à recolha dos dados e utilizando dois programas que permitem o tratamento dos dados obtidos de forma a obter o mapeamento e modelos 3D, tendo como principais objetivos os seguintes tópicos:

- Efetuar levantamento dos requisitos de hardware e software de uma solução de reconstrução 3D para cenários GPS-*denied*;
- Desenvolvimento de uma solução, o sistema *GeoTec*, que reúne vários tipos de sensores de forma a poder mapear o ambiente subterrâneo de uma mina;
- Análise e estudo das técnicas e algoritmos usados por dois *softwares* (RTAB_Map e Meshroom) cuja finalidade é realizar reconstrução 3D;
- Estudar a eficácia da estimação da trajetória efetuada pelo sistema, comparando com um *ground truth*, estimação essa obtida usando os dois programas já referidos (RTAB_Map e Meshroom);
- Comparar a eficiência de algumas técnicas de estimação de trajetórias;

1.3 Estrutura

Esta dissertação está organizada em 8 capítulos, sendo que no primeiro é feita uma contextualização do problema e são apresentadas as razões pelas quais surge este projeto.

No capítulo 2 é feito um levantamento de literatura quanto a técnicas de odometria visual, bem como alguns sistemas similares ao *GeoTec*.

No capítulo 3 são descritos alguns conceitos e métodos importantes para o desenvolvimento do sistema. O quarto capítulo descreve as ferramentas de software utilizadas para testar as capacidades do sistema.

Já no capítulo 5 é apresentada uma descrição detalhada de todos os passos necessários para o desenvolvimento do sistema *GeoTec*, desde a fase de projeto, escolha de sensores e componentes, arquiteturas de hardware e software, até ao desenho das estruturas mecânicas que fazem parte do sistema.

A implementação de todos os componentes e estruturas do sistema é descrito no capítulo 6. Já no capítulo 7 são apresentados alguns resultados da calibração dos parâmetros das câmaras e dos testes de todos os sensores, bem como resultados dos testes de campo, da realização do *ground truth* e reconstruções 3D.

Por fim no último capítulo, 8 são apresentadas as conclusões desta dissertação, bem como o trabalho futuro que possa vir a ser desenvolvido.

Esta página foi intencionalmente deixada em branco.

Capítulo 2

Estado da Arte

Habitualmente, as ferramentas de reconstrução 3D recebem como *input* um *set* de imagens e através de métodos de deteção e *matching* de *features* conseguem obter correspondências e relacionar assim os diferentes pontos de vista. Quando estas *features* se mantêm no plano da imagem ao longo do tempo, a sua trajetória pode ser relacionada geometricamente com a trajetória da câmara. Este processo é denominado de *Structure from Motion* (SFM), no entanto, se existir apenas informação visual, existe uma acentuada incerteza associada ao fator de escala global, podendo introduzir erros na estimação da trajetória da câmara.

Nesta secção serão abordadas as diferentes técnicas de reconstrução 3D que utilizam técnicas de visão e dados obtidos usando um *Inercial Motion Unit* (IMU) para a estimação da posição e atitude ao longo de um percurso efetuado em cenários *GPS-denied*, sendo também referidos alguns sistemas robóticos semelhantes, ou com propósitos similares ao sistema *GeoTec*.

2.1 Odometria Visual-Inercial

A tarefa de estimar o movimento de um robô através da combinação de imagens e medições inerciais provenientes de um *Inertial Measurement Unit* (IMU) denomina-se de Odometria Visual-Inercial (ou *Visual Inertial Odometry* - VIO). Vários estudos têm demonstrado que a conjugação destes dois sensores, para além de resultar numa estimação robusta e fiável [14, 15] do estado de um robô (posição, orientação e velocidade) traz diversas vantagens, evidenciadas de seguida, relativamente a outros sistemas de localização, tendo assim recebido bastante interesse na comunidade robótica.

2.1.1 Vantagens

Relativamente aos sensores utilizados em VIO, existem vantagens em ambos os casos. No que diz respeito ao sensor inercial, os avanços recentes no desenvolvimento de sensores inerciais baseados em sistemas microeletromecânicos (MEMS) tornou possível a produção de IMUs de baixo custo, tamanho reduzido e com excelente precisão, capazes de estimar a *pose* (entenda-se por *pose* a combinação da posição e orientação de um determinado robô) de sistemas como UAVs ou robôs móveis terrestres. No que toca às câmaras, o ganho fundamental da sua utilização como sensor são as medições ricas em informação e, tendo em conta que usando os métodos existentes atualmente para a extração de *features* é possível obter centenas de pontos de controlo por cada imagem, obtém-se uma boa eficácia na localização.

Adicionalmente, a principal vantagem de combinar uma câmara com um IMU está relacionada com o facto de os dois sensores se complementarem de forma a resolver as ambiguidades existentes nas suas medições para a estimação do movimento, que seriam impossíveis de resolver se os sensores fossem utilizados individualmente. Por um lado, através da medição de aceleração linear e velocidade angular de um corpo rígido por parte do IMU, presumivelmente, poder-se-ia determinar a variação da *pose* pela integração destas medições (quer dos acelerómetros, quer dos giroscópios), contudo, na prática verifica-se que os sensores inerciais são alvo de ruídos de baixa frequência que limitam a sua precisão após um certo período de tempo e, portanto, as observações de imagem são benéficas no sentido em que contrariam este erro acumulado na integração das medidas obtidas usando o IMU.

Por outro lado, quando existe perda de informação visual, quer por alterações bruscas de luminosidade, quer pela ofuscação da imagem provocada pelo movimento ou até pela existência de áreas com muito pouca textura, poderá levar a falhas na extração de *features* ou a estas serem observadas apenas durante curtos intervalos de tempo, introduzindo ambiguidades na estimação do movimento. Neste tipo de cenários, o sistema deve conseguir adaptar-se e lidar com os períodos em que não se observam *landmarks* ou onde exista ambiguidade no resultado das mesmas, desta forma a integração do IMU consegue melhorar drasticamente a performance da estimação, auxiliando nos seguintes aspetos:

- Remoção das descontinuidades na estimação do movimento resultantes das *features* que entram ou saem do campo de visão (*Field of view* - FOV) da câmara.
- Determinação de uma escala global.

- Geração de uma trajetória mais robusta na medida em aumenta a robustez do sistema a perdas de informação visual.

Por conseguinte, pode assim considerar-se que uma câmara em movimento funciona como um sensor exteroceptivo, isto é, permite adquirir informação do ambiente à volta do robô, medindo o aspeto e a geometria de um cenário tridimensional num fator de escala indeterminado. Já o IMU é considerado um sensor proprioceptivo, na medida em que é capaz de medir valores internos ao sistema, como o movimento. Com o IMU é possível processar a escala de medição proveniente da visão e estimar tanto a escala absoluta como os parâmetros que modelizam o movimento do sistema [16, 17]. Assim, estes sensores complementam-se, cobrindo as suas próprias limitações e fornecendo informação relevante para a produção de sistemas de navegação e/ou mapeamento.

2.1.2 Vulnerabilidades

Para além da vulnerabilidade associada às situações de elevada velocidade, explicada anteriormente, que exige demasiadas integrações do IMU e consequente integração de erro, em VIO existem ainda duas outras complicações:

- **Alto alcance dinâmico**: Nas câmaras tradicionais o alcance dinâmico é limitado, o que pode levar tanto à sub-exposição como sobre-exposição de significativas regiões da imagem, reduzindo drasticamente a quantidade de informação.
- **Inicialização do IMU**: Para conseguir estimar os parâmetros associados à *pose*, através da integração da informação do IMU, é necessário inicializar corretamente o sistema, isto é, a sua posição, orientação e velocidade iniciais. Quanto maior for a incerteza do estado inicial, maior dificuldade terá o estimador em convergir para valores corretos.

Para atenuar o problema do alcance dinâmico surgiram abordagens dedicadas a cenários altamente dinâmicos [18], ou até com câmaras com sensor de visão dinâmico (denominadas de *event cameras*) [19, 20, 21] que trazem uma grande vantagem neste tipo de situações tendo em conta que, ao contrário das câmaras tradicionais, cada pixel opera independente e assincronamente obtendo-se variações de intensidade entre imagens. No que diz respeito à inicialização do IMU, as soluções passam por colocar a plataforma robótica numa posição previamente conhecida, ou inicializar os estados com uso de um rector *GNSS* ou, em último caso, adotar condições iniciais com um determinado grau de incerteza, e tentar convergir para o estado verdadeiro ao longo do tempo.

2.1.3 Calibração

A fusão das medições visuais e inerciais só é possível se ambas estiverem no *frame* de navegação, tal com é evidenciado na figura 2.1. Para tal é necessário calibrar os extrínsecos (transformação de 6 graus de liberdade) de ambos os sensores (câmara e IMU). Se existir erro associado a esta calibração a performance do sistema de navegação será afetada, dada a introdução de um *bias* sistemático a cada estimação.

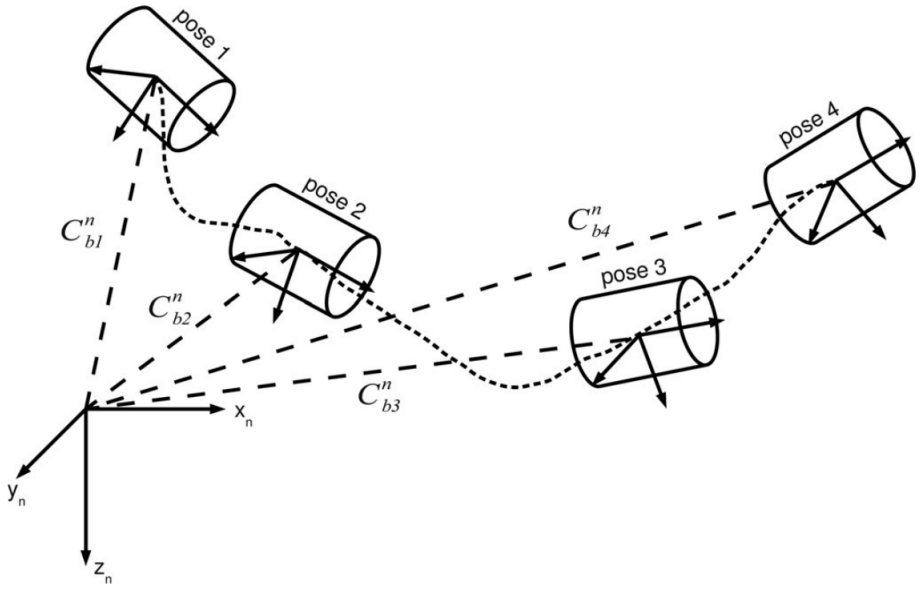


Figura 2.1: Exemplo de transformação (dada por C e representada através das retas a tracejado) das medições do IMU para o *frame* de navegação de forma a estimar a trajetória (representada a tracejado) [18]

Porém, o processo de calibração é normalmente complexo, demorado e inclusive terá de ser repetido caso um dos sensores sejam reposicionados ou alvo de stress mecânico. Para tentar facilitar este processo vários autores apresentaram diferentes métodos de calibração capazes de calcular a matriz de transformação sem necessidade de conhecer *a priori* as posições das *landmarks* [22, 23, 24] capazes até de se calibrarem em operação [25].

2.1.4 Metodologias

O estudo desenvolvido na área da VIO pode ser subseccionado em três parâmetros:

- **Dimensão da estimação (*poses* da câmara**: enquanto que os algoritmos de *smoothing* estimam todo o historial de *poses*, os algoritmos de filtragem e *fixed-lag smoothing* (com recurso a *sliding window* usam apenas o último estado (caso da filtragem) ou marginalizam os estados mais antigos (caso do *fixed-lag smoothing*).
- **Representação da incerteza na medição**: no caso do *Extended Kalman Filter* (EKF) é pela matriz de covariância, já no filtro de informação e técnicas de *smoothing* recorre-se à matriz de informação (inverso da matriz de covariância) ou à raiz quadrada da matriz de informação.
- **Número de linearizações ao modelo de observação**: nas abordagens de *smoothing* permite a linearização múltiplas vezes a cada medição, enquanto que no EKF as medições são processadas uma única vez.

A abordagem mais comum para lidar com a Odometria Visual-Inercial é a fusão multisensorial onde o IMU opera como um módulo independente de assistência à infraestrutura visual. No campo da robótica, as características inerentes à VIO, nomeadamente a estimação de uma *pose* de uma plataforma tridimensional, são comumente associadas ao SLAM (*Localization And Mapping*), tendo este problema sido bastante estudado e explorado pela comunidade robótica nos últimos anos.

No que toca à Odometria Visual (e SLAM, consequentemente), esta tem sido associada à filtragem [26], onde as medidas do IMU propagam o estado, e as medições visuais associadas às *features*/*keypoints* formam as observações, ou *update*. A maioria dos métodos de SLAM adotam técnicas recursivas de filtragem. Em [27] os autores propõem uma fusão em tempo real, baseada em EKF, com apenas uma câmara. Em [28] os autores apresentam os resultados de uma estimação por filtragem para VIO num *dataset* de longo percurso. Ambos os casos com erros inferiores a 0,5 % da distância total percorrida. Segundo os autores [29, 30, 31] o EKF é a metodologia mais apropriada para a fusão de sensores de visão com sensores inerciais. Para além dos inúmeros autores que privilegiam o uso do EKF para VIO [32, 33, 34], ainda nas técnicas de filtragem, existem abordagens com *Unscented Kalman Filter* (UKF) [22, 25]] para o problema da Odometria Visual-Inercial.

Porém, a complexidade das técnicas associadas à filtragem, tal como o EKF, cresce quadraticamente com o número de *landmarks* a estimar, e tendo em conta o elevado

volume de informação que as imagens têm atualmente, para cenários com mais de que uma câmara, a performance destes algoritmos pode ser limitada, e condicionar aplicações como sistemas de localização em tempo real. Uma alternativa à filtragem é o *smoothing*, já implementado por vários autores no âmbito da VIO [35, 36, 37] que se baseia na otimização de *factor graphs* [38] onde o IMU é preintegrado e otimizado juntamente com a informação visual aquando de uma observação/*update*, existindo ainda trabalhos com aplicação prática em tempo real [39].

Por um lado, as abordagens de filtragem compensam o problema da complexidade, na medida em que possibilitam ciclos mais rápidos de estimação, no entanto os erros associados à linearização em torno da última observação podem deteriorar a precisão. Por outro lado, as abordagens de *smoothing* baseadas em otimizações não lineares oferecem resultados exatos mas são computacionalmente mais exigentes. Observa-se assim que em VIO impõe-se um importante *trade-off* entre a precisão e eficiência.

Equitativamente à linha de trabalho e objetivos do sistema aqui apresentado, em [40] os autores focam-se em fornecer uma estimação robusta e *offline* da trajetória da câmara de modo a recuperar o seu movimento para gerar um espaço virtual desse percurso, através da fusão do IMU com os resultados do SFM.

2.2 Sistemas similares existentes

Existem alguns sistemas similares ao pretendido que se encontram já desenvolvidos, sendo que a maioria usa sensores laser ($LiDAR$), e não apresenta qualquer tipo de sensores de visão ou inerciais.

Alguns sistemas tentaram criar representações volumétricas 3D de ambientes subterrâneos com *scanners* laser 2D. Em Thrun et al. [41] são usados dois sensores laser 2D para adquirir dados 3D. Um sensor é montado horizontalmente e um verticalmente, sendo que o último obtém um *scan* vertical que é transformado em pontos 3D usando a *pose* atual do robô e que o outro sensor é usado para calcular essa mesma *pose*. A precisão dos pontos 3D obtidos depende dessa *pose* e da precisão do scanner utilizado.

Ferguson et al. [42] apresenta uma solução de arquitetura de software de um sistema robótico para mapeamento autónomo de minas abandonadas. O veículo usado, representado na figura 2.2, é equipado com dois sensores laser atuados, sendo que ao explorar e mapear, primeiro o robô movimenta-se guiado por *scans* laser 2D, alternando para fases em que o veículo se imobiliza e adquire um *scan* 3D do ambiente que o rodeia. Analisando estes *scans* o robô planeia uma trajetória que é depois executada usando scans 2D rápidos para determinar a posição relativa do robô em relação ao mapa 3D.

Esta aproximação é um pouco diferente da pretendida nesta dissertação, na medida em que o objetivo não é um sistema autónomo mas sim um sistema de *surveing* que recolha os dados necessários para realizar modelos 3D.

Figura 2.2: Veículo robótico usado por Ferguson et al. [42]

Já no caso de Thrun et al. [43], para além de ter sido utilizado o mesmo robô que no trabalho de Ferguson et al. [42], foi usada também uma plataforma robótica empurrada por humanos equipada com quatro sensores laser 2D. Estes fornecem informações sobre a secção transversal da mina à frente do veículo e a estrutura do solo e do teto. De forma a construir mapas consistentes de minas extensas, é apresentado um algoritmo para estimar correspondências globais e alinhar as trajetórias dos robôs. Este algoritmo permite recuperar mapas consistentes com várias centenas de metros de diâmetro, sem utilizar informações de odometria. A abordagem seguida baseia-se na correspondência de *scans* 2D para obter um mapa localmente consistente e num algoritmo de alinhamento global 2D para gerar mapas globais também eles consistentes. Os mapas resultantes e as trajetórias dos robôs formam a base para a integração das informações 3D, adquiridas por sensores adicionais apontados para o teto e para o chão da mina. Uma etapa final de otimização melhora ainda mais a consistência espacial do mapa 3D resultante da mina.

Nenhum destes sistemas robóticos apresenta sensores inerciais ou de visão de algum tipo, sendo este um dos seus defeitos, uma vez que obriga a que a localização dos veículos após deixarem o ponto inicial seja apenas estimada usando os *scans* laser.

Esta página foi intencionalmente deixada em branco.

Capítulo 3

Fundamentos Teóricos

3.1 Visão Computacional

O termo visão computacional pode ser descrito como o processo de replicação da visão humana usando diferentes tecnologias e equipamentos. Entre esses equipamentos encontram-se as câmaras, sendo que estas são o equipamento mais importante na área de visão computacional, na medida em que é possível extrair uma grande quantidade de informação através do processamento de uma imagem capturada.

Uma descrição simples de uma câmara pode ser feita separando-a em três componentes principais sendo que cada um tem funções diferentes. Estes são: o sensor ótico, a lente e o obturador (*shutter*). A lente é responsável por convergir os feixes de luz de modo a que estes incidam sobre o sensor ótico, sendo que este tem como objetivo converter a intensidade dos feixes de luz recebidos em pulsos elétricos. Por último, o shutter que existe entre os dois tem a função de regular a quantidade de luz que chegar ao sensor [44]. Na figura 3.1 está detalhada a posição e interação entre estes componentes.

3.1.1 Modelo *pinhole*

Uma imagem é uma projeção 2D de uma cena 3D no mundo real, e como já foi referido, contém informações sobre a cena capturada que são indispensáveis, e portanto necessário proceder à sua extração, quando se fala em visão computacional. Para obter essas informações, é necessário obter um modelo que represente a câmara e que descreva como um ponto 3D no mundo é transferido para o plano da imagem na forma de um ponto 2D.

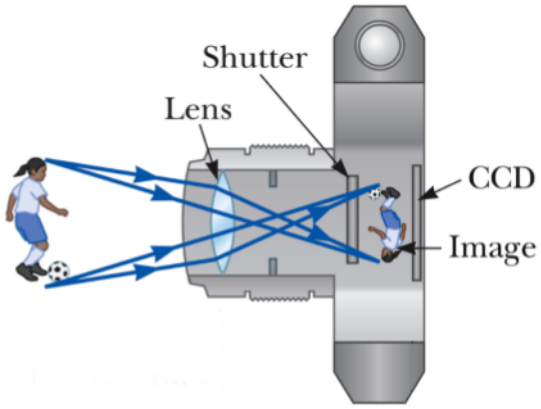


Figura 3.1: Componentes de uma câmara [44]

Existe um modelo relativamente simples e usado comummente em visão computacional, o ***Pinhole Camera Model.*** Na figura 3.2 está representado este modelo e a transformação entre um objeto 3D e um plano, sendo que este plano é o **plano da imagem**, o ponto ***O*** é o ***pinhole*** ou **centro da câmara**. A distância entre o plano da imagem e o *pinhole* é a distância focal, ***f*** [45]. Usando este modelo é possível assumir que um pixel na imagem (x e y), terá apenas uma representação no mundo (X, Y, Z).

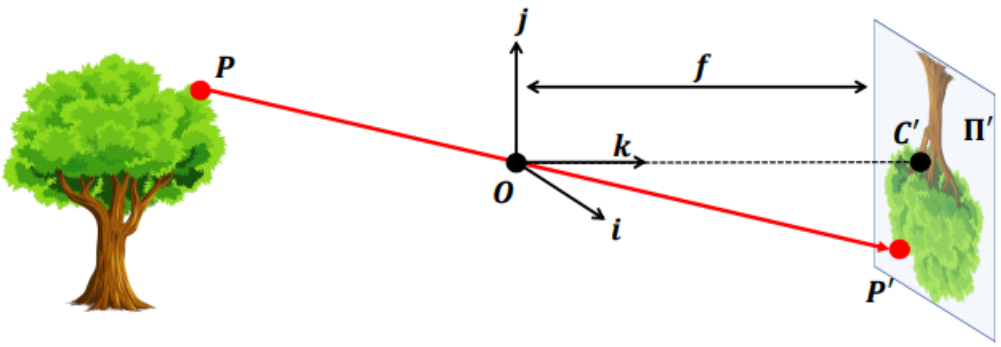


Figura 3.2: Modelo *pinhole* [45]

3.1.2 Ópticas e distorção

O modelo *pinhole* assume que a luz refletida pelos objetos do mundo passa por um ponto único sem sofrer alterações, ou seja assume que é capturada uma imagem perfeita.

Contudo, no mundo real, não é possível existir uma abertura infinitamente pequena. Um dos efeitos que uma maior abertura causa na imagem é a perda de definição. Embora se pense que uma abertura o mais pequena possível é sempre preferível, este pode não ser o caso, visto que com a diminuição da abertura a imagem fica mais nítida, mas mais escura. Na figura 3.3 está representado este efeito da mudança de tamanho da abertura na imagem [45].

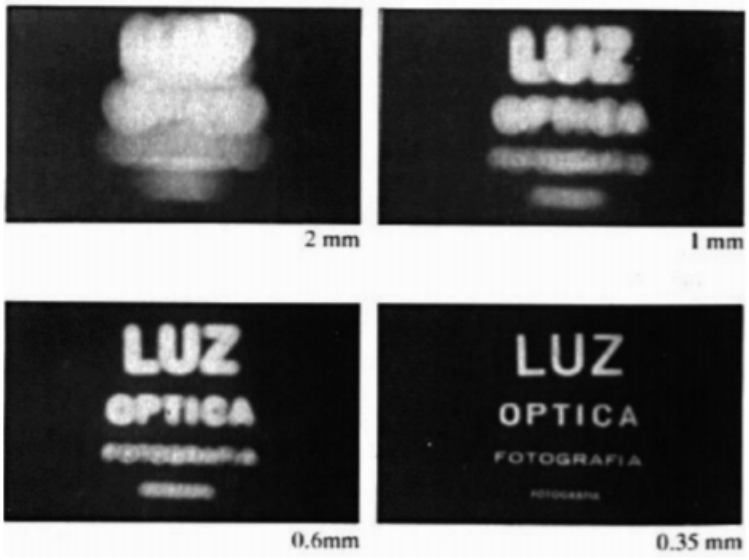


Figura 3.3: Efeitos da variação do tamanho da abertura numa imagem [45].

Este conflito entre nitidez e luminosidade é mitigado pela existência de ópticas que podem focar ou dispersar a luz. Para além de focar a luz no sensor, estas servem para proteger o sensor, podendo ser estáticas ou móveis para permitir ajustar o foco e o zoom da imagem. Para além disso as ópticas focam todos os raios de luz paralelos ao eixo óptico num ponto, o **ponto focal**, sendo que a distância entre o ponto focal e o centro da lente é conhecido por **distância focal**, $\boldsymbol{f}$ (fx, fy) [45].

A presença de uma óptica, introduz alguns efeitos de distorção que vão afetar a replicação do mundo na imagem, sendo que existem dois tipos de distorção: radial e tangencial [46].

A distorção radial ocorre quando os raios de luz sofrem uma refracção maior perto das extremidades das lentes do que no seu centro, provocando uma deformação do objeto no plano da imagem que se propaga do centro para as extremidades sendo que para modelizar esta distorção são usados os coeficientes de distorção radial.

No caso da distorção tangencial, esta deve-se ao facto de existirem desalinhamentos entre a lente e o sensor, isto é, a óptica e o sensor não estarem completamente paralelos ou alinhados, sendo que esta distorção é modelizada pelos coeficientes de distorção tangencial. Na figura 3.4 estão representados os dois tipos de distorção, tangencial à esquerda e radial à direita.

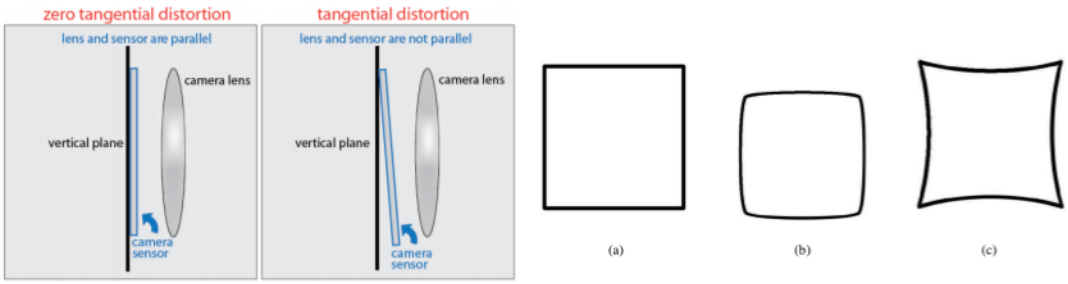


Figura 3.4: À esquerda: distorção tangencial [47], à direita: sem distorção (a), *barrel distortion* (b), *pincushion distortion* (c) [48]

Como já foi referido existe um conjunto de coeficientes que podem ser estimados que modelizam ambas as componentes de distorção, sendo estes representados geralmente pela equação 3.1. No caso da distorção radial os coeficientes são: k_1, k_2 e k_3 e na distorção tangencial são: p_1 e p_2. Sejam as coordenadas de um ponto na imagem original (x, y) e a sua posição na imagem corrigida (x_c, y_c), as equações que corrigem os *pixels* da imagem usando os coeficientes de distorção radial são a 3.2 e a 3.3 [49]. As equações 3.4 e 3.5 mostram como corrigir a distorção tangencial utilizando os coeficientes corretos.

$$Distortion_{coeficients} = [k_1, k_2, p_1, p_2, k_3] \tag{3.1}$$

$$x_c = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6] \tag{3.2}$$

$$y_c = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6] \tag{3.3}$$

$$x_c = x + [2p_1 xy + p_2(r^2 + 2x^2)] \tag{3.4}$$

$$y_c = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \tag{3.5}$$

3.1.3 Parâmetros intrínsecos

Os parâmetros intrínsecos de uma câmara permitem relacionar um ponto no mundo (referencial da câmara) com a sua projeção no plano da imagem e representam as características físicas do sensor. Entre estes parâmetros encontram-se: as componentes em x e y da distância focal, $\mathbf{f}$, as coordenadas (c_x e c_y) do centro focal, $\mathbf{c}$ e os parâmetros η e s, que modelizam o *aspect ratio* e o *skew* dos *pixels* da imagem. Como nas câmaras digitais de hoje em dia os *pixels* são quadrados e não apresentam *skew* estes dois últimos parâmetros são geralmente despresáveis ($\eta = 1$ e $s = 0$).

À matriz de projeção que inclui estes parâmetros intrínsecos e que relaciona um ponto no mundo com a sua projeção na imagem em coordenadas homogéneas dá se o nome de *camera Matrix*, K, e está representada em (3.6).

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & \eta f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

3.1.4 Parâmetros extrínsecos

Com a obtenção dos parâmetros intrínsecos, que expressam as características internas da câmara, é possível relacionar pontos na imagem, com pontos do mundo no referencial da câmara. Contudo, para estes dados extraídos poderem ser relacionados com outros, estes têm que ser transformados do referencial da câmara para outro referencial relativo ou global. Ao contrário dos parâmetros intrínsecos que apenas se alteram se houver mudanças na lente, os parâmetros extrínsecos são diferentes dependendo do referencial ao qual são relativos.

O objetivo passa então por relacionar a posição do sensor da câmara com o referencial necessário. Estes parâmetros podem ser divididos numa componente de rotação e outra de translação. A rotação é definida por uma matriz $\mathbf{R}$, 3x3, que representa as rotações sobre todos os 3 eixos de rotação (x,y,z) usando os 3 ângulos de *Euler*: *Roll* (ϕ), *Pitch* (θ) e *Yaw* (ψ) estando apresentada em 3.7.

$$R(\phi,\theta,\psi) = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \cos\psi\sin\phi\sin\theta - \cos\theta\sin\psi & \sin\psi\sin\phi\sin\theta + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\psi\sin\theta\cos\phi + \sin\phi\sin\psi & -\sin\phi\cos\psi + \sin\phi\sin\theta\cos\theta & \cos\theta\cos\phi \end{bmatrix} \tag{3.7}$$

Uma translação é representada por um vetor ,$\boldsymbol{t}$, e pode acontecer ao mesmo tempo que uma rotação ou não, sendo que se apenas uma simples translação for feita, nada mais é do que adicionar um *offset* nos eixos desejados. Na figura 3.5 encontra-se uma representação da rotação e translação entre os referenciais da câmara e do mundo.

$$t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{3.8}$$

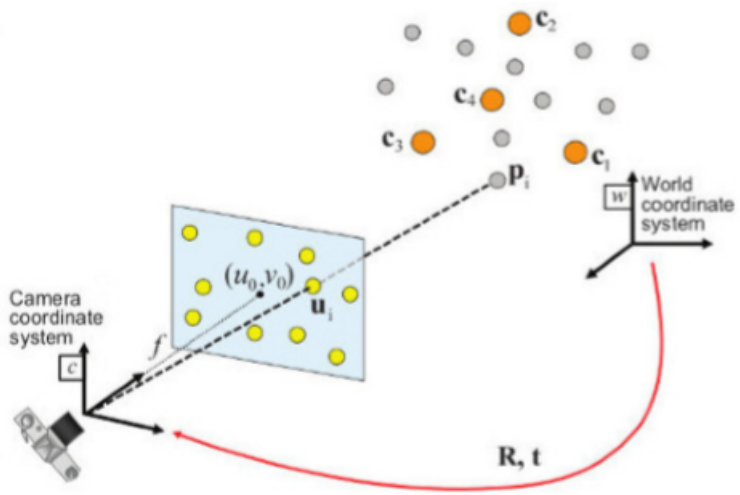


Figura 3.5: Rotação e translação entre referenciais [44]

Quando acontecem ao mesmo tempo uma rotação e translação existe uma forma de notação que facilita a manipulação de transformações, usando uma só matriz, transformação (**T**), em coordenadas homogéneas, representada em 3.9. Então para transformar um ponto *p*, num referencial, num ponto *p'*, noutro referencial, é apenas preciso respeitar a equação 3.10.

$$T = \Big[R|t\Big] = \begin{bmatrix} R_{3x3} & t_{3x1} \\ 0_{1x3} & 1 \end{bmatrix} \tag{3.9}$$

$$p' = T\ p \tag{3.10}$$

3.2 Projeção de objetos

Após obter os parâmetros intrínsecos e extrínsecos de uma câmara, é possível projetar um ponto no mundo, parte de um objeto, para o plano da imagem. A figura 3.6 mostra esta mesma projeção.

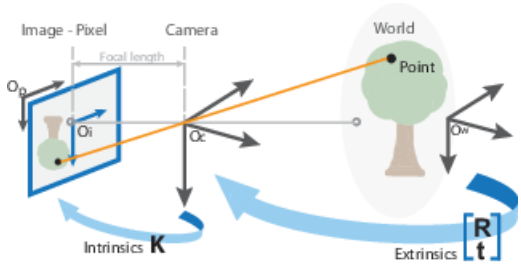


Figura 3.6: Projeção de um ponto do mundo para a imagem [45]

Primeiro é necessário obter a projeção das coordenadas do ponto 3D, $\boldsymbol{p}$, no referencial do mundo para coordenadas no referencial da câmara ($\boldsymbol{p'}$) utilizando os parâmetros extrínsecos (rotação e translação da câmara). Para tal é usada a equação 3.10. Depois é necessário projetar os pontos 3D no referencial da câmara em pontos 2D no plano da imagem usando os parâmetros intrínsecos (matrix K). Então o ponto p na imagem pode ser obtido através da equação 3.11 e 3.12 .

$$p = K\ T\ p' \tag{3.11}$$

$$p = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{3x3} & t_{3x1} \\ 0_{1x3} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.12}$$

3.3 Stereo

Usando apenas uma câmara, na mesma perspetiva em relação ao objeto, não é possível obter informações sobre a profundidade dos pontos no mundo. Para se conseguir obter essas informações é necessário utilizar a triangulação *stereo*. Para se realizar *stereo* é necessário que existam duas ou mais câmeras calibradas com pontos de vista diferentes do objeto que tenham um campo de visão com um mínimo de *overlap*, isto é, existir

alguma sobreposição entre os campos de visão de ambas as câmaras. Isto também é possível com apenas uma câmara, sendo que nesse caso é necessário que o objeto esteja estático e se obtenha várias imagens de diferentes prespetivas desse mesmo objeto (*Structure from Motion* (SFM)).

3.3.1 Geometria epipolar

Um conceito importante na realização de *stereo* é a geometria epipolar. Através deste conceito é possível triangular pontos 3D no mundo usando as imagens de ambas as câmaras.

A figura 3.7 tem como objetivo o auxilio à explicação deste conceito e representa duas câmeras que observam a mesma cena no mundo, com campos de visão que se sobrepõem, sendo ***O*** e ***O'*** os centros de cada câmara, ***X*** o ponto 3D no mundo, e os pontos 2D ***x*** e ***x'*** as projeções do ponto ***X*** na imagem da esquerda e da direita respetivamente.

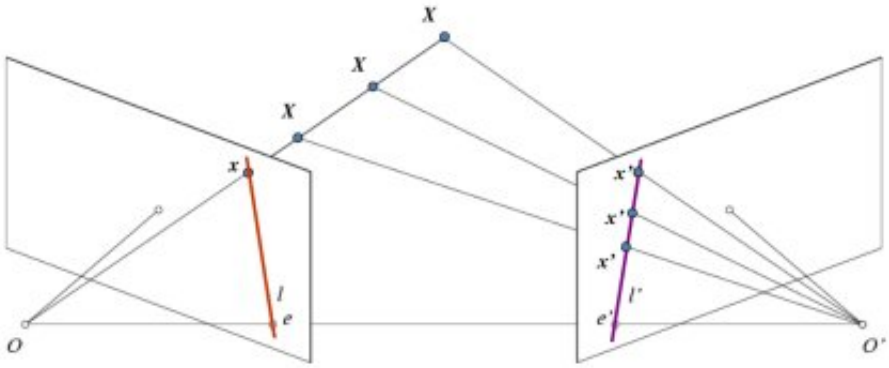


Figura 3.7: Geometria epipolar [50]

Usando apenas a imagem da esquerda é impossível saber a distância à qual o ponto ***X*** está da câmara, porque a localização deste pode ser qualquer uma ao longo do segmento de reta ***OX***, que a projeção no plano da imagem irá ser a mesma. Mas se considerarmos a imagem da direita também, o ponto ***X*** dependendo de onde esteja, vai ser projetado em coordenadas diferentes no plano da imagem da câmara da direita. Portanto o conceito geral é que com esta informação é possível triangular o ponto 3D, e saber a sua localização no mundo [50].

A projeção de todos os pontos do segmento de reta ***OX*** na imagem da direita, forma uma reta no plano da imagem direito que é chamada de **linha epipolar**, ***l'***. Isto

significa que o ponto $\boldsymbol{X}$ na imagem direita têm que estar contido ao longo da linha epipolar, limitando a procura deste ponto a uma linha, em vez de ter que ser pesquisada a imagem toda, o que é uma vantagem em termos de performance e precisão. Este conceito é chamado de **restrição epipolar**, sendo que existe para todos os pontos da imagem da esquerda, que vão ter linhas epipolares correspondentes na imagem da direita e vice versa.

A reta que une os centros de uma camâra a outra é chamada de *baseline* e os pontos de interceção com os planos de ambas as imagens é o epipólo, sendo que todas a linhas epipolares passam pelo epipólo.

Existem duas matrizes muito importantes no que toca à encontrar as linhas epipolares e os epipólos. A matriz fundamental, **F**, e a matriz essencial, **E**. A matriz essencial contém a informação sobre a translação e rotação da segunda câmara em relação à primeira em coordenadas globais. No caso da matriz fundamental, esta contém a mesma informação de rotação e translação, bem como os intrínsecos de cada uma das câmaras, de forma a ser possível relacionar as duas câmaras em coordenadas de *pixels*, isto é, esta matriz mapeia um ponto numa imagem numa linha epipolar na outra imagem. A matriz fundamental pode ser estimada usando pontos correspondentes entre as duas imagens, através do algoritmo de oito pontos [50].

3.4 Processamento de imagem

Uma imagem de alta resolução contém enormes quantidades de informação, sendo que a maioria é informação inútil. Para extrair toda a informação de uma imagem de forma a se retirar apenas as informações úteis, seria preciso analisar detalhadamente cada imagem, o que iria resultar num processo lento e custoso em termos computacionais. Para reduzir os tempos de computação foram desenvolvidas várias técnicas que se focam em extrair pontos de interesse na imagem, *features*, de uma maneira rápida e "superficial". Depois de identificados estes pontos, são analisados mais detalhada e profundamente de forma a extrair toda a informação possível que seja fidedigna.

3.4.1 Deteção de *features*

O conceito de *features* baseia-se numa zona da imagem que difere do que a rodeia, e que é de interesse para resolver tarefas de processamento de imagem. A escolha de *features* depende da finalidade da aplicação, o que leva a que existam algoritmos especializados em detetar certos tipos de *features*. Mudanças de intensidade de *pixels*,

gradientes de cor e texturas são as propriedades mais observadas de forma a detetar pontos de interesse. Depois de extraídas, estas têm que ser armazenadas juntamente com um descritor, que permita uma fácil identificação e comparação entre *features* de diferentes imagens.

Existem vários métodos de deteção de *features*, desde os mais simples aos mais elaborados, entre eles:

- *FAST corner detector*[51];
- *Binary Robust Independent Elementary Features* (BRIEF) [52];
- SURF [53];
- SIFT[54];
- ORB[55];

Nas subsecções seguintes vão ser descritos apenas alguns dos métodos referidos, sendo que foram escolhidos os mais relevantes.

3.4.2 *Scale-Invariant Feature Transform* (SIFT)

Os métodos de deteção de features tais como o Harris corner detector e o FAST, são métodos invariáveis com a rotação, isto é, mesmo que a imagem apareça rodada um canto não deixa de ser um canto. Isto pode não ser verdade se existir uma mudança de escala na imagem, o que origina uma necessidade de criação de um método que seja invariante com a escala da imagem. D.Lowe desenvolveu, em 2004, um método, o SIFT, que como o nome indica é um algoritmo que é invariante com a escala. Existem 5 passos principais no algoritmo SIFT [54]: *scale-space extrema detection*, *keypoint localization*, *orientation assignement*, *keypoint descriptor* e *keypoint matching*.

No primeiro passo, deteção de extremos *scale-space*, é feita uma filtragem de *scale-space*, onde é encontrada uma *Laplacian of Gaussian* (LoG) para a imagem com valores de σ distintos. Encontrando o máximo local na escala e no espaço, é obtida uma lista de valores (x, y, σ) que são potenciais pontos de interesse nas coordenadas (x, y) com um fator de escala σ. Como a LoG tem um custo computacional elevado, é usada uma aproximação, que é a diferença de Gaussianas. Esta diferença é usada em diferentes oitavas, como representado na figura 3.8. Depois de encontradas as diferenças de Gaussianas é feito outra procura por extremos nas imagens, como mostra a figura 3.9, onde cada

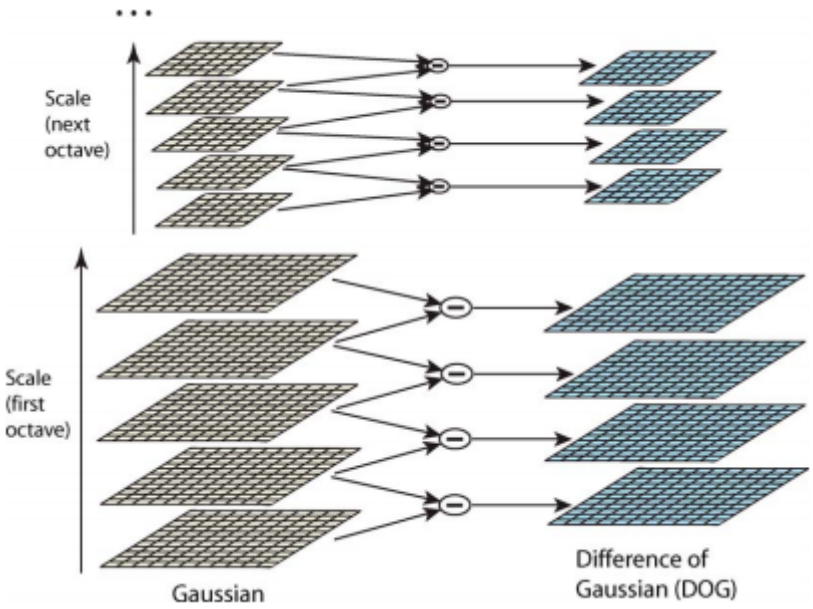


Figura 3.8: Deteção de extremos [54]

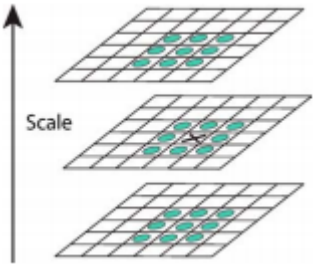


Figura 3.9: Comparação com *pixels* vizinhos [54]

pixel é comparado com os seus 26 vizinhos nas regiões 3x3 na escala atual e adjacentes (superior e inferior).

Depois de encontradas as potenciais localizações dos pontos de interesse, estas são refinadas usando expansões de séries de Taylor, também são removidos todos os *edjes* através de um detetor de Harris, resultando assim na eliminação de todos os pontos com contraste baixo e que sejam *edjes*, restando apenas os mais fortes pontos de interesse.

O passo seguinte é atribuir uma orientação a cada ponto de interesse para garantir a invariância à rotação da imagem. Uma região à volta do ponto é considerada, com

tamanho dependendo da escala do ponto, e de seguida é calculada a magnitude e direção dessa área.

De seguida é criado um descritor para o ponto de interesse, baseado nos vizinhos numa área de 16x16 à volta do ponto. É gerado um histograma de orientação de oito direções em cada sub-bloco de 4x4, sendo representado o descritor através de um vetor.

Por fim, para relacionar *features* em diferentes imagens, os pontos de interesse são correspondidos identificando os seus vizinhos mais próximos.

3.4.3 *Speeded-Up Robust Features* (SURF)

Uma das limitações do método SIFT é o elevado tempo de execução, o que o torna inviável para uso em aplicações em tempo real. Foi introduzido em 2006, um aperfeiçoamento ao método SIFT, um método que é uma versão mais rápida do mesmo. Isto foi atingido por aproximar a diferença de Gaussianas (que no método SIFT já era uma aproximação à LoG) a *Box Filters*.

A convolução utilizando *Box Filters* pode ser facilmente obtida com a ajuda de imagens integrais e pode ser realizada em paralelo para diferentes escalas com alto rendimento.

Para algumas aplicações onde não é necessário uma invariância à rotação, o método SURF permite ignorar a orientação das *features*, melhorando ainda mais a velocidade de processamento e mostrando grande robustez em rotações até 15°. Isto é chamado de U-SURF [56].

O descritor de *features* do método SURF pode ser estendido para uma dimensão de 128, em vez dos habituais 64, dobrando o numero de *features*, sem aumentar muito a complexidade computacional [56].

Como mostra a figura 3.10 este método, tal como as suas variantes, apresentam uma melhoria significativa em termos de tempo de computação em relação ao método SIFT, apresentando uma performance similar. O método SURF é bom a lidar com imagens desfocadas e com rotações, mas não tão eficaz no que toca a imagens com mudança de perspectiva e iluminação.

	U-SURF	SURF	SURF-128	SIFT
time (ms):	255	354	391	1036

Figura 3.10: Comparação dos tempos de execução de vários métodos na análise de uma imagem [53]

3.4.4 *Oriented FAST and Rotated BRIEF* (ORB)

Este método é uma mistura de dois métodos, o detetor de *features* FAST e o descritores BRIEF para melhorar o uso de memória e ter um rendimento mais alto em termos de deteção e correspondência de *features*. Foi pensado como uma alternativa mais rápida e precisa aos detetores patenteados referidos anteriormente, SIFT e SURF.

Para resolver os problemas de memória presentes nos métodos SIFT e SURF, foi usado o detetor de *features* FAST, para detetar pontos de interesse mais rapidamente com um uso de memoria mais reduzido, com a desvantagem de de ser um detetor que não contém uma componente de orientação.

Para obter a invariância à rotação, é proposto um método eficaz na obtenção da orientação do canto: É calculada a intensidade ponderada do centróide do fragmento da imagem que contém o canto no centro. A direção do vetor deste canto para o centróide fornece a orientação. Para melhorar a invariância da rotação, são calculados momentos com x e y, que devem estar numa região circular de raio r, em que r é o tamanho do fragmento [57].

Como já foi referido neste método são usados os descritores BRIEF, sendo que estes apresentam uma performance muito pobre com a existência de rotação. Então no método ORB são "rodados"os descritores BRIEF de acordo com a orientação dos pontos de interesse [57].

O método ORB é muito mais rápido do que os detetores SIFT e SURF [55], e os descritores ORB apresentam melhor performance que os do método SURF, tornando o ORB uma boa escolha para sistemas com pouca capacidade de processamento.

Esta página foi intencionalmente deixada em branco.

Capítulo 4

RTAB-Map & Meshroom

Neste capítulo vão ser descritos ambas as ferramentas usadas para realizar o mapeamento e reconstrução 3D. Vão ser abordadas as técnicas utilizadas, bem como todos os passos realizados para obter a reconstrução, tanto no programa *Meshroom*, como no *Real-Time Appearance-Based Mapping* (RTAB-Map).

4.1 RTAB-Map

O RTAB-Map é uma abordagem de diferentes variantes do problema de *Simultaneous Location and Mapping* (SLAM): RGB-D, *Stereo* e LiDAR baseado em grafos. É distribuído como uma biblioteca *open-source* desde 2013 e foi inicialmente conhecido pela abordagem de gestão de memória que permite a execução de operações de SLAM de larga escala ou longa duração, através da deteção de *loop-closures* baseada na aparência do mapa ("*appearance-based*"), em tempo real.

Recentemente (2019) o RTAB-Map foi alargado ao SLAM, o que permitiu a sua verdadeira aplicação prática para múltiplos robôs e plataformas móveis. Apesar de ter começado como uma biblioteca C++, atualmente está disponível em formato de *package* ROS[1] (*Robot Operating System*). Nas seguintes secções será evidenciada a evolução desta ferramenta nas suas principais contribuições: gestão de memória, *loop-closure* e SLAM.

4.1.1 Mecanismo de gestão de memória e *loop-closures*

O RTAB-Map surgiu de uma implementação de um método de gestão de memória que permitiu a deteção de *loop-closures* em tempo real. O problema do *loop-closure*

[1]Disponível em: `https://wiki.ros.org/rtabmap_ros`.

[58, 59] consiste na determinação de locais onde o robô/plataforma esteve anteriormente depois de um certo tempo de navegação, se esta determinação for assertiva é possível melhorar significativamente a precisão da estimação da *pose*.

A necessidade de um método de gestão de memória advém das características do problema de SLAM, isto é, à medida que se avança no mapa, o tempo de processamento para processar novas observações aumenta, dado a expansão, quer em tamanho, quer em complexidade, do mapa, o que pode condicionar a exequibilidade em tempo real. Assim, o RTAB-Map veio colmatar a inexistência de métodos de gestão de memória capazes de realizar *loop-closure* em tempo real para SLAM de longa duração ou longo percurso.

Inicialmente, a metodologia implementada consistia em manter as localizações recentes, bem como as localizações frequentemente observadas na memória de trabalho do robô (*Working Memory* (WM)), e transferir as restantes para a memória de longo prazo (*Long-Term Memory* (LTM)) [60]. Quando é encontrada uma correspondência entre a localização atual e uma localização presente na WM, é possível aceder à memória LTM, de modo a verificar associações e actualizá-las em caso afirmativo.

O critério utilizado para limitar as localizações existentes na WM (mantendo este valor constante ao longo do percurso) é o tempo de processamento, ou seja, é mantido o número máximo de localizações que permite satisfazer a restrição do tempo real. Assim, se o número de localizações no mapa tornar o tempo de processamento de cada ciclo superior ao *threshold* do tempo real esta abordagem transfere as localizações com menor probabilidade de causar *loop-closures* para a LTM de modo a que não sejam consideradas a cada ciclo de processamento. A memória de longo prazo é apenas verificada na existência de correspondências entre a localização atual e as presentes na memória de trabalho do robô, para que seja possível considerar/adquirir as localizações vizinhas.

Este modelo foi posteriormente consolidado [61], passando a representar cada localização por uma imagem de referência, um tempo de ocorrência (idade) e um peso, e relacionando todas as localizações num grafo, quer por proximidade (vizinhança), quer por correspondências de *loop-closure*. Embora as localizações da LTM não sejam consideradas na deteção de *loop-closures*, é importante decidir corretamente as que são enviadas para a LTM e, portanto, a transferência de localizações também foi alterada:

- **Abordagem inicial** - *First-In First-Out* (FIFO), descartam-se as observações mais antigas à medida que é necessário transferir para a LTM de modo a garantir a execução em tempo real.
- **Abordagem recente** - baseada na hipótese de que as localizações mais frequentemente observadas têm maior probabilidade de causar *loop-closures*.

É possível perceber que a abordagem inicial estabelece um número máximo de localizações que podem ser observadas durante a exploração e, no caso do tempo de processamento atingir o *threshold* do tempo real antes da deteção de *loop-closures*, poderia dar-se o caso de nunca serem encontradas correspondências. Já na abordagem recente, é escolhida uma amostra aleatória da WM, contudo, apenas são mantidas nesta memória as localizações de maior suscetibilidade a serem revisitadas. Esta probabilidade tem por base a hipótese previamente descrita, e é representada pelo peso, que é proporcional ao número de vezes que a localização foi observada. Assim, neste modelo, quando é necessário transferir uma localização da WM para a LTM, é escolhida a localização com menor peso (e mais antiga no caso de existirem múltiplas com o menor peso associado). Na figura 4.1 é demonstrada a abordagem recente para o modelo de gestão de memória.

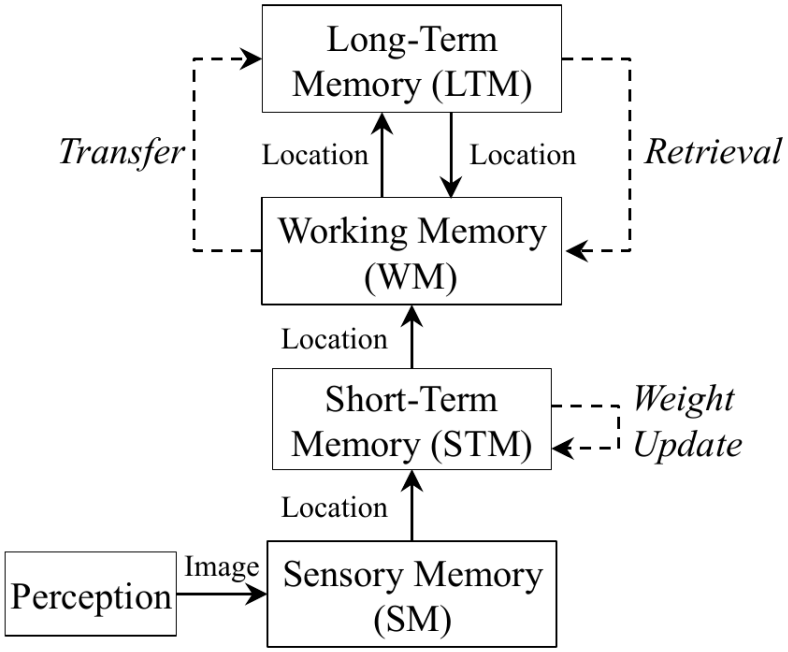


Figura 4.1: Mecanismo de gestão de memória do RTAB-Map [61]

- **Modelo de perceção:** Adquire a imagem e envia para a memória sensorial (*Sensor Memory* (SM)).
- **Memória Sensorial:** Examina a imagem e extrai *features* importantes para a deteção de *loop-closures*. De seguida cria uma localização associada à imagem e envia esta informação para a memória de curto prazo (*Short-Term Memory*

(STM)).

- **Memória de curto prazo:** Atualiza as localizações, nomeadamente o seu peso, isto é, se o processo "*Weight Update*"detetar que a nova localização é semelhante à anterior, funde as duas numa única localização, incrementando o seu peso associado. Esta componente de memória permite assim observar as similaridades entre imagens ao longo do tempo. O tamanho da STM varia com a velocidade e *frame rate* do robô, quando a memória fica totalmente preenchida, a última localização é enviada para a WM.

- **Memória de trabalho:** Atribui uma probabilidade, à localização recebida, de constituir um *loop-closure*, comparando-a com as restantes através de um filtro Bayesiano discreto.

- **Reaquisição (*Retrieval*):** Caso haja uma acentuada probabilidade de *loop-closure*, as localizações vizinhas de maior peso (que não estão na WM) são trazidas da LTM para a WM, aumentando assim a suscetibilidade de identificar *loop-closures* em localizações futuras.

- **Transferência (*Transfer*):** Este processo ocorre quando o tempo de deteção é superior ao *threshold* do tempo real, englobando a transferência para a LTM da localização menos observada (com menor peso) e mais antiga.

Este mecanismo de gestão de memória garante assim a execução em tempo real, independentemente da escala do ambiente a percorrer, dado que mantém constante as localizações a verificar a cada ciclo.

4.1.2 Extensão ao SLAM e suporte ROS

O crescimento do RTAB-Map permitiu a implementação de SLAM em múltiplas plataformas robóticas, sobretudo pelo facto de introduzirem diferentes abordagens: visual e baseado em LiDAR [62]. Até então, só existiam uma ou outra abordagem, impedindo a comparação de resultados entre elas.

Esta evolução foi possível porque o RTAB-Map é independente do tipo de odometria que é introduzida no sistema de navegação, ou seja, este *software* permite o *input* de informação proveniente de odometria clássica, recorrendo à informação incremental dos *encoders* das rodas, odometria visual, ou por LiDAR. Esta funcionalidade implica que o RTAB-Map possa ser utilizado para implementar os diferentes tipos de odometria, bem

como múltiplas abordagens em simultâneo, possibilitando uma comparação de diferentes configurações a aplicar num robô real.

A figura 4.2 ilustra o funcionamento da mais recente versão do nó ROS do RTAB-Map. Os *inputs* podem ser de imagem (*stereo* ou RGB-D) juntamente com um tópico de calibração (tipicamente denominado de *camera_calibration*), bem como um nó de odometria, de 3 ou 6 graus de liberdade, e uma transformação do referencial do sensor para o referencial base do robô. Podem ainda ser introduzidos tópicos com *scans* 2D ou *pointclouds* 3D de um LiDAR. Todas as mensagens disponíveis são sincronizadas com base no tempo ROS e enviadas para o algoritmo de *graph-SLAM*.

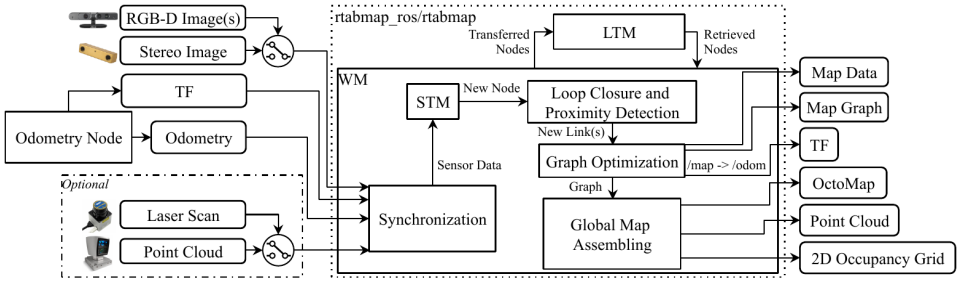


Figura 4.2: Diagrama de blocos do nó ROS *rtabmap* [62]

Esta versão evoluída do RTAB-Map, permite assim fornecer uma integração com o ROS, e consequentemente realizar operações de calibração, sincronização, bem como efetuar as transformações entre sensores, e gerar a informação gráfica 2D e 3D de cada um dos sensores e do mapa. O facto de ser uma aplicação de SLAM multi-sensorial possibilita que os utilizadores possam criar e testar os seus protótipos com diferentes configurações de sensores e capacidade de processamento, de forma a comparar a *performance* do mesmo para diversos cenários de aplicação.

4.2 Meshroom

O *Meshroom* é um *software* de reconstrução 3D baseado nos algoritmos fornecidos pela *framework AliceVision*. Esta aplicação permite executar as várias etapas da *pipeline* da fotogrametria (visíveis na figura 4.3) dado o seu sistema de nós que torna o utilizador capaz de executar cada um deles individualmente, explicando claramente qual o seu *input* e *output*.

Os principais objetivos deste programa consistem na possibilidade de que qualquer utilizador seja capaz de obter com facilidade modelos 3D a partir de um conjunto de

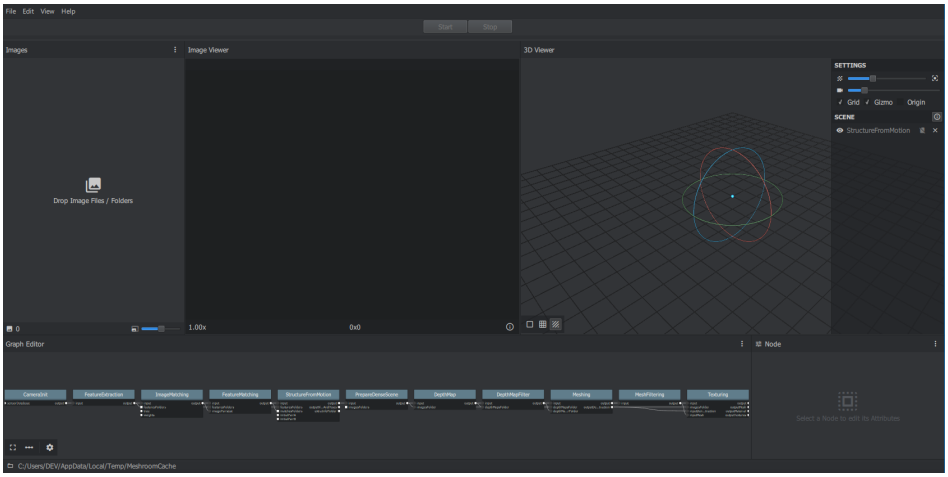


Figura 4.3: Interface gráfica do *Meshroom*.

imagens, bem como fornecer aos utilizadores mais avançados, como investigadores ou designers, uma solução às suas necessidades técnicas ou criativas.

4.2.1 *Framework* AliseVison

O *AliceVision* [63, 64] é uma *framework open source* de visão computacional que utiliza fotogrametria para reconstrução 3D, modelização de imagens e do movimento da câmara. A fotogrametria é a ciência que aborda métodos de aquisição de informação das componentes físicas de objetos e do ambiente em torno dos mesmos, através de medições provenientes de imagens. Esta metodologia permite recuperar a profundidade perdida pela projeção de uma cena tridimensional num plano 2D de uma imagem.

4.2.2 *Pipeline* da fotogrametria do *AliceVision*

O *AliceVision* contém os algoritmos de visão computacional necessários para efetuar um conjunto de operações relevantes à reconstrução 3D, sendo que estas operações estão inseridas numa *pipeline* denominada de "*pipeline* da fotogrametria"representada na Figura 4.4.

4.2.2.1 Extração de *features* (*Feature extraction*)

Nesta etapa da *pipeline* é importante encontrar conjuntos de pixeis que se mantenham no plano da imagem durante algum tempo no processo de aquisição para que, desta forma, uma *feature* tenha o mesmo descritor em diferentes imagens. O método

Figura 4.4: *Pipeline* da fotogrametria do *AliceVision* [65]

mais comum para a deteção de *features* é o SIFT (*Scale Invariant Feature Transform*) [66], cujo objetivo é identificar um grupo de áreas discriminativas numa determinada imagem que, ao serem comparadas com outras áreas de uma outra imagem permitem obter diferenças de rotação, translação e escala. Esta é uma das técnicas dos algoritmos do *AliceVision*, bem como o *Akaze* [67] e o CCTag. Esta *framework* efetua ainda uma filtragem de modo a controlar o número de *features* extraídas dado que a sua densidade pode variar drasticamente com as alterações da complexidade da textura obtida do meio.

4.2.2.2 Associação de imagens (*Image Matching*)

Nesta fase o objetivo passa por encontrar imagens que partilham o mesmo conteúdo, isto é, encontram-se a recolher informações sobre a mesma área num determinado cenário. No entanto, se o conteúdo for verificado através da análise à correspondência entre *features* o processo torna-se demasiado complexo. De forma a evitar esta situação, o *AliceVision* usa técnicas de recuperação de imagem que utilizam descritores compactos para encontrar conteúdo semelhante entre imagens. Um dos métodos usados para gerar o descritor de imagem é o *VocTree* (*Vocabulary Tree*) [68] que permite que os descritores das *features* sejam inseridas num esquema em árvore onde a classificação é realizada pela comparação entre os descritores em diferentes nós.

4.2.2.3 Associação de *features* (*Feature Matching*)

Depois de associar as imagens, o *AliceVision* foca-se em calcular as correspondências entre os diferentes pares de imagens gerados. Começa-se por calcular, por cada *feature* numa determinada imagem, uma lista de candidatos à correspondência numa outra imagem, por fotometria. No entanto, nesta fase existe um elevado número de *outliers*, sendo necessário remover maus candidatos. Esta remoção é realizada escolhendo os dois descritores mais próximos e aplicando um determinado *threshold* entre eles. As coordenadas das *features* são de seguida utilizadas para implementar um filtro geométrico (através de geometria epipolar), e as *features* resultantes são inseridas numa *framework* de RANSAC (*Random Sample Consensus*) [69] de forma a remover os *outliers*.

4.2.2.4 SFM (*Structure-from-Motion*)

O objetivo desta tarefa é calcular a relação geométrica entre as observações geradas pelo conjunto de imagens recebidas de modo a obter o modelo da estrutura 3D juntamente com a posição, orientação e calibração da câmara a cada medição. O processo de reconstrução é incremental, isto é, o SFM do *AliceVision* calcula uma reconstrução inicial a partir de duas observações, e a reconstrução é alargada à medida que se adicionam novas imagens.

A primeira operação executada nesta fase é a fusão entre todos os pares de imagens em "*tracks*"que representam pontos no espaço observados por múltiplas observações. De seguida o algoritmo escolhe um par de imagens inicial que levará à melhor reconstrução final possível, baseado nos seguintes requisitos:

1. O par deve maximizar o número de correspondências entre as restantes imagens.
2. O ângulo entre o par deve ser amplo o suficiente para fornecer informação geométrica relevante.

A partir do par de imagens escolhido é calculada a matriz fundamental, considerando uma delas como a origem do referencial, adicionalmente, e tendo em conta que se conhece a pose de duas imagens e a matriz fundamental, é possível triangular as *features* do plano da imagem para pontos 3D.

O passo seguinte trata-se de um algoritmo denominado de "*Best Views Selection*"cujo objetivo é, tal como o nome indica, escolher as imagens que contêm correspondências suficientes com as *features* já projetadas, obtendo-se assim novas observações e consequentemente novas *poses*. À medida que se faz a triangulação de novos pontos 3D, vão aparecendo novos candidatos para o algoritmo de "*Best Views Selection*", e o processo volta, recursivamente a projetar novas *features* e a remover *landmarks* que se consideram inválidas, até que não se observem medições.

4.2.2.5 Estimação da Profundidade (*Depth Map Estimation*)

Depois de obter as poses das câmaras no SFM, o algoritmo da *AliceVision* calcula o valor de profundidade de cada pixel através de *Semi-Global Matching* (SGM) [70]. Este processo começa por escolher as câmaras mais próximas, bem como um conjunto de planos paralelos baseados na interseção dos eixo ótico com os *pixels* das imagens vizinhas selecionadas, com vários candidatos a profundidade de cada pixel. De seguida estima-se uma relação de similaridade entre eles, e obtém-se uma acumulação de similaridades

através de uma relação ZNCC (*Zero Mean Normalized Cross-Correlation*). O valor de profundidade é finalmente obtido através de uma função de custo que encontra o mínimo local do volume acumulativo de similaridades e é inserido num mapa de profundidades.

4.2.2.6 Reconstrução (*Meshing*)

Nesta etapa, o objetivo é gerar uma superfície densa representativa do cenário observado pelas diferentes câmaras. Os mapas de profundidade são fundidos numa *octree*, onde os valores de profundidade compatíveis são unidos numa única célula. De seguida usam-se duas técnicas distintas de *meshing*: Triangulação de Delaunay e *Graph Cut Max-Flow* de modo a gerar um volume que representa a superfície. Finalmente aplica-se uma filtragem de Laplace para remover artefactos locais.

4.2.2.7 Obtenção de textura (*Texturing*)

O objetivo deste passo é atribuir uma textura à *mesh* gerada no passo anterior, implementando uma técnica de mapeamento UV [71]. Para cada triângulo da *mesh*, são criados candidatos à textura tenho em conta a informação de visibilidade presente nos vértices. As observações com um mau ângulo em relação à superfície são filtradas de modo a obter uma média aceitável para a textura dos pixeis.

Esta página foi intencionalmente deixada em branco.

Capítulo 5

GeoTec System

Tendo em conta a falta de uma estrutura capaz de suportar todos os sensores e as condições adversas no interior da mina, foi necessário criar uma estrutura que permita mapear os ambientes subterrâneos desejados. Neste capítulo é endereçado o levantamento dos requisitos do sistema que se pretende desenvolver e as opções de projeto tomadas para cada subsistema.

5.1 Requisitos do sistema

Considerando que o principal objetivo deste sistema é a recolha de dados que permitam o mapeamento do interior do ambiente subterrâneo, torna-se necessário que o sistema seja capaz de cumprir alguns requisitos, tais como:

- Integrar um conjunto de sensores de forma a permitir o mapeamento e reconstrução 3D do ambiente (IMU, LiDAR, par stereo);
- Ter uma unidade de processamento CPU com capacidade elevada de processamento e escrita em disco, para guardar os dados de todos os sensores. Essa unidade deverá também apresentar vários *Universal Serial Bus* (USB), portas série e ethernet de forma a permitir a conexão dos vários sensores do sistema.
- Sincronização dos dados obtidos por todos os sensores com o *clock* interno do PC;
- Ser mecânicamente robusta e leve de forma a permitir um fácil transporte;
- Ter uma estrutura modular e de fácil adaptação, para permitir alterações tanto na forma como em qualquer componente do sistema;

- Permitir o deslocamento do sistema dentro dos túneis e galerias, tendo para esse fim que se apresentar uma estrutura estreita e alta;

5.2 Arquitetura de Hardware

Na arquitetura de hardware do sistema, representada na figura 5.1, estão apresentadas todas a conecções entre os diferentes componentes. Estão representados todos os sensores do sistema, nomeadamente, IMU, LiDAR, GNSS e câmaras, tal como o *Central Processing Unit* (CPU), o módulo de sincronização/trigger e o armazenamento.

Todos os sensores apresentam diferentes interfaces e protocolos de comunicação, sendo que o IMU e LiDAR usam USB 2.0, as câmaras comunicam através de *Gigabit Ethernet* (GigE), e o GNSS está conectado ao PC por RS-232. O CPU do sistema será, então, responsável por configurar todos os sensores que o necessitem, adquirir os dados enviados por estes e armazená-los em memória.

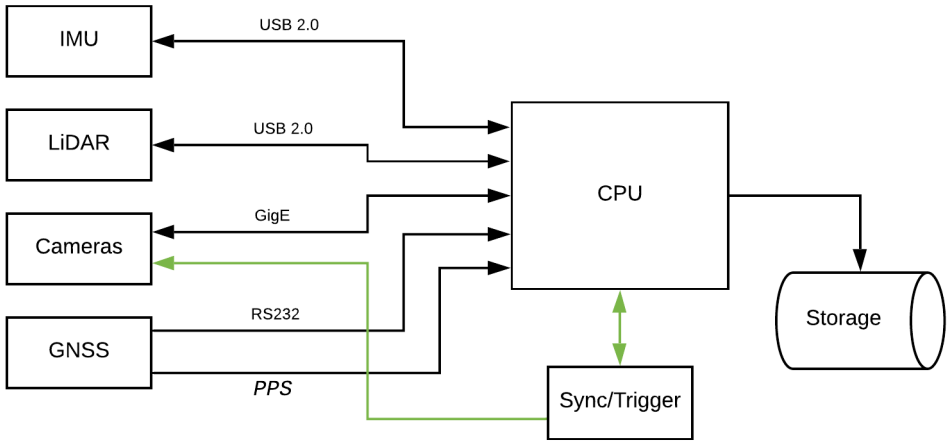


Figura 5.1: Arquitetura de hardware do sistema.

Já foi discutido anteriormente que a sincronização de todos os dados é de extrema importância. Para proceder à sincronização do relógio do PC, este recebe um sinal *Pulse Per Second* (PPS), vindo do recetor GNSS, numa das suas portas COM. Este sinal vai ser recebido e enviado ao programa de sincronização *Chrony*, cujo funcionamento se encontra descrito na secção 6.2, que se encarregará de sincronizar o relógio do CPU.

O módulo de sincronização/trigger receberá comandos do CPU, e será o módulo

responsável pelo trigger das câmeras, que será gerado por hardware como foi explicado na secção 5.3.2. Este módulo também será responsável por enviar o sinal de trigger para uma da portas COM do PC de forma a que os tempos nos quais o trigger foi gerado sejam recebidos pelo CPU e registados.

Este procedimento torna-se necessário para a sincronização exata das imagens, uma vez que, quando a câmara recebe o sinal de trigger, esta adquire imediatamente uma imagem naquele instante, mas, devido ao facto da transferência de dados não se mostrar instantânea entre a câmara e o PC, essa imagem não irá ser recebida no PC no instante no qual foi tirada. Isto provoca que o instante que é registado no PC quando a imagem é recebida não seja o mesmo que o instante no qual a imagem foi tirada. Portanto o sinal de trigger é enviado para o CPU para ser guardado e para posteriormente se poder proceder à associação dos tempos corretos de aquisição a cada imagem.

5.3 Opções de Projeto

De forma a cumprir os requisitos referidos anteriormente é necessário proceder a algumas opções de projeto, nesta sub-secção são fundamentadas essas escolhas.

5.3.1 Setup de sensores

Para se conseguir recolher dados suficientes para realizar o mapeamento e reconstrução 3D dos ambientes subterrâneos, foi idealizado o seguinte setup de sensores e componentes:

- IMU;
- CPU;
- LiDAR;
- Recetor GNSS;
- Duas câmaras idênticas para realizar stereo;
- Duas ópticas, uma para cada câmara;

As câmaras, que em conjunto com as lentes, serão usadas para realizar stereo, o LiDAR para recolher nuvens de pontos do ambiente e o IMU de maneira a fornecer dados da atitude do sistema, que eventualmente serão fundidos com os dados obtidos por odometria visual. No caso do recetor GNSS, este será usado para inicializar a posição do

sistema no mundo antes de entrar para o ambiente subterrâneo, altura na qual se torna inútil, visto que os sinais de satélite não penetram a superfície terrestre, sendo que, por último, a unidade de processamento será responsável por processar os dados enviados por todos os sensores.

Depois de decidido quais seriam os tipos de sensores base que o sistema teria que incorporar, foi necessário proceder à avaliação e escolha de um sensor específico dentro de cada tipo.

5.3.1.1 IMU

O sensor escolhido para unidade de medida inercial, foi o STIM300 da Sensonor. Este é um sensor robusto, compacto e leve, que apresenta um elevado desempenho. Este conta com três acelerómetros, três inclinómetros e três giroscópios *Micro-Eletric-Mechanical Systems* (MEMS) de elevada precisão.

Este IMU foi escolhido para ser integrado no sistema GeoTec, por apresentar uma relação qualidade/robustez elevada e um tamanho reduzido. Para além disso, este vem calibrado de fábrica de forma a compensar os valores medidos, consoante as variações de temperatura. Apresenta também uma estrutura metálica que para além de ser robusta o torna quase imune a perturbações magnéticas.

Na tabela 5.1 estão descritas mais algumas características do sensor, que se encontra representado na figura 5.2.

Figura 5.2: STIM 300 [1]

Destas características pode-se retirar que o sensor apresenta reduzido valor de instabilidade de bias dos giroscópios e acelerómetros, tal como baixo valor de ruído dos giroscópios.

Tabela 5.1: Características STIM 300 [1]

STIM 300	
Interface	USB 2.0
Peso	55g
Instabilidade de bias dos giroscópios	$(0.3^{\circ}/h)$
Instabilidade do bias dos acelerómetros	0.05mg
Ruído dos giroscópios	$(0.15^{\circ}/\sqrt{h})$
Escala de acelerações	$\pm$ 10g
Escala da taxa angular	$\pm 400^{\circ}/s$ de
Erro no bias dos giroscópios com a temperatura	10 $^{\circ}/h$

5.3.1.2 Unidade de processamento

Como foi referido nos requisitos, a unidade de processamento terá que apresentar uma capacidade de processamento elevada, bem como várias portas para comunicar com os diversos sensores do sistema. Outro requisito importante é o PC ser de reduzidas dimensões, de forma a não sobrecarregar o sistema.

A *board* selecionada foi a CAPA881 da Axiomtek. Esta é uma *Single-Board Computer* (SBC) com um processador *Intel© Core™ i5* de quarta geração, equipada com um disco de armazenamento *Solid State Drive* (SSD) de 256 GB, 8 GB de memória *Random Access Memory* (RAM) e várias portas USB 2.0 e 3.0, Ethernet e COM.

Na figura 5.3 está representado um exemplar da unidade de processamento escolhida, bem como na tabela 5.2 se encontram descritas algumas características deste modelo de SBC.

Uma vez que esta unidade de processamento apenas apresenta duas portas ethernet e visto que uma delas será usada para comunicar com o PC externo, de forma à permitir um maior número de ligações por ethernet foi necessário adicionar um switch ethernet de 5 portas.

5.3.1.3 Sensor óptico e lentes

No caso dos sensores ópticos, os requisitos mais relevantes são: alta resolução, elevado *frame rate*, permitir trigger externo e usar *Global Shutter* como método de captura.

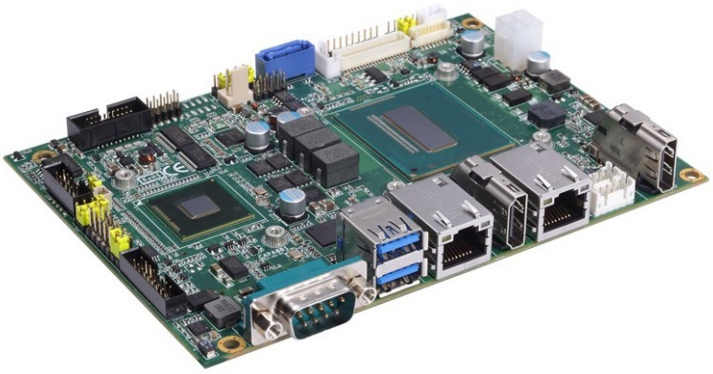


Figura 5.3: CAPA881 Axiomtek [2]

Tabela 5.2: Características CAPA881 [2]

CAPA881	
CPU	*Intel® Core™ i5* 4ª geração
Chipset	*Intel® HM86*
Memória	204-pin SO-DIMM DDR3L-1600
Basic Input/Output System (BIOS)	AMI
COM	2 x RS-232/422/485
USB	2 x USB 3.0 e 1 x USB 2.0
Ethernet	2 x 10/100/1000 Mbps
Display	2 x *High-Definition Multimedia Interface* (HDMI)

Outros requisitos óbvios, mas menos importantes, são: apresentar baixo consumo, ser o mais robusta e pequena possível.

Existe ainda a questão da interface de comunicação com a câmara, sendo que existem vários tipos de interfaces (GigE, USB 2.0, USB 3.0, FireWire, Camera Link).

As câmaras geralmente usadas no laboratório apresentam na sua maioria interface GigE ou USB 3.0, sendo que existem diferenças significativas entre estes dois tipos de interface, nomeadamente em termos de velocidade e taxa de transferência. Apesar da interface USB 3.0 apresentar valores mais elevados nesses dois aspetos, a interface escolhida foi GigE, devido ao facto de que USB 3.0 provoca interferências em sistemas

sensíveis a ruído, como IMUs e recetores GNSS.

Depois de definida qual seria a interface, a escolha do modelo da câmara tornou-se mais simples, visto que são necessários dois exemplares iguais que cumpram os requisitos falados. Os sensores escolhidos foram então, duas câmaras da FLIR, Blackfly GigE, modelo BFLY-PGE-31S4C-C.

Este modelo tem um sensor *Complementary Metal-Oxide-Semiconductor* (CMOS), Sony IMX265 de 1/1.8", uma resolução elevada (2048x1536), um frame rate máximo de 35 *Frames per Second* (FPS), 3.2 Megapixeis e *Global Shutter*. Na tabela 5.3 estão descritas mais algumas características do modelo, que se é apresentado na figura 5.4.

Figura 5.4: Modelo Blackfly GigE - BFLY-PGE-31S4C-C da FLIR [3]

Tabela 5.3: Características BFLY-PGE-31S4C-C [3]

BFLY-PGE-31S4C-C

Interface	GigE	Resolução	2048 x 1536
Peso	36g	Formato do sensor	1/1.8"
Frame Rate	35	Megapixels	3.2
Tipo de Montagem	C-mount	Método de Captura	*Global Shutter*
Tipo de Sensor	CMOS	Consumo	25 W

Depois de escolhidas as câmaras, procedeu-se à escolha de um modelo de lentes que fosse compatível com o das câmaras escolhidas, nomeadamente no tipo de montagem (C-mount) e resolução (3.2 megapixeis). As lentes encontradas foram as *GMTHR23514MCN*

da Goyo Optical. Estas possuem as características necessárias para poderem ser usadas em conjunto com o modelo de câmaras escolhidos. Na tabela 5.4 encontram-se algumas das características deste modelo de ópticas.

Tabela 5.4: Características ópticas GMTHR23514MCN [4]

GMTHR23514MCN			
Distância Focal	3.5 mm	*Minimum Object Distance* (MOD)	200 mm
Formato da lentes	1/2”	Campo de visão	103.34° x 76.54° x 131.4°
Resolução	3 MP	Tipo de Montagem	C-mount
Peso	100 g	Dimensões (DxL)	Ø31 x 31.75

5.3.1.4 LiDAR

O LiDAR escolhido para integrar o sistema, representado na figura 5.5, foi o *Hokuyo UTM-30LX*. Este é um *scanner* laser 2D com alcance de 30m e um ângulo de medição de 270º, compacto e leve, alinhando-se as suas características com os requisitos do sistema. Uma vez que este sensor apenas retorna uma linha 2D em cada varrimento, para se conseguir mapear ambientes 3D é necessário monta-lo de maneira a que, com o movimento do sistema, se consiga alinhar todos os scans de forma a que seja retornado uma nuvem de pontos 3D. No caso deste sistema irá ser montado na vertical a apontar para cima de maneira a mapear os túneis e galerias dos ambientes subterrâneos. Na tabela 5.5 estão apresentadas algumas características do sensor.

Figura 5.5: Hokuyo UTM-30LX [5]

Tabela 5.5: Características do UTM-30LX [5]

UTM-30LX			
Fonte laser	Díodo laser (λ=905nm)	Alcance de deteção	0,1 to 30m, 270°
Resolução Angular	0.25° (360°/1,440 passos)	Interface	USB 2.0
Tempo de scan	25msec/scan	Precisão (0.1 to 10m)	$\pm 30mm$
Ruído de funcionamento	Menos de 25dB	Peso	370 g

5.3.1.5 GNSS

O recetor GNSS escolhido para o sistema foi o UB482 da *Unicore Communications*. Este recetor, representado na figura 5.6, suporta todas constelações de sinal de satélite, como BDS B1/B2, GPS L1/L2, GLONASS L1/L2, Galileo E1/E5b, QZSS L1/L2 e permite usar *Real Time Kinematic* (RTK).

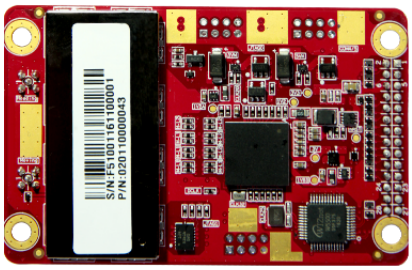


Figura 5.6: Recetor GNSS UB482 [6]

O UB482 adota um desing compacto e suporta navegação inercial, permitindo o uso de duas antenas, sendo que a primeira (*moving*) é usada para estimar a posição e a segunda (*heading*) é responsável pela orientação. Esta é estimada gerando um ângulo entre o vetor criado pelas duas antenas e o vetor que aponta para o norte verdadeiro. A comunicação pode ser feita por ethernet ou atráves das 3 interfaces *Universal Asynchronous Receiver-Transmitter* (UART), nível TTL, presentes no módulo. Na tabela 5.6 estão representadas algumas especificações do UB482.

Tabela 5.6: Especificações do recetor UB482 [6]

UB482	
Canais	432
Constelações	GPS, GLONASS Galileo, BDS QZSS
RTK	Horizontal: 1 cm + 1 ppm Vertical: 1.5 cm + 1 ppm
Interface	UART Ethernet
Dimensões	71 x 46 x 11.4 m
Peso	21 g

5.3.2 *Trigger* das câmaras por *Hardware*

Os sistemas de aquisição multi-câmaras capturam continuamente várias imagens de diferentes perspetivas. Estas imagens precisam de ser capturadas em sincronismo e de receber uma referência temporal (*timestamp*) para poderem ser identificadas cronologicamente e permitirem a correta associação de cada par de imagens. A aplicação destas no mapeamento e reconstrução 3D requer um nível de precisão elevado no sincronismo, para uma correta reconstrução dos cenários. As imagens de diferentes câmaras precisam de ser capturadas no mesmo instante, especialmente numa cena não estática. Neste contexto os mecanismos de sincronização são essenciais, uma vez que são responsáveis pela referência temporal das imagens capturadas.

Existem várias estratégias de sincronização conhecidas, sendo as duas mais importantes [7] :

- Hardware: Este método é baseado na geração de um sinal de trigger dedicado através de um módulo externo de hardware;
- Software: O software de aquisição controla os tempos de captura, mandando um comando específico de software para cada câmara;

Na tabela 5.7 encontram-se uma comparação dos dois métodos.

Um dos requisitos mais importantes do sistema é a sincronização dos dados dos

sensores, uma vez que para os dados recolhidos serem válidos para realizar a reconstrução 3D, é necessário que estejam todos sincronizados temporalmente.

Comparando ambos os métodos, o trigger por hardware torna-se a escolha clara, uma vez que garante uma sincronização com uma precisão na escala dos *micro* segundos. Uma das razões para esta escolha em relação ao *trigger* por software, é o facto dos computadores mostrarem uma capacidade limitada no controlo preciso e sincronizado de hardware, sendo que muitos protocolos de controlo contêm delays imprevisíveis. Uma outra razão é, numa situação em que o CPU estiver saturado, e a tarefa de sincronização não tiver uma prioridade elevada, pode acontecer que o CPU não consiga atender a tarefa nos tempos devidos, e por isso não apresenta a garantia que esta seja executada num tempo específico.

Para obter uma sincronização mais precisa possível, reduzindo também a ocupação do CPU, será utilizado um sistema de hardware externo para o trigger das câmaras. Um sistema parecido já se encontra desenvolvido no laboratório e o processo de adaptação deste será detalhado na secção 6.1.4.

Tabela 5.7: Comparação entre trigger por Hardware e Software [7].

Metric/Method	**Hardware Trigger**	**Software**
Accuracy	100/150 us	ms scale
Need Real-time OS	No	Yes
Implementation	HW	SW
Cost	Low	None
Difficulty	Low	High
Buses	All	All
Cameras	Any with external trigger	Any

5.3.3 *Framework* ROS

O ROS é um *Meta-Operating System*, ou seja, fornece os serviços que se espera de um sistema operativo, incluindo abstração de hardware, controlo de dispositivos de baixo nível, implementação de funcionalidades geralmente utilizadas, envio de mensagens entre processos e gestão de pacotes [72].

Portanto, o ROS é uma *framework*, cujo objetivo é criar um software funcional para todos os sistemas robóticos, fazendo apenas pequenas alterações no código.

O ROS foi originalmente desenvolvido em 2007 pela *Stanford Artificial Intelli-*

gence Laboratory (SAIL) com o apoio do projeto Stanford AI Robot. A partir de 2008, o desenvolvimento foi realizado principalmente pela Willow Garage, um instituto de investigação na área da robótica, com a colaboração de mais de 20 instituições.

A arquitetura do ROS foi projetada e dividida em três secções ou níveis de conceitos:

- O nível do sistema de arquivos (Filesystem);
- O nível do gráfico computacional (Computation Graph);
- O nível comunitário (Community).

No primeiro nível, um grupo de conceitos é usado para explicar como o ROS é formado internamente, a estrutura das pastas e os ficheiros mínimos necessários para funcionar.

O segundo nível, *Computation Graph*, é onde a comunicação entre processos e sistemas acontece. Nesta secção, são abordados os conceitos e os sistemas que o ROS tem, de modo a que este possa lidar com todos os processos e para que possa comunicar-se com mais de um computador.

O último nível é o nível comunitário onde são exploradas as ferramentas e conceitos para compartilhar conhecimento, algoritmos e código de qualquer *developer*. Este nível é importante, já que permite que o ROS possa continuar a crescer rapidamente, devido a um grande suporte da comunidade.

Nas próximas páginas será analisado o nível *Computation Graph*, visto que é a camada que tem mais interesse do ponto de vista de como funciona o ROS.

O ROS cria uma rede na qual todos os processos estão interligados. Qualquer nó no sistema pode aceder a esta rede, interagir com outros nós, transmitir dados para a rede e ver as informações que estes estão a enviar.

Os conceitos básicos neste nível são os nós, o Master, os parâmetros do servidor, as mensagens, os serviços, os tópicos e os bags, sendo que todos fornecem dados para o *graph* de maneiras diferentes:

- Nodes (nós): Os nós são executáveis que podem comunicar com outros processos, usando tópicos, serviços ou o servidor de parâmetros, e onde é realizada a computação. Normalmente, um sistema terá muitos nós com diferentes funções, visto que é preferível ter vários nós que forneçam apenas algumas funcionalidades, em vez de um nó complexo que realize tudo no sistema.

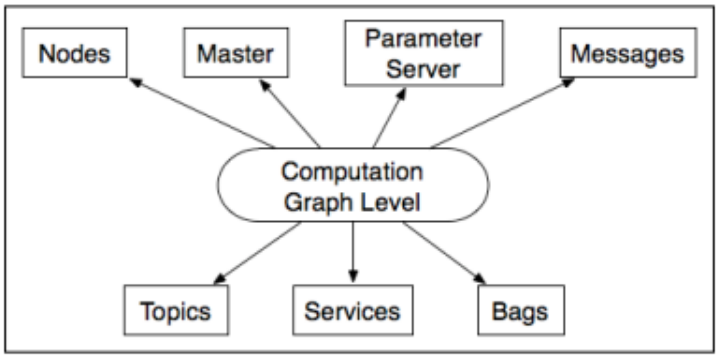


Figura 5.7: Computation Graph level [72].

- Master: O master fornece serviços de nomeação e registo para os restantes nós do sistema. A função do Master é permitir que os nós ROS se localizem mutuamente, de maneira a que estes possam comunicar uns com os outros. Se o ROS Master não existir no sistema, os nós não conseguem comunicar-se entre si e os serviços e mensagens não se encontram disponíveis.

- Parameter Server: O servidor de parâmetros é uma espécie de dicionário compartilhado por todos os nós e serviços, possibilitando o armazenamento de dados num local central.

- Messages: Um nó envia informações para outro nó usando mensagens que são publicadas por tópicos. A mensagem tem uma estrutura simples que usa tipos padrão (como inteiros, booleanos ou strings) ou tipos personalizados desenvolvidos pelo programador.

- Topics: As mensagens são encaminhadas através de um sistema de transporte do tipo publicador/subscritor. Um nó envia uma mensagem publicando-a num determinado tópico. O tópico é um nome usado para identificar o conteúdo da mensagem. Um nó que está interessado num determinado tipo de dados subscreverá o tópico apropriado. Pode haver vários publicadores e subscritores simultâneos para um único tópico, e um nó pode publicar e/ou subscrever vários tópicos. No geral, os publicadores e os subscritores não estão cientes da existência uns dos outros. Os tópicos em ROS podem ser transmitidos usando *Transmission Control Protocol / Internet Protocol* (TCP/IP) e ´*User Datagram Protocol* (UDP). O transporte baseado em TCP/IP é conhecido como TCPROS, sendo o transporte padrão, e o transporte baseado em UDP é conhecido como UDPROS.

- Services: O modelo publisher/subscriber é um paradigma de comunicação muito flexível, mas quando precisamos de uma solicitação ou uma resposta de um nó, não é possível fazer isso com tópicos. Para tal, os serviços dão-nos a possibilidade de interagir com os nós. Para além disso, os serviços devem ter um nome exclusivo.

- Bags: Um bag é um arquivo criado pelo ROS com o formato .bag, que permite guardar todas as informações das mensagens, tópicos, serviços e outros. Estes dados poderão ser usados mais tarde de modo a testar/verificar algoritmos e guardar os resultados de uma experiência.

Em suma, é possível compreender o porquê do ROS ser escolhido como ferramenta de desenvolvimento de software, já que todos os dados dos sensores apresentam o instante de tempo na qual estes foram adquiridos e cada informação dos sensores é armazenada no respetivo tópico, a taxa de saída dos sensores é preservada, é de fácil implementação e, através dos bags, é possível guardar todos os dados recebidos dos sensores, que são publicados no respetivo tópico, bem como o tempo de aquisição destes dados.

Por outro lado, se no futuro houver a necessidade de alterar o conjunto de sensores presentes no sistema, será apenas necessário desenvolver os nós que comunicam e obtêm os dados desses sensores, sem alterar os restantes nós já desenvolvidos.

5.4 Arquitetura de Software

A figura 5.8 mostra as várias camadas da arquitetura de software do sistema. Na parte inferior estão representados os vários sensores do sistema. No módulo acima dos sensores, encontra-se o sistema operativo do sistema, mais concretamente, Linux. Este *Operating System* (OS) será responsável por gerir os recursos de hardware e de software e fornecer serviços comuns para os vários processos.

Os drivers assumem uma função especial no kernel do Linux, já que são "caixas pretas"que fazem com que uma determinada peça de hardware responda a um programa de software. Os drivers permitem que os sistemas operativos e outros programas de computador acedam ao hardware sem precisarem de saber detalhes precisos de como o hardware é utilizado.

Na parte superior, na camada ROS, encontra-se representado os módulos (*nodes*) ROS. Estes serão responsáveis por realizar a leitura do seu sensor e publicar no respetivo tópico, que se encontram apresentados na secção 6.2. O nó *trigger* está responsável por detetar o sinal de trigger enviado pelo módulo de sincronismo, como referido na secção 5.2, para tornar possível a sincronização do par stereo.

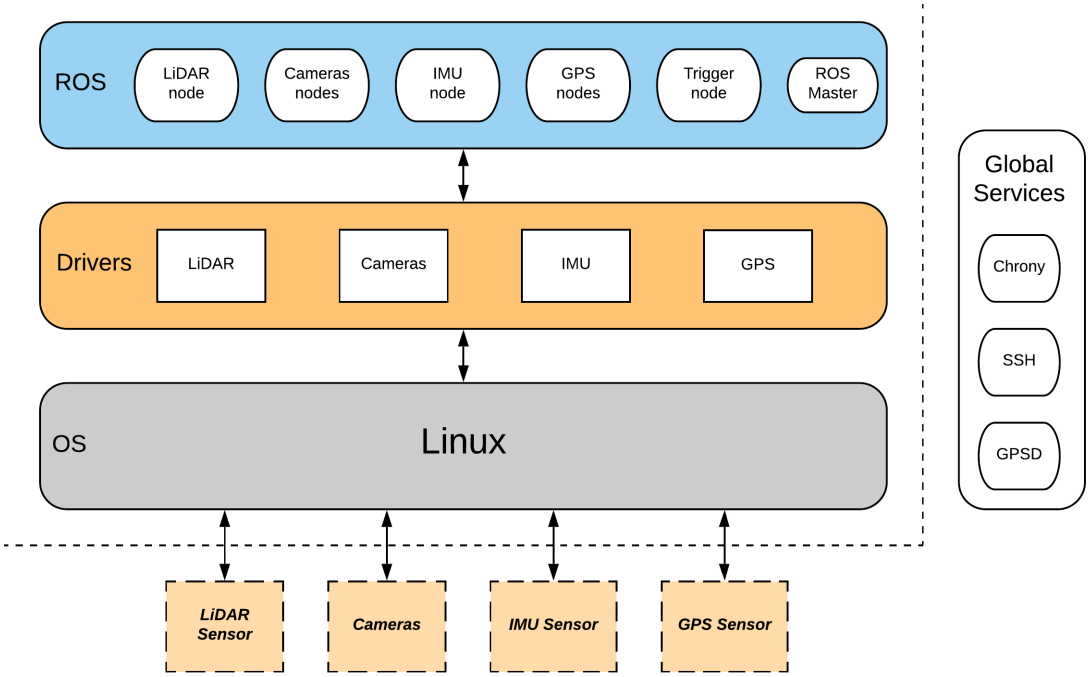


Figura 5.8: Arquitetura de software do sistema.

Por último, a camada de Global Services fornece vários serviços básicos para todos os módulos de software. A sincronização do relógio é obtida através de um servidor NTP, fazendo uso do Chrony, o protocolo de rede SSH permite a comunicação segura entre dois computadores, enquanto que o GPSD permite obter os dados dos recetores GNSS.

Esta página foi intencionalmente deixada em branco.

Capítulo 6

Implementação

Depois de projetado o sistema GeoTec, procedeu-se à sua implementação. Depois de ser validado em laboratório o sistema foi usado no primeiro local de testes do projeto MineHeritage, o Mosteiro e Mina de Tibães. Sobre os dados obtidos nesta missão de campo, foi realizado pós-processamento de forma a serem testadas algumas ferramentas que permitam validar a eficácia do sistema. Neste capítulo vão ser detalhados todos os componentes, tanto de *hardware* como de *software*, desde a estrutura mecânica do sistema, ao desenvolvimento de *Printed Circuit Board*s (PCBs), até à implementação do sistema.

6.1 Descrição Hardware

Nesta secção serão descritas as várias soluções de hardware projetadas, bem como explicado o funcionamento em mais detalhe de alguns módulos de hardware que já se encontram implementados. Primeiro será feita uma descrição da estrutura mecânica desenhada, posteriormente será exposto o desenvolvimento da PCB de distribuição de *power* do sistema, e para finalizar será feita uma descrição detalhada do funcionamento da placa de *trigger* das câmaras.

6.1.1 Estrutura Mecânica - Corpo principal

De forma a acomodar todos os sensores do sistema e os componentes necessários ao seu funcionamento, foi projetada uma estrutura mecânica. Esta foi desenhada com recurso ao software *SOLIDWORKS*, que permite ter uma boa representação do sistema tal como será na realidade. Para o material base desta estrutura, foi escolhido alumínio,

mais concretamente calhas de perfil *Bosch*, na figura 6.1. Este material apresenta várias vantagens:

- É suficientemente leve, sem comprometer a integridade e robustez do sistema;
- Permite que a estrutura seja modular, o que possibilita a alteração ou adição, rápida e fácil, de qualquer parte da estrutura;
- Devido ao seu design e peças de encaixe, garante alinhamentos e ângulos precisos;

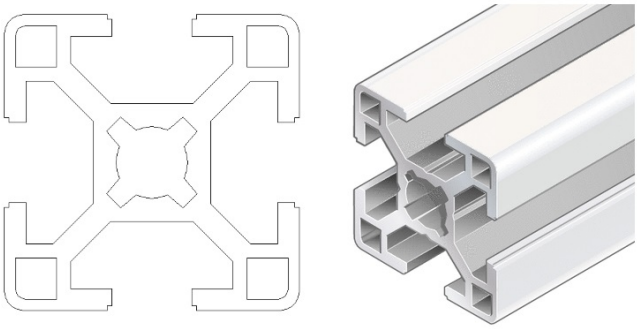


Figura 6.1: Perfil Alumínio Bosch 30 x 30 [73].

Existem vários tamanhos padrão de perfil sendo que neste caso foi escolhido o tamanho intermédio (30 x 30 mm) apresentando a melhor relação peso/resistência dos tamanhos disponíveis.

A estrutura mecânica completa do sistema pode ser visualizada na figura 6.2, esta pode ser decomposta em duas partes, sendo que uma é a frame base, um paralelepípedo com dimensões de 59 x 40 x 24 cm, e a outra é uma estrutura em "T"na horizontal, subida do frame base em 20 cm. Na figura 6.3 estão detalhadas as duas partes.

No interior do frame base existe um patamar intermédio que suporta uma placa de acrílico, como apresenta a figura 6.4, à esquerda. Este patamar tem como objetivo fixar alguns dos componentes do sistema, como o PC, o IMU, as baterias, a placa de distribuição de power, a placa de sincronização e o switch ethernet. À direita, na figura, é visível a distribuição espacial de todos os componentes referidos na placa de acrílico.

Na parte da frente deste patamar existe uma área mais reforçada estruturalmente com o intuito de fornecer uma zona estável para a fixação do IMU. Este sensor, devido à sua natureza mais sensível a perturbações e ruído, têm que estar isolado do resto dos componentes e é a razão principal pela qual o frame base é tão comprido. Na figura

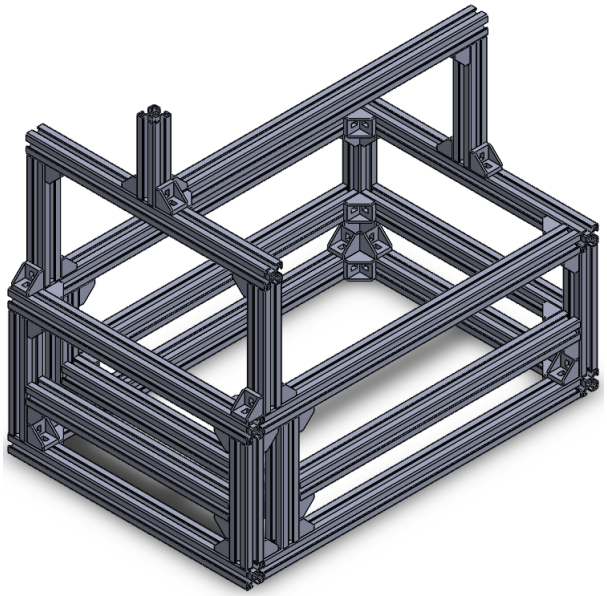


Figura 6.2: Estrutura mecânica completa.

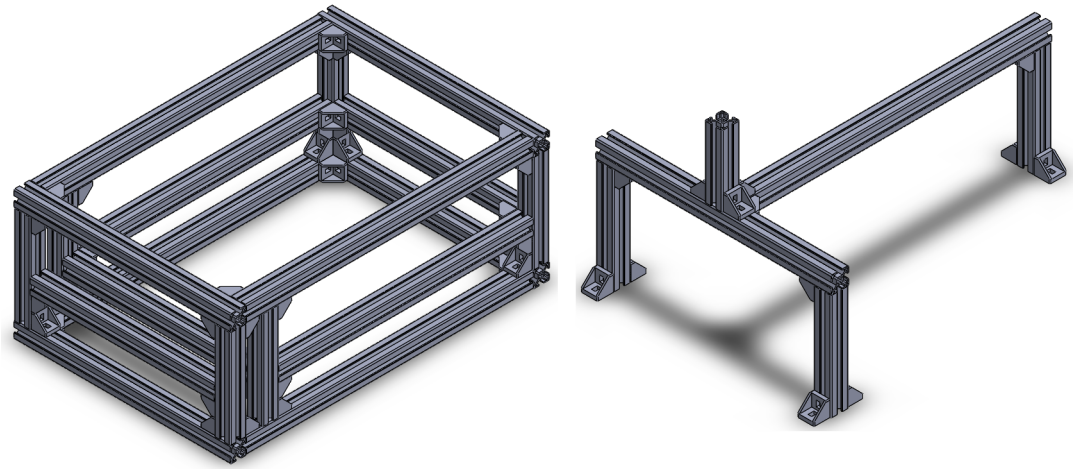


Figura 6.3: Estrutura mecânica: à esquerda frame base, à direita frame em "T".

6.4 é possível observar a zona referida, com e sem o *IMU*. Este é fixado numa peça em alumínio, desenhada para o encaixe perfeito do sensor, de forma a garantir o alinhamento e dissipar o calor gerado.

Na segunda parte da estrutura, o frame em "T", são colocados o resto dos sensores do sistema. Em cada ponta dos braços laterais é fixada uma câmara, que é instalada

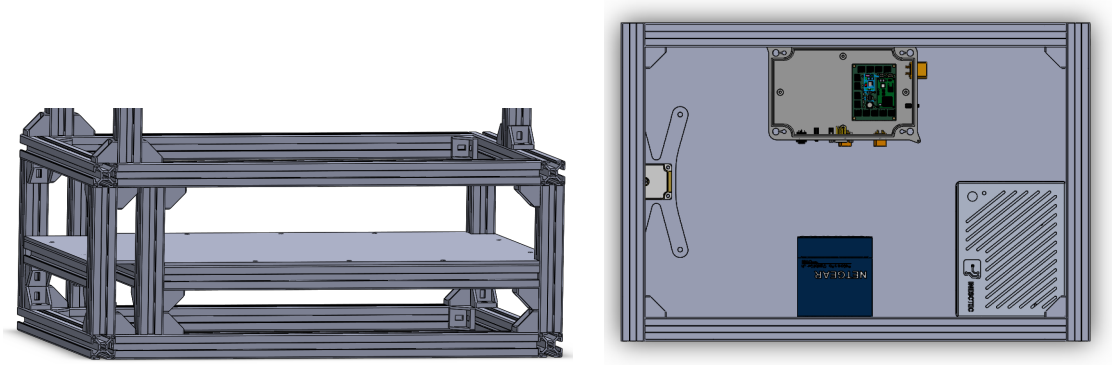


Figura 6.4: Esquerda: Patamar intermédio do frame base, sendo visível a zona reforçada para o IMU. Direita: Posicionamento dos componentes na placa de acrílico.

numa peça de alumínio, que permite alterar a *baseline* e o ângulo entre câmaras. Esta peça é composta por duas partes, sendo que numa delas, figura 6.5 lado esquerdo, é onde a câmara encaixa, e foi desenhada tendo em conta as dimensões da mesma, para permitir um encaixe robusto que garanta o alinhamento das câmaras. A outra parte, figura 6.5 lado direito, faz a coneção com o perfil bosch, garantindo também o alinhamento através das pregas que encaixam no rasgo do perfil, e permitindo o deslize desta ao longo do mesmo para proceder à alteração da *baseline* entre as câmaras.

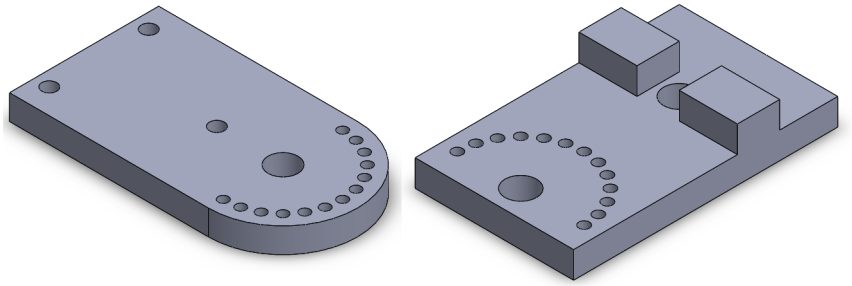


Figura 6.5: Peça para fixação das câmaras.

Ambas as partes são perfuradas numa da extremidades com vários furos espaçados 15º entre si, de forma a permitir a alteração do ângulo das câmaras em incrementos de 15º. A disposição destes furos é tal que em qualquer um dos valores de ângulos possíveis, existe sempre pelo menos dois pares de furos alinhados para permitir que, através da inserção de um parafuso em cada par, o ângulo entre ambas as partes se mantenha fixo.

Uma das razões para estas peças serem feitas de alumínio, para além das óbvias

(peso reduzido e robustez), foi o facto de estas servirem de dissipador para ajudar a dissipar o calor gerado pela câmaras, que neste tipo de câmaras é considerável. Na figura 6.6 é possível observar a junção das duas partes e a câmara.

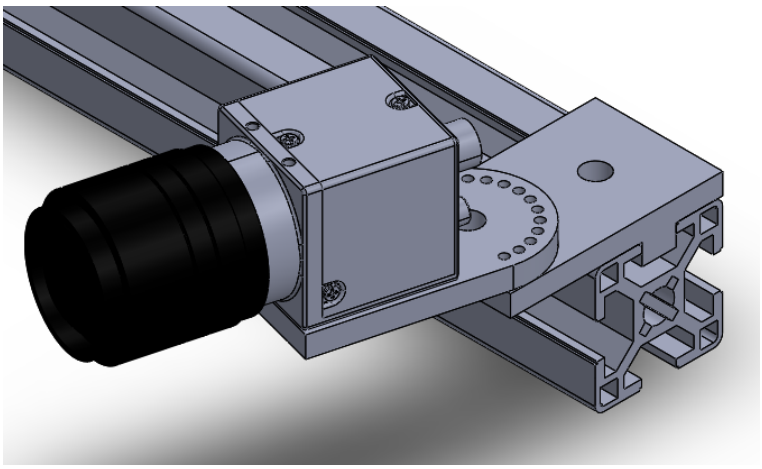


Figura 6.6: Conjunto câmara e peças de fixação.

Na intersecção do "T"eleva-se um perfil com 12 cm de altura para permitir a fixação do LiDAR na parte traseira, e o sistema de iluminação para as câmaras na parte frontal. O LiDAR encaixa numa peça de alumínio que faz a ligação à estrutura, este é colocado na vertical, apontado para cima, de forma a mapear o túnel com o movimento da estrutura, visto que é um sensor 2D.

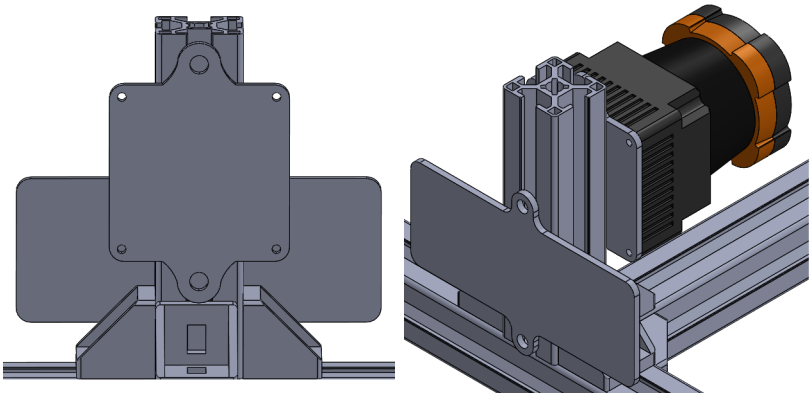


Figura 6.7: Peça de fixação da placa de iluminação e do LiDAR, à esquerda e conjunto completo com o sensor, à direita

A peça do sistema de iluminação das câmaras é de certa maneira parecida com a

do LiDAR no encaixe no perfil Bosh. Nesta peça, também feita em alumínio para servir de dissipador para o calor gerado, são coladas várias fitas *Light Emiting Diode* (LED) de alta potência de forma a iluminar o ambiente em frente às câmaras para permitir a utilização das mesmas em ambientes escuros e subterrâneos. Na figura 6.7 é visível, à esquerda, a peça de adaptação do LiDAR e à direita está representada uma vista do perfil de 12cm em conjunto com as peças e sensor.

O perfil perpendicular ao perfil frontal do "T", será usado para colocar antenas dos recetores GNSS para o sincronismo do relógio do CPU. A figura 6.8 mostra o sistema completo com todos os componentes, sendo visível o frame "T"e a placa de acrílico com todas as peças e sensores.

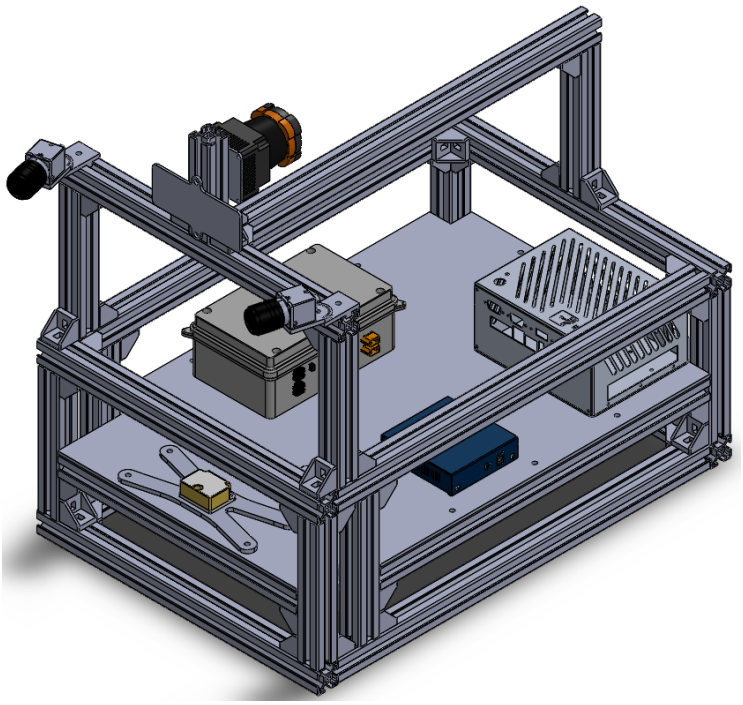


Figura 6.8: Estrutura mecânica completa com todos os componentes.

6.1.2 Estrutura Mecânica - Suporte para movimentação

De forma a facilitar o transporte da estrutura principal do sistema, em terrenos mais planos e não extremamente acidentados, foi pensado um suporte que permita um fácil acoplamento e desacoplamento do sistema principal, pois devido à morfologia variada dos túneis e galerias das minas, pode ser possível apenas o transporte e passagem

do sistema principal. Na figura 6.9 está representado o sistema desenhado.

Figura 6.9: Base do suporte mecânico para movimentação.

Este sistema foi pensado para permitir uma fácil movimentação, apesar de apresentar uma certa robustez contra superfícies com até 20 cm de água, e com rugosidade moderada. Para tal foram escolhidas uma rodas resistentes, com piso adequado para o pretendido. Foi também desenhada uma solução de direção de Ackermann, para permitir o controlo da direção ou até posteriormente a introdução de locomoção motorizada. Na figura 6.10 é possível observar a disposição dos componentes que compõem a direção de Ackermann.

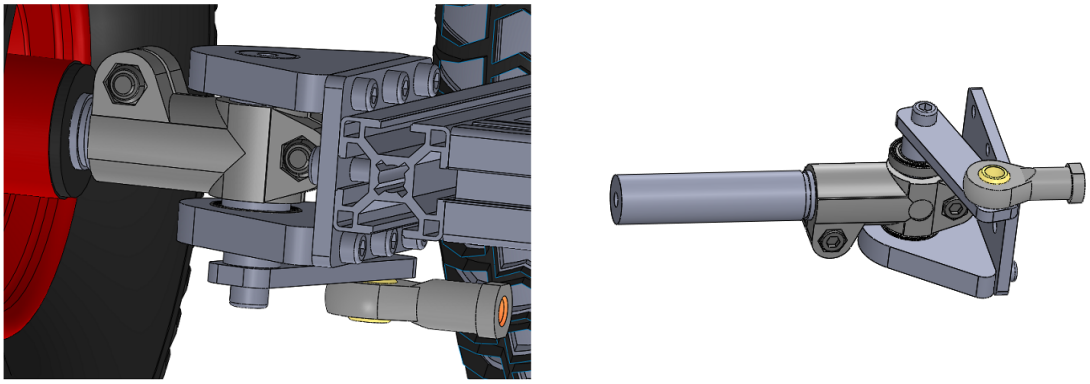


Figura 6.10: Esquerda: Montagem das rodas posteriores do suporte de locomoção. Direita: Pormenor da construção da direção de Ackermann.

A montagem consiste num veio horizontal que encaixa na roda, ligado a um veio

vertical por intermédio de uma peça de encaixe. Este veio vertical é introduzido entre duas peças de alumínio em contacto com um rolamento em cada uma das extremidades do veio. As duas peças de alumínio são seguras no lugar através de uma placa que faz a interface com a calha que forma o retângulo da base do carrinho. Numa das extremidades do veio vertical, existe uma outra peça que liga a uma junta rotativa que, através de um varão, faz a conexão com a outra roda dianteira e completa a geometria de Ackermann. O retângulo de base tem as mesmas dimensões que o retângulo da estrutura principal de modo a permitir o encaixe de uma estrutura na outra.

6.1.3 PCB de distribuição de energia

De forma a alimentar todos os componentes do sistema, e uma vez que a maioria exige níveis de tensão diferentes, torna-se essencial proceder à elaboração de uma PCB de distribuição de energia. Depois de uma análise aos requisitos de energia dos vários componentes do sistema, conclui-se que os níveis de tensão essenciais são: 24V e 12V. Os níveis de 5V e 3,3V foram também acrescentados para fornecer uma maior cobertura em termos de valores de tensão, permitindo também a fácil adição ao sistema de componentes futuros que necessitem destes níveis.

O valor de tensão das baterias que vão ser usadas no sistema é de 24V, e por isso os componentes que necessitam de 24V, no sistema apenas a placa de *trigger*, obtêm esse valor diretamente da bateria. A maioria dos sensores e componentes presentes no sistema usa 12V, nomeadamente o PC, as câmaras, o *switch ethernet* e o LiDAR. Devido a esta carga mais elevada de sensores neste valor de tensão, o conversor DC-DC que vai converter os 24V para 12V tem que ser capaz de apresentar uma corrente de saída elevada. Foi escolhido o *PTN78020HAH* da *Texas Instruments* que tem como corrente máxima 6 Amperes. Para converter dos 24V para os 5V é usado um conversor *muRata OKI-T/36W-W40 Series* que permite uma corrente de saída de 3A. No caso dos 3,3V como os dispositivos que geralmente usam este nível de tensão não apresentam necessidade de correntes muito elevadas, por isso foi colocado um simples regulador *Low-Dropout* (LDO) *LD1117S33TR*. Este tem como tensão de entrada 5V e portanto irá receber esse nível de tensão através do *muRata*.

Estando definidos os componentes principais da placa, procedeu-se ao desenho da mesma. Para tal foi utilizando o software de criação de placas de circuitos impressos, *KiCad*. O primeiro passo foi criar o esquema elétrico da PCB. Este consiste nos esquemas dos componentes já referidos, com a adição dos condensadores e resistências necessários para "selecionar"os níveis de tensão pretendidos, tanto no muRata como no PTN. Estes

componentes são necessários pois estes conversores têm uma gama alta de valores de saída (de 5V a 15,5V no caso do muRata e de 11.85V a 22V no caso do PTN), que é definido conforme a combinação de resistências e condensadores especificadas pelos fabricantes. Foram colocadas 3 saídas de 24V (diretas da bateria), 5 de 12V (uma vez que é o nível de tensão mais utilizado) e duas de cada valor de tensão restante (3,3V e 5V).

Depois do esquema elétrico completo, procedeu-se à elaboração do desenho da PCB. Na figura 6.11, à esquerda, é possível observar o resultado do processo de desenho da placa, onde se pode verificar a localização dos componentes, em particular de todos os conectores dos diferentes níveis de tensão. Já na mesma figura, à direita, está apresentada uma vista 3D da PCB com todos os componentes.

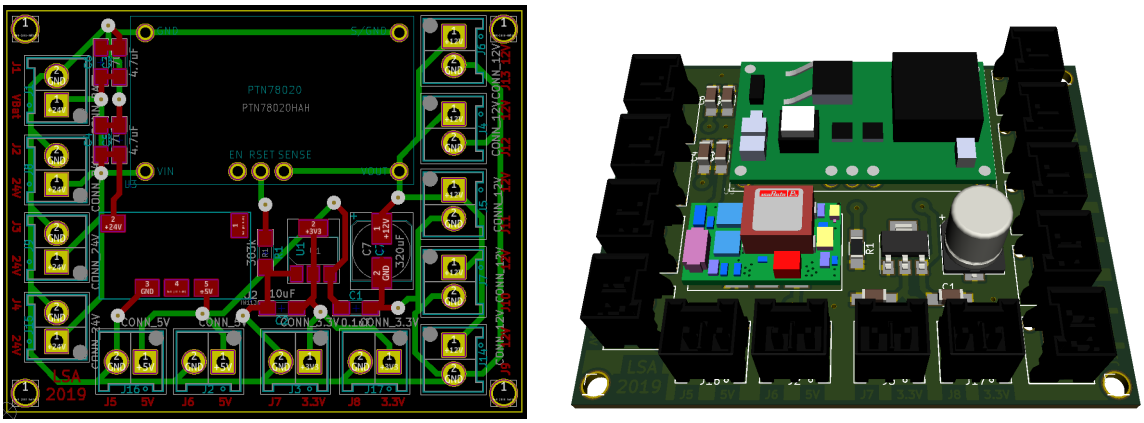


Figura 6.11: Esquerda: Desenho da PCB de distribuição de *power*. Direita: Vista 3D da PCB.

6.1.4 Placa de trigger

O disparo das câmaras será realizado por um módulo externo de hardware, para este módulo recorreu-se a um sistema desenvolvido no laboratório, o qual é possível adaptar para realizar este propósito.

Este sistema foi desenhado para o projeto UNEXMIN, e faz parte dos *Scructured Light Systems* (SLSs), encontrando-se um desses sistemas representado na figura 6.12. Este sistema contém 2 ou mais tipos de emissores de luz, em que um deles é necessariamente um emissor laser em linha, que roda sobre um eixo, e a sua posição absoluta é conhecida usando a leitura do encoder do motor, possibilitando assim realizar varri-

mentos. É também responsável por gerar os *triggers* dos emissores de luz, e/ou *triggers* externos segundo sequências de 1 segundo programáveis, sendo que é possível comunicar com a placa responsável por estas funções usando RS485, para receção e envio de dados, seja informações do posicionamento do laser, configurações, entre outros. Para o caso do sistema *Geo Tec* é apenas necessária a placa de controlo. Na figura 6.13 está representado a vista 3D da placa de *trigger* dos SLSs.

Figura 6.12: Sistema de Luz Estruturada [44].

Visto que esta placa gera sinais de *trigger* para vários usos, pode ser utilizado para dar *trigger* às câmaras. Este sistema tem oito canais de controlo disponíveis, funcionando a 24V, e como na aplicação pensada para este projeto será apenas necessário dar *trigger* às câmaras, este sistema consumirá, no máximo, 150 mA.

O sistema disponibiliza dois modos de funcionamento: sincronismo por PPS e sem sincronismo. No primeiro modo, este recebe um sinal PPS e sincroniza os disparos com esse sinal. No segundo modo, o sistema fica em espera e quando recebe o sinal para começar, executa a sequência pretendida.

Como já foi referido este sistema grava as sequências de *trigger* em intervalos de um segundo. Estas sequências são geradas num ficheiro de texto, através de uma interface de alto nível feita em ROS. Estas sequências também podem ser geradas pelo MATLAB, e são dependentes do número de sistemas a instalar. O ROS apenas gera os ficheiros, não enviando a sequência para o micro controlador presente na placa. Esta tarefa, neste momento, é realizada pelo MATLAB.

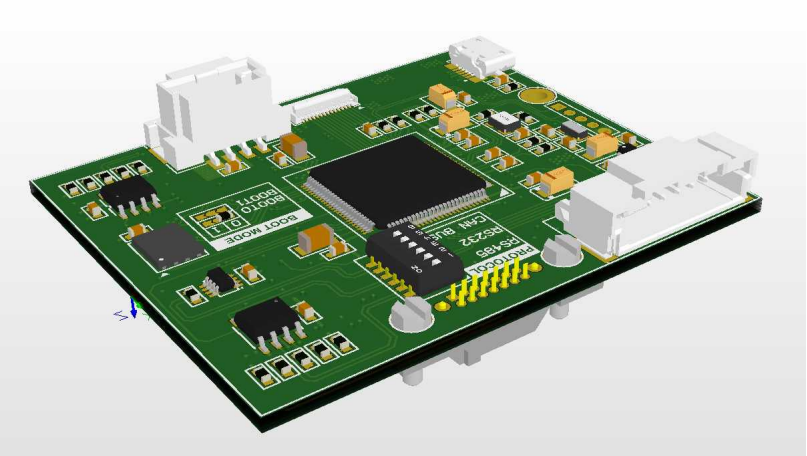


Figura 6.13: 3D da placa de *trigger* das câmaras.

Para gerar os sinais de *trigger* pretendidos para o sistema *Geo Tec*, foi desenvolvido um código em MATLAB que gera as sequências de um segundo de forma a que sejam gerados sinais iguais em três canais da placa. Dois destes sinais serão encaminhados cada uma para a respetiva câmara, e o terceiro para uma porta série do PC de forma a ser recebido no PC o tempo no qual foi capturada a imagem.

6.1.5 PCB de interface com o recetor GNSS

Para a integração do recetor GNSS UB482 no sistema já se encontrava desenvolvida no laboratório uma placa de interface que alimenta e contém todos os componentes necessários ao funcionamento do módulo, bem como permitir a comunicação por porta série das 3 UART do recetor.

A inclusão de um integrado FT4232HL na placa, um conversor de sinal TTL (UART) para USB de 4 canais, permite que a existência de uma interface de validação USB para além das portas séries originais. Neste caso a primeira porta série estará a enviar os dados *raw* da primeira antena, a segunda os dados da segunda antena, também *raw*, e a terceira fornecerá dados no formato NMEA.

Existe também um duplicador de sinal ao qual está ligado o sinal PPS do recetor, de modo a apresentar numa ficha da placa dois sinais PPS que podem ser usados para sincronização. Neste caso apenas um será usado para sincronizar o relógio do CPU. Na figura 6.14 está representado o desenho da placa de interface com o recetor GNSS.

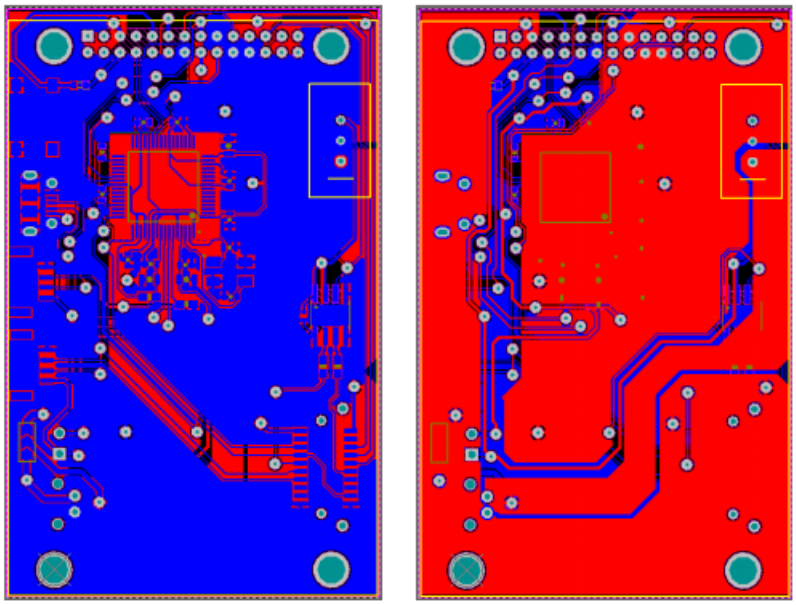


Figura 6.14: Placa de interface com o recetor GNSS.

6.2 Descrição de Software

Como já foi referido na subsecção 5.3, a aquisição dos dados dos sensores e o armazenamento destes será desenvolvido utilizando a *framework* ROS. Para tal, foram desenvolvidos ou usados nós para os vários sensores, de modo a realizar a aquisição da informação dada pelos mesmos.

Começando pelo LiDAR, existe um *package*, *hokuyo_node*, já desenvolvido pela comunidade ROS que permite fazer a aquisição e publicação num tópico dos dados do *Hokuyo*. Este suporta vários modelos de *Hokuyo*, incluindo o usado no sistema. O nome deste nó será *hokuyo* e o tópico */hokuyo/scan*, sendo que publica uma mensagem do tipo *LaserScan*.

Em termos de GNSS será usado um nó genérico, visto que a maioria dos recetores GNSS utilizam um protocolo padrão, neste caso o NMEA. Então, o nó *gnss* publica os dados de posição no tópico */gnss/fix*.

No caso do IMU, foi desenvolvido um nó para o *STIM 300*, visto a não existência de um *package* para este sensor. Este *package*, de nome *stim300* publica vários tópicos sendo o mais importante */imu/data*.

O nó responsável por receber o sinal de *trigger*, foi também desenvolvido parcialmente,

usando como base o *package* de Linux, *pps_tools*. Este permite uma fácil leitura de um sinal PPS que esteja a ser injetado num pino especifico (Carrier Detect) da porta série do PC, sendo que teve que ser adaptado para funcionar em ROS. Este nó publica no tópico */trigger/trigger_camera*.

Por último, o nó que irá obter as imagens das câmaras, *camera*, foi baseado no *package* já desenvolvido pela comunidade, *pointgrey_camera_driver*. Este nó publicará em vários tópicos, sendo os mais importantes: */image_raw* e *image_compressed*. Na tabela 6.1 estão representados os vários tópicos referidos, que são publicados pelo sistema.

Tabela 6.1: Lista de tópicos publicados

Sensor	Tópico ROS	Descrição
IMU (1000 Hz)	/imu/data	- Aceleração Linear (x,y,z) - Velocidade angular (x,y,z)
GNSS (5 Hz)	/gnss/fix	- Latitude - Longitude - Altitude
LiDAR (40 Hz)	/hokuyo/scan	- Ranges - Intensidades
Camera (6 Hz)	/left_camera/image_raw /right_camera/image_raw	- Imagens Raw
Camera trigger	/trigger/trigger_camera	- Timestamp

6.2.1 Sincronização do relógio do CPU

O *clock* dos computadores, servidores, estações de trabalho e dispositivos de rede não são inerentemente precisos. Geralmente, a maior parte destes relógios são definidos manualmente, apresentando um atraso elevado em relação ao tempo real e raramente são verificados depois de acertados. Muitos destes relógios são mantidos por uma bateria e um dispositivo de calendário-relógio que se pode atrasar até um segundo por dia. Portanto, para um sistema em que a sincronização tem de ser rigorosa, como é o caso, é impossível utilizar os relógios dos computadores sem que estes estejam sincronizados de forma precisa.

A sincronização do relógio lida com a compreensão da ordenação temporal de eventos produzidos por processos concorrentes. É útil para sincronizar remetentes e recetores de mensagens, controlar a atividade conjunta e o acesso simultâneo a objetos compartilhados. O objetivo é que vários processos não relacionados executados em diferentes máquinas estejam de acordo e sejam capazes de tomar decisões consistentes sobre a ordenação de eventos num sistema.

Atualmente, existem várias soluções para obter uma sincronização: NTP, GNSS, *Precision Time Protocol* (PTP), entre outros. O NTP é um protocolo de rede para sincronização de relógios entre vários sistemas computacionais, projetado por David L. Mills da Universidade de Delaware. Os objetivos deste protocolo são: permitir que os computadores estejam sincronizados com precisão com o *Coordinated Universal Time* (UTC); fornecer um serviço confiável que possa sobreviver a longas perdas de conetividade; permitir que os clientes sincronizem o seu relógio com frequência e, assim, compensem os *drifts* dos mesmos e, por último, fornecer proteção contra interferências externas.

Os servidores NTP são organizados em camadas, na qual a primeira camada contém os servidores primários, que são máquinas conectadas diretamente a uma fonte de tempo precisa, como o *Global Position System* (GPS), *Galileo Positioning System*, etc. A segunda camada contém os servidores secundários, sendo que estas máquinas são sincronizadas a partir dos relógios dos computadores da primeira camada, enquanto que a terceira camada contém os servidores que são sincronizados a partir dos secundários e assim por diante. Estas camadas juntas formam a sub-rede de sincronização, representada na figura 6.15, [74].

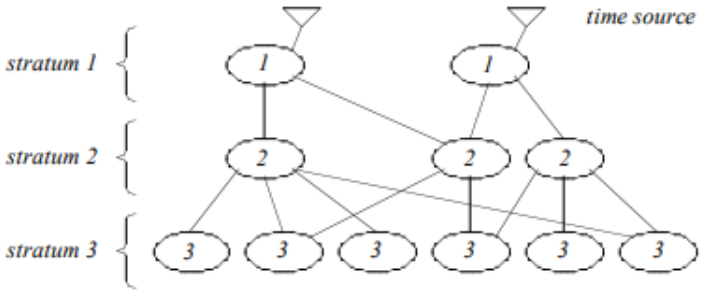


Figura 6.15: Sub-rede de sincronização do NTP.

O GPS é um dos sistemas de navegação por satélite disponíveis atualmente. Este é operado pelo Departamento de Defesa dos Estados Unidos, sendo que os sinais de posicionamento são produzidos por 24 satélites que estão distribuídos de tal forma que, pelo

menos quatro deles estão na linha de visão de qualquer ponto da Terra [75]. Os satélites transmitem informações do tempo dos seus relógios atómicos internos, juntamente com os dados da posição momentânea do recetor GNSS. A distância é calculada a partir do tempo que o sinal leva a percorrer a distância entre o satélite e a antena do recetor. Devido à elevada precisão do tempo fornecido pelo sistemas de navegação por satélite, este sistema pode ser usado para sincronizar o relógio de um sistema computacional.

O *chrony* é uma implementação versátil do Network Time Protocol, sincronizando o relógio do sistema com servidores NTP, relógios de referência (GNSS) e entrada manual. Este foi projetado para funcionar numa ampla variedade de condições, incluindo conexões de rede intermitentes, redes altamente congestionadas, mudanças de temperatura (os relógios comuns dos computadores são sensíveis à temperatura) e em sistemas executados em máquinas virtuais.

A precisão típica entre duas máquinas sincronizadas pelos servidores de Internet está na ordem de alguns milissegundos, enquanto que, usando um relógio de referência local, a precisão é tipicamente em dezenas de micro-segundos.

Neste software estão incluídos dois programas, o *chronyd* e o *chronyc*. O primeiro é um *daemon* (programa de computador que é executado como um processo em segundo plano) que pode ser iniciado no arranque do computador e o segundo é um programa de interface de linha de comando que pode ser usado para monitorizar o desempenho do *chronyd* e alterar alguns parâmetros, enquanto este está a ser executado.

De forma a ser usado o *chrony* para a sincronização do relógio do sistema *Geo Tec*, usando o sinal PPS fornecido pelo recetor GNSS é necessário um outro programa *gpsd*. Este programa é também um *daemon* que recebe dados de um receptor GNSS e re-publica esses estes dados de forma a poderem ser usados por várias aplicações. Assim, fornece uma interface unificada para receptores de diferentes tipos e permite o acesso simultâneo por vários processos.

Neste caso o que o *gpsd* está configurado para realizar é interpertar os dados NMEA do recetor GNSS e coloca-los numa *share memory* para poderem ser usados pelo *chronyd* para a sincronização do CPU do sistema.

6.3 Implementação GeoTec

Depois de estarem projetados todos os componentes e partes, desde a estrutura mecânica à parte electrónica, procedeu-se à montagem do sistema. Para tal, em primeiro lugar procedeu-se à elaboração da estrutura principal de perfil *Bosch*, sendo que os materiais foram encomendados a uma empresa especializada na área. Uma vez ob-

tidos, foi montada a estrutura de acordo com o que tinha sido previamente projetado. Para finalizar esta estrutura foi necessário maquinar a placa de acrílico que encaixa no suporte intermédio.

Depois de implementada a estrutura principal, passou-se à colocação e desenvolvimento de todos os componentes/sensores integrantes do sistema. Começando por maquinar as peças que garantem o encaixe e alinhamento das câmaras, como mostra a figura 6.16. Depois de ambas as câmaras estarem colocadas nos respetivos lugares, foram realizadas as peças de fixação do IMU, LiDAR, e sistema de iluminação. Colocados estes sensores nos respetivos sítios passou-se a fixar o PC e o switch à placa de acrílico.

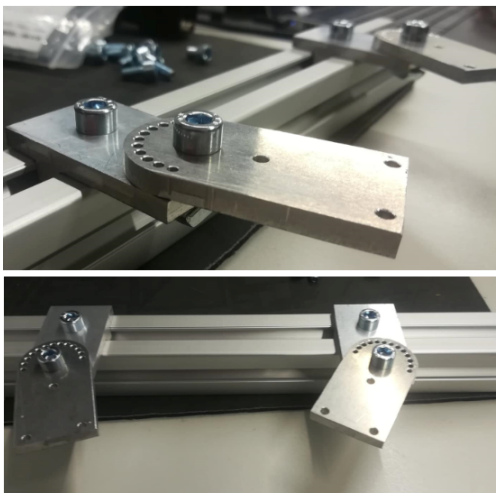


Figura 6.16: Peças de encaixe das câmaras.

De seguida procedeu-se à realização da PCB de distribuição de power do sistema, sendo que tanto esta placa como a placa de trigger das câmaras, necessitavam um sítio que permitisse a sua fixação, bem como alguma proteção contra salpicos e/ou pó. A solução encontrada foi alterar uma caixa estanque para albergar no seu interior as duas placas, e apresentar no seu exterior diferentes tipos de conetor de forma a possibilitar a ligação de todos os componentes do sistema. Na figura 6.17 está apresentada a caixa com as placas no seu interior, bem como a interface de conetores apresentada.

Depois da caixa ser fixada no acrílico da estrutura, foram feitas todas as ligações necessárias, e foram realizados alguns testes para verificar que tudo se encontrava operacional. Estando a estrutura principal completamente validada e testada, passou-se à construção e implementação de todos as peças e componentes necessários para realizar a estrutura de movimentação.

Figura 6.17: Caixa que contêm a placa de power e a PCB de *trigger*.

De forma a fazer com que a estrutura principal, quando montada no suporte de movimentação, ficasse ao nível do peito de uma pessoa, foi acrescentado um paralelepípedo (também ele feito de perfil de alumínio Bosch) ficando assim completa a estrutura de movimentação. Na figura 6.18 encontra-se apresentado um pormenor da direção de Ackermann, bem como a estrutura de movimentação completa.

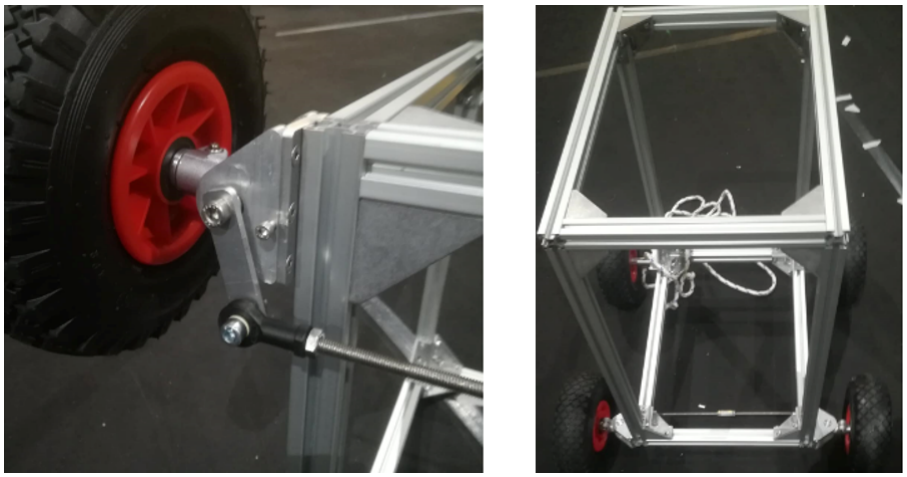


Figura 6.18: Esquerda: Pormenor da direção de Ackermann. Direita: Suporte de movimentação completo.

Para permitir o encaixe das duas estruturas, a principal e a de movimentação, foram

feitas quatro peças que garantem o alinhamento e o fácil desacoplamento entre ambas. A figura 6.19 mostra duas vistas diferentes do sistema *GeoTec* completo.

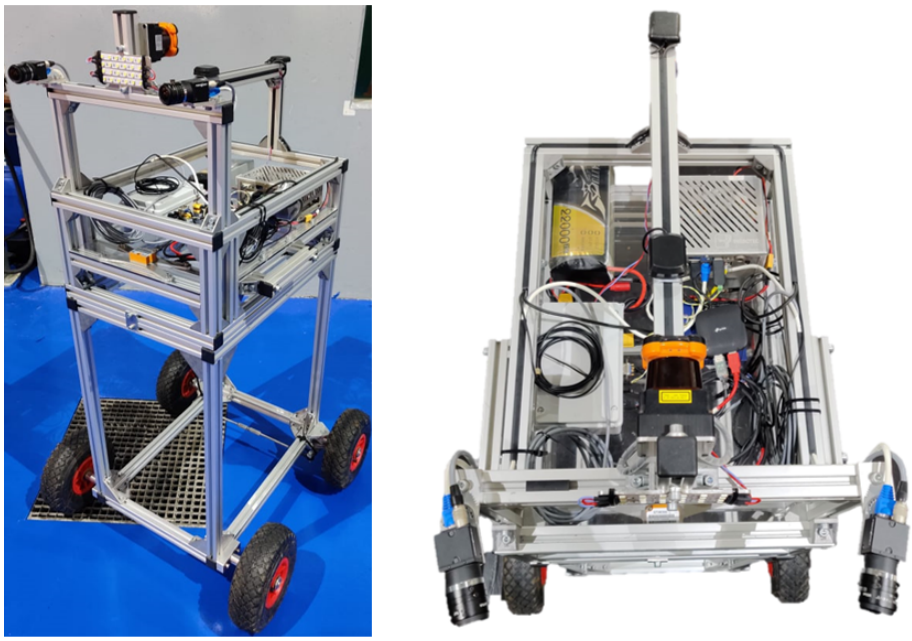


Figura 6.19: Sistema *GeoTec* completo.

De forma a realizar um *log* de *ground truth*, foi necessário adicionar dois suportes para permitir a montagem das duas antenas do recetor GNSS. Estas antenas têm que ser montadas o mais altas possível para melhorar a qualidade do sinal captado. Em funcionamento normal o sistema não necessita das antenas tão altas estando estas colocadas ao nível das câmaras, visto que apenas servem para inicializar e sincronizar o PC, antes da entrada para o ambiente subterrâneo provocar a perda do sinal. A necessidade da realização do *ground truth* será explicada no capítulo seguinte. Na figura 6.20 está representado o sistema preparado para a realização desse *log*.

Figura 6.20: Sistema *GeoTec* preparado para a realização do *ground truth*.

Esta página foi intencionalmente deixada em branco.

Capítulo 7

Resultados

Ao longo deste capítulo são apresentados os resultados do trabalho desenvolvido. Neles estão incluídos os testes de validação do sistema e sensores, bem como uma descrição dos primeiros testes em ambiente real do sistema no âmbito do projeto *MineHeritage*, passando pela apresentação do método de calibração usado para obter os parâmetros intrínsecos e extrínsecos das câmaras, até aos resultados obtidos através do processamento dos dados por cada programa, e a comparação com o *ground truth*.

7.1 Testes de validação do sistema e sensores

Depois de implementado o sistema, foram realizados vários testes para validar o funcionamento de todos os componentes. Usando o software desenvolvido referido previamente, primeiro procedeu-se ao teste de validação da aquisição de imagens pelas câmaras (figura 7.1), de seguida verificou-se também o correto funcionamento do IMU e do LiDAR.

No passo seguinte procedeu-se ao teste do recetor GNSS. Primeiro foi testada a receção de dados NMEA e raw de cada uma das antenas além da receção do sinal PPS numa das portas série do PC. Na figura 7.2 estão apresentados dois *print screen* que mostram, respetivamente, a receção dos dados NMEA usando o programa *gpsmon*, em cima, e a receção do sinal PPS, usando o programa *pps_test* do *package pps_tools* em baixo.

Depois de testado o correto funcionamento do recetor GNSS, procedeu-se à sincronização do *clock* do CPU com o tempo fornecido pelo recetor. Como já foi referido esta sincronização é feita usado o programa *chrony* sendo que para o correto funcionamento

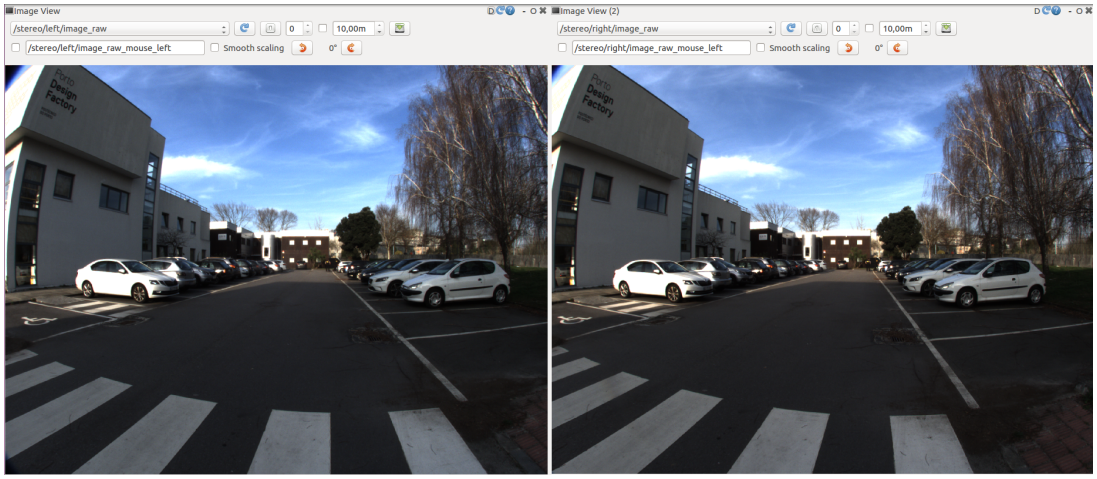


Figura 7.1: Exemplo de aquisição de imagens no exterior do laboratório.

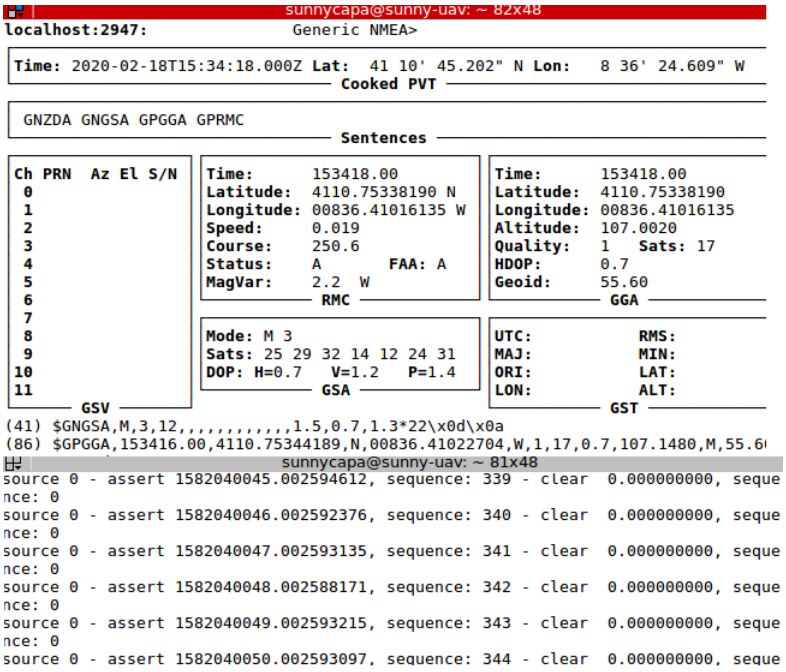


Figura 7.2: Cima: Receção dos dados NMEA. Baixo: Receção do sinal PPS.

deste, é necessário proceder a algumas alterações de configurações, nomeadamente: indicar qual é a *shared memory* na qual o *gpsd*, o software que efetua a ligação entre os dados recebidos no PC vindos do recetor GNSS e os fornece para o *chrony*, disponibiliza o sinal PPS e indicar que se pretende realizar a sincronização usando os dados *NMEA* para uma

sincronização inicial e posteriormente usar o sinal *PPS*. Na figura 7.3 estão representados os outputs do programa *chronyc traking* e *chronyc sources*, que mostram a sincronização do sistema e o erro de sincronização. À esquerda, na figura é possível verificar que o *clock* do CPU está sincronizado pelo sinal PPS e que se encontra 0,000039546 segundos adiantado ao tempo NTP. À direita, o programa *chronyc sources* mostra as duas fontes de sincronização, o sinal NMEA e o sinal PPS, sendo que estão detalhadas as diferenças de tempos entre as fontes e o tempo NTP. No caso do sinal NMEA, esta diferença é de 54*ms* e no sinal PPS é de 1232µs. Isto realça uma vez mais que a sincronização usando o sinal PPS é superior em termos de precisão em relação à sincronização feita apenas usando dados NMEA, garantindo uma precisão de sincronização na ordem dos micro segundos.

```
                          sunnycapa@sunny-uav: ~ 81x23
Every 2,0s: chronyc tracking                          Tue Feb 18 15:31:24 2020

Reference ID    : 80.80.83.0 (PPS)
Stratum         : 1
Ref time (UTC)  : Tue Feb 18 15:30:59 2020
System time     : 0.000039546 seconds fast of NTP time
Last offset     : +0.000040559 seconds
RMS offset      : 0.012397435 seconds
Frequency       : 16.648 ppm slow
Residual freq   : +0.267 ppm
Skew            : 93.031 ppm
Root delay      : 0.000000 seconds
Root dispersion : 0.003767 seconds
Update interval : 16.0 seconds
Leap status     : Normal
                          sunnycapa@sunny-uav: ~ 82x23
Every 2,0s: chronyc sources                           Tue Feb 18 15:31:25 2020

210 Number of sources = 2
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
#- GPS                           0   4   377     6    +54ms[  +54ms] +/-  250ms
#* PPS                           0   4   377     6  +1232us[+1743us] +/-  397us
```

Figura 7.3: Cima: Output do programa *chronyc tracking*. Baixo: Output do programa *chronyc sources*

O teste seguinte foi verificar se os tempos de *trigger* das câmaras estavam a ser corretamente recebidos pelo PC. Isto é feito utilizando o mesmo programa que foi usado para receber o sinal PPS do recetor GNSS. Através da ferramenta de análise de *rosbags*, rqt_bag é possível observar as diferenças de tempo entre o instante no qual o *trigger* foi dado e o instante no qual as imagens foram recebidas, reforçando a importância de receber o este sinal no PC. Usando o nó ROS desenvolvido que adquire e publica o tempo de *trigger* das câmaras, e gravando em conjunto com os tópicos nos quais são publicados as imagens de cada câmara, é possível realizar um *rosbag* de teste para verificar a diferença de tempos, como mostra a figura 7.4. Nela é possível verificar que

os tempos de *trigger* (neste teste o sinal *trigger* foi colocado a 4 FPS) são constantes, e que a receção das imagens acontece instantes depois do sinal *trigger* ser gerado.

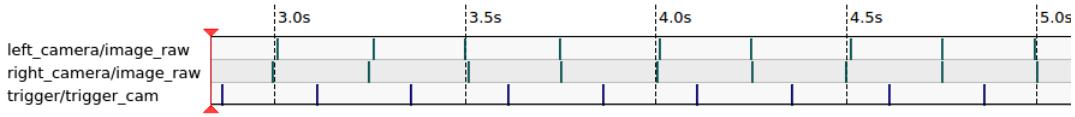


Figura 7.4: Diferença de tempos entre aquisição e receção das imagens.

7.2 Calibração dos parâmetros intrínsecos e extrínsecos

De forma a ser possível o uso das imagens obtidas utilizando o *GeoTec* para fins de mapeamento 3D, odometria visual e slam, é necessário obter os parâmetros intrínsecos e extrínsecos dos sensores ópticos, isto é, calibrar as câmaras. De forma a permitir esta calibração foi tirado um *log* de calibração com o sistema. Isto significa capturar imagens, das duas câmaras em simultâneo, de um padrão de xadrez com tamanho e número de quadrados conhecidos, em todas as zonas de distorção da imagem, com diferentes rotações e orientações do padrão de forma a permitir uma boa calibração.

Depois para realizar a calibração foi usada a toolbox *Stereo Camera Calibrator*, (representada na figura 7.5) do programa MATLAB. Esta, através da introdução das imagens de calibração referidas em cima, permite a calibração tanto dos parâmetros intrínsecos de cada câmara, quer os parâmetros extrínsecos de uma câmara em relação à outra.

A toolbox, depois de receber as imagens, o tamanho das arestas dos quadrados do padrão e o número de cantos interiores (horizontal e vertical, no caso do alvo usado 7x4), deteta primeiro os cantos entre quadrados, e rejeita todas as imagens nas quais o padrão não seja detetado. Depois de escolhidos o número de coeficientes de distorção radial a serem estimados (2 ou 3), a estimação ou não dos coeficientes de distorção tangencial e a estimação do *skew* dos *pixels*.

Depois de estimados os parâmetros intrínsecos e extrínsecos de ambas as câmaras, a aplicação permite observar os erros de reprojeção dos cantos (como mostra a figura 7.6), e permite também retirar as imagens com maior erro, podendo ser analisadas para perceber ao que se deve o erro. Depois de retiradas estas imagens é feita uma recalibração, permitindo assim realizar este processo até os níveis de erro serem dentro do aceitável.

Na figura 7.7 estão representados os parâmetros intrínsecos e extrínsecos obtidos através da calibração. È possível reparar na matriz de extrínsecos que a *baseline* obtida

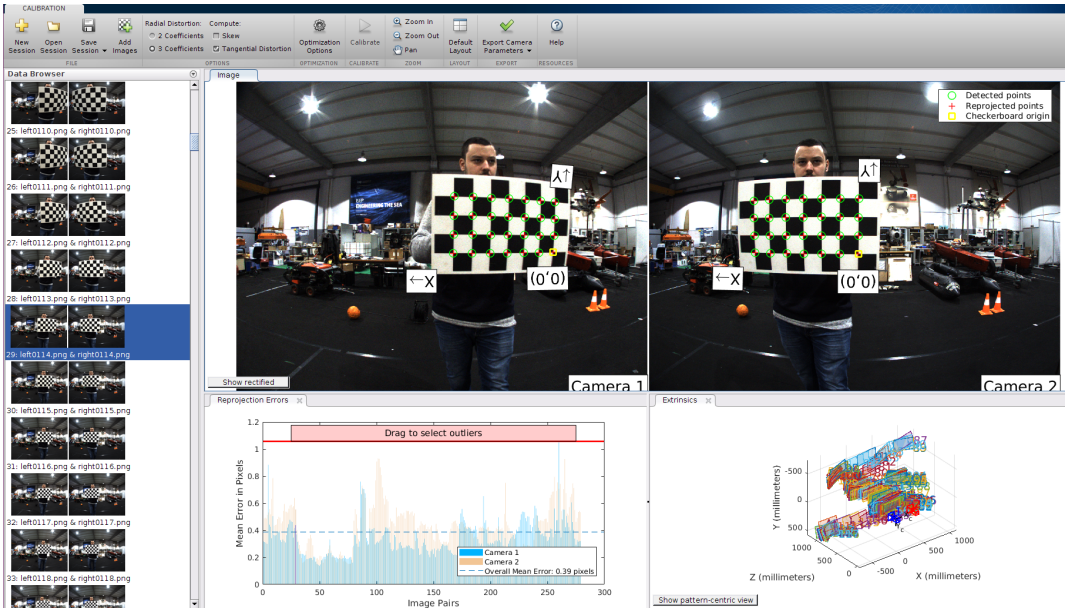


Figura 7.5: *Toolbox* de calibração stereo MATLAB.

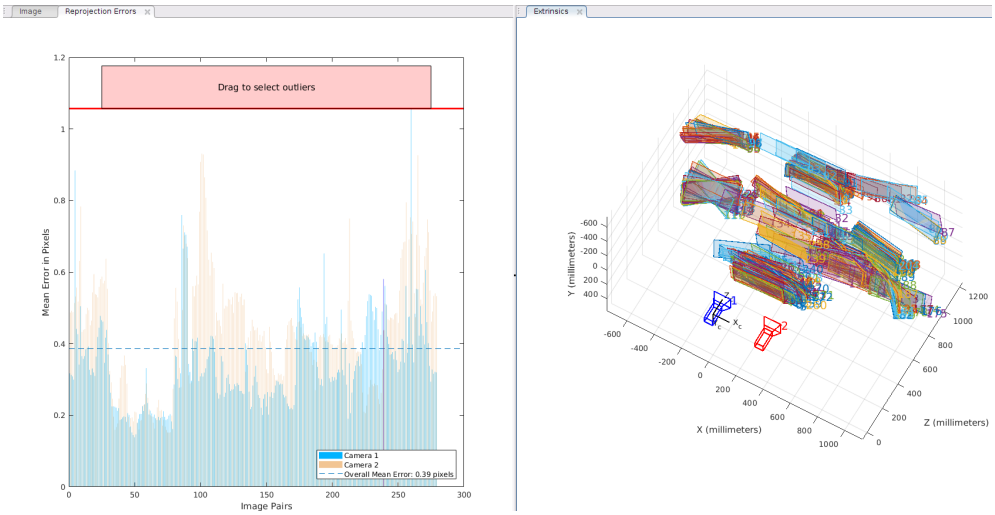


Figura 7.6: Pormenor dos erros de reprojeção da calibração.

pela calibração é de 369mm, o que está concordante com o valor obtido por CAD.

stereoParams_flat

1x1 stereoParameters

Property	Value
CameraParameters1	*1x1 cameraParameters*
CameraParameters2	*1x1 cameraParameters*
RotationOfCamera2	[0.9999 -0.0022 -0.0146;0.0023 1.0000 0.0038;0.0146 -0.0039 0.9999]
TranslationOfCamera2	[-369.7680 0.8494 3.3174]
FundamentalMatrix	[-4.4764e-09 -2.9631e-06 0.0032;-1.8450e-06 1.2680e-06 0.3502;0.0014 -0.3476 -4.3763]
EssentialMatrix	[-0.0050 -3.3142 0.8620;-2.0676 1.4198 369.7744;-0.0292 -369.7663 1.4119]
MeanReprojectionError	0.3869
NumPatterns	279
WorldPoints	*28x2 double*
WorldUnits	'millimeters'

Figura 7.7: Parâmetros extrínsecos e intrínsecos obtidos através da calibração.

7.3 Testes *MineHeritage* na Mina das Aveleiras, Tibães

O primeiro teste de campo do sistema *GeoTec*, coincidiu com o primeiro local de testes do projeto *MineHeritage*. Este fica situado em Braga, noroeste de Portugal, e é uma mina desativada que se encontra dentro do território pertencente ao Mosteiro de Tibães. Esta mina, apesar de ter como nome oficial, mina das Aveleiras, é vulgarmente conhecida por mina de Tibães, e foi explorada na primeira metade do século 20 para a extração de volfrâmio durante mais de 23 anos, até ter sido desativada. As características desta mina, entre as quais a zona envolvente na qual está inserida, fazem deste local um ótimo candidato a ponto de interesse do projeto *MineHeritage*. Na figura 7.8 está representado o traçado geral da mina das Aveleiras, bem como, a vermelho, a parte do traçado que foi mapeada na missão.

Na figura 7.9 estão apresentadas algumas imagens adquiridas no interior das diversas galerias existentes na mina das Aveleiras, usando o sistema *GeoTec*.

Este local é um ponto de interesse no projeto *MineHeritage*, devido à sua natureza histórica, e do património geo-mineiro que apresenta. No escopo do projeto é necessário a obtenção de dados que permitam a reconstrução das galerias e túneis da mina, sendo que para tal pode ser usando o *GeoTec*, mas também o mapeamento das áreas circundantes à mina, que neste caso é de interesse, o mosteiro de Tibães. Para mapear com a melhor resolução este meio envolvente, foi usado um scanner laser de alta resolução, do fabricante *FARO*, modelo *Focus 3D*.

Os resultados preliminares obtidos a partir do trabalho de campo realizado na Mina de Tibães foram apresentados e discutidos na Primeira Conferência organizada pela Ordem dos Engenheiros Técnicos e ISEP, intitulada "*4GEO — resources, materials, technologies and environment*", que decorreu em Novembro de 2019. Posteriormente,

Figura 7.8: Esquerda: Traçado geral da mina das Aveleiras (adaptado de Chaminé et al. 2017)[76] sobreposto pela parte mapeada (tracejado a vermelho). Direita: *GeoTec system* na entrada da mina.

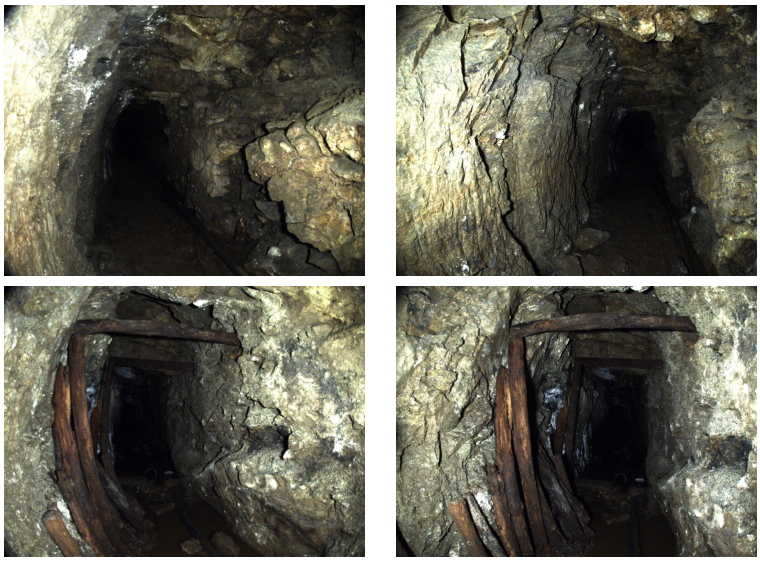


Figura 7.9: Exemplo de algumas imagens obtidas nos testes em Tibães (à esquerda imagens da câmara esquerda e à direita imagens da câmara direita).

os resultados foram processados e publicados numa Edição Especial na *Springer ASTI series*[1] sendo que mais detalhes poderão ser consultados em [77].

[1] + info: https://www.springer.com/series/15883

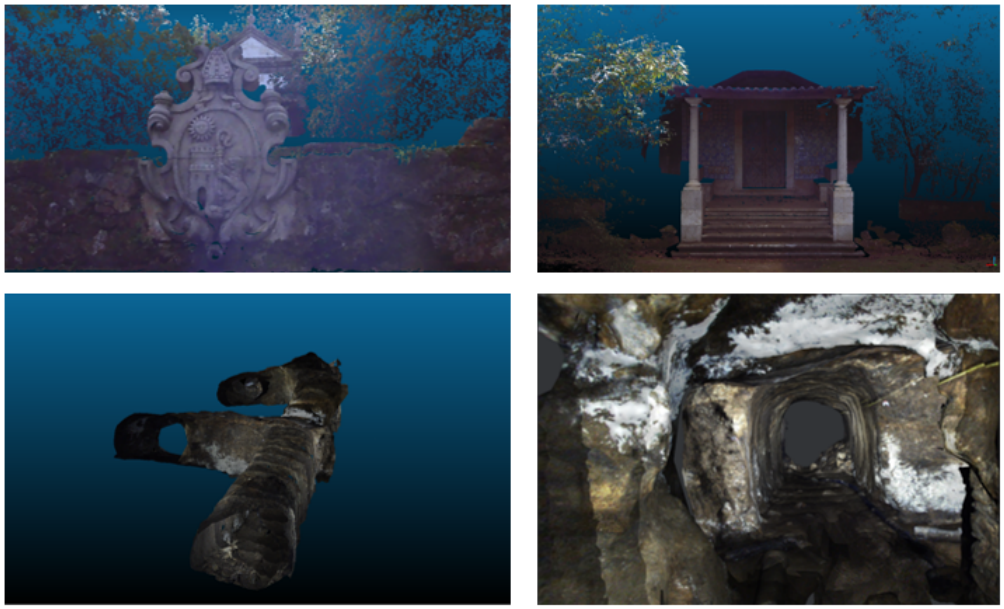


Figura 7.10: Resultados preliminares apresentados na conferência 4GEO.

Na figura 7.10 estão apresentados alguns dos resultados preliminares divulgados na conferência, as duas imagens superiores representam as nuvens de pontos obtidos com o scanner laser, nomeadamente o pormenor de um brasão numa das fontes do escadório do mosteiro e a parte frontal da capela no topo do mesmo escadório. No caso das imagens inferiores, estas representam a reconstrução 3D da intercepção das duas galerias principais, e o pormenor do interior dessa intercepção. Estas reconstruções foram obtidas usando o *software*, Meshroom.

7.4 Ground Truth

De forma a obter uma base de comparação para a eficácia da estimação das trajectórias obtidas usando ambos os programas, procedeu-se à realização de um *ground truth*, isto é, um *log* de teste, feito no exterior do laboratório, no qual foi adquirido o sinal de cada uma das antenas do recetor GNSS, em conjunto com os dados do IMU, e as imagens sincronizadas de ambas as câmaras. Na figura 7.11 encontram-se alguns exemplos de imagens obtidas neste *log*.

Depois de realizado esse teste, procedeu-se ao pós-processamento dos dados obtidos, nomeadamente do recetor *GNSS* e do sistema inercial, através do software *NovAtel Way-*

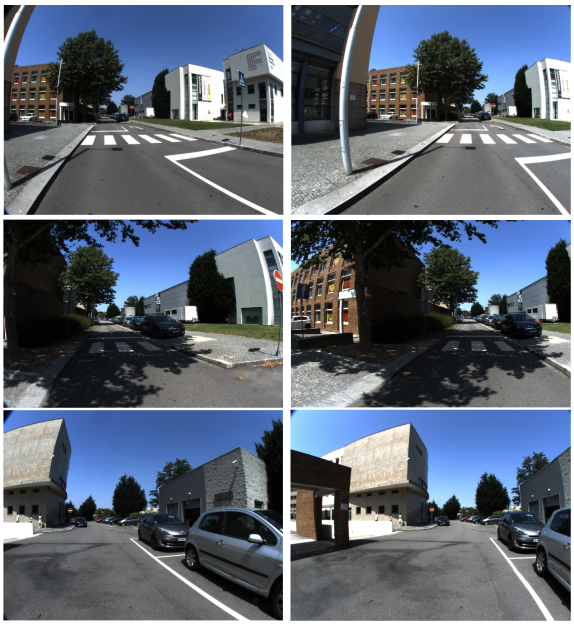


Figura 7.11: Exemplo de alguns pares de imagens do *ground truth*

point Inertial Explorer, que permite realizar a fusão sensorial entre ambos os sensores, resultando em valores de posição, velocidade e aceleração medidos com alta precisão. Estes dados tratados, são a melhor aproximação possível às posições reais que o sistema descreveu e compõem o *ground truth* de comparação. Na figura 7.12 encontra-se apresentada a trajectória obtida através do pós-processamento, representada com a ajuda do programa MATLAB. Já na figura 7.13 está representada à esquerda a posição em cada eixo, e à direita as rotações (*roll*, *pitch*, *yaw*).

De forma a auxiliar a compreensão da qualidade do sinal GPS ao longo do percurso efetuado, a figura 7.14 apresenta o percurso preenchido com cores diferentes consoante a qualidade do sinal. A escala de qualidade varia de nível um a seis, sendo que nível 6 é má qualidade nível 1 qualidade máxima. Destes dados é possível concluir que as partes do percurso nas quais a qualidade do sinal GPS é inferior perfazem 6,16% do percurso total.

Para realizar a comparação das trajetórias estimadas por cada um dos programas, Meshroom e RTAB-Map, com o *ground truth* obtido é necessário proceder à extração dos valores das *poses* em cada programa.

No caso do RTAB-Map, este fornece uma opção para a extração imediata das poses, apresentado várias alternativas quanto aos formatos de exportação. Já no caso do Mesh-

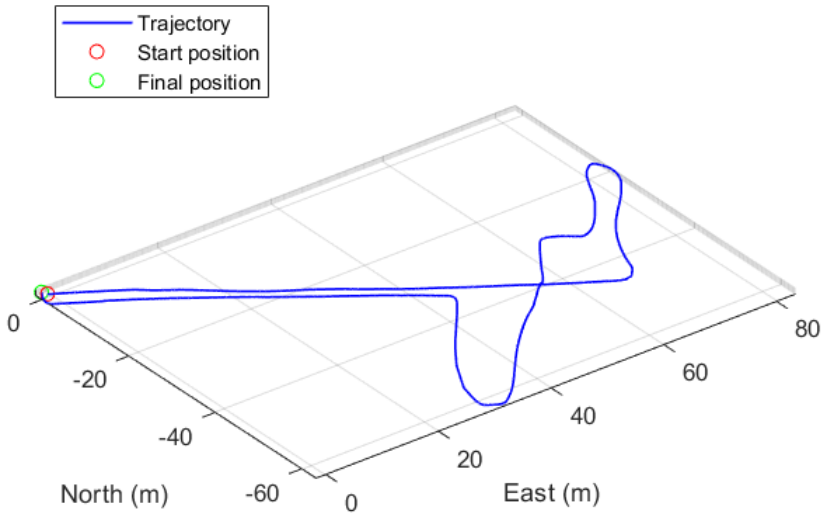


Figura 7.12: Trajetória pós-processada realizada no *ground truth*.

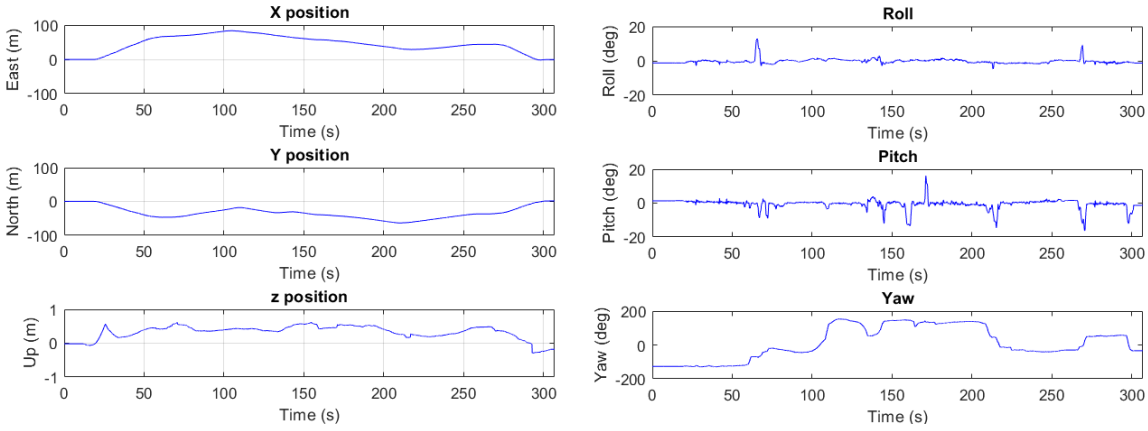


Figura 7.13: Esquerda: Posição em cada um dos eixos da trajetória de *ground truth*. Direita: Rotações de Euler em graus(*roll*, *pitch*, *yaw*) da trajetória de *ground truth*.

room, este não apresenta uma maneira direta para a extração destes dados, tendo sido necessário o desenvolvimento de um programa que interprete os ficheiros intermédios criados pelo bloco *Structure from Motion* e devolva os valores das *poses* num ficheiro *.txt*.

Depois de obtidos estes ficheiros com as *poses* estimadas por cada um dos *softwares*, estes foram inseridos no MATLAB, sendo efetuadas as mudanças de referenciais necessárias para a correta apresentação dos dados, resultando na figura 7.15 que apresenta

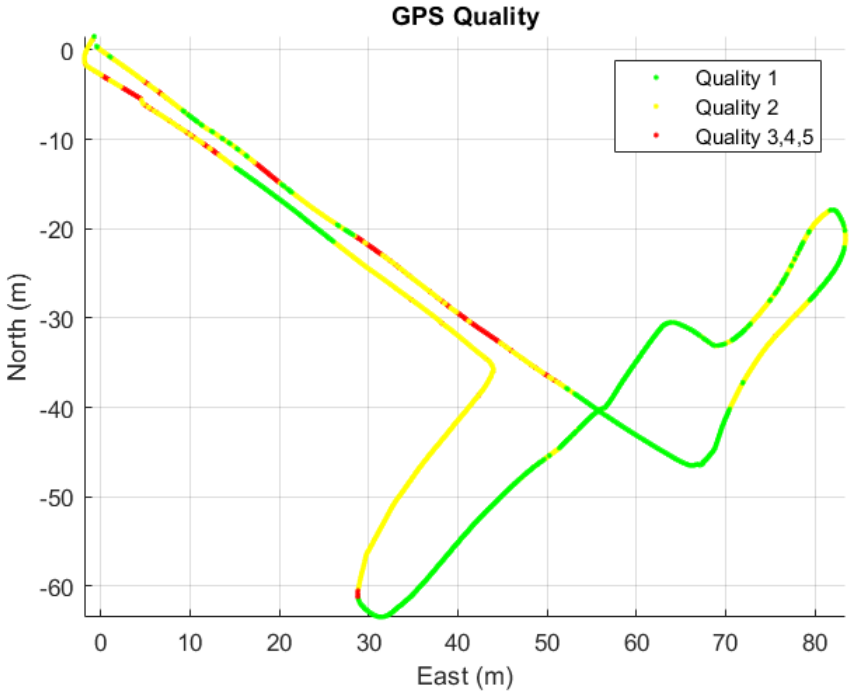


Figura 7.14: Qualidade do sinal GPS durante o percurso realizado no *ground truth*

as duas trajetórias estimadas em comparação à trajetória do *ground truth*.

Como é possível observar, existem erros substanciais na estimação de trajetória em ambos os programas. No caso da estimação do RTAB-Map as diferenças devem-se ao reduzido número de *features* usadas devido ao facto de ser um software desenhado para ser executado em tempo real. Outro fator importante é a existência de poucos *loop closures*, que se deve em parte ao facto do erro já se apresentar tão elevado quando o sistema se encontra em sítios onde já esteve antes, que o programa não associa as features retiradas naquele instante às retiradas na outra passagem pelo mesmo local por considerar que, através do desvio de trajetória, não se encontra no local.

Já no caso do meshroom este apresenta menos diferenças para o *ground truth*, que se deve em parte ao facto de ser um programa que não é executado em tempo real e como tal não necessita de reduzir o tempo de computação, extraindo um elevado número de *features* em cada imagem (cerca de 10000). Isto ajuda a uma aproximação maior à trajetória real. No início a trajetória apresenta diferenças mínimas até chegar a um ponto no qual esta começa a divergir substancialmente mais e é por isso que a posição final é tão diferente da posição final do *ground truth*.

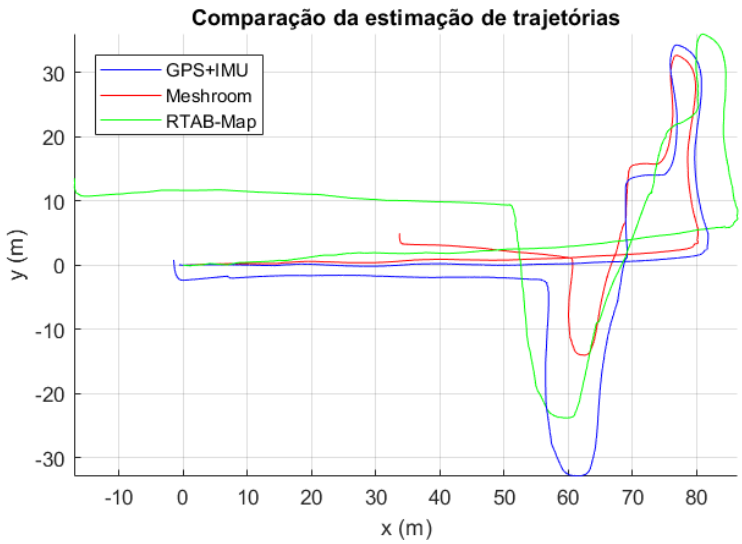


Figura 7.15: Comparação da estimação da trajetória, usando o RTAB-Map e Meshroom, com o *ground truth*.

Na figura 7.16 está apresentado em pormenor a diferença entre a estimação das trajetórias nos primeiros 45 metros de *log*. Nesta fase inicial foi realizada uma trajetória o mais reta possível, ao longo do eixo dos *xx*, como comprova o gráfico do *ground truth*, a azul.

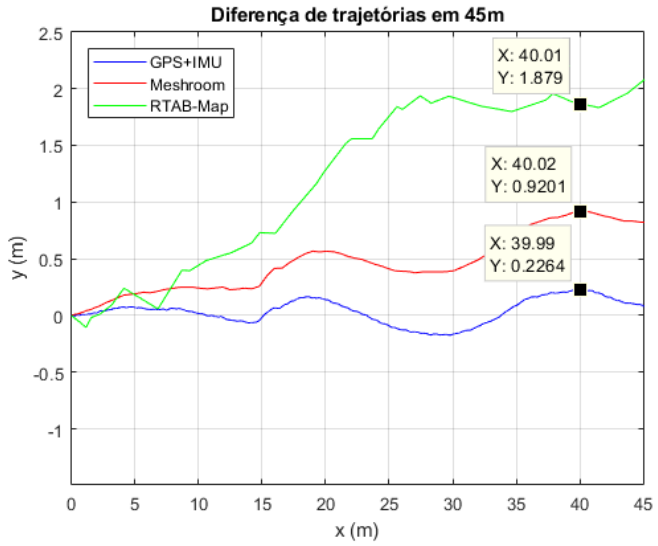


Figura 7.16: Estimação da trajetória nos primeiros 40m do *ground truth*.

Na mesma figura são também visíveis os valores de y aos 40 metros, que corresponde a 37,8 segundos de log. Pela subtracção do valor do *ground truth* nos valores do RTAB-Map e Meshroom, obtém-se o erro acumulado em 40 metros de trajétória. No caso do Meshroom, é uma diferença de apenas 0.694 metros em relação ao *ground truth*, sendo que no caso do RTAB-Map, está diferença apresenta-se mais elevada, 1.653 metros.

7.5 Reconstruções 3D usando o Meshroom

Nesta secção são apresentados alguns resultados de reconstruções obtidas através do programa Meshroom, usando os dados recolhidos na mina de Tibães e os dados do *log* de *ground truth*.

Na figura 7.17 é visível a interface gráfica do Meshroom com as imagens adquiridas no *log* de *ground truth* encontrando-se já completamente processados todos os blocos de código do programa como se pode verificar na parte inferior da figura. É possível também verificar o número de imagens utilizadas (no caso do *grund truth* foram adquiridas 2316 imagens sendo metade de cada câmara), bem como, no lado direito da figura, é possível observar as poses da trajetória descrita por cada câmara em conjunto com a nuvem de pontos obtida.

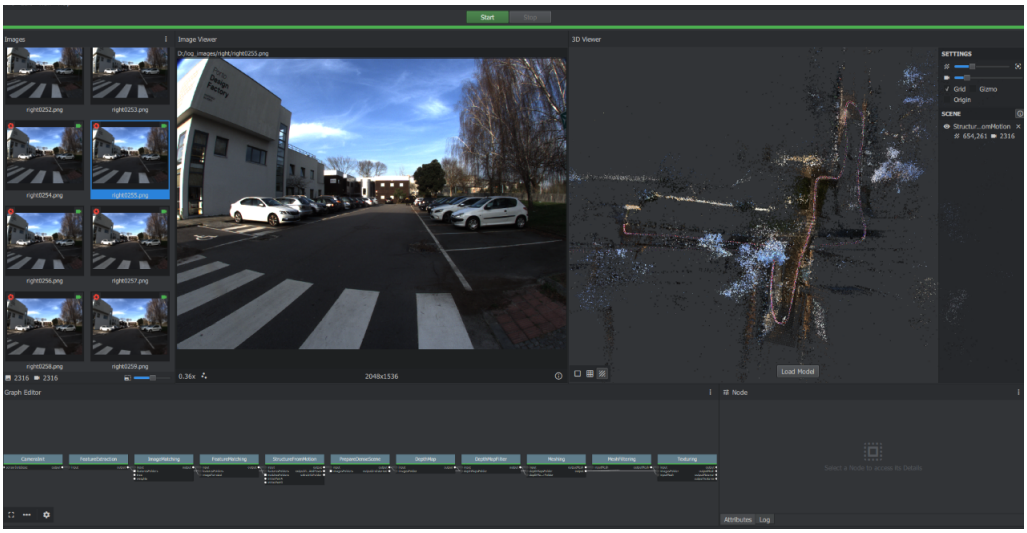


Figura 7.17: Interface gráfica Meshroom com dados processados.

O meshroom permite observar as *features* que foram extraídas em cada imagem (neste caso recorrendo ao método SIFT), com dois modos de visualização, apenas com pontos

no local de cada *feature* ou com quadrados orientados, estes mostram a orientação e magnitude atribuída a cada ponto de interesse detetado. A figura 7.18 mostra uma imagem com a representação de *features* através de quadrados orientados, sendo que no caso desta imagem foram detetadas 6679 pontos de interesse.

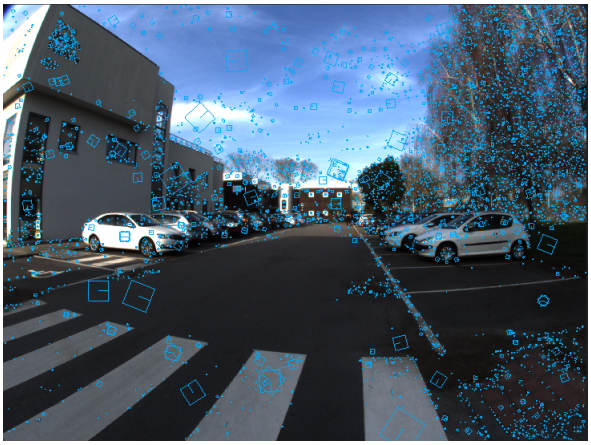


Figura 7.18: Imagem com os descritores SIFT representados por quadrados orientados.

No que toca a reconstrução e modelo 3D o Meshroom apresenta tanto a nuvem de pontos obtida depois de realizar SFM, bem como a *mesh* texturizada obtida depois de realizar o passo de *meshing* e *texturing*. No caso deste conjunto de imagens foi gerada uma nuvem de pontos com 654.261 pontos e o modelo resultante contém 1.604.920 triângulos. Na figura 7.19 encontram-se alguns exemplos de partes do modelo 3D gerado.

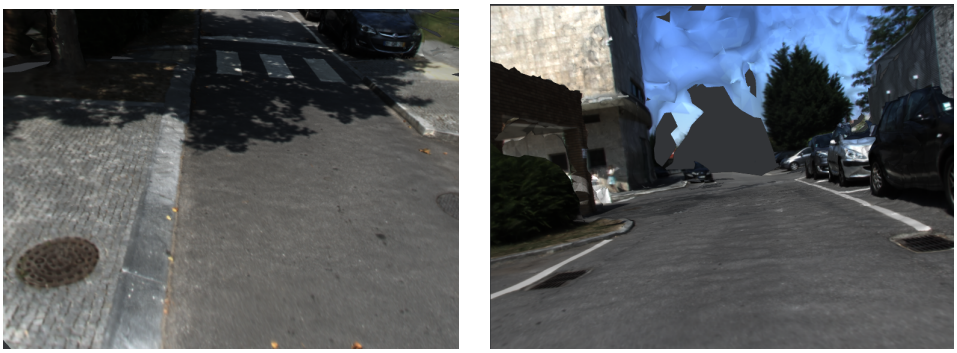


Figura 7.19: Exemplos da reconstrução 3D usando as imagens do *ground truth*.

Processando da mesma forma os dados adquiridos nos testes na mina das Aveleiras, é possível obter a reconstrução 3D dos túneis e galerias percorridos com o sistema. A

figura 7.20 mostra uma imagem exemplo com e sem as *features* identificadas. Neste caso foram extraídos o limite máximo imposto de pontos de interesse, 10000.

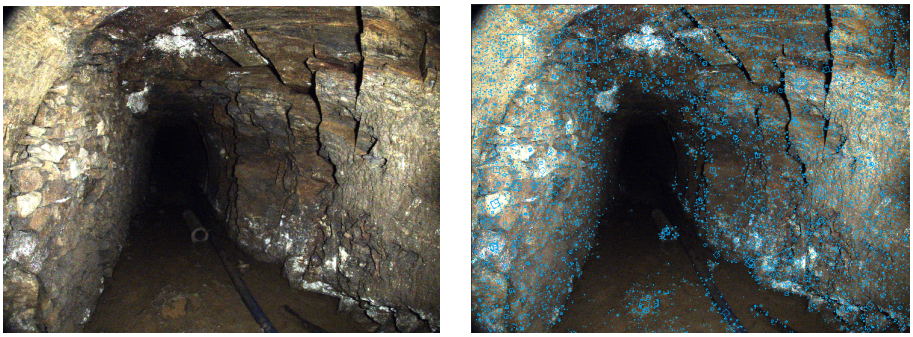


Figura 7.20: Esquerda: Imagem obtida do interior da mina. Direita: Imagem com representação das *features* SIFT extraídas.

No caso da figura 7.21 está apresentada a nuvem de pontos gerada representando o túnel principal da mina sendo possível observar a *pose* das câmaras ao longo do percurso realizado. Por fim na figura 7.22 é possível observar algumas partes da reconstrução 3D obtida do túnel principal da mina.

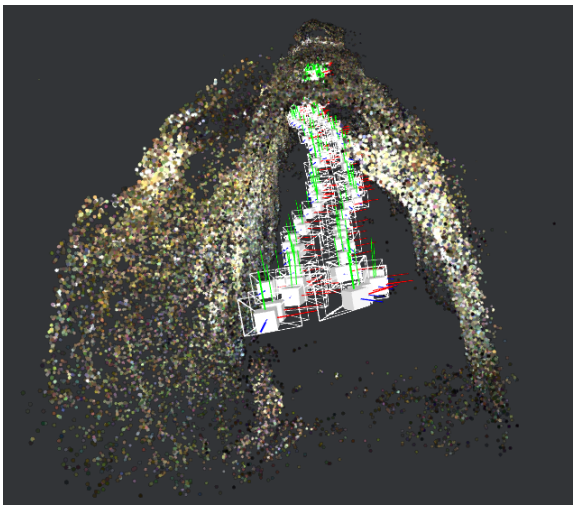


Figura 7.21: Nuvem de pontos obtida, sendo possível observar as poses estimadas das câmaras.

A reconstrução 3D da mina apresenta resultados promissores e com elevado nível de detalhe, sendo possível observar claramente vários artefactos, desde tubos de cimento e

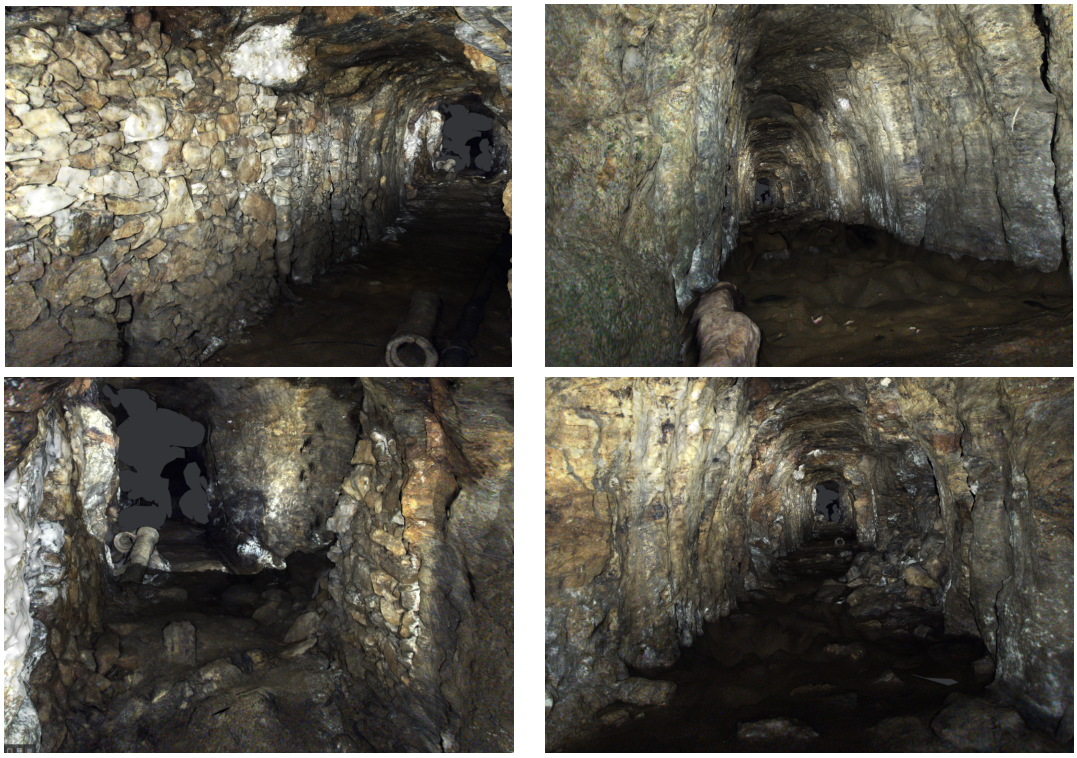


Figura 7.22: Exemplos da reconstrução 3D usando as imagens adquiridas na mina.

destroços no chão da mina, à intersecção entre o túnel principal e uma galeria secundária, e até à parte do túnel em que uma parede foi construída pelos mineiros, provavelmente devido a uma derrocada ou para reforçar estruturalmente a galeria, notando-se claramente a divisão entre os pequenos blocos que fazem parte da parede. Em parte estes resultados devem-se ao elevado número de *features* extraídas em cada imagem, permitindo a obtenção de uma nuvem de pontos muito densa e por consequência uma reconstrução mais fiel à realidade. Como é de se esperar para atingir este nível de qualidade o software precisa de ser executado durante longos períodos de tempo, chegando mesmo, para casos com valores de imagens elevados (mais de 2000), a demorar dias a realizar o processamento completo. Este é um dos aspetos negativos deste programa, não sendo de todo possível executa-lo em tempo real. No caso deste projeto, como o objetivo é apenas recolher os dados para posteriormente serem pós processados, esta desvantagem não acarreta um impacto demasiado elevado.

7.6 RTAB MAP

De forma a usar o *package* em ROS, *rtabmap_ros*, foi necessário proceder ao desenvolvimento de um *node*. Este é responsável por ler os parâmetros de calibração das câmaras e guardá-los num ficheiro com o formato *YAML*, publicando-os no tópico *camera_info* de modo a que o nó *stereo_image_proc* receba as imagens e os parâmetros das câmaras e realize a retificação das imagens. Retificar é um processo de transformação das imagens adquiridas de forma a que as linhas epipolares fiquem paralelas, de modo a simplificar a procura de *features* na outra imagem, uma vez que reduz a área de procura para uma linha de pixeis em vez da imagem toda. Na figura 7.23 está apresentado um exemplo de um par de imagens retificadas pelo nó *stereo_image_proc*.

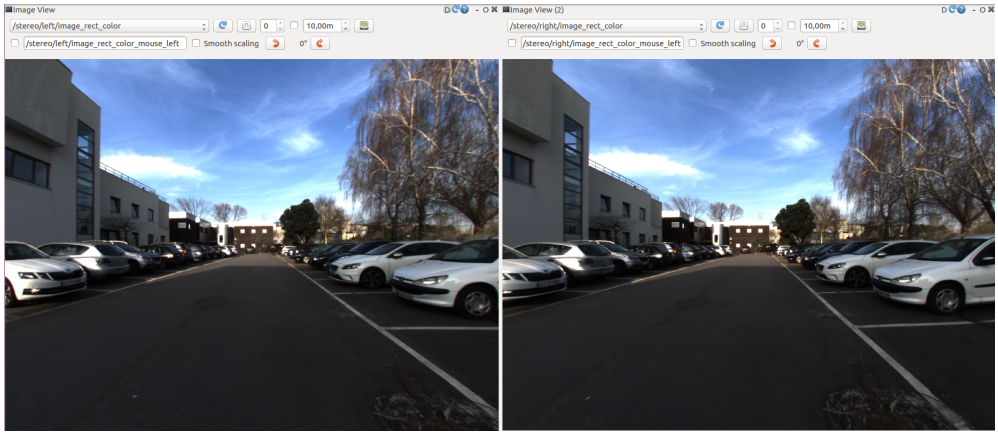


Figura 7.23: Par de imagens retificadas.

O package *rtabmap_ros* subscreve-se aos tópicos das imagens retificadas e dos parâmetros das câmaras, e depois de processar os dados das imagens do *ground truth* obtém se a trajetória e nuvem de pontos apresentada na figura 7.24. Nela é possível observar a existência de *loop-closures* (linhas a roxo), notando se a reduzida quantidade destes, como referido na secção da comparação com o *ground truth*. Na figura 7.25 está representada um exemplo de uma imagem na qual é visível a localização dos pontos de interesse extraídos usando o método SURF.

As reconstruções obtidas encontram-se apresentadas na figura 7.26 sendo possível observar que a deteção de poucas *features* afeta a densidade da reconstrução, apresentando-se esta com várias lacunas.

Não existem resultados da reconstrução e mapeamento 3D usando os dados adquiridos em ambiente subterrâneo, não existindo nenhuma reconstrução aceitável das galerias da

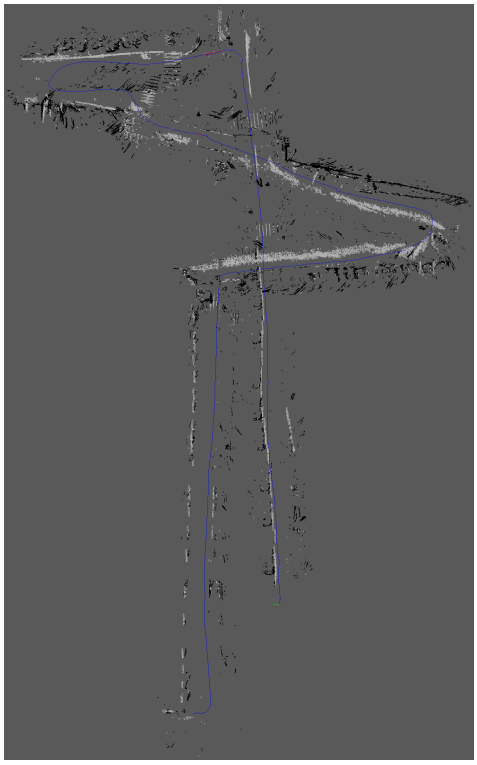


Figura 7.24: Trajetória e nuvem de pontos obtida usando o *software* RTAB-Map.

Figura 7.25: Imagens com *features* extraídas usando o método SURF.

mina das Aveleiras. Isto deve-se ao facto de, mesmo após inúmeras tentativas, o nó ROS

Figura 7.26: Reconstruções 3D obtidas usando o *software* RTAB-Map.

que efetua a odometria visual se "perder"ao entrar na mina, uma vez que não consegue detetar *features* em número suficiente para realizar a odometria. A causa pode ser derivada não da deficiência do algoritmo de deteção do *software*, mas por outros fatores, visto que o RTAB-map é um *software* com um número muito elevado de parâmetros que podem influenciar esta questão. De forma a resolver esta questão seria necessário tentar perceber a origem da fraca deteção de *features* no interior da mina.

Esta página foi intencionalmente deixada em branco.

Capítulo 8

Conclusão e Trabalho Futuro

Esta dissertação incide no desenvolvimento de uma plataforma de *surveing* que permite a recolha de dados essenciais para a realização de mapeamento e reconstruções 3D de ambientes subterrâneos. Este sistema contribuiu para a obtenção de dados de minas históricas desativadas no âmbito do projeto *MineHeritage*.

O trabalho desenvolvido resultou na construção do sistema *GeoTec*, desenvolvido de raiz quer a nível eletrónico quer a nível mecânico. O seu desenvolvimento teve como base um estudo profundo dos requisitos necessários, a nível de hardware e software. O *GeoTec* é um sistema robótico que permite recolher dados de visão, inerciais e de LiDAR, resultando na elaboração de reconstruções e mapas 3D em cenários GPS-*denied*. Para proceder ao desenvolvimento e implementação do sistema foi efetuado um estudo e levantamento dos requisitos, tanto de *hardware* como de *software*.

De forma a mostrar a utilidade dos dados recolhidos pelo sistema foram também estudados os algoritmos e técnicas usadas por dois *softwares* (RTAB-Map e Meshroom) que realizam reconstrução e mapeamento 3D através de odometria visual, procedendo-se à alteração/criação dos módulos necessários para a sua correta utilização.

Para analisar a eficácia de cada um dos programas no que toca à estimação da localização e trajetória foi realizado um *ground truth*, que permite obter uma estimação fiel à verdadeira trajetória descrita pelo sistema, como base de comparação com as trajetórias estimadas pelos programas.

Após os primeiros testes preliminares de laboratório para testar o correto funcionamento de todos os sensores, o sistema foi testado em contexto real. Tendo em conta a necessidade da utilização do sistema GeoTec no projeto *MineHeritage*, este foi testado numa das localizações de teste do projeto, uma mina desativada denominada por mina das Aveleiras, no mosteiro de Tibães em Braga. Foram recolhidos dados da mai-

oria do traçado acessível da mina, que permitiram apresentar resultados no que toca a reconstruções 3D e modelos da mina.

O desenvolvimento deste sistema, e os resultados preliminares obtidos no primeiro local de testes do projeto *MineHeritage* foram apresentados e discutidos na conferência "*4GEO — resources, materials, technologies and environment*", que decorreu em Novembro de 2019. Desta conferência resultou uma publicação numa Edição Especial na *Springer ASTI series* com o titulo "*GeoTec, a system for 3D mapping, imaging and laser scanner technology in underground environment (Tibães Mine, NW Portugal*".

Relativamente aos resultados obtidos utilizando o *software RTAB-Map*, existe uma deficiência a nível de reconstruções do ambiente subterrâneo. De forma a melhorar este ponto, seria interessante como trabalho futuro realizar uma análise mais profunda da causa do problema do *node* ROS de que realiza a odometria não conseguir detetar *features* suficientes no interior da mina para realizar odometria visual.

Usando os conhecimentos adquiridos ao longo da realização desta dissertação surge, como investigação futura, a necessidade de desenvolver uma versão mais leve e compacta do sistema, que permita o seu uso como uma mochila, para o transporte em túneis mais estreitos e de difícil acesso, uma vez que uma das limitações do sistema desenvolvido é a impossibilidade da sua introdução em túneis mais estreitos e acidentados. Outro desenvolvimento futuro poderá ser melhorar os modelos obtidos, procedendo ao desenvolvimento de algoritmos que integrem os dados do sistema inercial.

Bibliografia

[1] Sensonor. Stim 300, 2020. https://www.sensonor.com/products/inertial-measurement-units/stim300/, acedido: 16/01/2020.

[2] Axiomtec. Capa881, 2020. http://www.axiomtek.fr/Default.aspx?MenuId=Products&FunctionId=ProductView&ItemId=6425&C=CAPA881&upcat=270, acedido: 16/01/2020.

[3] FLIR Systems. Blackfly gige, 2020. https://www.flir.com/products/blackfly-gige?model=BFLY-PGE-31S4C-C, acedido: 16/01/2020.

[4] Goyo Optical Goyo North America. Gmthr23514mcn, 2020. http://goyonorthamerica.com/lenses/gmthr23514mcn/, acedido: 16/01/2020.

[5] Hokuyo. Distance data output/utm-30lx, 2020. https://www.hokuyo-aut.jp/search/single.php?serial=169, acedido: 16/01/2020.

[6] Unicore Communications. Ub482 - gps/bds/glonass/galileo multi-frequency high-precision rtk and heading board, 2020. http://www.unicorecomm.com/en/product/content_1605.html, acedido: 16/01/2020.

[7] Daniel Thiel, Maurıcio Goulart, Silvia Botelho, and R Bem. Hardware and software infrastructure to image acquisition using multiple cameras. *Universidad Federal do Rio Grande–FURG*, 2004.

[8] Historic England. Photogrammetric applications for cultural heritage. guidance for good practice, 128 p, 2017.

[9] Karl T Bates, Frank Rarity, Phillip L Manning, David Hodgetts, Bernat Vila, Oriol Oms, Àngel Galobart, and Robert L Gawthorpe. High-resolution lidar and photogrammetric survey of the fumanya dinosaur tracksites (catalonia): implications for the conservation and interpretation of geological heritage sites. *Journal of the Geological Society*, 165(1):115–127, 2008.

[10] F Nex and F Rinaudo. Photogrammetric and lidar integration for the cultural heritage metric surveys. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(Part 5):490–495, 2010.

[11] Fabio Remondino. Heritage recording and 3d modeling with photogrammetry and 3d scanning. *Remote sensing*, 3(6):1104–1138, 2011.

[12] Inga Siebke, Lorenzo Campana, Marianne Ramstein, Anja Furtwängler, Albert Hafner, and Sandra Lösch. The application of different 3d-scan-systems and photogrammetry at an excavation—a neolithic dolmen from switzerland. *Digital Applications in Archaeology and Cultural Heritage*, 10:e00078, 2018.

[13] Young Hoon Jo and Seonghyuk Hong. Three-dimensional digital documentation of cultural heritage site based on the convergence of terrestrial laser scanning and unmanned aerial vehicle photogrammetry. *ISPRS International Journal of Geo-Information*, 8(2):53, 2019.

[14] Dennis Strelow and Sanjiv Singh. Online motion estimation from image and inertial measurements. 2003.

[15] Jonathan Kelly, Srikanth Saripalli, and Gaurav S Sukhatme. Combined visual and inertial navigation for an unmanned aerial vehicle. In *Field and Service Robotics*, pages 255–264. Springer, 2008.

[16] Gabriel Nützi, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Fusion of imu and vision for absolute scale estimation in monocular slam. *Journal of intelligent & robotic systems*, 61(1-4):287–299, 2011.

[17] Agostino Martinelli. Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *IEEE Transactions on Robotics*, 28(1):44–60, 2011.

[18] Todd Lupton and Salah Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28(1):61–76, 2011.

[19] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. 2017.

[20] Beat Kueng, Elias Mueggler, Guillermo Gallego, and Davide Scaramuzza. Low-latency visual odometry using event-based feature tracks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 16–23. IEEE, 2016.

[21] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based visual inertial odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5391–5399, 2017.

[22] Jonathan Kelly and Gaurav S. Sukhatme. Visual-inertial simultaneous localization, mapping and sensor-to-sensor self-calibration. *2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation - (CIRA)*, pages 360–368, 2009.

[23] Jorge Lobo and Jorge Dias. Relative pose calibration between visual and inertial sensors. *The International Journal of Robotics Research*, 26(6):561–575, 2007.

[24] Faraz M Mirzaei and Stergios I Roumeliotis. A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation. *IEEE transactions on robotics*, 24(5):1143–1156, 2008.

[25] Jonathan Kelly and Gaurav S Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *The International Journal of Robotics Research*, 30(1):56–79, 2011.

[26] Juan Li, Juan A Besada, Ana M Bernardos, Paula Tarrío, and José R Casar. A novel system for object pose estimation using fused vision and inertial data. *Information Fusion*, 33:15–28, 2017.

[27] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572. IEEE, 2007.

[28] Eagle S Jones and Stefano Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *The International Journal of Robotics Research*, 30(4):407–430, 2011.

[29] Jing Chen, Wei Liu, Yongtian Wang, and Junwei Guo. Fusion of inertial and vision data for accurate tracking. In *Fourth International Conference on Machine Vision (ICMV 2011): Machine Vision, Image Processing, and Pattern Analysis*, volume 8349, page 83491D. International Society for Optics and Photonics, 2012.

[30] Arif Tanju Erdem and Ali Özer Ercan. Fusing inertial sensor data in an extended kalman filter for 3d camera tracking. *IEEE Transactions on Image Processing*, 24(2):538–548, 2014.

[31] Mary Alatise and Gerhard P Hancke. Pose estimation of a mobile robot using monocular vision and inertial sensors data. In *2017 IEEE AFRICON*, pages 1552–1557. IEEE, 2017.

[32] Mingyang Li and Anastasios I Mourikis. High-precision, consistent ekf-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[33] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2016.

[34] Stefan G Chroust and Markus Vincze. Fusion of vision and inertial data for motion and structure estimation. *Journal of Robotic Systems*, 21(2):73–83, 2004.

[35] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[36] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. Georgia Institute of Technology, 2015.

[37] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[38] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[39] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2016.

[40] S-H Jung and Camillo J Taylor. Camera trajectory estimation using inertial sensor measurements and structure from motion results. In *Proceedings of the 2001 IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, volume 2, pages II–II. IEEE, 2001.

[41] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 321–328. IEEE, 2000.

[42] David Ferguson, Aaron Morris, Dirk Haehnel, Christopher Baker, Zachary Omohundro, Carlos Reverte, Scott Thayer, Charles Whittaker, William Whittaker, Wolfram Burgard, et al. An autonomous robotic system for mapping abandoned mines. In *Advances in Neural Information Processing Systems*, pages 587–594, 2004.

[43] Sebastian Thrun, Dirk Hahnel, David Ferguson, Michael Montemerlo, Rudolph Triebel, Wolfram Burgard, Christopher Baker, Zachary Omohundro, Scott Thayer, and William Whittaker. A system for volumetric robotic mapping of abandoned mines. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 3, pages 4270–4275. IEEE, 2003.

[44] Eduardo José Pinto Soares. Sistema de perceção 3d subaquático com projetor laser rotativo: Calibração e reconstrução tridimensional. Master's thesis, Instituto Superior de Engenharia do Porto, 2018.

[45] Kenji Hata and Silvio Savarese. Cs231a course notes 1: Camera models, 2020. `https://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf`, acedido: 20/01/2020.

[46] Mario Sabbatelli. 3d vision-based perception and modelling techniques for intelligent ground vehicles. 2015.

[47] MathWorks. What is camera calibration?, 2020. `https://www.mathworks.com/help/vision/ug/camera-calibration.html`, acedido: 20/01/2020.

[48] John Mallon and Paul F. Whelan. Precise radial un-distortion of images. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1 - Volume 01*, ICPR '04, page 18–21, USA, 2004. IEEE Computer Society.

[49] OpenCV. Camera calibration with opencv, 2020. `https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html`, acedido: 20/01/2020.

[50] OpenCV. Epipolar geometry, 2020. `https://docs.opencv.org/master/da/de9/tutorial_py_epipolar_geometry.html`, acedido: 20/01/2020.

[51] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[52] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

[53] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

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

[55] Paul L Rosin. Measuring corner properties. *Computer Vision and Image Understanding*, 73(2):291–307, 1999.

[56] OpenCV. Introduction to surf (speeded-up robust features), 2020. `https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html`, acedido: 20/01/2020.

[57] OpenCV. Orb (oriented fast and rotated brief), 2020. `https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html`, acedido: 20/01/2020.

[58] Paul Newman and Kin Ho. Slam-loop closing with visually salient features. In *proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 635–642. IEEE, 2005.

[59] Kin Leong Ho and Paul Newman. Loop closure detection in slam by combining visual and spatial appearance. *Robotics and Autonomous Systems*, 54(9):740–749, 2006.

[60] Mathieu Labbé and François Michaud. Memory management for real-time appearance-based loop closure detection. In *2011 IEEE/RSJ international conference on intelligent robots and systems*, pages 1271–1276. IEEE, 2011.

[61] Mathieu Labbe and Francois Michaud. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics*, 29(3):734–745, 2013.

[62] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[63] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Adaptive structure from motion with a contrario model estimation. In *Asian Conference on Computer Vision*, pages 257–270. Springer, 2012.

[64] Michal Jancosek and Tomás Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR 2011*, pages 3121–3128. IEEE, 2011.

[65] AliceVision. Alicevision - photogrammetric computer vision framework, 2020. `https://alicevision.org/`, acedido: 30/01/2020.

[66] Ives Rey Otero. *Anatomy of the SIFT Method*. PhD thesis, 2015.

[67] Pablo F Alcantarilla and T Solutions. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Trans. Patt. Anal. Mach. Intell*, 34(7):1281–1298, 2011.

[68] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168. Ieee, 2006.

[69] 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.

[70] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814. IEEE, 2005.

[71] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM transactions on graphics (TOG)*, 21(3):362–371, 2002.

[72] Aaron Martinez and Enrique Fernández. *Learning ROS for robotics programming*. Packt Publishing Ltd, 2013.

[73] rexroth A Bosh Company. Strut profile 30x30, 2020. `https://www.boschrexroth.com/en/xc/products/product-groups/assembly-technology/basic-mechanic-elements/strut-profiles/strut-profiles-slot-8-modular-dimensions-30/30x30#`, acedido: 16/01/2020.

[74] Paul Krzyzanowski. Clock synchronization. *Lectures on distributed systems*, 2002, 2000.

[75] Bernhard Sterzbach. Gps-based clock synchronization in a mobile, distributed real-time system. *Real-Time Systems*, 12(1):63–75, 1997.

[76] H.I. Chaminé, M.E. Lopes, J.F Trigo, and M.J. Costa Dias. As minas de tibães: Um património hidrogeológico, geomineiro e monástico the tibães mines: a hydrogeological, geomining and monastic heritage. Laboratory of Cartography and Applied Geology (LABCARGA), Geotechnical Department, School of Engineering of Porto, Portugal, 2017.

[77] A. Pires, A. Dias, P. Rodrigues, P. Silva, L. Sousa, T. Santos, A. Oliveira, A. Ferreira, J. Almeida, A. Martins, H.I. Chaminé, and E. Silva. Geotec, a system for 3 d mapping, imaging and laser scanner technology in underground environment (tibães mine, nw portugal). In *Advances in Geoengineering, Geotechnologies, and Geoenvironment for Earth Systems and Sustainable. Georesources Management – Proceedings of the 1st Conference on Georesources, Geomaterials, Geotechnologies and Geoenvironment (4GEO), [H.I. Chaminé & J.A. Fernandes (Editors)]*. Springer ASTI Series, in prep, 2020.