Tiago Manuel Rua Matos Talhada

Perceção 3D utilizando uma câmara estéreo

Tiago Manuel Rua Matos Talhada

Perceção 3D utilizando uma câmara estéreo

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Mecânica, realizada sob orientação científica de Vítor Santos, Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro.

O júri / The jury

Presidente / President

Prof. Doutor Jorge Augusto Fernandes Ferreira
Professor Auxiliar da Universidade de Aveiro

Vogais / Committee

Prof. Doutor Alexandre José Malheiro Bernardino
Professor Auxiliar do Instituto Superior Técnico

Prof. Doutor Vítor Santos
Professor Associado da Universidade de Aveiro (orientador)

Agradecimentos / Acknowledgements

Em primeiro lugar, gostaria de agradecer ao Professor Doutor Vítor Santos pela orientação, por toda a ajuda disponível e pela motivação no trabalho. Um agradecimento especial ao Miguel Oliveira e ao Jorge Almeida pelos conhecimentos transmitidos e pelas preciosas dicas de programação. Um obrigado aos meus colegas de mestrado pelas horas de gargalhadas, pelo companheirismo e pela boa disposição sempre presente no laboratório. Queria deixar também um agradecimento especial à minha irmã Daniela pelas dicas na prática de bom Português e pela correção ortográfica. Por fim, mas não menos importante, aos meus pais um enorme obrigado pelo esforço incansável que têm feito para garantir a minha formação.

Palavras-chave

Visão estéreo, Perceção, Segmentação euclidiana, Crescimento de regiões, Mecanismos de atenção, Sistemas de apoio à condução.

Resumo

Este trabalho consiste na obtenção e processamento de dados tridimensionais provenientes de um sensor estéreo, com o objetivo principal de desenvolver um mecanismo de perceção para sistemas de apoio à condução. O processo é iniciado com a calibração das câmaras do sensor a fim de obter dados com a melhor qualidade possível, para facilitar o posterior processamento, cuja etapa inicial é baseada na filtragem da nuvem de pontos para remoção de ruído e redução da quantidade de dados. A etapa seguinte consiste na determinação do conjunto de pontos que representam o plano do chão na nuvem de pontos, para que seja possível estimar a área navegável do veículo e consequentemente o conjunto de obstáculos que se encontram nessa área. A finalidade é proceder à segmentação dos diversos obstáculos relativamente ao plano do chão e dentro da zona navegável de duas formas distintas: a primeira abordagem consiste na aplicação do algoritmo de segmentação euclidiana à nuvem de pontos do sensor estéreo, e a segunda consiste na implementação de um algoritmo de crescimento de regiões para aplicação aos dados bidimensionais segmentados provenientes de sistemas de perceção laser 2D. Os métodos desenvolvidos são testados em condições reais e são analisadas situações de falha, vantagens e desvantagens da aplicação de um método relativamente ao outro.

Keywords Stereo vision, Perception, Euclidean clustering, Region growing, Attention mechanisms ,Driver assistance systems.

Abstract

This work covers the achievement and processing of three-dimensional data from a stereo sensor, with the main objective to develop a perception mechanism to support driver assistance systems. The process begins with camera calibration, to obtain the point cloud of best possible quality to favour the further processing, whose initial step is based on filtering the point cloud to reduce noise and down sample the data. The next step is the determination of the set of points that represent the ground in the point cloud, so it is possible to estimate the navigable area of the vehicle and consequently the number of obstacles and location within that area. The purpose is to do the clustering of the various obstacles within the navigable area relatively to the ground plane which was done using two different methods: the first approach involves the application of Euclidean cluster segmentation on stereo point cloud, and the second consists of the implementation of a region growing algorithm applied on two-dimensional segmented data from laser scans. The methods performed are tested under real conditions and failure situations, advantages and drawbacks of their applications are analysed and compared.

Conteúdo

1 Introdução **1**
- 1.1 Enquadramento . . . 1
  - 1.1.1 O projeto ATLAS . . . 1
- 1.2 Ambiente de Programação ROS . . . 2
- 1.3 Apresentação do problema e objetivos . . . 4
- 1.4 Estrutura da Dissertação . . . 4

2 Sistema de aquisição estéreo **5**
- 2.1 Descrição do sensor . . . 5
- 2.2 Integração com o ROS . . . 5
- 2.3 Calibração . . . 7
  - 2.3.1 Modelo de câmara e propriedades óticas . . . 7
  - 2.3.2 Procedimento . . . 8
  - 2.3.3 Calibração na Bumblebee xb3 . . . 9
  - 2.3.4 Verificação da calibração . . . 10
- 2.4 Cálculo estéreo . . . 14
  - 2.4.1 Princípio de funcionamento . . . 14
  - 2.4.2 Correspondência estéreo . . . 14
  - 2.4.3 Cálculo da disparidade . . . 15
  - 2.4.4 Parâmetros . . . 18
  - 2.4.5 Reconstrução tridimensional . . . 20

3 Processamento de nuvens de pontos **23**
- 3.1 Filtros aplicados e transformações . . . 23
  - 3.1.1 Transformações geométricas . . . 23
  - 3.1.2 *Voxel Grid* . . . 24
  - 3.1.3 Filtros de distâncias . . . 26
- 3.2 Deteção do plano do chão . . . 26
- 3.3 Segmentação de obstáculos . . . 30
  - 3.3.1 Cálculo do perímetro navegável . . . 30
  - 3.3.2 Segmentação por distância euclidiana . . . 31
  - 3.3.3 Algoritmo de crescimento de regiões . . . 33

4 Resultados e discussão **37**
- 4.1 Resultados obtidos . . . 37
- 4.2 Eficiência do cálculo estéreo . . . 38
- 4.3 Segmentação do plano do chão . . . 39

4.4 Segmentação euclidiana e crescimento de regiões 40

5 Conclusões e Trabalho futuro **45**
5.1 Conclusões finais 45
5.2 Trabalho futuro 46

Lista de Tabelas

2.1 Resultados da verificação: Profundidade e superfície. 11
2.2 Resultados da verificação: Altura, Largura 12
2.3 Parâmetros do algoritmo *Block Match* 19

Lista de Figuras

1.1 ATLASCAR (Protótipo à escala real) [Atlas, 2011]. 1

2.1 *Bumblebee XB3* com três câmaras. 5
2.2 *Baselines* da xb3 6
2.3 Modelo de câmara *pinhole* [Wöhler, 2009]. 7
2.4 Interface do software de calibração. 8
2.5 Exemplo de resultado da calibração estéreo. 9
2.6 Montagem da verificação: Profundidade e superfície. 10
2.7 Montagem da verificação: Altura, Largura. 11
2.8 *Speckling* em pinos de marcação de estrada. 13
2.9 *Speckling* em arestas de obstáculos. 13
2.10 Par de imagens estéreo. 14
2.11 Correspondência estéreo. 15
2.12 Mapa de disparidade. 16
2.13 Exemplo visual do efeito da alteração de parâmetros. 19
2.14 Geometria do sensor estéreo [Bradski and Kaehler, 2008]. 20
2.15 Relação entre disparidade e profunidade [Bradski and Kaehler, 2008]. . . . 21
2.16 Exemplo de nuvem de pontos (Reconstrução tridimensional). 22

3.1 Sistemas de coordenadas dos sensores. 24
3.2 Exemplo de aplicação de vários tamanhos de *voxel grid*. 25
3.3 Exemplo de filtragem com aplicação de *voxel grid* e filtro de distâncias. . . 26
3.4 Geometria dos planos: veículo (B) e chão (A). 28
3.5 Exemplo da aplicação do algoritmo na presença de múltiplos planos. . . . 29
3.6 Exemplo da aplicação do algoritmo de classificação da zona navegável. . . 31
3.7 Exemplo da aplicação da segmentação euclidiana. 33
3.8 Exemplo da aplicação do algoritmo de crescimento de regiões. 35

4.1 Mapa do percurso efetuado. 38
4.2 Frequências do cálculo estéreo (*short* e *wide*). 39
4.3 Frequências da segmentação do plano do chão. 40
4.4 Taxas de publicação com *voxels* de 20 cm. 41
4.5 Taxas de publicação com *voxels* de 10 cm. 41
4.6 Taxas de publicação com *voxels* de 5 cm. 42
4.7 Falha na aplicação da segmentação euclidiana. 43
4.8 Falhas de registo nos sensores. 43

Capítulo 1

Introdução

1.1 Enquadramento

1.1.1 O projeto ATLAS

O projeto ATLAS surgiu no departamento de Engenharia Mecânica da Universidade de Aveiro para a participação em competições de robótica. Com o sucesso obtido nas provas, surgiu a necessidade de criar um protótipo à escala real com o objetivo de estudo e desenvolvimento de soluções de apoio à condução. Para tal, adaptou-se o veículo *Ford Escort* equipado com alguns sensores e modificado mecanicamente para ser capaz de realizar algumas operações sem a necessidade da presença do condutor. O ATLASCAR constitui o protótipo utilizado na elaboração deste trabalho (Figura 1.1).

Figura 1.1: ATLASCAR (Protótipo à escala real) [Atlas, 2011].

Atualmente, o veículo encontra-se equipado com diversas unidades sensoriais das quais fazem parte as seguintes:

- Câmara estéreo - Câmara *FireWire* capaz de transmitir três imagens em simultâneo com resolução de 1280x960 a uma frequência de 16fps. As imagens são processadas em computador para gerar dados tridimensionais;

- Visão foveada - Sistema de visão com duas câmaras semelhantes mas com diferentes óticas. Cada câmara tem uma distância focal e ângulo de visão distintos: enquanto uma fornece uma visão geral do ambiente, a outra fornece grande detalhe acerca de um determinado obstáculo. As câmaras encontram-se suportadas por um mecanismo rotativo que permite orientá-las em diversas direções. Este sistema é particularmente importante no desenvolvimento de mecanismos de atenção;

- Laser 3D - A solução de laser 3D aplicada é baseada num laser 2D instalado num suporte rotativo capaz de gerar dados tridimensionais utilizados para caraterizar o ambiente;

- Lasers 2D - Dois lasers equipam o para-choques frontal do veículo, um do lado esquerdo e outro do direito. Os lasers permitem detetar imediatamente obstáculos que se encontrem quer à frente quer nas laterais do veículo, desde que se encontrem no plano de ação do laser. Um terceiro laser está instalado no tejadilho do veículo direcionado para a estrada com o objetivo de proceder à reconstrução tridimensional de um mapa local ao longo do tempo;

- IMU - O IMU (*Inertial measurement unit*) é um dispositivo eletrónico que contém acelerómetros e giroscópios para medir orientações e acelerações;

- Outros sensores de baixo nível - Outros sensores estão atualmente instalados para medições de baixo nível, tais como *encoder* de velocidade, sensores de pressão nos pedais, monitorização de suspensão, entre outros.

A utilização destes sensores requer um fornecimento de energia relativamente elevado, por isso surgiu a necessidade de instalar um segundo alternador no veículo juntamente com um inversor AC/DC para inversão de sinal e ainda uma UPS (Uninterruptible Power Supply) para a gestão de toda a energia elétrica do sistema.

1.2 Ambiente de Programação ROS

Para facilitar a interação entre os diversos sensores e controladores de baixo nível, utiliza-se atualmente a plataforma ROS (*Robot Operating System*), de arquitetura modular. Esta plataforma permite a criação e gestão de diversos módulos de software e garante a comunicação entre eles de uma forma estruturada e robusta [ROS, 2012].

O ROS é constituído por uma plataforma de programação *open source* para robôs, criada em 2007 para o desenvolvimento do projeto STAIR (*STanford Artificial Intelligence Robot*) [Morgan Quigley, 2007]. A partir dessa data o grupo de desenvolvimento de hardware e software *open source Villow Garage* tem assegurado a continuidade do projeto [Quigley et al., 2009].

O sistema operativo ROS foi desenvolvido com a intenção de cumprir os seguintes objetivos [Quigley et al., 2009]:

- Facilitar o desenvolvimento modular do software;

- Constituir uma plataforma gratuita e *open source*;

- Fornecer ferramentas para implementação de código como servidor de parâmetros de configuração, visualizador de mensagens, auto geração de documentação, entre outros;

- Suportar várias linguagens de programação (atualmente *C++*, *Python*, *Octave* e *LISP* [Quigley et al., 2009]);

- Constituir um sistema operativo leve e robusto;

- Permitir a utilização *peer-to-peer*.

O ambiente ROS apresenta diversas facilidades, como por exemplo, a possibilidade de gravação de mensagens de nodos em disco, um facto interessante na pois permite a posterior análise e desenvolvimento de software em laboratório com dados reais. Por exemplo, é possível gravar as imagens de uma câmara em disco e posteriormente reproduzi-las como se fossem novamente publicadas pela câmara e desenvolver o software de processamento de imagem *offline*. A plataforma ROS está implementada de acordo com a nomenclatura de nodos, mensagens, tópicos e serviços [Quigley et al., 2009]. Os nodos são os processos que fazem a computação do sistema, e que constituem os diversos módulos do software. Com a arquitetura modular, é necessário que os nodos comuniquem entre si, processo efetuado através de mensagens. Uma mensagem constitui uma estrutura de dados específica com os tipos de dados primitivos das linguagens de programação (*integer*, *floating point*, *boolean*, entre outros). Uma mensagem pode ser composta por outras mensagens ou por vetores de mensagens e podem constituir estruturas de dados bastante complexas. Para que um determinado nodo possa subscrever ou publicar uma determinada mensagem no sistema, é necessário atribuir-lhe um nome. Desta forma, o nome de uma mensagem constitui um tópico. É possível que possam existir diferentes nodos a publicar e subscrever no mesmo tópico, bem como o mesmo nodo publicar e subscrever diversos tópicos. A comunicação entre mensagens é feita sob a filosofia publicar/subscrever que, embora seja bastante flexível, não é adequada para situações em que seja necessário o uso de RPC(*Remote Procedure Call*) [Quigley et al., 2009]. Por isso, na constituição do sistema ROS existem os serviços para comunicações do tipo pedido/resposta. Desta forma, um serviço constitui um par de mensagens: uma para o pedido e outra para a resposta.

O software baseado em ROS é organizado em *packages*. A definição de *package* é um pouco complexa, pois depende da aplicação a que se destina e o seu grau de complexidade. Contudo uma *package* consiste obrigatoriamente num diretório que contém um ficheiro XML com uma descrição geral e com a informação sobre dependências de software [Quigley et al., 2009]. Normalmente, no diretório de uma *package* existem diversos sub diretórios para organizar a informação e separar, por exemplo, ficheiros de código fonte de ficheiros executáveis para que o software possa ser desenvolvido de uma forma organizada. Para exemplificar o conceito, existem *packages* destinadas à comunicação com sensores, à perceção, navegação ou planeamento de trajetórias.

Desta forma, para garantir uma boa continuidade do software desenvolvido, é de extrema importância garantir a sua organização e documentação e nesse sentido a plataforma ROS assume um papel importante.

1.3 Apresentação do problema e objetivos

Este trabalho enquadra-se no âmbito de desenvolvimento de sistemas avançados de perceção com visão estéreo. Este tipo de sistemas é essencial no controlo de veículos autónomos para caraterização do ambiente que rodeia o veículo. No contexto do projeto ATLAS, na área de desenvolvimento de sistemas de apoio à condução, pretende-se abordar a temática da visão estéreo em dois níveis principais. Numa primeira etapa, pretende-se avaliar e caraterizar a qualidade do sistema de aquisição disponível e analisar possibilidades que podem ser tomadas para melhoramento do sistema. Numa segunda fase do trabalho pretende-se desenvolver uma aplicação capaz de interpretar os dados tridimensionais obtidos para segmentação de obstáculos e respetiva caraterização.

A intenção é desenvolver um algoritmo capaz de segmentar obstáculos em relação ao plano do chão. No final, a aplicação deverá ser capaz de indicar os obstáculos que se encontrem dentro da zona navegável do veículo, desde que estes se encontrem na zona de alcance da câmara.

1.4 Estrutura da Dissertação

Este trabalho encontra-se dividido em cinco capítulos distintos.

No primeiro capítulo, descreve-se sucintamente o enquadramento do trabalho no âmbito de desenvolvimento de sistemas avançados de apoio à condução. Descrevem-se também os principais objetivos desta dissertação de Mestrado.

No segundo capítulo, descreve-se o modo de procedimento de calibração do sensor estéreo, bem como o procedimento a efetuar para a obtenção de mapas de disparidade e reconstruções tridimensionais.

Ao longo do capítulo três, descrevem-se os diversos mecanismos de processamento e segmentação de nuvens de pontos. Neste capítulo, descrevem-se os métodos adotados para segmentação do chão, e abordam-se duas metodologias distintas para a posterior segmentação de obstáculos: segmentação da nuvem de pontos através da distância euclidiana, e segmentação através da aplicação de um algoritmo de crescimento de regiões a pontos "semente".

No quarto capítulo apresentam-se os resultados obtidos ao longo de um percurso de navegação, em que são comparados os métodos de segmentação aplicados, e são enumeradas algumas situações de falha das aplicações.

Por último, no quinto capítulo apresentam-se algumas propostas de continuação do trabalho bem como as conclusões finais da dissertação.

Capítulo 2

Sistema de aquisição estéreo

2.1 Descrição do sensor

O sensor estéreo utilizado é a *Bumblebee XB3* da *PointGrey*. Este sensor carateriza-se essencialmente por possuir três câmaras.

A utilização de várias câmaras no mesmo sensor tem como objetivo a simulação do sistema de visão binocular em que a partir de duas imagens obtidas no mesmo cenário é possível criar uma representação tridimensional do espaço. Esta representação é calculada mediante o conhecimento prévio dos parâmetros de calibração das câmaras e da distância entre elas (*baseline*). O facto deste sensor possuir três câmaras permite combiná-las em dois pares distintos, ou seja, obter dois *baselines* (Figura 2.1).

Figura 2.1: *Bumblebee XB3* com três câmaras.

A principal vantagem na utilização deste dispositivo é a possibilidade de calcular a disparidade com dois pares de imagens. Dependendo da aplicação pretendida pode-se utilizar o *baseline* de 120 milímetros ou de 240 milímetros, ou se preferível em simultâneo [PointGrey, 2011]. A utilização de um *baseline* maior torna-se vantajosa quando se pretende reconhecer propriedades do ambiente num alcance de profundidade mais elevado e obter cenários mais distantes, uma vez que para um *baseline* fixo o erro de medição de disparidade aumenta quadraticamente com a profundidade [Gallup, 2008].

2.2 Integração com o ROS

De forma a integrar a câmara *bumblebee xb3* na plataforma ROS (*Robot Operating System*), foi necessário desenvolver um nodo capaz de comunicar com a câmara através da interface IEEE-1394b e publicar os respetivos tópicos de imagem para o processamento estéreo posterior.

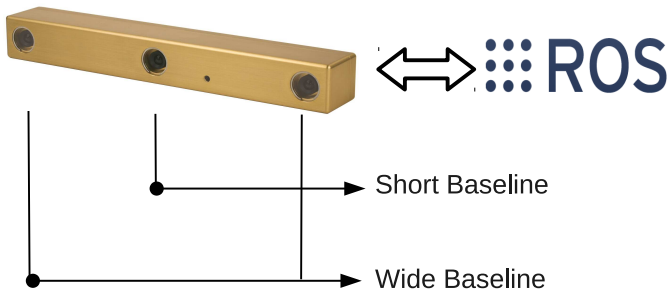


Figura 2.2: *Baselines* da xb3

O diagrama 2.2 representa duas combinações possíveis para *baselines*. A lente da esquerda em combinação com a central formam o *short baseline* e com a da direita formam o *wide baseline*.

O nodo responsável pela leitura de imagens da câmara é baseado em bibliotecas escritas em C para a comunicação *FireWire* para obtenção de imagens que são posteriormente convertidas para o formato do OpenCv para que seja possível efetuar algumas operações como a diminuição da resolução das imagens, ou filtragem. O nodo é capaz de publicar para o ambiente ROS os seguintes tópicos:

- $/short/left/image\_raw$;
- $/short/left/camera\_info$;
- $/short/right/image\_raw$;
- $/short/right/camera\_info$;
- $/wide/left/image\_raw$;
- $/wide/left/camera\_info$;
- $/wide/right/image\_raw$;
- $/wide/right/camera\_info$.

Cada tópico acima enumerado com a terminação $image\_raw$ representa o conjunto de píxels obtidos diretamente das câmaras respetivas, cuja resolução pode ser ajustada quer para 1280x960, quer para 640x480. Os tópicos do tipo $camera\_info$ contém dados relativos às matrizes das câmaras e dos *baselines*, nomeadamente as matrizes intrínsecas, de retificação e de projeção, e ainda os coeficientes de distorção. Esta estrutura de dados é essencial para poder executar as diversas ferramentas do ROS destinadas ao processamento estéreo. Normalmente, não são utilizadas imagens com resolução muito elevada, uma vez que quanto maior a dimensão das imagens maior será o custo computacional envolvido. Neste caso, as imagens originais do dispositivo, com uma resolução de 1280x960, foram ajustadas para 640x480, após passagem de um filtro gaussiano de 5x5 e de rejeitar as colunas e linhas pares.

2.3 Calibração

A calibração consiste na determinação da relação entre os referenciais do plano da imagem, da câmara e do mundo. [Wöhler, 2009]. Este passo é necessário para assegurar a continuidade do trabalho uma vez que é responsável pela determinação da relação entre o mundo e o sensor. As subsecções seguintes descrevem as etapas necessárias para efetuar a calibração de câmaras, e como a calibração pode ser feita para o caso específico da *Bumblebee Xb3*.

2.3.1 Modelo de câmara e propriedades óticas

O modelo de câmara adotado é o *pinhole* no qual a câmara é representada pelo seu centro ótico, situado entre o espaço tridimensional e o plano da imagem; e pelo eixo ótico, perpendicular ao plano da imagem e que passa pelo centro ótico. O ponto de interseção entre o eixo ótico e o plano da imagem denomina-se por ponto principal. A distância do ponto principal ao centro ótico é designada por distância principal [Wöhler, 2009]. A Figura 2.3 ilustra o modelo de câmara *pinhole* para melhor compreensão da geometria envolvida.

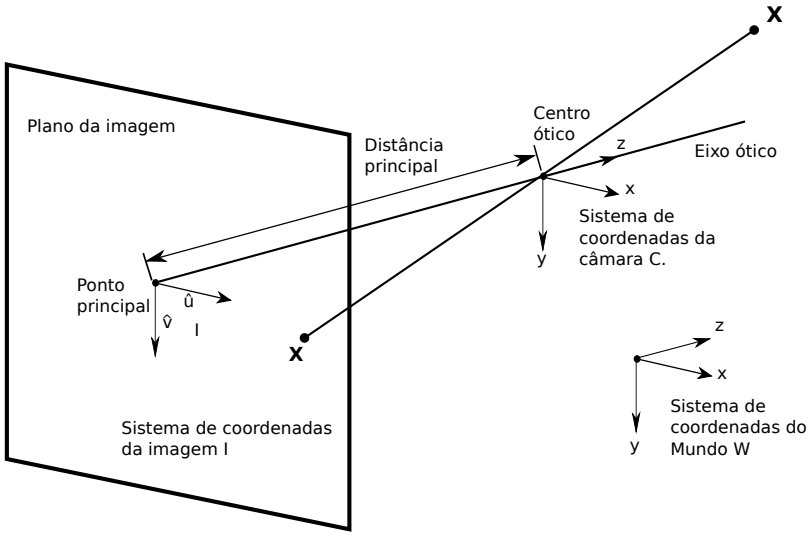


Figura 2.3: Modelo de câmara *pinhole* [Wöhler, 2009].

A transformação entre a lente da câmara e o plano da imagem (2.3) é caraterizada pela matriz intrínseca da câmara (expressão 2.1),

$$A = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

Os coeficientes α_x e α_y representam a distância focal, o coeficiente γ é o coeficiente de distorção entre x e y, e os parâmetros u_0 e v_0 representam o ponto principal [Wöhler, 2009]. A matriz intrínseca é independente do cenário em que a câmara se encontra e uma vez estimada pode ser reutilizada, desde que a distância focal não se altere (no caso de câmaras com zoom).

Para a descrição do movimento entre a câmara e o cenário, utiliza-se a matriz dos parâmetros extrínsecos, que representa uma translação e uma rotação, ou seja, a transformação entre o referencial do mundo e o referencial da câmara 2.3.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}. \tag{2.2}$$

Desta forma é possível estabelecer a relação entre as coordenadas de pontos do referencial do mundo W e o referencial da imagem I (Figura 2.3), através da expressão 2.3.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \times R \times \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.3}$$

A determinação das matrizes intrínsecas e extrínsecas é essencial para poder efetuar uma representação de imagens coerente. Desta forma, é necessário recorrer a um procedimento de calibração de câmaras para estimar todos os parâmetros envolvidos pela aplicação inversa da equação 2.3, ou seja, determinar os parâmetros da câmara a partir de dados conhecidos do mundo.

2.3.2 Procedimento

A calibração pode ser feita recorrendo a um módulo de ROS desenvolvido propositadamente para câmaras estéreo (*stereo_image_proc*). Teoricamente, a calibração pode ser feita com um algoritmo de visão sobre um objeto qualquer, contudo, o módulo de calibração do ROS está preparado para reconhecer zonas de elevado contraste, normalmente um padrão em xadrez de fundo branco com 8x6 zonas de reconhecimento. A dimensão do tabuleiro de calibração é configurável, contudo, normalmente utiliza-se a dimensão A0. Para proceder à calibração, basta executar o módulo de software configurado com as medidas do tabuleiro e deslocá-lo em frente à câmara de forma a obter imagens de diferentes perspetivas. A Figura 2.4 ilustra a interface do software de calibração utilizado. Como se pode reparar na imagem, o módulo reconhece os cantos do tabuleiro e sabendo a distância entre eles é possível determinar os parâmetros de calibração.

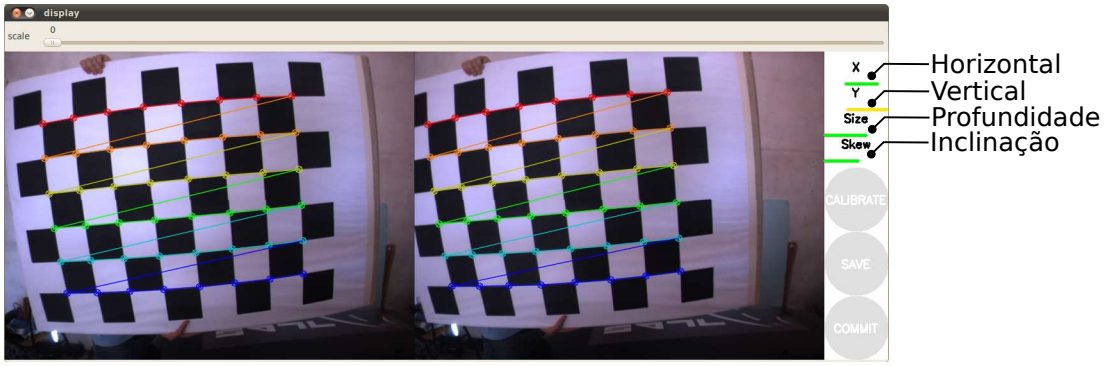


Figura 2.4: Interface do software de calibração.

Para obter uma calibração de boa qualidade da câmara não basta mover o tabuleiro aleatoriamente, deve garantir-se que o software retira exemplares por todo o campo de visão horizontal e vertical (X e Y respetivamente). É necessário também variar o ângulo de visão, ou seja, inclinar o tabuleiro nas diversas direções. Por fim, deve-se ainda garantir que o tabuleiro é visto em todo o campo de profundidade, ou seja, aproximar e afastar o máximo possível da câmara. Para facilitar esta tarefa, o módulo de software apresenta quatro barras de progresso que indicam o estado de aquisição de dados para calibração. Para os campos de visão horizontal, vertical e de profundidade as barras são representadas por X, Y e *size*, respetivamente. A barra *skew* representa os ângulos de rotação do tabuleiro. (Figura 2.4). Quando as barras de progresso estão no máximo, ou próximo, executa-se o algoritmo de calibração para processar as imagens e estimar todos os parâmetros de calibração.

Ao fim de alguns segundos é possível verificar os parâmetros de calibração resultantes bem como as imagens retificadas e alinhadas.

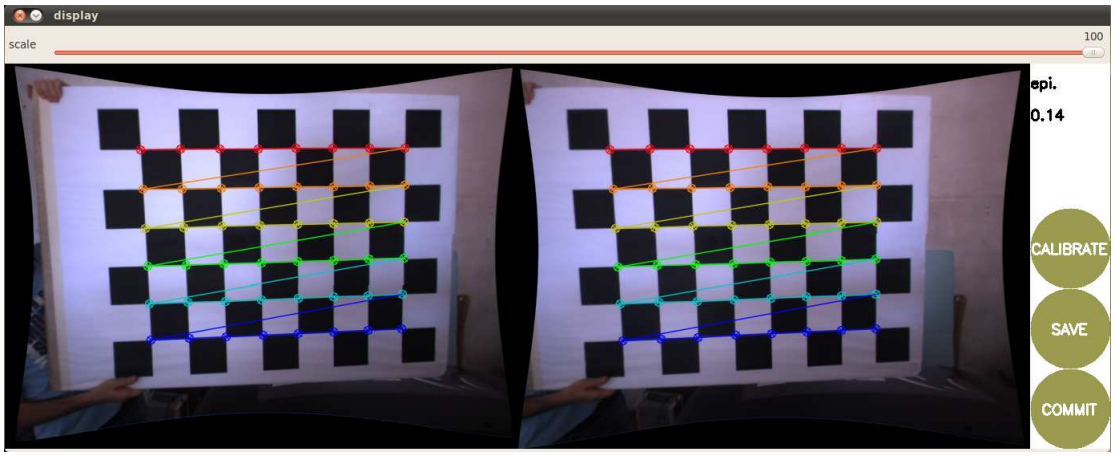


Figura 2.5: Exemplo de resultado da calibração estéreo.

A Figura 2.5 exemplifica um resultado de calibração de um par estéreo. Para além das matrizes intrínsecas e extrínsecas, o módulo calibra a geometria das imagens, de forma a que cada pixel da imagem da esquerda corresponda exatamente com a mesma coordenada em Y na imagem da direita. Finalmente, guardam-se os resultados em ficheiros de extensão *.yaml* para posterior utilização.

2.3.3 Calibração na Bumblebee xb3

Como mencionado na secção 2.1, o sensor estéreo utilizado possui três câmaras distintas, o que resulta em dois pares estéreo diferentes (Figura 2.2). Este facto constitui um pequeno problema quando é necessário calibrar a câmara através do módulo disponível em ROS, uma vez que o software está preparado e otimizado para câmaras estéreo convencionais (duas lentes), e não para câmaras mais complexas como esta. Uma forma de ultrapassar esse problema seria refazer o software de calibração para a *xb3*, processo que não foi realizado por ser demasiado moroso. Numa fase inicial, optou-se por calibrar um par estéreo de cada vez, como se fossem duas câmaras estéreo individuais. Contudo, a sobreposição de dados dos dois pares revelou alguma discrepância, de certa forma causada

pela utilização de imagens de calibração diferentes. Na tentativa de resolver o problema observado com este procedimento, experimentou-se a calibração em simultâneo, ao executar duas instâncias do software e desta forma garantir que as imagens para o cálculo dos coeficientes fossem semelhantes. A segunda abordagem de calibração, embora não tenha resolvido totalmente o problema, reduziu significativamente as diferenças obtidas relativamente ao cálculo de disparidades.

2.3.4 Verificação da calibração

Depois de realizadas as calibrações, é necessário testar a qualidade e o erro obtido no processo. Para o caso particular da XB3, é ainda necessário testar quais as diferenças entre os *baselines* e conferir a sua precisão. Esta verificação é necessária para estimar o *baseline* mais adequado para cada situação.

Para validar corretamente os resultados de calibração, é necessário recorrer a uma verificação direta dos resultados com medições, ou seja, por *ground truth*. Essa verificação consiste numa montagem experimental, onde se colocam alguns objetos de dimensões e distâncias conhecidas para poder comparar a informação dada pelo sensor com a informação real. Para tal, duas experiências foram realizadas: uma para a verificação de distâncias em profundidade e regularidade de superfícies; e outra para verificação de distâncias em largura e altura. A Figura 2.6 esquematiza a forma como foi montado o cenário para a avaliação dos dados da câmara em profundidade e regularidade de superfícies.

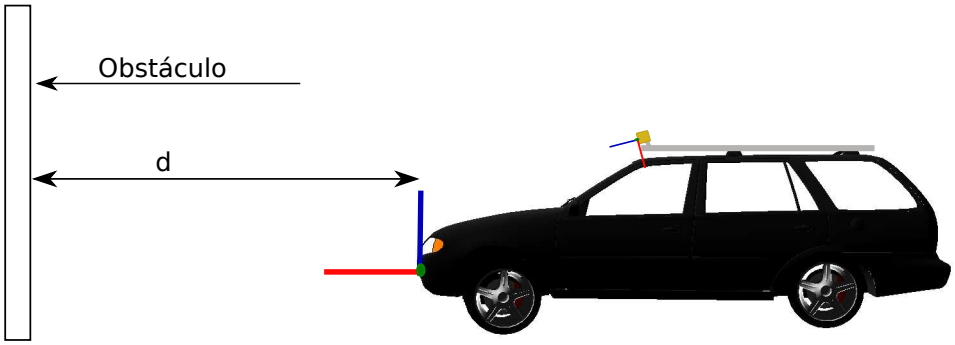


Figura 2.6: Montagem da verificação: Profundidade e superfície.

O primeiro ensaio consistiu essencialmente na avaliação da distância em profundidade de um determinado obstáculo da câmara e na qualidade das superfícies. Neste caso, a câmara foi direcionada paralelamente para uma parede plana, fazendo variar a distância d (Figura 2.6). Desta forma, é possível comparar a eficácia de medição dos *baselines*, saber qual o mais adequado e qual o limite onde a informação é credível. A experiência permitiu ainda avaliar a regularidade de superfícies, já que se tratava de uma parede plana. A Tabela 2.1 descreve sucintamente as observações verificadas ao longo da experiência.

Desta atividade experimental, verificou-se que o *wide baseline* permite resultados mais precisos em toda a gama de observações. Contudo, para distâncias inferiores a aproximadamente cinco metros, o *short baseline* permite obter uma nuvem de pontos com maior densidade, o que pode ser vantajoso para certas aplicações, por exemplo, situações em que se pretenda uma grande definição de pontos para caraterização de

		Baseline	
		short	**wide**
Distância d (m)	5	Boa densidade na nuvem de pontos. Precisão na distância. Obstáculo plano.	Pouca densidade na nuvem de pontos. Precisão na distância. Obstáculo plano.
	10	O obstáculo não fica representado paralelamente ao eixo do veículo como seria suposto.	O obstáculo permanece paralelo ao eixo do veículo. Precisão na distância. Obstáculo plano.
	15	Obstáculo deformado. Falta de paralelismo ao eixo do veículo.	Alguma distorção na representação. Planeza do obstáculo reduzida.
	20	Dificuldade em perceber o obstáculo. Excesso de deformação.	Obstáculo com bastante distorção mas percetível.

Tabela 2.1: Resultados da verificação: Profundidade e superfície.

objetos.

Avaliada a eficácia da medição em profundidade, realizou-se um método semelhante para caraterizar e avaliar a medição nas restantes dimensões, isto é, largura e altura. No segundo ensaio colocaram-se dois objetos semelhantes e de dimensões conhecidas, afastados de dois metros numa linha paralela ao eixo do veículo. A Figura 2.7 representa a montagem efetuada.

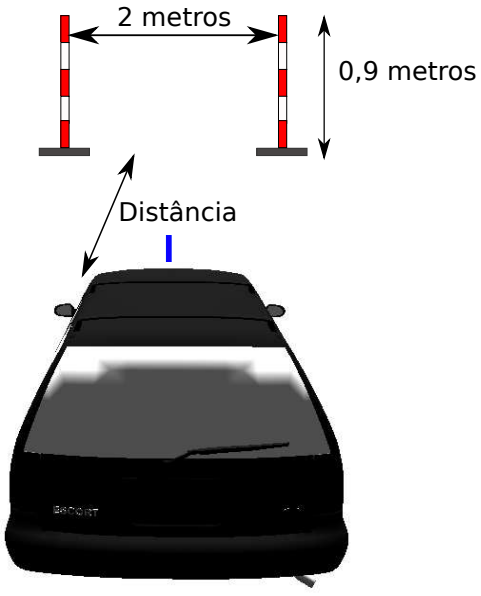


Figura 2.7: Montagem da verificação: Altura, Largura.

O objetivo deste ensaio consistiu na avaliação da medição da distância horizontal entre os dois objetos colocados, bem como a sua altura. De forma equivalente ao ensaio anterior, realizaram-se quatro testes a distâncias variáveis. Na Tabela 2.2, estão listados de forma resumida os resultados dos ensaios.

Os resultados qualitativos desta experiência foram semelhantes aos do ensaio anterior. O *wide baseline* revelou resultados mais coerentes, sobretudo a distâncias superiores a

		Baseline	
		short	**wide**
Distância (m)	5	Distâncias e altura próximas, com um erro aproximado de cerca de $\pm 0,10(m)$.	Distâncias e altura próximas, com um erro aproximado de cerca de $\pm 0,05(m)$.
	10	Os obstáculos apresentam alguma falta de paralelismo. Erros da ordem dos $\pm 0,15(m)$	Distâncias e altura próximas, com um erro aproximado de cerca de $\pm 0,08(m)$.
	15	Excesso de *speckling*. Não é possível medir a altura com precisão. Erros de cerca de $\pm 0,30(m)$	Ligeiro efeito de *speckling*, erros da ordem dos $\pm 0,12(m)$
	20	Excesso de *speckling* e irregularidades. Os obstáculos acabam por se fundir com o chão.	Excesso de *speckling* para objetos de pequenas dimensões. Erro aproximado de $\pm 0,20(m)$

Tabela 2.2: Resultados da verificação: Altura, Largura

dez metros. O *short baseline* mostrou melhor definição nas superfícies para distâncias inferiores a 5 metros, uma mais valia quando se pretende realizar o reconhecimento de superfícies para deteção de obstáculos. Os ensaios realizados, permitiram verificar a qualidade do processamento estéreo obtido, bem como a dificuldade que existe em efetuar medições, sobretudo para distâncias elevadas em que o erro no cálculo de disparidades é maior.

Medições em altura e *speckling*

O fenómeno de *speckling*, mencionado na Tabela 2.2, resulta da aplicação do algoritmo estéreo sobre as imagens. O fenómeno descreve-se pela representação inclinada de obstáculos em vez da representação vertical que seria de esperar. A Figura 2.8 traduz a forma como o fenómeno ocorre e a sua variação consoante a distância. Ao analisar a imagem, observa-se um efeito de *speckling* mais elevado no obstáculo assinalado a vermelho do que a verde, por aquele se encontrar mais distante. O efeito surge em zonas onde, no mesmo agrupamento de píxels, existem zonas próximas e distantes, ou seja, nas arestas das superfícies e nos objetos finos. Nestes últimos (como um poste, um pino de marcação de estrada ou qualquer objeto com a mesma estrutura), o efeito de *speckling* apresenta tendência para se agravar, e este tipo de obstáculos deixa de ser representado na vertical. Os motivos pelos quais este efeito ocorre encontram-se detalhados adiante, na subsecção 2.4.3 sobre cálculo de correspondência estéreo.

A Figura 2.8 ilustra o fenómeno de *speckling* em pinos de marcação de estrada para distâncias elevadas.

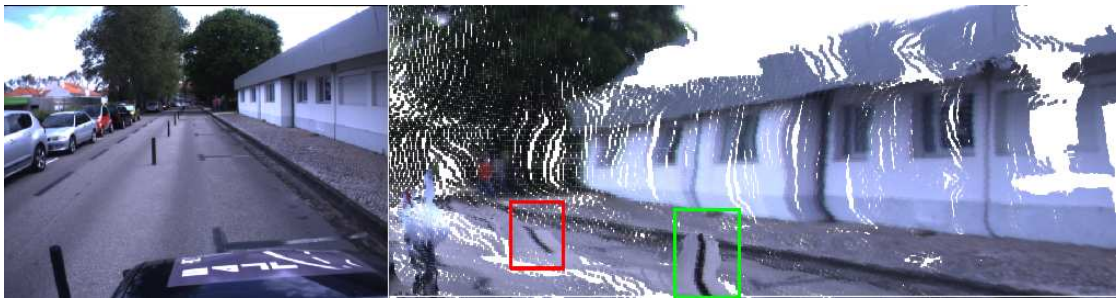


Figura 2.8: *Speckling* em pinos de marcação de estrada.

Figura 2.9: *Speckling* em arestas de obstáculos.

O efeito manifesta-se também em contornos de obstáculos como exemplificado na Figura 2.9. Neste caso, o efeito pode ser grave uma vez que o conjunto de pontos na nuvem que resultam do *speckling* unem os diversos obstáculos, o que dificulta o processo de segmentação.

Os ensaios realizados acerca dos *baselines* da câmara aplicada especificamente no ATLASCAR permitiram observar que:

- O *short baseline* apresenta resultados credíveis apenas para distâncias inferiores a cerca de cinco metros a partir da zona frontal do veículo (para choques);
- O *wide baseline* fornece uma maior precisão em todo o alcance da nuvem de pontos, contudo, para distâncias mais próximas, a densidade de pontos é menor;
- A partir de cerca dos vinte metros, a informação torna-se demasiado distorcida para ambos os *baselines* e deixa de ser credível.

Para finalizar esta fase de classificação dos *baselines*, pode-se afirmar que para o uso particular no ATLASCAR pretende-se a medição e classificação de obstáculos a distâncias superiores a cerca de três metros, por isso é de maior relevância a utilização do *wide baseline*. A utilização ideal da câmara nestas circunstâncias, seria o *short baseline* até cerca de cinco metros e a do *wide baseline* dos cinco aos vinte metros. Contudo, essa utilização implica um custo computacional bastante mais elevado, uma vez que é necessário recorrer ao cálculo estéreo de duas nuvens de pontos em vez de uma, e à partida descartar a maior parte da informação. Em termos comparativos, utilizar os dois *baselines* implica dispender mais do dobro do tempo que seria necessário para processar

os dados de um *baseline*, já que é necessário calcular o dobro de disparidades e processar as nuvens de pontos para fundir a informação. Essa utilização condiciona à partida a eficácia do processamento a nível computacional.

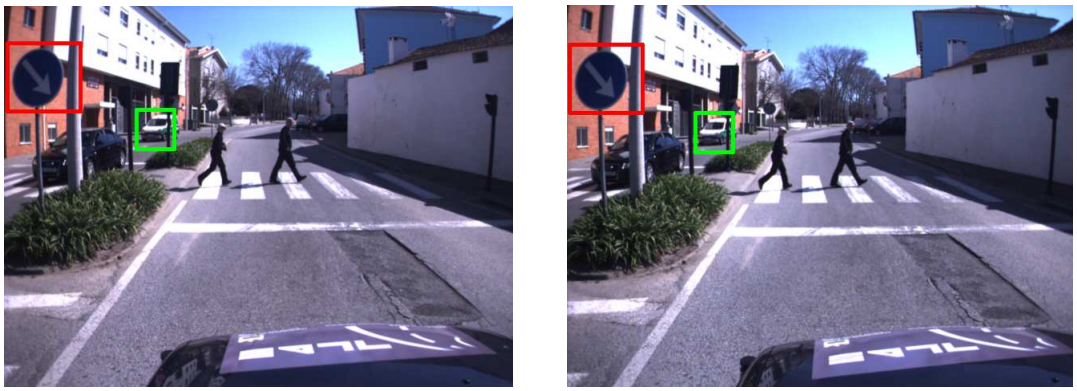
2.4 Cálculo estéreo

2.4.1 Princípio de funcionamento

Nas secções anteriores, abordaram-se as questões de aquisição de dados do sistema sensorial em causa e a respetiva calibração; nesta secção descreve-se a forma como a informação de imagens do sensor pode ser convertida para dados tridimensionais para posterior análise.

A visão estéreo consiste na interpretação de dados de duas ou mais imagens obtidas do mesmo cenário de pontos de vista diferentes. O princípio de funcionamento é semelhante ao que acontece com a visão humana, na qual cada um dos olhos fornece uma perspetiva diferente do cenário visualizado, o que dá uma perceção tridimensional do ambiente que nos rodeia.

Sabendo a diferença de posição entre duas câmaras (*baseline*) e as suas propriedades óticas, é possível estimar a localização de objetos em profundidade num determinado cenário. O princípio consiste em aplicar a diferença de coordenadas das duas imagens, sabendo que quanto mais próximo um determinado obstáculo estiver da câmara, maior será a diferença em píxels nas duas imagens obtidas, ou seja, maior será a sua disparidade.



(a) Esquerda. (b) Direita.

Figura 2.10: Par de imagens estéreo.

Atente-se, por exemplo, aos blocos de píxels assinalados a diferentes cores na figura 2.10. Como é fácil compreender, a diferença de coordenadas de píxels nas duas imagens será maior no objeto assinalado a vermelho (objeto mais próximo), que no objeto assinalado a verde (objeto mais longínquo).

2.4.2 Correspondência estéreo

A corresponência estéreo consiste no método de determinação de semelhanças entre as duas imagens para o cálculo da disparidade. Para uma melhor compreensão da correspondência estéreo, tome-se a titulo de exemplo a figura 2.10, obtida pela câmara estéreo

a bordo do ATLASCAR. Para determinar a correspondência de píxels e o respetivo mapa de disparidade, divide-se a imagem da esquerda em zonas. Atente-se à zona assinalada a vermelho na ilustração 2.11 (figura 2.11(a)). Para cada zona nesta imagem, procura-se a zona de maior correspondência na imagem da direita, numa banda horizontal à mesma altura do bloco de procura (figura 2.11(b)). Este método designa-se por *Block Match* e é baseado na geometria epipolar, ou seja, exige que as imagens estejam perfeitamente alinhadas na horizontal, o que implica que o par estéreo seja previamente calibrado conforme a descrição na secção 2.3. O método encontra-se implementado no módulo ROS, *stereo_image_proc*, cuja descrição detalhada do funcionamento encontra-se detalhado adiante.

(a) Esquerda. (b) Direita.

Figura 2.11: Correspondência estéreo.

A essência do problema no uso da visão estéreo está na relação inversa entre a qualidade de mapas de disparidade obtidos e o custo computacional envolvido no processo de cálculo. Torna-se necessário jogar com estes dois fatores de forma a encontrar a melhor relação entre qualidade e custos computacionais. Dos diversos algoritmos de cálculo de zonas de correspondência, existem situações comuns que frequentemente geram zonas de correspondência mal calculada [Szeliski and Zabih, 2000]. A seguir enumeram-se as principais:

- Superfícies refletoras de luminosidade;
- Regiões com falta de textura que causam ambiguidade;
- Descontinuidades de profundidade nas arestas dos objetos, especialmente quando os objetos são finos;
- Zonas oclusas, quando um objeto se encontra a ocludir parte de outro. Na verdade os algoritmos deveriam detetar zonas de oclusão quando as arestas dos obstáculos se intersetam mas normalmente não o fazem [Szeliski and Zabih, 2000].

2.4.3 Cálculo da disparidade

A partir dos princípios descritos anteriormente, é possível obter uma imagem com diferentes tonalidades em que cada tom representa uma distância diferente. Esta representação

constitui um mapa de disparidade, exemplificado na figura 2.12, onde claramente se reconhece que os tons mais claros representam objetos mais próximos da câmara e os tons mais escuros representam objetos mais distantes.

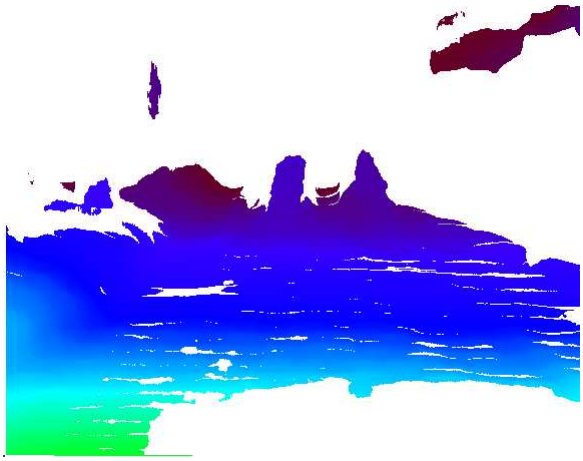


Figura 2.12: Mapa de disparidade.

Com particular interesse na aplicação de visão estéreo no ATLASCAR, pretende-se um algoritmo com elevada eficiência para processamento em tempo real. A forma mais eficaz de calcular estéreo é a utilização da geometria epipolar para reduzir ao máximo as zonas de procura de semelhanças entre pares de imagens [Bradski and Kaehler, 2008]. Esta geometria é de extrema importância porque torna possível a procura apenas numa dimensão em vez de duas, e para além de reduzir o custo computacional envolvido, tem a vantagem acrescida de reduzir as zonas que poderiam induzir o erro de disparidade [Bradski and Kaehler, 2008].

Em ambiente de programação ROS existe um nodo bastante otimizado para cálculo estéreo baseado em ferramentas do OpenCv, denominado por *stereo_image_proc*, cujas etapas do algoritmo enunciam-se pela seguinte ordem [Bradski and Kaehler, 2008]:

1. Remoção matemática da distorção causada pelas lentes, para obtenção de imagens distorcidas;

2. Ajuste de ângulos e distâncias entre lentes para obtenção de imagens alinhadas na horizontal e retificadas;

3. Cálculo da correspondência, para obtenção de um mapa de disparidade de diferenças de coordenadas horizontais;

4. Conversão do mapa de disparidades para um mapa de profundidades por triangulação.

O funcionamento dos algoritmos de distorção e retificação de imagem são semelhantes aos usados para outras câmaras que não as estéreo, portanto não são especificados aqui os detalhes acerca da sua utilização. Uma descrição mais detalhada pode ser encontrada em *Learning OpenCV* [Bradski and Kaehler, 2008]

Para o cumprimento do terceiro tópico mencionado na lista acima, é necessário recorrer os seguintes passos.

1. Pré filtragem das imagens para normalizar o brilho e enaltecer a textura das superfícies;

2. Cálculo da disparidade através do algoritmo da soma das diferenças absolutas;

3. Pós filtragem para reduzir as zonas de má correspondência.

A pré filtragem consiste na normalização da imagem, em que se faz percorrer uma janela de determina dimensão sobre a imagem entre 5x5 (min) a 21x21 (max), em que o pixel central I_c é substituído por:

$$I_c = min[max(I_c - I, -I_{cap}), I_{cap}], \tag{2.4}$$

em que I é o valor médio calculado na janela e I_{cap} é um limite numérico, normalmente com valor constante igual a trinta [Bradski and Kaehler, 2008]. A dimensão da janela a percorrer é especificada pelo utilizador, e o seu valor constitui um dos parâmetros de configuração do nodo ROS (*prefilter_size*). De seguida, é necessário aplicar o método de deteção de semelhanças entre as imagens. O método consiste na aplicação de uma janela (5x5 até 21x21) SAD (*Sum of absolute difference*) em imagens retificadas e monocromáticas. Esta varíável denomina-se por janela de correlação (*correlation_window_size*) e a sua dimensão é também um parâmetro definido pelo utilizador. Este método é interessante e largamente utilizado em aplicações que exigem um tempo de processamento reduzido, uma vez que apenas envolve cálculos de somas e subtrações [Bradski and Kaehler, 2008]. Para melhor compreender o método, considere-se uma zona de dimensão 3x3 (píxels) proveniente da imagem da esquerda da câmara estéreo, que corresponde a um modelo; e outra zona 3x5 (píxels) obtida da imagem da direita da câmara que representa a zona de procura na mesma linha epipolar. Os valores dos píxels estão representados por números inteiros de zero a nove para simplificação do exemplo.

Modelo
$$\begin{bmatrix} 2 & 5 & 5 \\ 4 & 0 & 7 \\ 7 & 5 & 9 \end{bmatrix}$$

Zona de Procura
$$\begin{bmatrix} 2 & 7 & 5 & 8 & 6 \\ 1 & 7 & 4 & 2 & 7 \\ 8 & 4 & 6 & 8 & 6 \end{bmatrix}$$

Existem apenas três localizações onde o modelo apresentado pode encaixar na zona de procura, na esquerda, no centro ou na direita. Para saber qual a melhor zona de correspondência, calcula-se a soma das diferenças absolutas. Por exemplo, a começar pela esquerda: $|2-2| = 0$, $|4-1| = 3$, $|7-8| = 1$ e os restantes resultados obtendo as diferenças, para as três possibilidades.

Esquerda
$$\begin{bmatrix} 0 & 2 & 0 \\ 3 & 7 & 3 \\ 1 & 1 & 3 \end{bmatrix}$$

Centro
$$\begin{bmatrix} 5 & 0 & 3 \\ 3 & 4 & 5 \\ 3 & 1 & 1 \end{bmatrix}$$

Direita
$$\begin{bmatrix} 3 & 3 & 1 \\ 0 & 2 & 0 \\ 1 & 3 & 4 \end{bmatrix}$$

Por fim, basta obter a soma de todas as diferenças, que resulta em: 20, 25 e 17, para a esquerda, centro e direita, respetivamente. Conclui-se que a zona da direita é a mais semelhante do modelo, porque a soma das diferenças absolutas é a menor. Ao aplicar este princípio a todas as zonas da imagem, é possível completar o mapa de disparidade.

Para finalizar o método, procede-se a uma pós filtragem de dados, que visa excluir do mapa regiões de falha no método de cálculo de correspondência. A primeira etapa de filtragem consiste na remoção das zonas onde a textura é menor que um determinado valor (normalmente 12) [Bradski and Kaehler, 2008]. Este valor consiste apenas num limite numérico aplicado diretamente ao resultado da soma das diferenças absoluta. O cálculo de correspondência por blocos é limitado na obtenção de bons resultados nos limites dos objetos, isto porque a janela de correspondência agrega píxels do objeto e píxels do fundo em simultâneo. Nesses limites, ao calcular a soma das diferenças absolutas, ocorre o efeito denominado por *speckling*, semelhante ao demonstrado na secção acerca de calibração. (figuras 2.8 e 2.9). Na tentativa de redução de maus cálculos de disparidade aplica-se um limite (*uniqueness_ratio*) em percentagem à correlação de píxels dado pela expressão:

$$uniqueness\_ratio \geq \frac{match\_val - min\_match}{min\_match}, \tag{2.5}$$

em que o valor de $match\_val$ é o valor da correspondência determinada pela aplicação da soma das diferenças absolutas e o valor de min_match é o valor mínimo da função de correspondência.

A etapa final consiste na aplicação de uma janela (5x5 até 21x21) píxels sobre o mapa de disparidade. Desde que os valores mínimo e máximo de disparidades dentro dessa janela respeitem um limite (normalmente igual a 4) [Bradski and Kaehler, 2008], a correspondência é permitida. No caso da correspondência não ser admitida, essa zona fica por preencher no mapa de disparidade, ou seja, é uma zona indeterminada.

2.4.4 Parâmetros

O módulo existente em ROS para cálculo estéreo permite a utilização de um servidor de parâmetros estéreo. A tabela 2.3 enumera os parâmetros configuráveis em estéreo, os valores padrão e estimados, e a forma em que eles podem influenciar a qualidade estéreo resultante [Bradski and Kaehler, 2008].

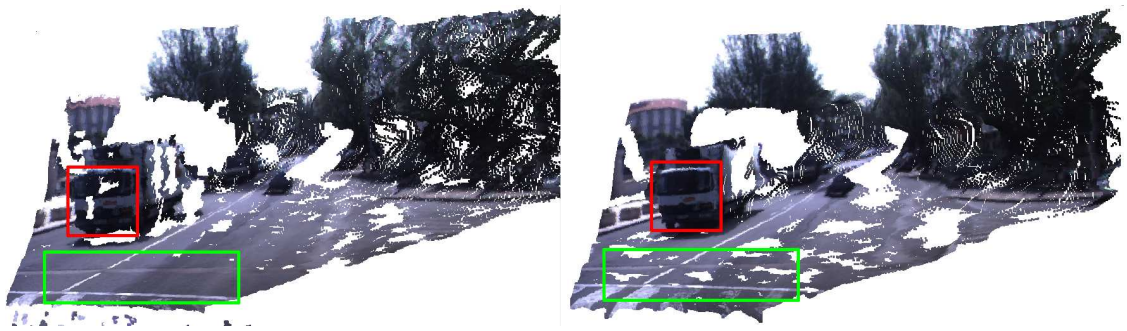
Não existem parâmetros que satisfaçam os requisitos da aplicação pretendida em todas as situações, nem existe uma forma matemática para os poder determinar de forma coerente [Bradski and Kaehler, 2008]. Os valores a utilizar dependem da aplicação pretendida, das condições de luminodidade e do alcance que se pretende obter [Bradski and Kaehler, 2008]. Contudo, o servidor de parâmetros do ROS permite ajustar os valores do processo de cálculo estéreo, em que o critério é o *feedback* visual do mapa de disparidades ou da nuvem de pontos obtida. Dos parâmetros ajustados, salienta-se a alteração do *prefilter_size* e do *prefilter_cap* para aumentar a normalização da imagem, ditada pela equação 2.4. Uma vez que o veículo circula em ambiente exterior, onde a fonte de iluminação é o sol, a aplicação de um filtro de maior dimensão ajuda no sentido de normalizar a iluminação em superfícies refletoras, apesar de aumentar ligeiramente os requisitos de processamento. Por outro lado, diminuiu-se ligeiramente o fator *disparity_range* para reduzir o campo de procura estéreo e desta forma diminuir a computação. O limite de textura exigido foi também aumentado significativamente (*texture_threshold*), o que revelou a eliminação na nuvem de pontos de zonas de disparidade mal calculadas. Por fim, aumentaram-se também os limites dos parâmetros *speckle_size* e *speckle_range*. O aumento destes limites conduziu à remoção de alguns pontos que influenciam o *speckling* explicado na subsecção 2.3.4 que é um fator importante, já que

	Parâmetro	Descrição	Valor padrão	Valor utilizado
Pré filtragem	*prefilter_size*	Reduz diferenças na iluminação por normalização da imagem.	9×9	21×21
	prefilter_cap	Constante de normalização de imagem.	30	63
	correlation_window_size	Tamanho da janela de correspondência.	15×15	15×15
	min_disparity	Disparidade mínima.	0	0
	disparity_range	Campo de procura de correspondência estéreo.	64	32
Pós filtragem	*uniqueness_ratio*	Limite em percentagem. A correlação só é aceite se o valor da disparidade calculada for reduzido.	10	10
	texture_threshold	Valor mínimo da textura para calcular a disparidade.	10	100
	speckle_size	Área máxima para remoção de *speckling*.	100	400
	speckle_range	Campo máximo de disparidade aceite entre píxels ligados.	4	15

Tabela 2.3: Parâmetros do algoritmo *Block Match*

o principal objetivo é a segmentação da nuvem de pontos, e neste caso, quanto menos pontos houver nas fronteiras entre segmentos maior será a eficácia da segmentação.

A figura 2.13 ilustra um exemplo visual de como a alteração dos parâmetros estéreo da tabela 2.3 influenciam a nuvem de pontos obtida.



(a) Valor padrão. (b) Valor utilizado.

Figura 2.13: Exemplo visual do efeito da alteração de parâmetros.

A alteração dos parâmetros permite obter uma qualidade superior nas superfícies frontais representadas, como no pormenor assinalado a vermelho na figura 2.13, que resulta da alteração dos valores dos parâmetros de pré filtragem (*prefilter_size* e *prefilter_cap*). Por outro lado, as alterações revelaram um pouco de perda de densidade na nuvem de pontos em zonas de baixa textura, como por exemplo, na estrada (zona assinalada a verde nas figuras 2.13(a) e 2.13(b)), derivado à alteração dos parâmetros relativos aos limites de textura (*texture_threshold*). Desta forma, verifica-se que não existe um conjunto de

parâmetros capaz de satisfazer todas as aplicações com estéreo, e mesmo dentro de uma aplicação, os valores utilizados podem não ser sempre os mais adequados. Contudo, neste trabalho em particular, os pontos relativos à estrada podem ser aproximados a um plano como explicado adiante na secção 3.2, e é de especial interesse que os obstáculos sejam representados com a melhor qualidade possível. Deste modo, optou-se por utilizar os parâmetros modificados em vez dos valores padrão.

2.4.5 Reconstrução tridimensional

Com os dados de disparidade, de cor e das propriedades óticas das câmaras, torna-se possível medir distâncias em profundidade de pontos e proceder a uma reconstrução tridimensional do cenário envolvente. Para obter a reconstrução tridimensional do cenário obtido por um sensor estéreo, é necessário garantir uma geometria semelhante à representada na figura 2.14 de forma a aplicar o algoritmo de *Bouguet* [Bradski and Kaehler, 2008], que consiste na reprojeção de pontos para o espaço tridimensional.

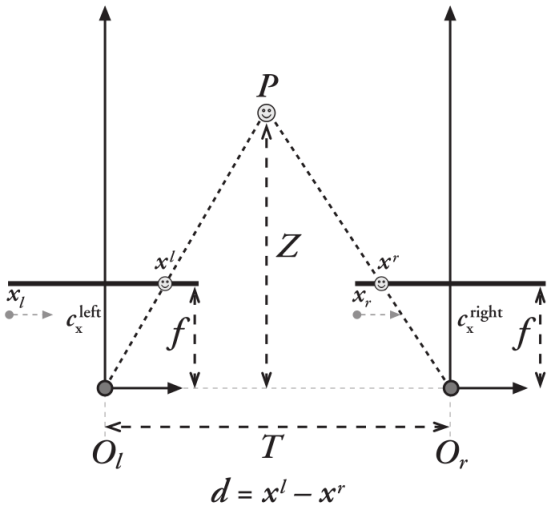


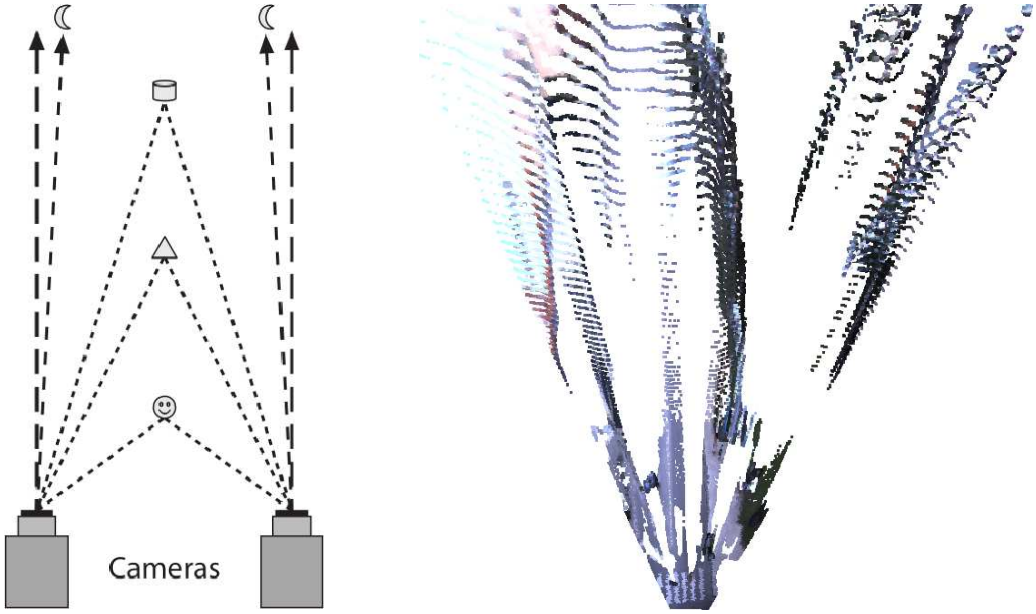
Figura 2.14: Geometria do sensor estéreo [Bradski and Kaehler, 2008].

A figura 2.14 representa o modelo *pinhole*, semelhante à representação 2.3, generalizado para câmaras estéreo. Para compreender a triangulação envolvida no cálculo da reprojeção, considere-se que é possível encontrar um objeto qualquer P no mundo físico nas duas câmaras do sensor estéreo em que as suas coordenadas no eixo x correspondem a x_l e x_r para a imagem da esquerda e da direita respetivamente (figura 2.14). Desta forma, a disparidade é dada por $d = x_l - x_r$, assumindo a geometria epipolar das imagens, e pode-se efetuar a triangulação das coordenadas para estimar a profundidade Z, segundo a equação 2.6 [Bradski and Kaehler, 2008].

$$\frac{T - (x^l - x^r)}{Z - f} = \frac{T}{Z} \Rightarrow Z = \frac{f \times T}{x^l - x^r} = \frac{f \times T}{d}. \tag{2.6}$$

Uma vez que a profundidade Z é inversamente proporcional à disparidade d, cria-se uma relação não linear entre estes dois termos. Quando a disparidade é próxima de zero, pequenas diferenças na disparidade correspondem a grandes diferenças na profundidade.

Por outro lado, quando a disparidade é elevada, pequenas diferenças de disparidade não mudam muito a profundidade. A principal consequência deste facto é que os sistemas de visão estéreo apenas apresentam resolução de profundidade para objetos relativamente perto do sensor. A figura 2.15(b) representa uma nuvem de pontos vista de topo. Atente-se que quanto maior for a profundidade, maior o efeito provocado pela relação não linear entre os termos.



(a) Triângulação de coordenadas.

(b) Efeito na nuvem de pontos.

Figura 2.15: Relação entre disparidade e profunidade [Bradski and Kaehler, 2008].

Ao conhecer a disparidade d e a profundidade Z calculadas por triangulação, é possível reprojetar o mapa de disparidade para a obtenção das coordenadas X e Y dos pontos no referencial do mundo. Desta forma, para um ponto qualquer representado pelas coordenadas x e y no plano, a reprojeção desse ponto no espaço é dado pela equação 2.7 [Bradski and Kaehler, 2008].

$$Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}, \tag{2.7}$$

em que a matriz Q é dada por:

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{-1}{T_x} & \frac{(c_x - c_x^{'})}{T_x} \end{bmatrix}. \tag{2.8}$$

Os parâmetros definidos pela matriz 2.8 são relativos à câmara da esquerda à exceção de $c_x^{'}$, que representa a coordenada em x na imagem da direita. Desta forma, de acordo

com a figura 2.14, Tx representa a distância entre as câmaras, c_x e c_y representam as coordenadas do ponto principal, e finalmente, f representa a distância focal. Desta forma é possível estimar as coordendas dos pontos tridimensionais dados por $(X/W, Y/W, Z/W)$.

Para finalizar, a figura 2.16 representa um cenário tridimensional obtido a partir do mapa de disparidade da figura 2.12, das propriedades da câmara obtidas na calibração (secção 2.3) e de uma das imagens a cores da câmara (figura 2.10).

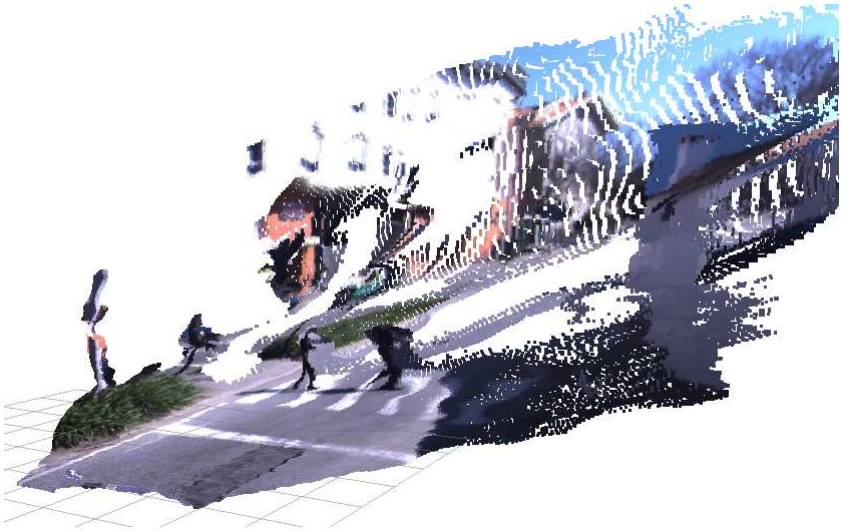


Figura 2.16: Exemplo de nuvem de pontos (Reconstrução tridimensional).

Capítulo 3

Processamento de nuvens de pontos

Este capítulo procura enquadrar e descrever os passos necessários para a obtenção e segmentação de nuvens de pontos. A programação em ambiente ROS (*Robot Operating System*) permite a inclusão inclusão de outras ferramentas *open source* ao projeto, com bastante facilidade, com particular interesse neste trabalho, o OpenCv para processamento de imagens, e a biblioteca de funções escrita em *C++* PCL(*Point Cloud Library*). Esta última é largamente utilizada neste trabalho, já que contém uma vasta gama de ferramentas para tratamento de dados tridimensionais.

3.1 Filtros aplicados e transformações

Retomando o exemplo da nuvem de pontos apresentada na Figura 2.16, obtida numa viagem do ATLASCAR, é fácil notar que se tratam de dados relativamente complexos no que respeita à informação que fornecem. A primeira observação a fazer está na quantidade de pontos na nuvem. Um *frame* da nuvem de pontos provenientes diretamente da câmara a uma resolução de 640x480 píxels resulta numa nuvem com 307200 pontos. Esta quantidade de informação é elevada para processamento em tempo real, o que implica a redução de dados como primeira etapa do processamento. Por outro lado, apesar dos ajustes efetuados aos parâmetros estéreo (secção 2.4.4), muitas vezes as nuvens de pontos obtidas apresentam zonas com ruído, oriundas de cálculo de más correspondências. Portanto, a etapa de filtragem de informação é importante para a obtenção de dados coerentes.

3.1.1 Transformações geométricas

Antes de proceder aos diversos mecanismos de tratamento de dados, é necessário estabelecer os sistemas de coordenadas de referência. No interesse deste trabalho, pretende-se localizar obstáculos relativamente ao plano do chão, independentemente da localização da câmara e da sua orientação no veículo. Por este motivo, surge a necessidade da utilização de uma transformação geométrica do referencial de coordenadas do sensor para o referencial de coordenadas situado na parte frontal do veículo, (*/atc/vehicle/center_bumper* na Figura 3.1).

A equação 3.1 permite transformar todos os pontos do referencial original para o

referencial de destino,

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{12} & t_1 \\ r_{21} & r_{22} & r_{22} & t_2 \\ r_{31} & r_{32} & r_{32} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{3.1}$$

em que as coordenadas x,y e z representam um ponto no referencial da câmara e as coordenadas x',y' e z' representam as coordenadas desse mesmo ponto no referencial do veículo. A Figura 3.1 representa todos os referenciais dos sensores do ATLASCAR e a sua nomenclatura.

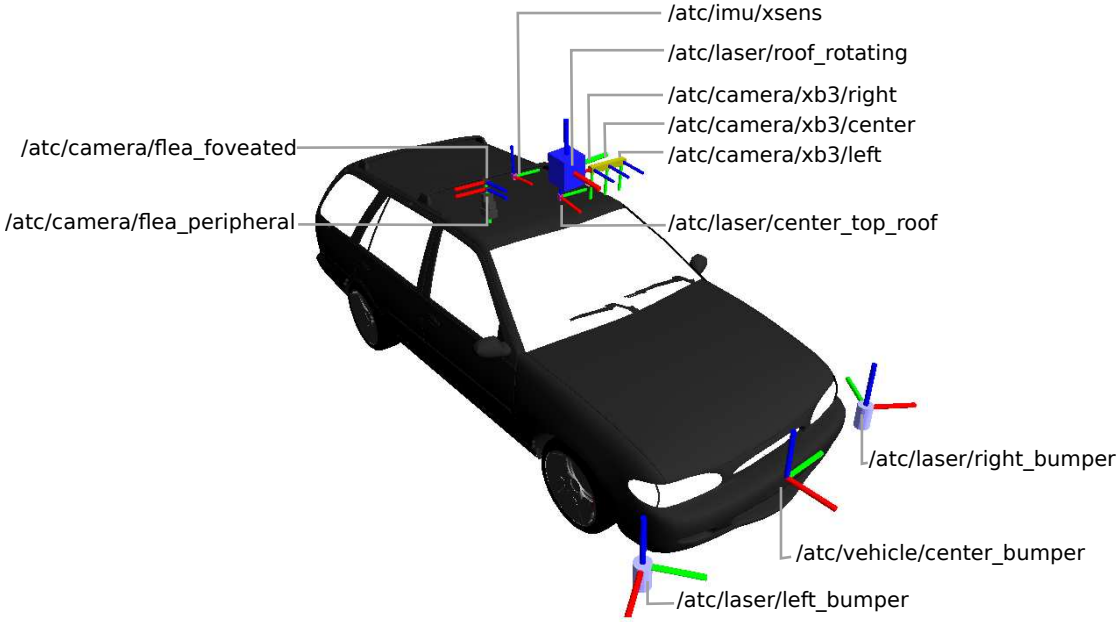


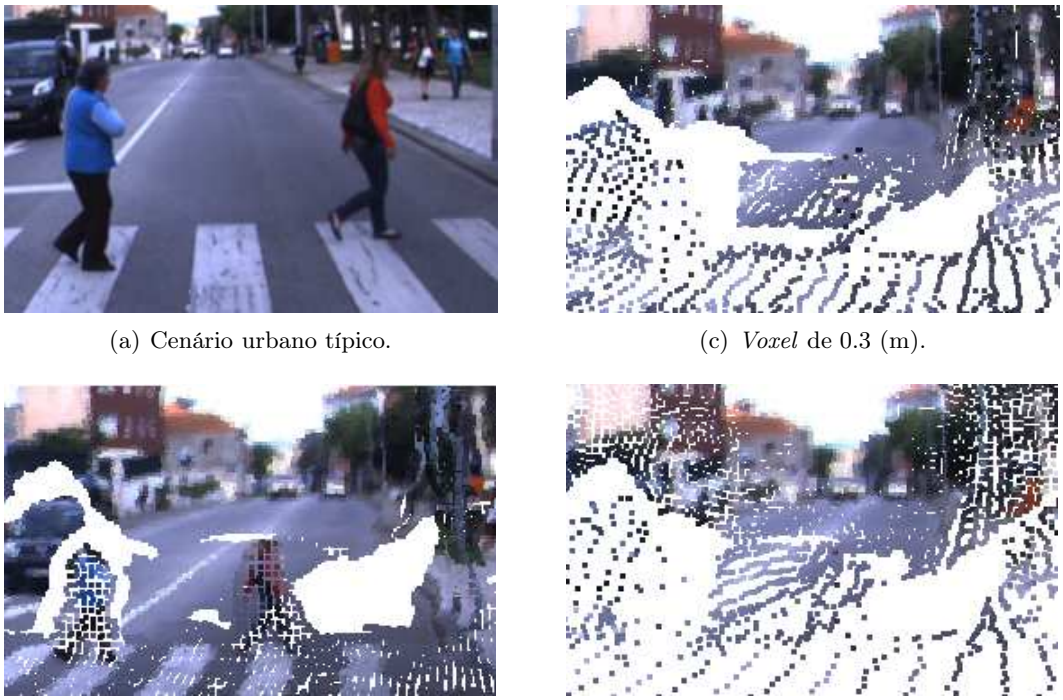
Figura 3.1: Sistemas de coordenadas dos sensores.

A programação de um nodo capaz de efetuar esta operação é essencial para os passos seguintes, já que facilita a medição de coordenadas. Uma vantagem na utilização desta transformação é que, se por algum motivo for necessário deslocar a câmara para outra posição, basta atualizar as transformações dos referenciais da câmara para o referencial situado no centro do para-choques frontal (Figura 3.1), eliminando a necessidade de eventuais alterações nos nodos seguintes de processamento. Outra vantagem na utilização de um referencial comum para representação de dados dos diversos sensores é a possibilidade de comparação e análise entre a informação de cada sensor.

3.1.2 *Voxel Grid*

A primeira etapa de filtragem inclui a utilização de uma *voxel grid*, que consiste na aplicação de volumes elementares (*voxels*) tridimensionais sobre todo o espaço ocupado pela nuvem de pontos, em que para cada uma todos os pontos são aproximados pelo centroide do volume elementar. Esta aproximação permite que se faça uma redução da densidade dos dados, preservando dentro de certos limites a estrutura e suavidade das

superfícies. Como é de prever, quanto maior for a dimensão dos *voxels* do filtro a aplicar na nuvem de pontos, menor será a quantidade de pontos depois da aplicação do filtro. Contudo, convém lembrar que se deve manter uma quantidade de informação suficiente que permita uma segmentação coerente para a distinção dos diversos obstáculos. A Figura 3.2 ilustra alguns resultados da aplicação de filtros desta natureza em nuvens de pontos retiradas da câmara estéreo durante uma viagem do ATLASCAR.



(a) Cenário urbano típico.

(c) *Voxel* de 0.3 (m).

(b) *Voxel* de 0.1 (m).

(d) *Voxel* de 0.5 (m).

Figura 3.2: Exemplo de aplicação de vários tamanhos de *voxel grid*.

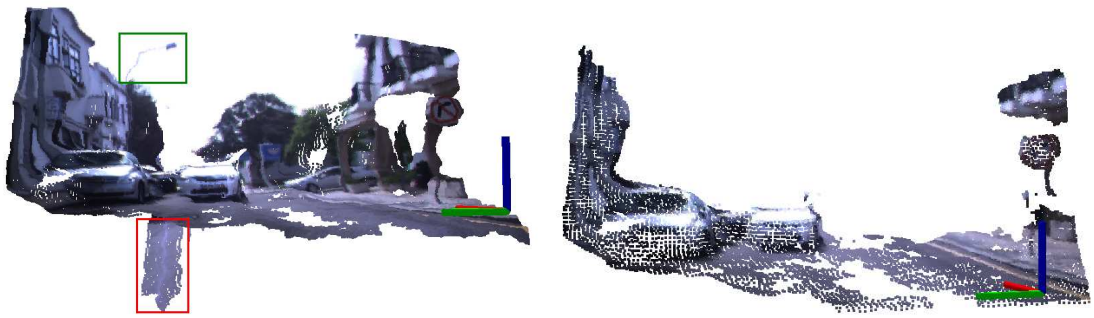
A Figura 3.2 permite compreender um pouco melhor o efeito da aplicação deste método. É possível reparar que se gera um compromisso entre a quantidade e qualidade de informação. Se o filtro aplicado constituir *voxels* de proporções elevadas, por exemplo de 0.5 metros, perde-se qualidade de dados (repare-se na quantidade de pontos que representa uma pessoa na Figura 3.2(d)), uma vez que conduz a um número elevado de aproximações. Por outro lado, se a dimensão do filtro for muito reduzida, o tempo de processamento pode não justificar a sua aplicação. A utilização deste filtro é bastante eficaz e permite reduzir drasticamente a quantidade de pontos. Para uma estimativa, um filtro com *voxels* de dez centímetros aplicado à nuvem de pontos (307200 pontos), permite reduzir para cerca de 50000 pontos (Figura 3.2(b)). A principal desvantagem da aplicação deste tipo de filtragem, para além da perda na qualidade de definição das superfícies, é que o número de pontos final obtido não permanece constante ao longo do tempo, dependendo do número de aproximações que a operação de filtragem possa efetuar.

Desta forma, o tamanho da *voxel* que forma o filtro deve ser influenciada pela relação que se pretende obter entre a qualidade da informação e o processamento.

3.1.3 Filtros de distâncias

Na subsecção 2.3.4, relativa a verificação de calibrações, concluiu-se que a qualidade de informação a partir de, aproximadamente, vinte metros apresenta tendência para se distorcer e deixa de ser credível. Tal facto sugere, de imediato, a construção de um filtro de remoção de todos os pontos que se encontrem para além dessa distância. Atente-se ao pormenor assinalado a verde na Figura 3.3(a), que representa o conjunto de pontos de um poste localizado a uma distância superior a vinte metros do veículo. Este tipo de informação deve ser descartado, quer pela falta de interesse no processamento de informação em alturas elevadas, quer pela incerteza que se obtém no cálculo de disparidade. Outro aspeto que convém referir é no caso de existirem, claramente, pontos que resultaram de disparidades mal calculadas. Tal facto ocorre com bastante frequência nos ambientes em que existem superfícies refletoras, (normalmente veículos de pintura metalizada), que refletem a luz solar e originam zonas, normalmente, abaixo do solo (assinalada a vermelho na Figura 3.3(a)). Este tipo de efeito pode ser eliminado da mesma forma, ao recorrer a filtros de distâncias. Por isso, construiu-se um filtro baseado num volume, em que foram descartados todos os pontos que não lhe pertencessem.

A Figura 3.3 representa um exemplo de aplicação após a filtragem de uma nuvem de pontos com *voxels* de tamanho igual a 0.1 (m) e com o filtro de distâncias.



(a) Nuvem de pontos em bruto (*wide baseline*). (b) Nuvem de pontos filtrada (*wide baseline*).

Figura 3.3: Exemplo de filtragem com aplicação de *voxel grid* e filtro de distâncias.

Com o procedimento de filtragem aplicado é possível reduzir significativamente a quantidade de pontos das nuvens, facto que constitui um passo essencial para as etapas posteriores de segmentação. Com o filtro de distâncias aplicado, consegue-se reduzir dos cerca de 50 mil pontos obtidos com as *voxels* para cerca de 15 mil pontos. Em termos de implementação, as dimensões da *voxel grid* e do volume do filtro de distâncias podem ser alteradas em qualquer momento, recorrendo à parametrização disponibilizada pelo ROS.

3.2 Deteção do plano do chão

Depois de filtrar devidamente a nuvem de pontos, é necessário segmentá-la, segundo a continuação do trabalho. O princípio de segmentação tridimensional consiste na partição da nuvem de pontos em diferentes zonas conhecidas, para simplificação do problema, de forma semelhante ao que acontece em problemas de visão convencionais.

Para ilustrar o conceito de segmentação envolvido, retome-se a título de exemplo a Figura 3.2(a) que ilustra um cenário característico obtido quando o ATLASCAR está em movimento. Para proceder à devida segmentação da nuvem de pontos, é necessário recorrer à análise da informação que é comum aos diversos obstáculos. Neste caso, o conjunto de pontos que une a maioria dos obstáculos, correspondem aqueles que representam o chão. Desta forma, é necessário criar um procedimento capaz de efetuar a segmentação dos pontos do chão, para que deixe de existir contacto entre os pontos que representam os diversos objetos. Esta forma de segmentação tem a vantagem de permitir a redução da quantidade de dados, já que, na maioria dos casos, os pontos que representam o chão constituem uma grande percentagem de informação. Note-se que, caso seja possível segmentar o conjunto de pontos que representam o chão P_c numa nuvem de pontos inicial P, então é possível obter uma nuvem de pontos P_o com a informação que representa os diversos obstáculos pela subtração das duas, isto é, $P_o = P \cap \overline{P_c}$. Desta forma, P_o deve constituir um conjunto de pontos de diversos obstáculos, sem que haja contacto entre eles. Por este motivo, a primeira etapa de segmentação consiste na separação dos pontos do chão dos restantes.

A primeira abordagem para remoção dos pontos do chão da nuvem de pontos que poderia ser adotada consiste em descartar todos os pontos de coordenada z abaixo de um determinado valor (Figura 3.1). Este método simplista tem a vantagem de ser bastante rápido, contudo só funcionaria para os casos em que o veículo se encontre paralelo ao plano do chão, o que limitaria o processo à partida. Uma forma para contornar o problema, seria monitorizar o estado da suspensão do veículo, para estimar uma transformação geométrica que compensasse desvios de orientações do veículo relativamente ao plano do chão como em rotundas, lombas de quebra de velocidade, depressões no piso, ou qualquer outra situação semelhante. Visto que o sistema de monitorização da suspensão encontra-se ainda em fase de desenvolvimento, é necessário recorrer a outra metodologia para a deteção do plano do chão.

Outra forma de proceder à segmentação do plano do chão é através da aplicação do modelo RANSAC (*RANdom SAmple Consensus*). A aplicação do método consiste na consideração de um modelo teórico, neste caso um plano, e de uma amostra de dados que pode ser aproximada a esse modelo, neste caso, a nuvem de pontos [Center et al., 1980]. Este método é iterativo e proporciona resultados corretos com uma determinada probabilidade, influenciada diretamente pelo número de iterações [Rusu and Cousins, 2011].

O princípio do algoritmo consiste na análise aleatória de amostras de pontos da nuvem e pelo teste hipotético dos parâmetros do modelo [Rusu and Cousins, 2011]. O resultado do algoritmo é aceite quando a quantidade de pontos que representa o modelo é considerada suficientemente grande. Por ser uma forma rápida e robusta de estimar geometrias em nuvens de pontos, a biblioteca PCL possui uma classe escrita em C++ para determinação de alguns modelos matemáticos através do método RANSAC. O algoritmo existente é capaz de devolver os coeficientes (a,b,c e d) do do maior plano na nuvem de pontos, representado através da equação 3.2:

$$ax + by + cz + d = 0. \tag{3.2}$$

Considere-se a representação da Figura 3.6, em que o plano A representa o plano do chão e B que representa o plano do veículo.

A partir da equação 3.2, é possível determinar a transformação geométrica entre o

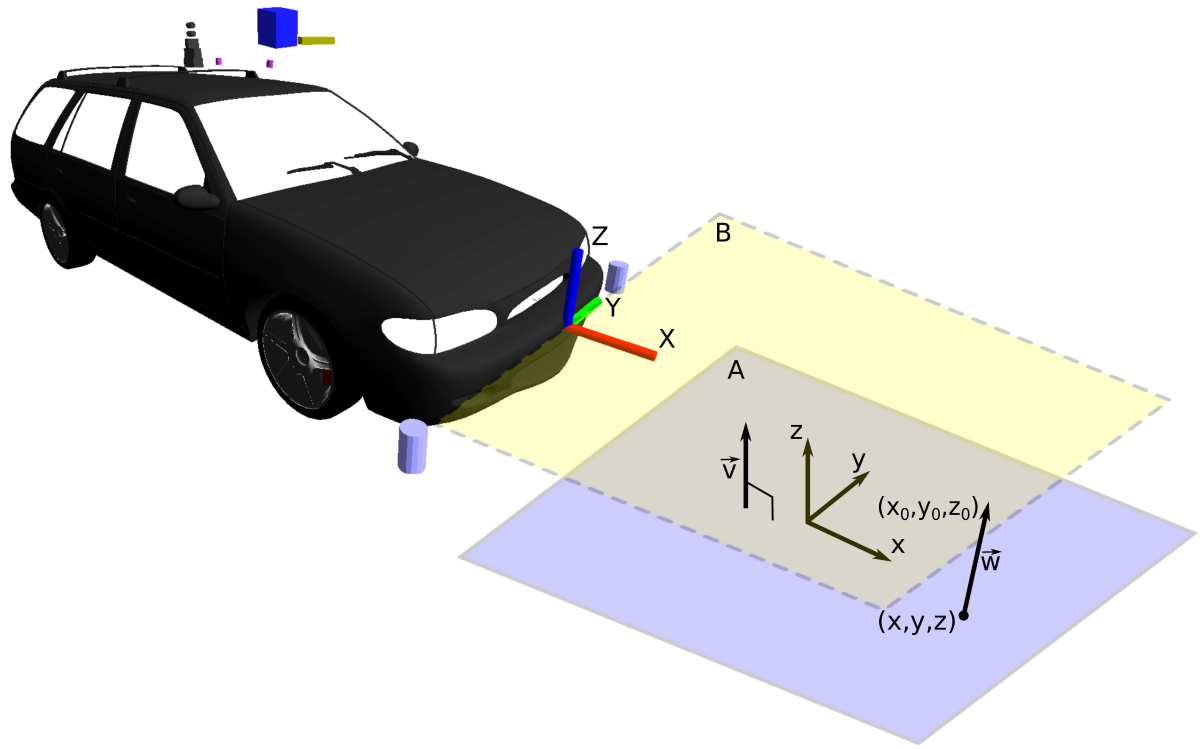


Figura 3.4: Geometria dos planos: veículo (B) e chão (A).

referencial XYZ do carro e o referencial xyz do chão (Figura 3.6). Para tal, considere-se o vetor $\overrightarrow{v}$ normal ao plano A dado pelos coeficiente a,b,c e d da equação 3.2:

$$\overrightarrow{v} = \begin{bmatrix} a \\ b \\ c \end{bmatrix},$$

e um vetor $\overrightarrow{w}$ dado por:

$$\overrightarrow{w} = -\begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix},$$

em que x_0,y_0,z_0 representam coordenadas de um ponto genérico representado no referencial XYZ. Então a distância D do ponto (x_0, y_0, z_0) ao plano é dada pela seguinte expressão [Gottwald and Gellert, 1989]:

$$D = \frac{|\overrightarrow{v}.\overrightarrow{w}|}{||\overrightarrow{v}||},$$

substituindo as respetivas variáveis a distância D pode ser determinada pela equação 3.3.

$$D = \frac{|a \times x_0 + b \times y_0 + c \times z_0 + d|}{\sqrt{a^2 + b^2 + c^2}}. \tag{3.3}$$

A partir da projeção da origem do referencial XYZ no plano A e de dois pontos conhecidos $(x_1, 0, 0)$ e $(0, y_1, 0)$, é possível determinar a localização do referencial xyz e a orientação dos ângulos de *Euler* (RPY). O ângulo R é estimado através do ângulo entre os dois vetores $\overrightarrow{Z}$ e $\overrightarrow{z}$, enquanto que o ângulo P é determinado pelo ângulo entre os vetores $\overrightarrow{X}$ e $\overrightarrow{x}$. A expressão 3.4 permite obter o ângulo entre dois vetores $\overrightarrow{v}$ e $\overrightarrow{w}$ [Gottwald and Gellert, 1989].

$$\alpha = \arccos\left(\frac{\overrightarrow{v}.\overrightarrow{w}}{||\overrightarrow{v}|| \times ||\overrightarrow{w}||}\right). \tag{3.4}$$

Repare-se que o terceiro ângulo de *Euler* (Y) não necessita de ser determinado pela equação 3.4, podendo assumir imediatamente o valor nulo, já que a orientação entre os vetores $\overrightarrow{Y}$ e $\overrightarrow{y}$ não é relevante para a determinação dos pontos da estrada, ou seja, os dois vetores são paralelos.

Com base no referencial xyz, é possível criar um volume com uma margem para extração dos pontos que fazem parte do chão, completando esta etapa de trabalho. A Figura 3.5 representa um exemplo da extração de todos os pontos que se encontrem a uma margem de dez centímetros do plano médio estimado pelo método RANSAC.

(a) Cenário com vários planos. (b) Nuvem de pontos filtrada e segmentada.

Figura 3.5: Exemplo da aplicação do algoritmo na presença de múltiplos planos.

Apesar da eficácia e robustez do método, há algumas ressalvas na sua aplicação que convém salientar. Em primeiro lugar, apesar do método ser significativamente rápido na execução computacional, não é possível estimar o tempo que o algoritmo exige, já que o método é iterativo. Outra desvantagem da aplicação, é que o método só consegue determinar uma solução. Isto quer dizer que, se na nuvem de pontos fornecida existirem dois planos candidatos, o método pode não devolver a solução ótima. Tal facto pode consistir um problema, no caso de existir um plano maior na nuvem de pontos que não o do chão. Considere-se o exemplo da Figura 3.5(a), que representa o deslocamento de um camião de traseira e lateral planas a uma distância reduzida do veículo. Neste caso, é provável que o algoritmo devolva os coeficientes do plano que se aproximam à parte traseira ou lateral do camião, em vez de devolver os coeficientes do plano médio do chão. Por isso, antes de validar o resultado da aplicação do método RANSAC, são avaliadas as coordenadas e ângulos do referencial xyz obtido. Se os valores não respeitarem determinadas margens, o resultado é rejeitado e é considerado o último referencial determinado. A partir do referencial estimado é possível determinar um volume de extração de pontos

do chão, com uma margem configurável relativamente ao plano médio determinado pelo método RANSAC. Desta forma, é possível obter uma aplicação capaz de extrair de forma significativamente robusta o conjunto de pontos que pertencem ao chão, o que constitui uma etapa fundamental na segmentação de nuvens de pontos.

3.3 Segmentação de obstáculos

Após a aplicação da metodologia de segmentação da nuvem de pontos através da análise de um plano (secção 3.2), é possível obter duas nuvens de pontos distintas: uma com todos os pontos que pertencem ao chão; e outra com os restantes que caraterizam o ambiente circundante. A etapa seguinte consiste na análise dos dados obtidos anteriormente, para dar continuidade ao processo de segmentação. O objetivo nesta etapa é classificar o conjunto de pontos que caraterizam o ambiente em duas partes distintas: uma que contém todos os pontos dentro da zona navegável, e outra que contém os pontos que estão fora dessa zona. Esta segunda segmentação tem especial importância na medida em que permite uma primeira classificação dos obstáculos. No final, o algoritmo deve ser capaz de distinguir, por exemplo, um peão a atravessar numa passadeira de um edifício que se encontre fora da área navegável.

3.3.1 Cálculo do perímetro navegável

Neste trabalho considera-se por zona navegável o contorno da zona vista pela câmara, num determinado instante, para a qual o veículo se pode deslocar. Considere-se uma nuvem de pontos P segmentada de acordo com a metodologia aplicada anteriormente, em que se obtém o conjunto de pontos do plano do chão P_c e o conjunto de pontos resultante da subtração, $P_o = P \cap \overline{P_c}$. As etapas seguintes consistem numa abordagem adotada para determinar a nuvem de pontos dentro da zona navegável.

1. Projeção dos pontos de P_c no plano médio do chão (previamente obtidos pelo método RANSAC);
2. Cálculo do *convex hull* da nuvem projetada para estimar o perímetro navegável;
3. Criação de um volume de extração com base no *convex hull*.

A Figura 3.6 representa a aplicação da metodologia descrita. O conjunto de pontos representado a azul é determinado através da projeção de todos os pontos no plano estimado pelo método RANSAC (secção 3.2) que representa o plano do chão. Esta projeção é interessante, porque permite o cálculo do *convex hull* do conjunto de pontos representado na Figura 3.6(b) por pequenas esferas alaranjadas. Um *convex_ hull* de um conjunto de pontos P é definido como a menor geometria convexa que contém P. Então, se P é finito o *convex_ hull* pode ser definido pela expressão 3.5, tal que para todos os pontos $p_i \in P$ [Dobkin and Huhdanpa, 1996]:

$$Ap_i + b \leq \overrightarrow{\mathbf{0}}, \tag{3.5}$$

em que cada linha de A e cada elemento de b definem a equação de um segmento de reta do *convex_ hull*. O conjunto de pontos definido pelo *convex_ hull* constitui aquilo que pode ser designado neste trabalho como perímetro navegável. A determinação deste

(a) Imagem. (b) Nuvem de pontos filtrada e segmentada.

Figura 3.6: Exemplo da aplicação do algoritmo de classificação da zona navegável.

perímetro permite estimar um volume que divide a nuvem de pontos em duas partes distintas: uma com o conjunto de pontos dentro da zona navegável, e outra com os pontos fora da zona navegável (Figura 3.6(b) a vermelho e a verde, respetivamente). Esta abordagem traduz uma etapa de segmentação interessante porque permite, em determinadas situações, distinguir e classificar a nuvem de pontos.

3.3.2 Segmentação por distância euclidiana

Até esta etapa, a metodologia de segmentação consiste numa abordagem de divisão de toda a nuvem de pontos em três zonas conhecidas, nomeadamente a zona do chão e dos pontos dentro e fora da zona navegável. Os métodos descritos anteriormente constituem uma forma de segmentação para simplificação do problema. Contudo, até esta fase não é possível obter informação acerca do número de obstáculos existentes, nem da sua localização e geometria. Desta forma, é necessário recorrer a uma segmentação a outro nível, para determinar a quantidade de obstáculos que se encontram dentro da zona navegável e as propriedades de cada um. Surge então a necessidade de analisar as distâncias entre o conjunto de pontos no espaço euclidiano. Define-se por distância euclidiana d ao comprimento de um segmento de reta que une dois pontos, o que matematicamente pode ser escrito através da expressão 3.6, em que $p = (p_1, p_2, p_3, ..., p_n)$ e $q = (p_1, p_2, p_3, ..., p_n)$ [StattSoft, 2012].

$$d = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}, \tag{3.6}$$

Para realizar uma segmentação num ponto de vista euclidiano, é necessário recorrer à decomposição espacial do conjunto de dados para os poder analisar quanto à sua proximidade.

De uma forma geral, o sistema tem de detetar segmentos e distingui-los. Matematicamente, um segmento pode ser definido da seguinte forma: Se o conjunto de pontos de entrada for representado por P, então o subconjunto $O_i = \{p_i : p_i \in P\}$ é distinto de

$O_j = \{p_j : p_j \in P\}$ se:

$$min||p_i - p_j|| \geq d_{th}, \forall i, j; \tag{3.7}$$

em que d_{th} é um limite máximo imposto de distância euclidiana [Rusu, 2009]. A equação 3.7 traduz que, se a distância mínima entre um conjunto de pontos $p_i \in P$ e outro conjunto $p_j \in P$ é maior que um determinado valor, então os pontos que pertencem a p_i pertencem a um segmento O_i e os pontos do conjunto p_j pertencem a outro segmento O_j, distinto do anterior.

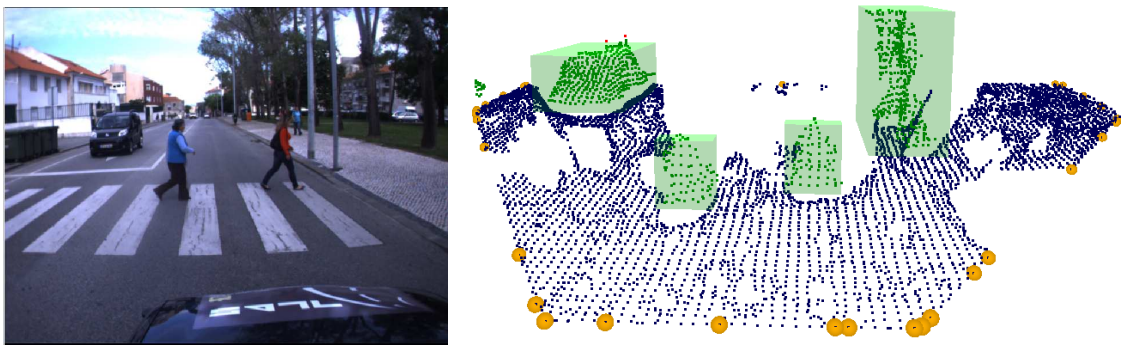
Implementação

A definição dos conceitos de distância euclidiana e de segmento é de especial importância para a compreensão do algoritmo de segmentação. Para proceder à implementação do algoritmo, é necessário recorrer à análise de vizinhança de pontos. Calcular e analisar as distâncias diretamente entre todos os pontos numa nuvem de pontos desorganizada, é uma tarefa extremamente exigente a nível computacional. Desta forma, surge a necessidade de representar a nuvem de pontos numa forma estruturada. A biblioteca de funções permite a utilização de uma representação em árvore (*kd-tree*), cujo método de procura é baseado na biblioteca de funções escritas em *C++ FLANN* (*Fast Library for Approximate Nearest Neighbor*), que possibilita a procura rápida de vizinhança de pontos [Rusu and Cousins, 2011]. As etapas do algoritmo são as seguintes, considerando P a nuvem de pontos a segmentar [Rusu, 2009]:

1. Criar uma representação estruturada em *kd-tree* da nuvem de pontos;
2. Criar uma lista vazia de segmentos C, e uma lista de pontos que têm de ser verificados Q;
3. Para cada ponto $p_i \in P$ efetuar os seguintes passos:
   (a) adicionar p_i à lista de pontos que têm de ser verificados, Q;
   (b) para cada ponto $p_i \in Q$ proceder da seguinte forma:
      i. procurar o conjunto de pontos vizinhos (p_i^k) de p_i dentro do raio $r < d_{th}$;
      ii. para cada vizinho (p_i^k) de p_i, verificar se o ponto já foi processado, e caso contrário adicioná-lo à lista Q;
   (c) Quando a lista de todos os pontos em Q estiver processada, adicionar Q à lista de segmentos C e apagar todos os pontos em Q;
4. O algoritmo termina quando todos os pontos $p_i \in P$ estão processados e fazem agora parte da lista de segmentos C.

A aplicação do algoritmo encontra-se implementada na biblioteca de funções PCL que devolve a lista de índices dos segmentos C. Ao proceder à extração da lista de índices C na nuvem de pontos inicial P e com a aplicação conjunta das ferramentas disponibilizadas pelo ROS para visualização, pode-se obter uma representação do volume ocupado pelos segmentos. A visualização dos segmentos é efetuada a partir do cálculo do centroide de cada segmento da lista C, e dos valores mínimo e máximo determinados no conjunto de

pontos. Com esta informação, representa-se um marcador no visualizador de mensagens do ROS. A Figura 3.7 representa a segmentação euclidiana de uma situação típica de navegação urbana.



(a) Imagem. (b) Nuvem de pontos filtrada e segmentada.

Figura 3.7: Exemplo da aplicação da segmentação euclidiana.

Apesar da eficácia da aplicação de segmentação por distância euclidiana, há algumas restrições do método que convém referir. A principal desvantagem é que o algoritmo exige que todos os pontos que existem na nuvem sejam processados. Tal facto pode constituir um problema por dois motivos principais; em primeiro lugar nem todos os pontos são informativos acerca do ambiente que rodeia o veículo, e em segundo lugar o processamento de todos os pontos da nuvem implica um custo computacional relativamente elevado.

3.3.3 Algoritmo de crescimento de regiões

A análise ao algoritmo de segmentação euclidiana conduziu a definição de uma outra forma de proceder à segmentação da nuvem de pontos. O método consiste na aplicação da segmentação euclidiana a dados bidimensionais, e, de seguida, proceder à reconstrução tridimensional a partir de um algoritmo de crescimento de regiões.

O ATLASCAR está equipado com dois lasers *Sick LMS100* situados no para choques frontal do veículo, um do lado esquerdo e outro do lado direito (Figura 3.1). Os lasers são capazes de publicar informação para o sistema ROS a uma taxa de $50Hz$, um valor maior que o da nuvem de pontos do sensor estéreo (cerca de $5Hz$ dependendo do processamento do computador utilizado). Uma vantagem em utilizar a informação bidimensional fornecida pelos lasers é que a quantidade de pontos obtida é significativamente menor, logo o processamento deste tipo de informação é mais reduzido que os dados de sensores tridimensionais. Desta forma, foi proposta a realização de uma aplicação que partisse dos dados bidimensionais fornecidos pelos lasers e, aplicar o algoritmo de crescimento de regiões para obter posteriormente a representação tridimensional de obstáculos a partir dos dados estéreo. Considere-se por P_L e P_R as nuvens de pontos fornecidas pelos lasers esquerdo e direito, respetivamente, em que a ordem de operações do algoritmo proposto se enumera da seguinte forma:

1. "Soma"das nuvens de pontos dos dois lasers, $P = P_L \cup P_R$;
2. Aplicação de um filtro de distâncias à nuvem de pontos P de acordo com os dados do sensor estéreo (secção 3.1.3);

3. Aplicar a segmentação euclidiana aos dados do laser consoante o método descrito na secção 3.3.2 para a obtenção de uma lista de segmentos C;

4. Para cada segmento C, obtido na etapa anterior, aplicar o algoritmo de crescimento de regiões para obtenção da representação tridimensional.

Implementação do algoritmo

Ao contrário do algoritmo de segmentação por distância euclidiana, o algoritmo de crescimento de regiões não se encontra implementado na biblioteca PCL. Desta forma, surge a necessidade da implementação de uma classe escrita em $C++$ capaz de aplicar o cálculo do crescimento de regiões. Tendo em conta a notação anterior e denominando por P_S o conjunto de pontos do sensor estéreo, a descrição matemática do algoritmo pode ser listada da seguinte forma:

1. Criar uma representação estruturada em *kd-tree* da nuvem de pontos do sensor estéreo P_S;

2. Criar um vetor de pontos processados T e um vetor de pontos que precisam de ser verificados Q;

3. Para cada ponto $p_j \in P_L$ efetuar as seguintes operações:

   (a) Procurar por todos os vizinhos de $p_j^k \in P_S$ num raio $r < r_{th}$, em que r_{th} é um limite imposto;

   (b) Adicionar todos os pontos p_j^k encontrados anteriormente à lista de pontos que têm de ser verificados Q;

4. Para cada ponto $p_i \in Q$ realizar as seguintes operações:

   (a) Verificar se o ponto já foi processado, ou seja, $p_i \in T$;

   (b) Se $p_i \notin T$:

      i. Procurar por todos os vizinhos de p_i num raio $r < r_{th}$ e adicioná-los à lista de pontos que necessitam de ser procurados Q;

      ii. Adicionar p_i à lista de pontos processados T;

5. O algoritmo termina quando todos os pontos na lista Q estão verificados, ou seja, quando a região deixa de crescer.

Esta foi a estrutura adotada para a implementação do algoritmo, já que dispensa a representação da informação do laser em *kd-tree*. Desta forma, a criação da lista que constitui as sementes do algoritmo de crescimento de regiões é determinado pela procura dos vizinhos de P_L em P_S, e o algoritmo de crescimento de regiões começa a partir desse ponto.

Neste caso, a aplicação do algoritmo de crescimento de regiões é feita para cada segmento de forma individual. Contudo, a sua aplicação pode ser feita para vários segmentos em simultâneo, desde que estejam espaçados de uma distância superior a r_{th}.

A Figura 3.8 representa um exemplo da aplicação do algoritmo no cenário da Figura 3.7(a). Os pontos representados a vermelho representam a informação dos dados dos lasers sobre os quais o método é aplicado para a determinação dos segmentos tridimensionais.

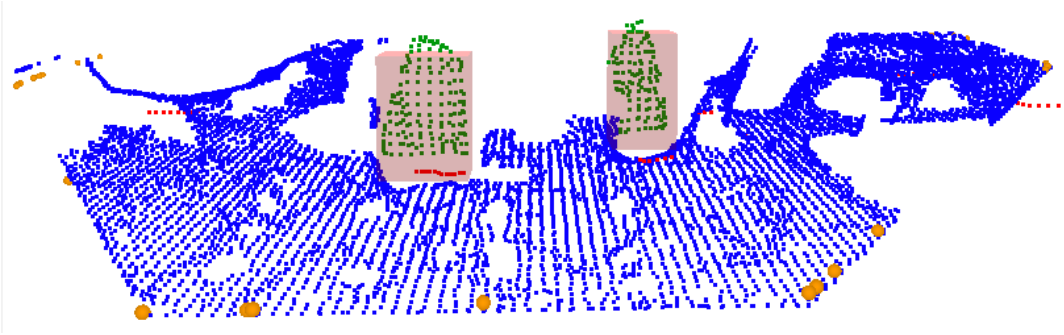


Figura 3.8: Exemplo da aplicação do algoritmo de crescimento de regiões.

Através da fusão dos sensores, é possível estimar o volume ocupado pelos objetos encontrados de uma forma bastante credível. A utilização de informação proveniente de diversos sensores apresenta uma vantagem fundamental que é a repetibilidade de informação, ou seja, se um determinado conjunto de pontos P_0 é detetado pelo sensor A, e um outro conjunto P_1 é detetado pelos sensores A e B, a probabilidade da informação de P_1 corresponder à realidade é relativamente superior à de P_0.

Capítulo 4

Resultados e discussão

4.1 Resultados obtidos

No capítulo 2 foi abordada a questão de calibração de câmaras estéreo bem como o caso particular da câmara utilizada, *Bumblebee XB3*. Quanto ao equipamento, a sua prestação é notável, já que é possível extrair o conteúdo de três imagens em simultâneo com alta resolução a uma taxa relativamente elevada (secção 2.1). Contudo, os problemas surgem do lado da computação, já que nem todas as imagens que o sensor disponibiliza são registadas em computador, e nem todas as imagens obtidas podem ser processadas em tempo real. Por isso, surge a necessidade de avaliar qualitativamente e quantitativamente os resultados obtidos ao nível de aquisição de informação.

Ao longo do capítulo 3, relativo ao processamento tridimensional, tomaram-se algumas escolhas e desenvolveram-se mecanismos para a segmentação de obstáculos. Contudo, no âmbito do enquadramento do projeto, surge a necessidade de avaliar os resultados dos diversos métodos adotados, a nível de situações de falha e de execução computacional.

Desta forma, com o intuito de avaliar o desempenho do software, pretende-se neste capítulo efetuar algumas medições e comparações, bem como enumerar algumas situações de falhas observadas. Na sequência do trabalho, pretende-se estabelecer qual a fiabilidade dos métodos e quais as suas limitações. Por isso, para efetuar uma análise quantitativa aos dois modelos, realizaram-se algumas experiências ao longo de um percurso com uma duração de cerca de 110 segundos em ambiente urbano na cidade de Aveiro, representado na Figura 4.1) retirada do serviço *google maps*. O objetivo é usufruir da ferramenta disponibilizada pelo ROS para acumulação de mensagens em disco ao longo do tempo, para poder reproduzir o mesmo percurso várias vezes e, assim, efetuar uma análise comparativa entre os métodos utilizados. Para a avaliação dos resultados, pretende-se registar algumas frequências de publicação de tópicos dos nodos. Desta forma, o sistema operativo ROS (*Robot Operating System*) apresenta um papel fundamental na obtenção de resultados, já que permite a avaliação dos métodos abordados exatamente nas mesmas condições.

Figura 4.1: Mapa do percurso efetuado.

Assim, a avaliação é efetuada após a recolha de dados, e os resultados obtidos foram processados num computador de laboratório com as seguintes características principais:

- Processador - *Quadcore Intel(R)Core(TM)* i3 CPU 530 @ 2.93GHz;
- Memória RAM disponível - 4096 Mb;
- Sistema Operativo - Ubuntu 11.04 (*"Natty Narwhal"*);
- Kernel - *Linux 2.6.38-15-generic.*

Com o hardware disponível pretende-se avaliar o desempenho dos módulos desenvolvidos. Em primeiro lugar, apresenta-se uma avaliação do processamento de mapas tridimensionais efetuados através do módulo disponibilizado pelo ROS (*stereo_image_proc*), para obter uma perceção do custo computacional do processamento estéreo. De seguida, avalia-se o nodo de deteção do plano do chão desenvolvido que envolve a aplicação do modelo RANSAC, seguido de duas extrações: a extração dos pontos que pertencem ao chão, e a extração do conjunto de pontos na zona navegável, conforme a descrição detalhada na secção 3.2. Por fim, pretendem-se avaliar e comparar os métodos de aplicação da segmentação euclidiana combinada com o algoritmo de crescimento de regiões e verificar algumas situações relevantes.

4.2 Eficiência do cálculo estéreo

Para avaliar o processamento estéreo a nível computacional, executou-se um teste para o *wide baseline*, utilizado largamente para exemplos e resultados deste trabalho. Em

primeiro lugar, registaram-se as médias da frequência da publicação do par de imagens estéreo, representadas pela linha azul na Figura 4.2. De seguida, num novo ensaio, registou-se a frequência de publicação de resultados por parte do nodo de cálculo estéreo do ROS para o tópico relativo à nuvem de pontos, ilustrada a vermelho no gráfico da Figura 4.2.

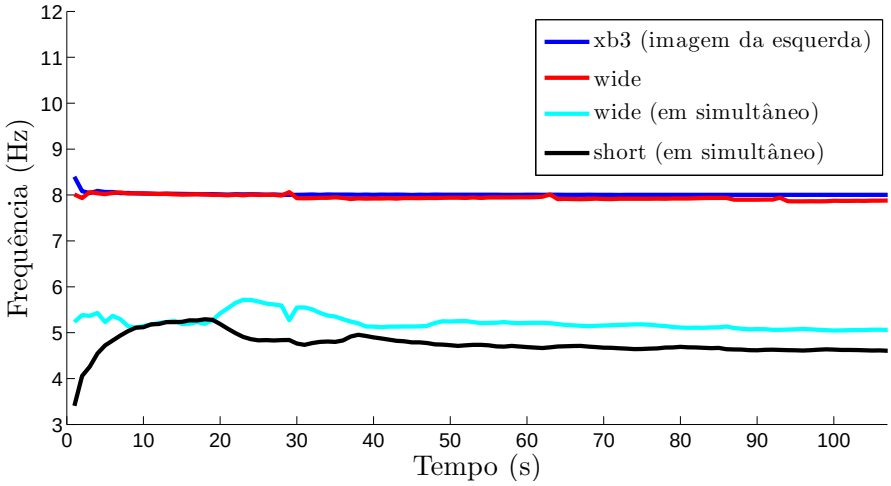


Figura 4.2: Frequências do cálculo estéreo (*short* e *wide*).

No ilustração 4.2 é possível reparar na elevada eficiência do processamento estéreo, quando efetuado apenas para um *baseline*. A média de frequência registada para a imagem da esquerda da câmara foi cerca de 8.01 Hz, enquanto que a frequência registada para a nuvem de pontos processada foi cerca de 7.99 Hz. Estes resultados representam uma eficiência (rácio de frequências) de 99.8%, isto significa que quase todas as mensagens foram processadas. Contudo, surge a necessidade de avaliar também a eficiência do processamento estéreo, quando se pretende utilizar os *baselines wide* e *short* em simultâneo. Nesta situação, as frequências apresentaram-se bastante mais reduzidas, sendo de 4.76 Hz para o *wide baseline* e de 5.22 Hz para o *short baseline*, que representam eficiências na ordem dos 59% para os dois casos. Os resultados obtidos comprovam o custo computacional envolvido quando se pretende utilizar os dois *baselines* em simultâneo.

4.3 Segmentação do plano do chão

Na ordem de seguimento do trabalho, é necessário efetuar uma avaliação semelhante à anterior, mas para a segmentação do plano da estrada, cálculo da zona navegável e posterior extração dos pontos. Os resultados obtidos encontram-se ilustrados na Figura 4.3, em que a curva assinalada a azul representa a nuvem de pontos de entrada no nodo, neste caso uma nuvem de pontos filtrada com uma *voxel grid* de dez centímetros de resolução e com um filtro de distâncias que descarta todos os pontos a uma distância superior a vinte metros. A curva a vermelho representa a nuvem de pontos com a informação relativa aos obstáculos dentro da zona navegável.

Os resultados globais obtidos foram de uma publicação a 5.69 Hz para a nuvem de pontos filtrada que representa uma eficiência no processo de 71.2%. A frequência média

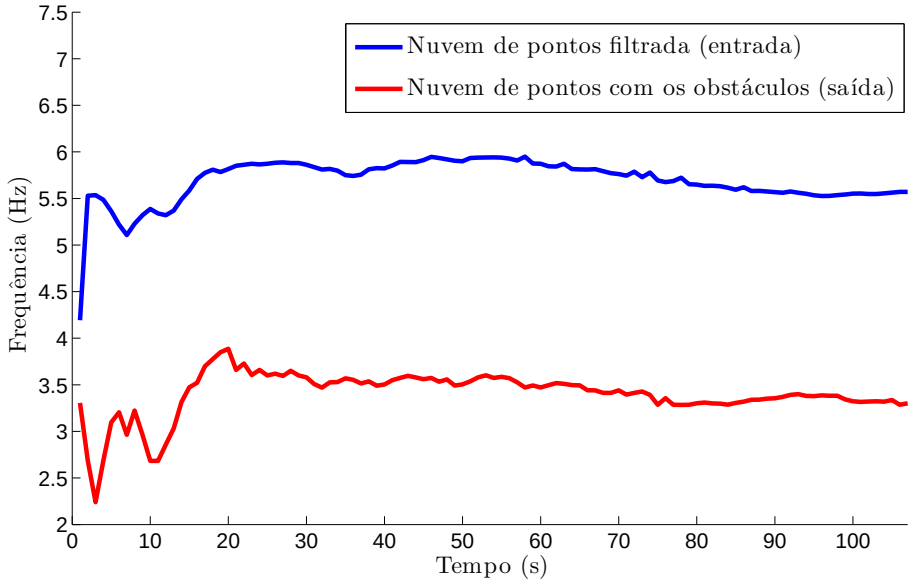


Figura 4.3: Frequências da segmentação do plano do chão.

de publicação do nodo de segmentação do plano da estrada, cálculo da zona navegável e posterior extração apresenta uma frequência de publicação de 3.40 Hz, ou seja uma eficiência de processamento de 59.8%.

4.4 Segmentação euclidiana e crescimento de regiões

Ao longo da secção 3.3 foram descritas as etapas necessárias para a segmentação de obstáculos dentro da zona navegável do veículo, em que numa é aplicado o algoritmo de segmentação euclidiana à nuvem de pontos tridimensional do sensor estéreo, e na outra o método é aplicado a dados bidimensionais dos lasers e propagado posteriormente para a informação tridimensional.

Para o percurso ilustrado, apresentam-se os resultados obtidos ao longo do tempo para as taxas de publicação dos tópicos para os dois métodos de segmentação, com o objetivo de determinar qual o mais eficaz e analisar de que forma a quantidade de dados envolvida pode influenciar o processo.

A Figura 4.4 ilustra os resultados obtidos das diversas frequências de publicação de tópicos para o ambiente ROS. Para este resultado, aplicou-se um filtro *voxel* para redução da quantidade de dados com uma resolução de vinte centímetros, que resultou numa média de 873 pontos a processar por cada *frame*. Claramente, neste caso, a prestação da segmentação euclidiana aplicada diretamente ao dados tridimensionais revelou-se um pouco mais eficaz, com uma média global de publicação de 3.90 Hz do resultado final ao longo de todo o percurso, muito próximo da frequência de entrada (3.92 Hz). A aplicação do algoritmo de crescimento de região apresentou uma frequência um pouco mais reduzida com uma média de 3.66 Hz. Os resultados justificam-se na medida em que o número de pontos médios a processar é significativamente reduzido. Desta forma, devido ao tempo que é necessário para o processamento dos dados do laser para aplicação do método de

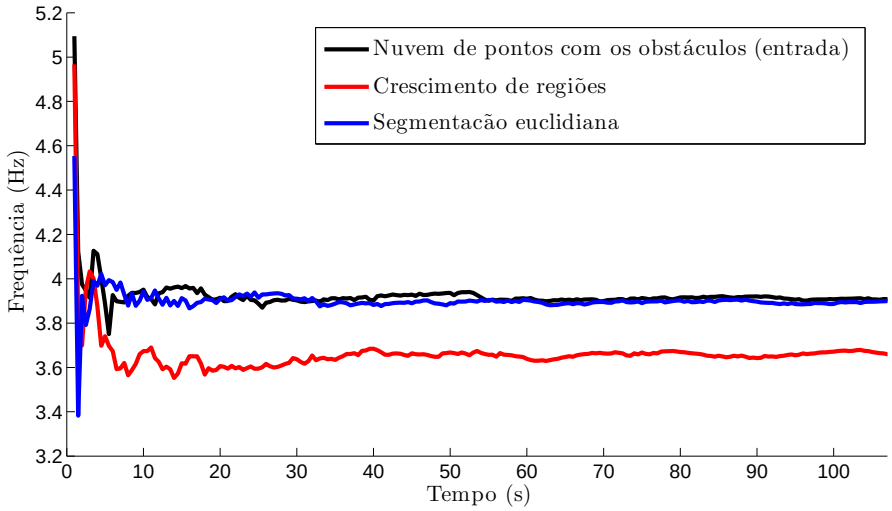


Figura 4.4: Taxas de publicação com *voxels* de 20 cm.

crescimento de regiões, não se justifica a sua aplicação. Contudo, a aplicação de filtros com *voxels* de dimensão elevada resulta numa nuvem de pontos com pouca resolução e, portanto, com pouco detalhe na descrição das superfícies, como no exemplo da Figura 3.2. Por isso, efetuou-se um segundo teste para uma resolução da grelha do filtro com metade da dimensão anterior, representado pela Figura 4.5.

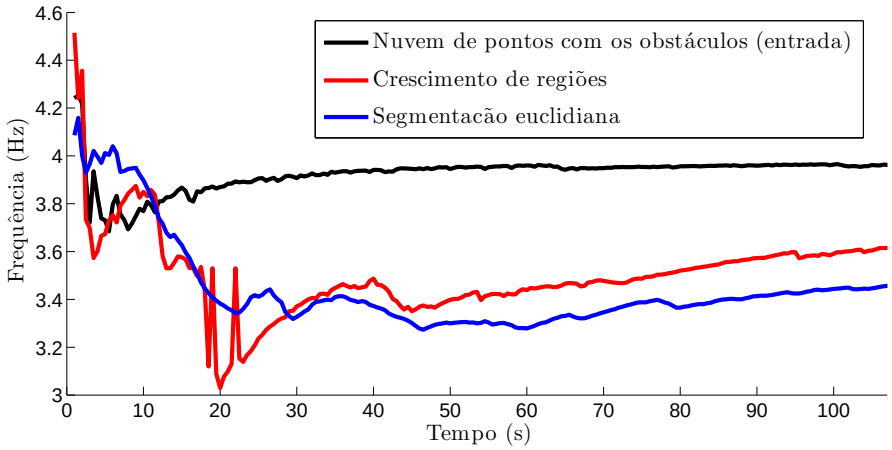


Figura 4.5: Taxas de publicação com *voxels* de 10 cm.

Neste caso, os resultados revelaram-se um pouco diferentes dos anteriores. A aplicação de *voxels* de dez centímetros resulta na obtenção de nuvens de pontos com cerca de 2825 pontos, que representa cerca do triplo da informação do teste anterior. Neste teste os valores registados revelaram-se mais próximas, com valores médios globais de 3.5 Hz para o algoritmo de crescimento de regiões e 3.45 Hz para a segmentação euclidiana. Contudo, na fase inicial, a segmentação euclidiana aplicada diretamente à nuvem de pontos proveniente do sistema estéreo apresenta taxas de publicação superiores. Estas

zonas representam localizações onde se encontram obstáculos de dimensões elevadas que levam a que o algoritmo de crescimento de regiões exija mais tempo para a procura de vizinhança de pontos. Os resultados obtidos até aqui levaram ao procedimento de um terceiro teste para nuvens de pontos mais densas. Contudo, a utilização de um *voxel grid* com uma resolução maior que cerca de dez centímetros não permite um processamento de toda a informação em tempo real na máquina utilizada. Por isso, pretende-se tirar vantagem do sistema de reprodução de mensagens do ROS que permite a publicação de tópicos a velocidades configuráveis, como se fosse em "câmara lenta"e desta forma é possível estimar quais seriam as frequências de publicação de resultados ao aplicar um filtro *voxel* de cinco centímetros. A Figura 4.6 ilustra o gráfico obtido nesta situação.

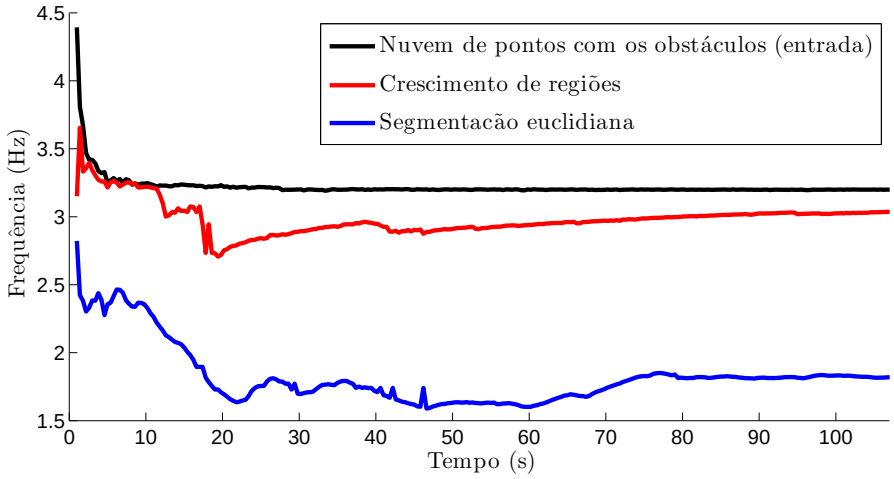
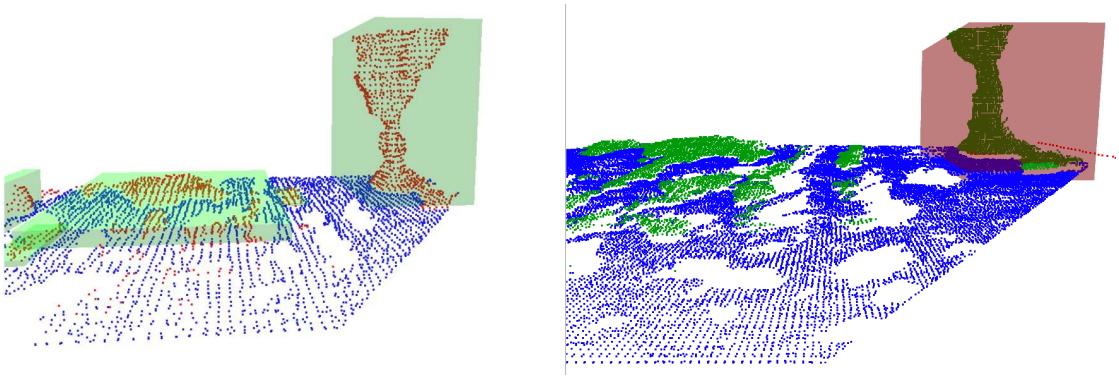


Figura 4.6: Taxas de publicação com *voxels* de 5 cm.

Os resultados obtidos nesta experiência demonstraram-se efetivamente distintos dos restantes. Repare-se que neste caso a média de pontos por *frame* obtidos é cerca de 6976, um número bastante elevado que exigiu uma redução da velocidade para cerca de quarenta por cento, para que o processador pudesse efetuar todos os cálculos necessários. Esta quantidade de dados resultou numa inversão dos resultados obtidos até agora. Repare-se que, neste caso, o tempo necessário para a execução do algoritmo de segmentação euclidiana representada a azul na Figura 4.6 torna-se bastante consumidor. Este facto deve-se à necessidade de processamento de todos os pontos da nuvem da câmara estéreo, contrariamente ao que acontece com a aplicação do método de crescimento de regiões.

Um problema da aplicação da visão estéreo para perceção tridimensional, é a tendência para a distorção das superfícies para distâncias relativamente elevadas, de acordo com a relação inversa e não linear entre a disparidade e profundidade (subsecção 2.4.5). Por isso, o algoritmo de crescimento de regiões apresenta um papel importante nesse sentido, já que o alcance dos sensores laser é, normalmente, superior ao do sensor estéreo, e desta forma, os obstáculos que por vezes são detetados erradamente pelo sensor estéreo podem ser descartados. A Figura 4.7 representa um exemplo típico em que os maus resultados no cálculo estéreo influenciam o resultado da segmentação euclidiana, enquanto que o resultado da aplicação do método de crescimento de regiões apresenta um resultado mais coerente nesse sentido.

Neste caso, a sombra causada por uma árvore influencia o cálculo estéreo e a posterior

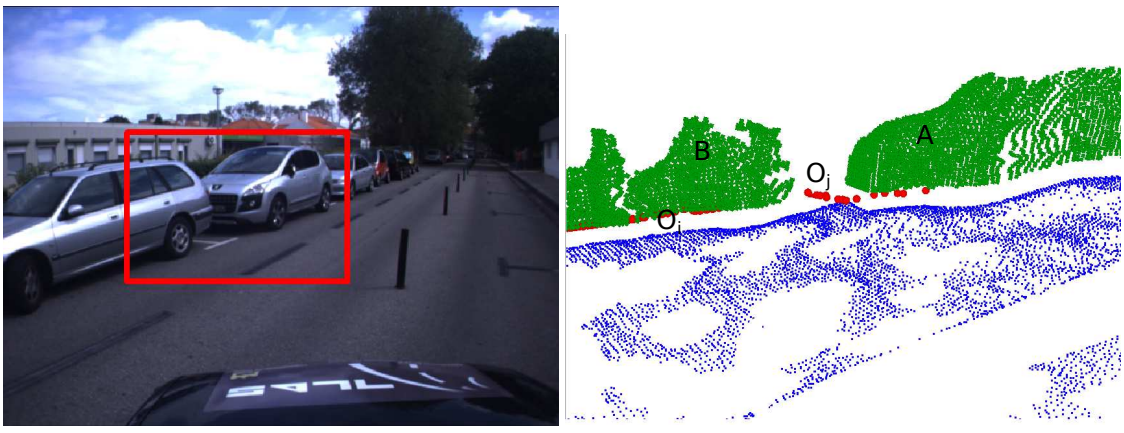


(a) Segmentação euclidiana. (b) Crescimento de regiões.

Figura 4.7: Falha na aplicação da segmentação euclidiana.

segmentação. A Figura 4.7(a) ilustra a aplicação da segmentação euclidiana aos dados tridimensionais do sensor estéreo, em que o algoritmo apresenta regiões de falha ao detetar obstáculos em cima da estrada que não existem. Neste tipo de situações, a aplicação do algoritmo de crescimento de regiões à informação do laser revela resultados mais coerentes em que o único obstáculo representado é comum aos dados dos lasers e da câmara.

Outro aspeto que se verifica correntemente, é a questão do registo dos sensores no sistema. Os métodos de avaliação de distâncias nos sensores são relativamente diferentes, enquanto que o sensor estéreo efetua uma medição indireta, o laser efetua uma medição direta e por vezes os obstáculos encontrados pelos lasers e pela câmara não se encontram devidamente nas mesmas coordenadas. Na maioria dos casos, este facto não constitui um problema de grande ordem, uma vez que, para a aplicação do método, nem todos os pontos considerados "semente"para a aplicação do crescimento de regiões têm necessariamente de se encontrar alinhados com os dados do sensor estéreo, desde que respeitem o limite definido por r_{th} (subsecção 3.3.3). A Figura 4.8 ilustra uma situação de curvatura do veículo em que o efeito se agrava.



(a) Zona de falha. (b) Nuvem de pontos.

Figura 4.8: Falhas de registo nos sensores.

A zona assinalada a vermelho na Figura 4.8(a) representa uma situação típica em

que se verificam falhas no registo dos sensores. Este tipo de situações torna-se crítico quando os pontos "semente"são ambíguos, ou seja, o algoritmo pode crescer para a região errada e induzir em erro. Considerem-se os segmentos O_i e O_j representados na Figura 4.8(b) pelos pontos a vermelho do laser, e os conjuntos de pontos A e B provenientes do sensor estéreo. Ao aplicar o algoritmo de crescimento de regiões o grupo de "sementes"O_i expande-se para A tal que: $O_i = \{p_i : p_i \in A\}$ e O_j expande-se para A e B, tal que $O_j = \{p_j : p_j \in A \cap B\}$. Neste caso, resultam dois segmentos, um constituído pelos pontos de A e outro pelos pontos de $A \cap B$, em vez de conter somente os pontos de B. Esta situação, representa uma das falhas mais críticas da aplicação do algoritmo de crescimento de regiões, e por este motivo é necessário proceder a um registo coerente dos diversos sensores do sistema.

Capítulo 5

Conclusões e Trabalho futuro

5.1 Conclusões finais

O processo de calibração aplicado à câmara e ao módulo de processamento estéreo permitiu melhorar significativamente a qualidade dos dados obtidos para esta aplicação. A execução de uma calibração rigorosa e o estudo da influência dos parâmetros estéreo revelaram-se de facto um passo importante para o processamento tridimensional. Assim, foi bem sucedida a implementação de um algoritmo de crescimento de regiões que se demonstrou uma forma interessante de proceder à segmentação, especialmente quando a quantidade de informação é elevada.

O trabalho efetuado nesta dissertação de Mestrado não se encontra isolado do enquadramento de aplicação em sistemas de apoio à condução. A implementação de um algoritmo de crescimento de regiões permite a continuação do estudo de mecanismos de atenção. Por exemplo, com a aplicação conjunta de algoritmos de seguimento de obstáculos em movimento (*Motion Target Tracking*) aplicados aos lasers, é possível proceder à posterior reconstrução tridimensional para o reconhecimento de obstáculos em movimento em torno do veículo. Desta forma, é importante o estudo e a continuação do desenvolvimento de sistemas de perceção tridimensional, não só para o uso em sistemas de apoio à condução, mas também para sistemas de vigilância ou robótica doméstica. A aplicação atual de sistemas de perceção tridimensionais enfrenta ainda o problema computacional envolvido. Infelizmente, apesar dos resultados obtidos serem satisfatórios, a aplicação de metodologias de segmentação tridimensional em tempo real para veículos em movimento, sobretudo a grandes velocidades, é ainda um problema por resolver. Contudo, o desenvolvimento tecnológico tem provado a sua existência e o aparecimento de processadores mais rápidos e robustos não é novidade nos dias de hoje. Assim sendo, o trabalho aqui desenvolvido não apresenta uma conclusão final, mas demonstra aquilo que deve ser continuado ao nível de sistemas de visão tridimensional e as áreas e situações que devem ser exploradas para o seu melhoramento.

5.2 Trabalho futuro

Com a realização deste trabalho, pode-se concluir que ainda muito pouco se tem efetuado na área de perceção tridimensional. A visão estéreo é uma área vasta, com muito por explorar, não só no desenvolvimento de algoritmos para aplicações em nuvens de pontos, mas também no uso de mapas de disparidade sem recorrer à reprojeção tridimensional. Esta pode ser uma frente de trabalho futuro com interesse em explorar. A principal vantagem no uso de mapas de disparidade, é que eles permitem obter informação tridimensional a partir de imagens, ou seja, sem sair do campo bidimensional. Tal facto pode constituir uma vantagem, sobretudo no que diz respeito a tempos de computação.

Um aspeto relevante a salientar, é no uso da câmara estéreo (*Bumblebee XB3*). Na verdade, atualmente, não se obtém o máximo partido do hardware a partir do uso do software disponibilizado pelo ROS (*Robot Operating System*). O principal motivo é a necessidade de replicação da informação de uma câmara (a da esquerda) para poder proceder ao cálculo estéreo dos dois *baselines* (*wide* e *short*). Uma proposta de trabalho futuro a realizar neste sentido seria o desenvolvimento de software para o cálculo e calibração de sensores estéreo com três câmaras em linha. Coloca-se ainda o estudo do desenvolvimento de um modelo para proceder ao cálculo estéreo a partir das três imagens em vez de utilizar os dois pares separadamente, ou seja, obter um único mapa de disparidade sem recorrer ao cálculo de duas nuvens de pontos, que prejudica o desempenho do computador, como demonstrado no capítulo de resultados.

Quanto ao processamento tridimensional, o trabalho realizado constitui uma etapa naquilo que pode ser feito para manobras e navegação de veículos autónomos. Fica em aberto a análise de descritores PFH (*Point Feature Histograms*) e VFH (*Viewpoint Feature Histograms*). Estes métodos constituem algoritmos avançados de perceção e distinção de obstáculos e podem ser úteis para o reconhecimento de peões, veículos ou sinais de trânsito.

Bibliografia

[Atlas, 2011] Atlas (2011). Atlas project. Obtido de: http://atlas.web.ua.pt. Consultado em: Novembro de 2011.

[Bradski and Kaehler, 2008] Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. Software that sees. O'Reilly.

[Center et al., 1980] Center, S. I. A. I., Fischler, A., and Bolles, R. (1980). *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*. Technical note. SRI International.

[Dobkin and Huhdanpa, 1996] Dobkin, D. P. and Huhdanpa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*.

[Gallup, 2008] Gallup, D. (2008). Variable baseline/resolution stereo. *Computer Vision and Pattern Recognition*.

[Gottwald and Gellert, 1989] Gottwald, S. and Gellert, W. (1989). *The VNR concise encyclopedia of mathematics*. Van Nostrand Reinhold.

[Morgan Quigley, 2007] Morgan Quigley, Eric Berger, A. Y. N. (2007). Stair: Hardware and software architecture. *AAAI 2007 Robotics Workshop*.

[PointGrey, 2011] PointGrey (2011). Bumblebee stereo vision camera systems. Obtido de: http://www.ptgrey.com. Ficha técnica.

[Quigley et al., 2009] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2009). Ros: an open-source robot operating system. *ICRA Workshop on Open Source Software*.

[ROS, 2012] ROS (2012). Ros. Obtido de: http://www.ros.org/wiki. Consultado em: Fevereiro de 2012.

[Rusu, 2009] Rusu, R. B. (2009). *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Technische Universitaet Muenchen.

[Rusu and Cousins, 2011] Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

[StattSoft, 2012] StattSoft (2012). Stattsoft: Electronic statistics textbook. Obtido de: http://www.statsoft.com/textbook/cluster-analysis. Consultado em: Maio de 2012.

[Szeliski and Zabih, 2000] Szeliski, R. and Zabih, R. (2000). An experimental comparison of stereo algorithms. *Computer science*.

[Wöhler, 2009] Wöhler, C. (2009). *3D Computer Vision*. Springer Science & Business Media.