Hugo Filipe de Oliveira Gomes da Fonseca Queirós

Licenciado em Engenharia Electrotécnica e de Computadores

Exploração da Point Cloud Library Aplicada à Percepção em Sistemas Autónomos

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores
Área de Especialização de Sistemas Autónomos

Orientador : Eduardo Alexandre Pereira da Silva, Prof. Doutor, Instituto Superior de Engenharia do Porto

Co-orientador : José Miguel Soares de Almeida, Prof., Instituto Superior de Engenharia do Porto

Novembro, 2012

Instituto Superior de Engenharia do Porto

Exploração da Point Cloud Library Aplicada à Percepção em Sistemas Autónomos

"You, the people, have the power, the power to create machines, the power to create happiness! You, the people, have the power to make this life free and beautiful, to make this life a wonderful adventure."
Sir Charles Chaplin (The Great Dictator)

Agradecimentos

Em primeiro lugar gostaria de prestar o meu profundo agradecimento aos meus pais e irmão, que sempre me apoiaram nesta atribulada viagem.

Em segundo lugar, gostaria de agradecer à minha segunda família, Amélia, Elisabete, Zé, Lena, Rui e à minha afilhada Joana pela compreensão da minha ausência e apoio prestado ao longo da minha vida.

Agradeço também o apoio incondicional prestado pelos meus padrinhos, José Fernandes e Elisa, que estiveram presentes sempre que foi necessário.

Gostaria de endereçar um especial agradecimento ao meu colega de curso Filipe Ferreira, pela amizade e disponibilidade em ouvir-me e ajudar.

Pretendo também agradecer a todos os meus colegas do LSA, em especial à Maria Costa, Pedro Gonçalves, João Teixeira, Paulo e Vando, pelos momentos de boa disposição e companhia que proporcionaram ao longo de pouco mais de dois anos, em noites sem dormir e fins-de-semana a trabalhar.

Por último agradeço ao meu co-orientador Prof. José Almeida e orientador Prof. Doutor Eduardo Silva e a todo conjunto de pessoas diariamente presentes no LSA por proporcionarem um ambiente em que o conhecimento pode sempre chegar mais longe do que seria espectável.

Resumo

A navegação e a interpretação do meio envolvente por veículos autónomos em ambientes não estruturados continua a ser um grande desafio na actualidade. Sebastian Thrun, descreve em [Thr02], que o problema do mapeamento em sistemas robóticos é o da aquisição de um modelo espacial do meio envolvente do robô.

Neste contexto, a integração de sistemas sensoriais em plataformas robóticas, que permitam a construção de mapas do mundo que as rodeia é de extrema importância. A informação recolhida desses dados pode ser interpretada, tendo aplicabilidade em tarefas de localização, navegação e manipulação de objectos.

Até à bem pouco tempo, a generalidade dos sistemas robóticos que realizavam tarefas de mapeamento ou *Simultaneous Localization And Mapping* (SLAM), utilizavam dispositivos do tipo *laser rangefinders* e câmaras stereo. Estes equipamentos, para além de serem dispendiosos, fornecem apenas informação bidimensional, recolhidas através de cortes transversais 2D, no caso dos rangefinders.

O paradigma deste tipo de tecnologia mudou consideravelmente, com o lançamento no mercado de câmaras RGB-D, como a desenvolvida pela PrimeSense™[1] e o subsequente lançamento da Kinect[2], pela Microsoft® para a Xbox 360 no final de 2010. A qualidade do sensor de profundidade, dada a natureza de baixo custo e a sua capacidade de aquisição de dados em tempo real, é incontornável, fazendo com que o sensor se tornasse instantaneamente popular entre pesquisadores e entusiastas.

Este avanço tecnológico deu origem a várias ferramentas de desenvolvimento e interacção humana com este tipo de sensor, como por exemplo a *Point Cloud Library*[3] [RC11] (PCL). Esta ferramenta tem como objectivo fornecer suporte para todos os

[1] PrimeSense - www.primesense.com/en/solutions/solsensor
[2] Kinect - www.xbox.com/en-US/kinect
[3] PCL - www.pointclouds.org

blocos de construção comuns que uma aplicação 3D necessita, dando especial ênfase ao processamento de nuvens de pontos de *n* dimensões adquiridas a partir de câmaras RGB-D, bem como scanners laser, câmaras *Time-of-Flight* ou câmaras stereo.

Neste contexto, é realizada nesta dissertação, a avaliação e comparação de alguns dos módulos e métodos constituintes da biblioteca PCL, para a resolução de problemas inerentes à construção e interpretação de mapas, em ambientes *indoor* não estruturados, utilizando os dados provenientes da Kinect.

A partir desta avaliação, é proposta uma arquitectura de sistema que sistematiza o registo de nuvens de pontos, correspondentes a vistas parciais do mundo, num modelo global consistente.

Os resultados da avaliação realizada à biblioteca PCL atestam a sua viabilidade, para a resolução dos problemas propostos. Prova da sua viabilidade, são os resultados práticos obtidos, da implementação da arquitectura de sistema proposta, que apresenta resultados de desempenho interessantes, como também boas perspectivas de integração deste tipo de conceitos e tecnologia em plataformas robóticas desenvolvidas no âmbito de projectos do Laboratório de Sistemas Autónomos (LSA).

Palavras-chave: *Point Cloud Library*, *Kinect*, mapeamento

Abstract

The navigation and interpretation of the environment by autonomous vehicles in unstructured environments continues to be a major challenge nowadays. Sebastian Thrun, in [Thr02], states that the problem of robotic mapping is "acquiring a spatial model of a robot's environment".

In this context, the integration of sensory systems in robotic platforms enabling the construction of a world map of the environment is of utmost importance. The information collected from the data can be interpreted, having applicability in localization, navigation and manipulation of objects tasks.

Up until very recently, most robotic systems that performed mapping or Simultaneous Localization And Mapping (SLAM) tasks, used devices like laser rangefinders and stereo cameras. These devices, in addition to being expensive, only provide, in the case of rangefinders, two-dimensional information collected through 2D cross-sections.

The paradigm of this type of technology, shifted considerably, with the market launch of RGB-D cameras, developed by PrimeSense™ and the subsequent launch of the Kinect, by Microsoft® for the XBox 360, in the late 2010. The depth sensor quality, given the low cost nature and its capability for real time data acquisition, is unavoidable, causing the sensor to become instantaneously popular among researchers and enthusiasts.

This technological breakthrough has led to several development and human interaction tools with this sensor type, like the Point Cloud Library [RC11] (PCL). This framework aims to provide support for all the common building blocks that a 3D application needs, giving special emphasis to the processing of point clouds of n dimensions obtained from RGB-D cameras, laser scanners, as well as Time-of-Flight cameras or stereo cameras.

In this context, is performed in this dissertation, the evaluation and comparison of some of the modules and methods that constitute the PCL library, for the

resolution of inherent problems to the building and interpretation of maps, in unstructured indoor environments, using the Kinect data.

From this assessment, it is proposed a system architecture that systematizes point clouds registration, corresponding to partial world viewpoints, in a consistent global model.

The evaluation results, performed to the PCL library, attest its viability for the solving of the proposed problems. Proof of their viability, are the obtained practical results obtained, from the implementation of the proposed system architecture, which presents interesting performance results, as well as a good prospects of integrating this type of technology and concepts in robotic platforms developed in the Laboratório de Sistemas Autónomos (LSA) scope.

Keywords: *Point Cloud Library*, *Kinect*, mapping

Conteúdo

Lista de Figuras

Lista de Tabelas

Listagens

Acrónimos

AR	*Augmented Reality*
BRIEF	*Binary Robust Independent Elementary Features*
BSD	*Berkley Software Distribution*
CE-RSAC	*Correspondence Estimation - Rejection SAC*
CPU	*Central Processing Unit*
CRF	*Conditional Random Fields*
CVFH	*Clustered Viewpoint Feature Histogram*
DoG	*Difference of Gaussians*
EGI	*Extended Gaussian Images*
ELCH	*Explicit Loop Closing Heuristic*
ESF	*Ensemble of Shape Functions*
FPFH	*Fast Point Feature Histograms*
FLANN	*Fast Library for Approximate Nearest Neighbors*
GIA	*Greedy Inicial Alignment*
GPU	*Graphic Processing Unit*
ICP	*Iteractive Closest Points*
INS	*Inertial Navigation System*
LIDAR	*LIght Detection And Ranging*
LMEDS	*Least Median of Squares*
LMS	*Laser Measurement Systems*
LSA	Laboratório de Sistemas Autónomos
MLESAC	*Maximum LikeLihood Estimation SAmple Consensus*
MLS	*Moving Least Squares*
MSAC	*M-Estimator SAmple Consensus*
NaN	*Not a Number*
NARF	*Normal Aligned Radial Features*
NDT	*Normal Distribution Transform*

OpenCV *Open Source Computer Vision*
OpenMP *Open Multi Processing*
OpenNI *Open Natural Interaction*
PCD *Point Cloud Data*
PCL *Point Cloud Library*
PR2 *Personal Robot 2*
PFH *Point Feature Histograms*
PLY *PoLYgon File Format*
PROSAC *Progressive SAmple Consensus*
RANSAC *RANdom SAmple Consensus*
REIN *REcnognition INfrastruture*
RIFT *Rotation Invariant Feature Transform*
RMLS *Robust Moving Least Squares*
RMSAC *Randomized MSAC*
ROS *Robotic Operating System*
RRANSAC *Randomized RANdom SAmple Consensus*
SAC *SAmple Consensus*
SAC-IA *SAmple Consensus Initial Alignment*
SDK *Software Development Kit*
SHOT *Signature of Histograms of OrienTations*
SIFT *Scale Invariant Feature Transform*
SLAM *Simultaneous Localization And Mapping*
SPFH *Simplified Point Feature Histogram*
SSD *Sum of Squared Differences*
STL *STereoLithography*
SURF *Speeded Up Robust Feature*
SVD *Singular Value Decomposition*
SVM *Support Vector Machines*
TBB *Intel Threading Buildind Blocks*
TOF *Time-of-Flight*
VFH *Viewpoint Feature Histograms*
VTK *Visualization ToolKit*

Lista de símbolos e notações

A seguinte lista contêm os símbolos matemáticos e notações, utilizados com mais frequência ao longo desta dissertação.

$\boldsymbol{p_i}$	ponto 3D com coordenadas geométricas $\{x_i, y_i, z_i\}$ (é utilizada também a notação $\boldsymbol{p}$ por questões de simplicidade).
$\vec{\boldsymbol{n}}_{\boldsymbol{i}}$	normal estimada de uma superfície no ponto $\boldsymbol{p_i}$, com orientação $\{n_{x_i}, n_{y_i}, n_{z_i}\}$.
$\boldsymbol{\mathcal{P}_n = \{p_1, p_2, p_3, ...\}}$	nuvem com nD pontos (é utilizada também a notação $\mathcal{P}$ por questões de simplicidade).
$\boldsymbol{\mathcal{P}^k}$	conjunto de pontos $\boldsymbol{p_j}$, $(j \leq k)$, localizados na vizinhança k de um dado ponto $\boldsymbol{p_i}$.
$\boldsymbol{r}$	raio de uma esfera para a determinação da vizinhança $\mathcal{P}^k$ de um ponto.
$\bar{\boldsymbol{p}}$	centroide de um conjunto de pontos $\mathcal{P}$, $$\bar{p} = \frac{1}{k} \cdot \sum_{i=1}^{k} p_i$$
$\vec{\boldsymbol{z}}$	eixo z de um dado sistema de coordenadas.

1

Introdução

Conteúdo

1.1 Motivação

No contexto da robótica, uma das grandes áreas de estudo e desenvolvimento dos últimos anos, é a de sistemas sensoriais que possibilitem o aumento da capacidade de percepção das plataformas robóticas nas quais estão integrados. A construção de mapas tridimensionais do meio envolvente em ambientes não estruturados, é uma importante tarefa dos sistemas autónomos, tendo aplicações na localização, navegação e manipulação de objectos. A utilização destes conceitos fornece um recurso muito importante, podendo dar informação relevante sobre o meio em que um veículo autónomo está inserido aumentando a sua capacidade de percepção do meio.

A grande maioria dos sistemas robóticos que executam tarefas de mapeamento, utilizam sensores que recolhem informação através apenas de cortes transversais 2D do mundo que os rodeia. Este facto decorre de que, até à bem pouco tempo, a

aquisição de informação 3D com alta qualidade, era extremamente dispendiosa ou limitava seriamente os movimentos dos robôs. Por esse motivo, o foco da investigação centrava-se muito em scanners laser, para solucionar problemas relacionados com o mapeamento ou SLAM, embora também existam alguns métodos que utilizam câmaras stereo [BRSM+09].

Em 2010, verificou-se uma importante mudança de paradigma na área da percepção sensorial tridimensional com o advento da Kinect[1], uma câmara RGB-D, construída com base na tecnologia desenvolvida pela PrimeSense™[2], fazendo com que o desenvolvimento nesta área ocorresse a um ritmo muito maior do que até à data. Esta mudança verificou-se, devido à capacidade deste tipo de sensores adquirirem dados a uma frequência elevada e ao seu baixo custo, tornando-se instantaneamente popular entre pesquisadores e entusiastas. A Kinect, por exemplo, permite a aquisição de dados a uma frequência até 30 Hz, contendo informação de profundidade e imagem.

Com isto, verificou-se o surgimento de várias ferramentas de desenvolvimento, que incorporam na sua génese, o processamento de nuvens de pontos adquiridos por dispositivos de percepção tridimensional, tendo especial destaque o surgimento da biblioteca Point Cloud Library [RC11] (PCL), em 2011.

O surgimento destas novas ferramentas teve um forte impacto na comunidade robótica, verificando-se o rápido desenvolvimento de novos algoritmos e aposta na melhoria do desempenho computacional dos já existentes, tentando promover o melhor desempenho possível no sentido de serem implementados em sistemas robóticos em ambientes de construção humana, por natureza muito dinâmicos.

Estes avanços tecnológicos, aliados ao aumento da capacidade de processamento por parte dos CPUs e GPUs e futura evolução das câmaras RGB-D, irá seguramente num futuro não muito longínquo, culminar no uso massivo deste tipo de sensores em sistemas robóticos.

O desafio nesta área é o de encontrar soluções que permitam o rápido mapeamento de um qualquer cenário e a sua interpretação. Isto permite que os sistemas robóticos tenham a capacidade de detectar, reconhecer, localizar e reconstruir geometricamente os objectos do meio no qual estão inseridos, na execução de tarefas como navegação ou manipulação de objectos.

Neste contexto, é realizada nesta dissertação, a avaliação de alguns dos módulos e métodos concorrentes que constituem a biblioteca PCL, para a resolução de problemas inerentes à construção e interpretação de mapas, em ambientes *indoor* não estruturados, utilizando a informação adquirida a partir da Kinect.

Tendo como base esta avaliação, é proposta uma arquitectura de sistema que

[1] Kinect - www.xbox.com/en-US/kinect
[2] PrimeSense - www.primesense.com

sistematiza o registo de nuvens de pontos provenientes da Kinect, correspondentes a vistas parciais do mundo, numa única representação global consistente.

A avaliação efectuada à biblioteca PCL e os seus resultados, atestam a sua viabilidade para a resolução dos desafios acima descritos. Os resultados práticos obtidos, das implementações dos métodos propostos, apresentam resultados de desempenho interessantes, quer no seu peso computacional, quer nos resultados qualitativos. Com isto, perspectivam-se boas hipóteses de estes conceitos, métodos e tecnologia ser integrada em plataformas robóticas desenvolvidas no âmbito de projectos do Laboratório de Sistemas Autónomos (LSA).

1.2 Problema

Em [Thr02], o problema do mapeamento através de sistemas autónomos é descrito como sendo o da aquisição de um modelo espacial do meio envolvente do robô. O relevo do terreno, a presença de objectos dinâmicos, obstáculos ou a inexistência de referências de localização, são algumas das imensas variáveis dinâmicas que descrevem a complexidade do mundo real.

Em cenários *indoor* não estruturados, o facto de não existir um prévio conhecimento desse mundo e dos objectos que lá se encontram, colocam sérios desafios aos sistemas de percepção 3D actualmente existentes.

O aparecimento de câmaras RGB-D de baixo custo como a Kinect, apresenta um bom recurso para a aquisição de dados tridimensionais, mas compreende ainda um nível considerável de ruído, fazendo com que o problema se torne de extrema dificuldade de resolução.

Normalmente, existem também problemas na obtenção de dados provenientes de certos tipos de materiais constituintes de objectos ou superfícies e em alguns casos de condições ambiente, o que pode resultar na existência de buracos nesses espaços do cenário virtual criado. Para a construção de modelos tridimensionais consistentes é necessário, na maior parte das vezes, recolher informação proveniente de várias nuvens de pontos, adquiridas de várias posições no espaço, de forma a se conseguir um modelo global diminuindo as zonas de oclusão, o que nem sempre será possível.

Tendo um conjunto de nuvens de pontos, descrevendo cada uma delas uma região do espaço em ambientes *indoor*, a sequência lógica para a resolução do problema é a obtenção de uma representação única do espaço. Este problema pode ser exemplificado pela Figura 1.1.

A informação recolhida do espaço amostrado, é de extrema utilidade, podendo ser interpretada para ser utilizada em tarefas de navegação ou manipulação de objectos. Neste caso, o desafio prende-se com a construção de mapas funcionais ou

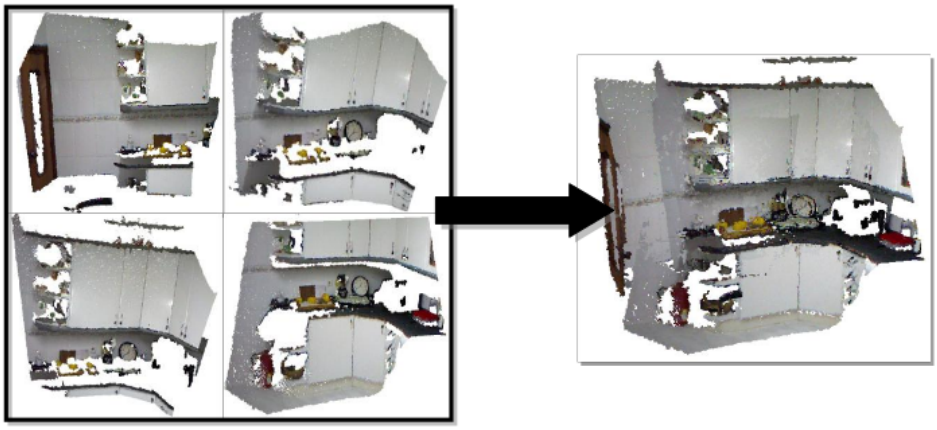


Figura 1.1: Criação de uma representação única do espaço.

semânticos, onde a informação recolhida é simplificada e os objectos classificados. Um exemplo dessa interpretação e classificação de objectos representados por conjuntos de nuvens de pontos, foi proposto em [Rus09], podendo ser demonstrado pela Figura 1.2[3].

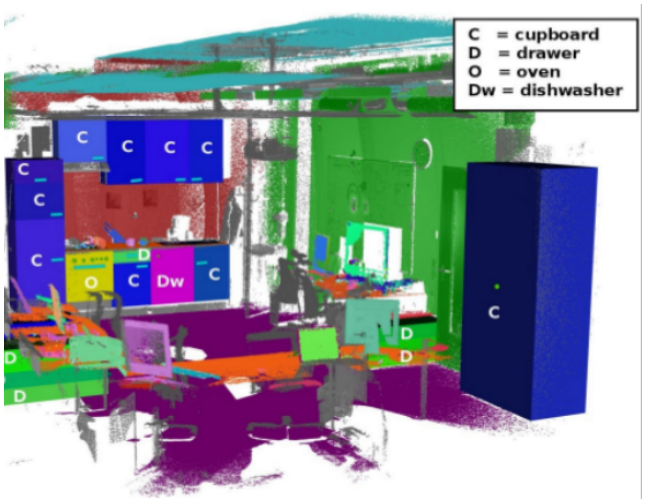


Figura 1.2: Criação de um mapa funcional.

Posto isto, o principal foco desta dissertação, assenta principalmente na resolução computacional de problemas relacionados com o processamento de nuvens de pontos adquiridas pela Kinect em ambientes *indoor* não estruturados e conceitos ligados ao registo de cada nuvem de pontos num único mapa global consistente. A resolução deste problema está intimamente relacionado com a avaliação efectuada à biblioteca PCL.

Os resultados obtidos nesta abordagem inicial são bastante promissores, perspectivando assim mais trabalhos futuros nesta área.

[3]imagem retirada de [Rus09]

1.3 Objectivos e Contribuições

Esta dissertação, tal como referido anteriormente, propõe-se a avaliação da biblioteca PCL para a resolução de problemas relacionados com o registo de nuvens de pontos num mapa global consistente e com a interpretação dos mesmos, com o intuito de se conseguir proceder à construção de mapas de ambientes *indoor* não estruturados, utilizando os dados adquiridos a partir da Kinect.

São várias as etapas a cumprir por forma a identificar correctamente os passos que levem à obtenção de resultados.

Os objectivos deste trabalho podem então ser resumidos nos seguintes pontos:

1. Identificar os módulos e métodos existentes na biblioteca PCL, relevantes para o objectivo deste trabalho.

   (a) Avaliar esses métodos, construindo simples aplicações que comparem métodos concorrentes e os testam, tendo em conta o seu custo computacional e a sua pertinência no contexto desta dissertação.

2. Definição de uma arquitectura de sistema que permita a obtenção de um mapa global consistente de um qualquer cenário *indoor* não estruturado.

   (a) Implementar e testar a arquitectura de sistema idealizada em vários cenários reais.

1.4 Estrutura do documento

Além deste capítulo, esta dissertação contempla mais outros 5 capítulos e 2 partes, estruturados da seguinte forma:

- Capítulo 2 - elabora-se o levantamento do estado da arte na área dos sistemas sensoriais existentes actualmente que promovem o mapeamento tridimensional do meio, do software que incorpora processamento de dados provenientes desses sistemas e trabalhos relacionados com o tema desta dissertação, o registo de nuvens de pontos e a semantização de cenas.

- Parte I - Análise do PCL

  ▷ Capítulo 3 - descreve-se de uma forma geral a biblioteca *Point Cloud Library*, elaborando sobre a sua estrutura e organização, bem como algumas das especificidades dos módulos que a compõe.

- ▷ Capítulo 4 - caracterizam-se alguns dos métodos existentes de alguns dos módulos que o PCL compreende, tendo em conta a sua relevância para o tema abordado nesta dissertação.

- Parte II - Mapeamento - testes e aplicações

  - ▷ Capítulo 5 - Descreve-se e avalia-se de uma forma prática, a solução encontrada para o registo de nuvens de pontos.

- Capítulo 6 - elabora-se o balanço do trabalho realizado face aos objectivos da dissertação, sendo mencionados alguns dos aspectos a abordar como trabalho futuro, bem como outras vertentes a explorar.

2

Estado da Arte

Conteúdo

Neste capítulo serão abordados os temas que enquadram a dissertação a realizar, nomeadamente conceitos básicos de sistemas de percepção tridimensional, exemplos deste tipo de sensores, exemplos de aplicações dos dispositivos em plataformas robóticas e o estado da arte dos processos de mapeamento e respectivos algoritmos e correspondentes estudos nesta área.

2.1 Conceitos básicos de aquisição de dados

No contexto dos sistemas de mapeamento orientados a plataformas robóticas existem três categorias de sensores que permitem a obtenção de dados tridimensionais: sensores que fazem uso das técnicas de triangulação, estimando a correspondência entre pontos obtidos no mesmo *timestamp*, entre dois sensores diferentes com conhecimento prévio dos seus parâmetros intrínsecos e extrínsecos; sensores baseados em sistemas *time-of-flight* que estimam o tempo que a luz ou som demora a percorrer o espaço desde a sua emissão até à sua recepção após ser reflectida por uma superfície; e, por último, sensores que utilizam o método de luz estruturada caracterizados por projectarem luz mediante um padrão conhecido, reconstruindo as superfícies tridimensionalmente de um cenário através da deformação do padrão de luz emitido.

Estas três categorias de sensores podem ainda ser divididos em dois tipos: sensores activos e sensores passivos. Sensores activos normalmente projectam algum tipo de luz ou som numa superfície lendo posteriormente a sua reflexão através de dispositivos receptores. Fazem parte deste tipo de sensores dispositivos como os sistemas LIDAR ou LMS, radares, câmaras *Time-of-Flight* (TOF), sonares ou sensores que usam luz estruturada.

Sensores passivos não emitem qualquer tipo de luz ou som no meio que os rodeiam, ao invés disso "lêem" a luz ambiente reflectida pelas superfícies de objectos. O processo denominado estereoscopia, em que duas câmaras fazem uso da técnica de triangulação (câmaras *stereo*), com um princípio de funcionamento muito parecido com o da visão humana, é um exemplo deste tipo de sensores.

2.1.1 Triangulação

Os sistemas sensoriais que fazem uso da técnica de triangulação estimam a distância (***d***) baseando-se na equação 2.1:

$$d = \frac{fL}{x} \tag{2.1}$$

em que $\boldsymbol{f}$ representa a distância focal dos dois sensores, $\boldsymbol{L}$ a distância entre os sensores (baseline) e $\boldsymbol{x}$ a variável correspondente à leitura do sinal com um dado desvio.

No caso do uso desta técnica recorrendo a câmaras *stereo* esta fórmula pode ser simplificada, tendo o aspecto da equação 2.2:

$$d = \frac{fL}{\|x_1 - x_2\|} \tag{2.2}$$

em que x_1 e x_2 representam os pontos correspondentes entre as duas câmaras.

2.1.2 Time-of-Flight

A estimação da distância por parte deste tipo de sensores, é efectuada, sabendo a que velocidade o sinal emitido é propagado e servindo-se de dispositivos de grande precisão na medição do tempo exacto que esse sinal demora entre a sua emissão e recepção. A distância $\boldsymbol{d}$ pode então ser estimada com recurso à equação 2.3.

$$d = \frac{\nu.t}{2} \tag{2.3}$$

em que $\boldsymbol{\nu}$ representa a velocidade do sinal emitido ($\nu = 343^{m}/_{s}$ para velocidade do som e $\nu = 299.792.458^{m}/_{s}$ para a velocidade da luz) e $\boldsymbol{t}$ o tempo despendido entre a emissão e recepção do sinal.

Como referido anteriormente, devido ao valor elevado da velocidade da luz, é necessário um dispositivo extremamente preciso. Por exemplo, para medir distâncias com precisão de 1 cm de precisão o tempo de medida deverá ser na ordem dos picosegundos (equação 2.4).

$$\Delta t = \frac{\Delta s}{c} = \frac{0,010m}{299.792.458^{m}/_{s}} = 33,4 \times 10^{-12}s \tag{2.4}$$

2.1.3 Luz estruturada

O método de luz estruturada tira partido de projecções de luz com um padrão conhecido, em que se obtém uma imagem tridimensional através da deformação do padrão emitido. De um sistema como este, fazem parte um projector e um sensor de imagem, como se pode verificar na figura 2.1[1].

Os padrões de luz emitidos por este género de dispositivos diferem quanto ao espectro de luz. Existiam até à bem pouco tempo, somente projectores que emitiam luz no espectro visível. Estes sofrem o problema de só serem eficazes em ambientes muito controlados em termos de iluminação.

[1] imagem retirada de [Tec]

Alternativamente, surgiram recentemente dispositivos que possibilitam a projecção de luz fora do espectro visível, como luz infravermelha, caracterizando-se por serem menos intrusivos e poderem operar em condições normais de luz ambiente.

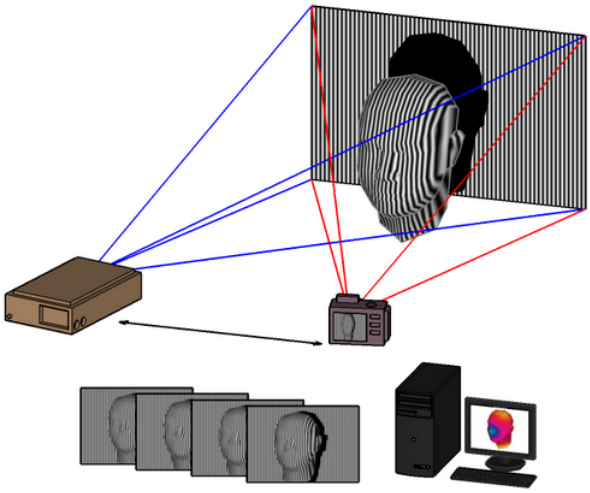


Figura 2.1: Princípio de funcionamento - luz estruturada.

Existem várias técnicas possíveis na emissão dos padrões de luz. No artigo [Gen11], estas técnicas estão classificadas e caracterizadas pormenorizadamente em duas categorias: múltiplas imagens e imagem única. Estas, estão ainda sub-divididas em cinco diferentes categorias: projecções sequenciais, variação contínua de padrões, indexação de linhas, indexação de grelhas e sistemas híbridos. Na figura 2.2[2] encontra-se a descrição resumida dessa classificação.

Para uma descrição mais pormenorizada deste tipo de tecnologia, sugere-se a consulta da referência acima assinalada.

2.2 Dispositivos de Aquisição de Informação Tridimensional

Nas últimas décadas tem-se vindo a assistir a um rápido desenvolvimento de vários equipamentos que visam a aquisição de informação tridimensional de um determinado cenário. Devido ao seu grande potencial na aplicação em plataformas robóticas, estas ferramentas foram rapidamente identificadas como um excelente método de percepção do mundo.

De seguida apresentam-se alguns dos equipamentos mais conhecidos e utilizados na área da robótica.

[2]imagem retirada de [Gen11]

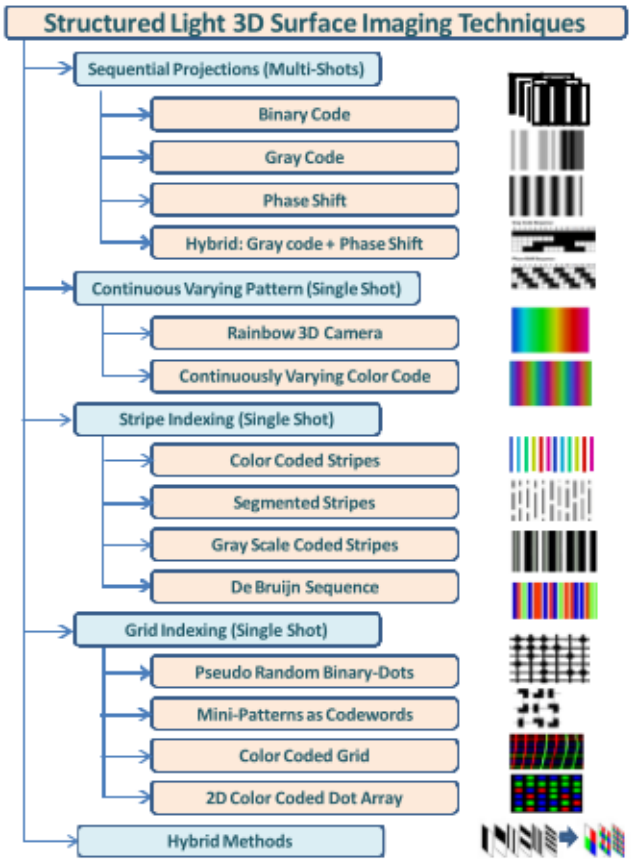


Figura 2.2: Classificação dos padrões de luz emitidos por projectores de luz estruturada.

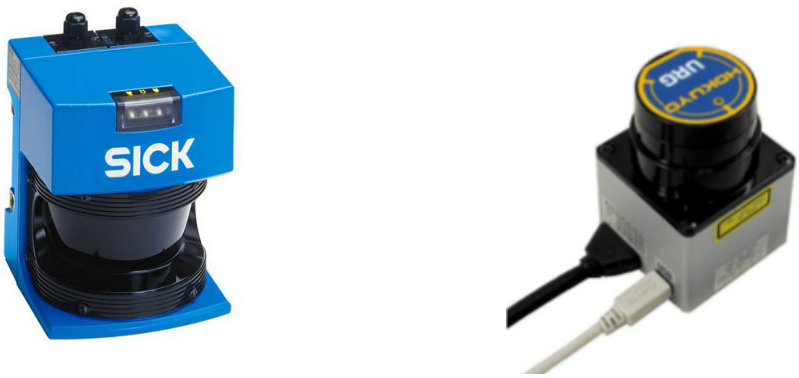
2.2.1 Laser Rangefinders (LIDAR)

Este tipo de equipamentos têm tido nos últimos anos, uma ampla integração em diversos projectos na área dos sistemas autónomos. Exemplos destes dispositivos são o: SICK LMS 200, o Hokuyo URG-04LX (figura 2.3[3]) e os Velodyne HDL-32E/HDL-64E (figura 2.4[4]).

O Sick LMS 200 é fabricado pelo grupo empresarial SICK[5] , tendo como características um alcance até 80 metros, ângulo de visão de 180^o com resolução angular de 0,5^o, aquisição de dados à taxa de 75 *scans* bidimensionais por segundo com uma resolução de 10 mm, 185x156x210 mm de dimensões externas, 4,5 kg de peso e um consumo estimado de 20 W.

Por seu lado, o Hokuyo URG-04LX, fabricado pela empresa Hokuyo[6], tem dimensões (50x50x70 mm) e peso bem mais reduzidas (0,16 kg), alcance máximo de 4,0 metros, ângulo de visão de 240^o com resolução angular de 0,36^o, aquisição de dados à taxa de 10 *scans* bidimensionais por segundo com uma resolução de 1 mm, e

[3]Imagens retiradas de [wll] e [Rob]
[4]Imagens retiradas de [Hiz]
[5]SICK - www.sick.com
[6]Hokuyo - www.hokuyo-aut.jp



(a) SICK LMS 200 (b) Hokuyo URG 04LX

Figura 2.3: SICK LMS 200 e Hokuyo URG 04 LX.

um consumo estimado de 4,0 W. Um estudo comparativo entre os dois dispositivos pode ser consultado em [KTCS09].

Normalmente, para obter mapas tridimensionais, este tipo de sistemas laser de aquisição de dados bidimensionais, são colocados em unidades rotativas ou em braços robóticos. Servindo-se da cinemática destas unidades, é possível auferir múltiplos varrimentos de planos 2D, possibilitando depois a conversão numa representação 3D consistente.

(a) Velodyne HDL 32E. (b) Velodyne HDL 64E.

Figura 2.4: Velodyne HDL 32E e Velodyne HDL 64E

O Velodyne HDL-64E e o HDL-32E (figura 2.4) foram especialmente desenvolvidos pela empresa Velodyne[7] para a sua integração em sistemas de navegação autónoma, estando por entre os dispositivos mais populares.

O Velodyne HDL-64E tem um vasto campo de visão de 360º HFOV (Horizontal Field of View) e 26,8º VFOV (Vertical Field of View) com uma taxa de rotação de 5 ou 15 Hz seleccionável pelo utilizador, aquisição de dados à taxa de 1,3 milhões de pontos por segundo, um alcance entre 50 a 120 metros dependente da reflectividade dos obstáculos e uma precisão <2 cm.

[7]Velodyne - velodynelidar.com/lidar/lidar.aspx

O Velodyne HDL-32E difere do anterior equipamento descrito, pelo seu menor tamanho e estrutura robusta, como também na qualidade das suas características. Tem um campo de visão horizontal de 360^o e $40{,}0^o$ de campo de visão vertical com uma taxa de rotação de 10 Hz, taxa de aquisição de dados de 700 mil pontos por segundo e um alcance de 70 metros com uma precisão de ± 2 cm. Na figura 2.5[8] é apresentado um exemplo de uma nuvem de pontos de uma área urbana adquirida por um destes dispositivos.

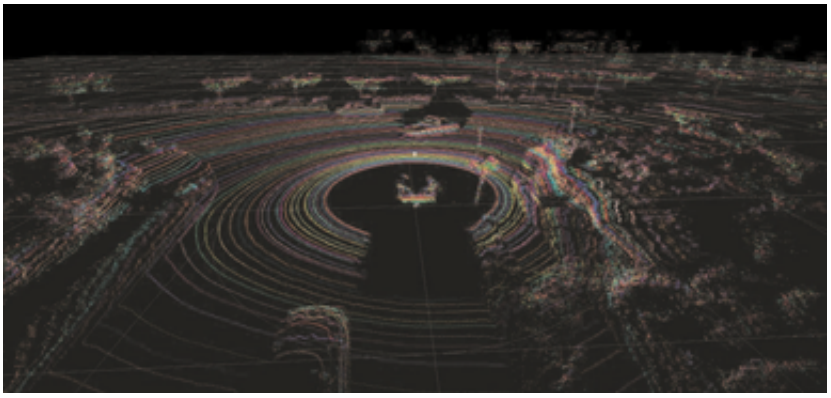


Figura 2.5: Nuvem de pontos recolhida pelo Velodyne de uma área urbana.

Em sistemas como o Hokuyo e o Sick, a principal desvantagem é a sua baixa taxa de aquisição de dados e a necessidade de construção de unidades rotativas que permitam acompanhar essa taxa com velocidades similares por forma a obter nuvens de pontos 3D.

Os Velodyne, por seu lado, distinguem-se da concorrência pelo facto de terem vários emissores e receptores laser, em vez de um laser com espelhos rotativos. Permitem também recolher um significativo número maior de pontos num espaço limitado de tempo.

2.2.2 Câmaras *Time-of-Flight*

Uma câmara *Time-of-Flight* caracteriza-se por ser um dispositivo de medição de profundidade que calcula a distância quantificando variações que um sinal de luz emitido encontra quando reflecte em superfícies presentes num cenário. Com este tipo de sistemas é possível auferir a aquisição de dados à taxas até 30 Hz, permitindo a sua utilização em aplicações que necessitam destas altas frequências de captura de dados.

Um exemplo destas câmaras é o SwissRanger 4000 (figura 2.6[9]), fabricado pela empresa Mesa Imaging[10] , que oferece uma frequência de aquisição de dados de 30

[8]Imagem retirada de [Col]
[9]Imagem retirada de [Vena]
[10]Mesa Imaging - www.mesa-imaging.ch

Hz, tem um alcance entre 0,1 e 10,0 m, precisão entre 10 e 15 mm, dimensões de 65x65x68 mm e um peso de 0,470 kg.

Figura 2.6: SwissRanger SR 4000.

Em relação a dispositivos como o SICK e o Hokuyo apresentado na secção 2.2.1, este tipo de câmaras têm a vantagem de conseguirem atingir taxas de aquisição de dados significativamente maiores e de dispensarem posterior processamento dos pontos obtidos em cada plano 2D por forma a obter uma mapa 3D consistente. Ainda assim, têm algumas desvantagens, particularmente, no maior ruído verificado na amostragem, menor densidade de pontos e muito maior propensão a sofrerem efeitos de *veiling* [MDH+08].

2.2.3 Câmaras RGB-D

As câmaras RGB-D servem-se da tecnologia de luz estruturada, descrita na secção 2.1.3, para a obtenção de mapas 3D de um cenário. Um dos exemplos deste tipo de equipamentos é o usado na realização desta dissertação, a Kinect (figura 2.7[11]), construída com base na tecnologia desenvolvida pela PrimeSense, e distribuída pela Microsoft para a XBox 360.

(a) Microsoft Kinect. (b) Interior da Kinect.

Figura 2.7: Microsoft Kinect

A Kinect possui 4 microfones, 2 câmaras (uma RGB com resolução de 1600x1200 e outra de infravermelhos com resolução de 640x480), um projector de luz estruturada no espectro infravermelho, uma memória DDR2 SDRAM da Hynix de 64 MB,

[11] imagens retiradas de [Kit] e [Venb]

um motor de *tilt*, um acelerómetro de 3 eixos e um chip PS1080-A2 desenvolvido pela PrimeSense que processa a informação adquirida antes de a Kinect transmitir o *output* desta para o dispositivo com o qual está conectada, uma XBox ou um computador. Tendo um alcance entre 0,40 e 8 metros consegue, a distâncias pequenas, uma precisão de 3 mm, sendo que esse valor vai crescendo consoante a distância da superfície observada.

2.3 Aplicações em Plataformas Robóticas

São vários os sistema robóticos, nos quais foram integrados os mais variados dispositivos de mapeamento.

Nesta secção tenta-se demonstrar vários tipos de plataformas, com diversos objectivos finais e design, para que se perceba o impacto que este tipo de tecnologia tem na área da robótica.

2.3.1 Kurt3D

O Kurt 3D (figura 2.8[12]) é uma plataforma robótica semi-autónoma, desenvolvido no *Institute of Computer Science* da universidade de Osnabruck na Alemanha [Nü09], construído com objectivo de desempenhar operações de busca e salvamento e de vigilância em ambientes urbanos.

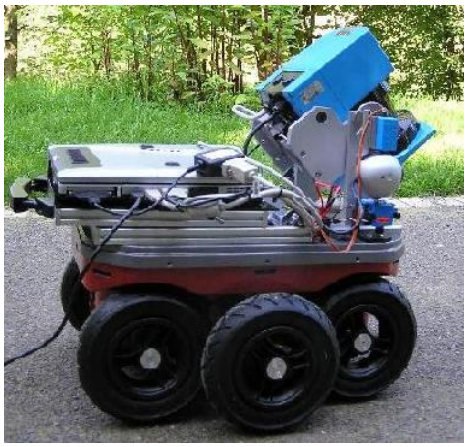


Figura 2.8: Kurt3D.

Este projecto tem três objectivos a cumprir:

6D SPLAM construção de um mapa, tendo em conta que 3 tarefas devem correr num ciclo permanente:

[12]Imagem retirada de [oCS]

- Planeamento e deslocamento para o próximo ponto de aquisição de dados;
- Localização da plataforma no mapa já construído;
- Aquisição e registo de um novo conjunto de dados consistente com o modelo prévio.

Busca e Salvamento dotar o robô de capacidades para navegação em ambientes de busca e salvamento.

Cooperação de Robôs construir mapas 3D de forma eficiente com a colaboração de vários robôs.

2.3.2 Stanley

O Slanley [TMD+06] é uma plataforma robótica, baseado no Volkswagen Touareg R5 (figura 2.9[13]), desenvolvida pela Universidade de Standford para participar no desafio DARPA (*Defense Advanced Research Projects Agency*). Em 2005 o Stanley venceu a prova disputada no deserto de Mohave na Califórnia, percorrendo 132 milhas em menos de 7 horas sem qualquer intervenção humana.

(a) Stanley. (b) Mapa criado pelo Stanley.

Figura 2.9: Stanley e nuvem de pontos gerada representando o mapa do mundo

O controlo de aceleração e do travão é executado via interface electrónico. Um motor DC acoplado à coluna de direcção promove o controlo da movimentação do veículo. O Stanley tem ainda um GPS, um radar, e 5 laser *rangefinders* colocados no tejadilho, dispostos com diferentes ângulos de inclinação para cobrir toda a frente do veículo, tendo um alcance de 25 metros. A informação proveniente destes sensores é fundida para construir mapas tridimensionais do mundo, sendo utilizada na navegação autónoma do veículo. A trajectória é gerada em função do ponto de destino, dos obstáculos detectados e do relevo do terreno.

[13]Imagens retiradas de [TMD+06]

2.3.3 PR2

O Personal Robot 2 (PR2)[14] (figura 2.10[15]) é um robô, desenvolvido pela Willow Garage, desenhado para ser um robô pessoal em ambientes em que haja interacção com seres humanos. Tem a capacidade de navegar nesses ambientes e destreza para alcançar e manipular objectos que lá se encontrem. O seu hardware e software é aberto à comunidade integrando o ROS e os pacotes disponíveis para este sistema operativo. No entanto, o preço cobrado pela plataforma completa é muito alto, rondando os 310.000 euros.

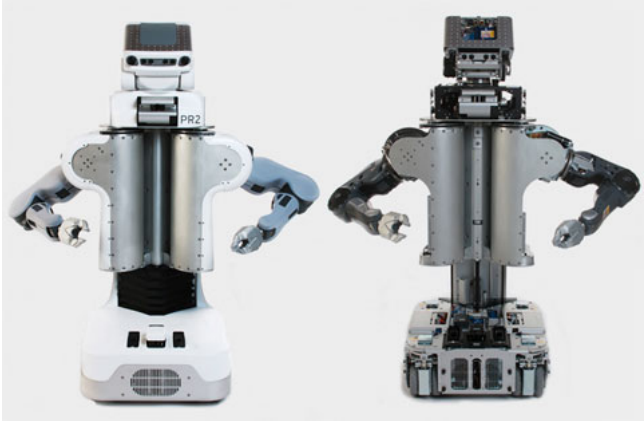


Figura 2.10: PR2.

Em termos de poder de processamento, integra: 2 servidores com processadores Quad-Core i7 Xeon, 24 GB de memória, um disco externo removível com 1,5 TB e um disco interno com 500 GB. Para se movimentar utiliza uma base omnidirectional com 4 rodízios. Para comunicações possui gigabit Ethernet, rede *EtherCAT* e *wireless* incluindo WiFi e Bluetooth. Como sistema de energia inclui um *pack* de baterias Lion de 1.3 kWh com autonomia de 2 horas em normal funcionamento do robô, podendo ser carregadas enquanto são executadas tarefas. Incorpora também, 2 braços robóticos com 4 graus de liberdade tendo cada um deles um pulso com 3 graus de liberdade e uma pinça com 1 grau de liberdade. Integra também uma grande panóplia de sensores, posicionados em diversas partes do robô:

- cabeça - uma Microsoft Kinect, uma câmara color gigabit Ethernet de 5 Megapixel, um sistema *stereo* de 2 câmaras *color* Ethernet para tarefas de visualização do ambiente que rodeia o robô, um sistema stereo de 2 câmaras monocromáticas Ethernet para tarefas de manipulação de objectos;
- zona dos ombros - um Hokuyo UTM-30LX e um IMU Microstrain 3DM-GX2;
- antebraço - uma câmara Ethernet;

[14] PR2 - www.willowgarage.com/pages/pr2/
[15] Imagem retirada de [NYT]

- pinça - um acelerómetro de 3 eixos, um conjunto de sensores de pressão posicionados nas pontas da pinça e um LED de calibração;
- base - um Hokuyo UTM-30LX.

2.3.4 TurtleBot

O Turtlebot[16] (figura 2.11[17]) é uma plataforma robótica de baixo custo desenvolvida pela Willow Garage, com o objectivo de fornecer uma plataforma autónoma, por forma a executar principalmente, tarefas de mapeamento e navegação.

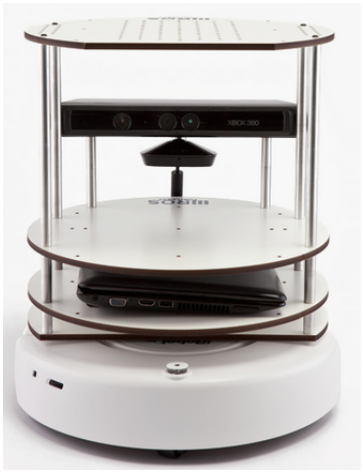


Figura 2.11: Turtlebot.

O robô traz com ele um *software open-source*, o Turtlebot SDK, com capacidade de ser utilizado em tarefas de mapeamento e navegação. Em termos de Hardware, o Turtlebot integra a Microsoft Kinect, um IRobot Create, um *pack* de baterias de 3000 mAh Ni-MH e um giroscópio de eixo único com 150 graus/segundo. A encomenda do robô poderá também integrar um *laptop* ASUS 1215N com um processador Intel® Atom™ D525 Dual Core, 2 GB de memória, uma placa gráfica NVIDIA® ION™ e um disco rígido de 250 GB.

2.4 Software orientado ao processamento de nuvens de pontos

No mercado existem poucos softwares orientados especificamente ao processamento de nuvens de pontos adquiridos pelos diversos equipamentos apresentados na secção 2.2. O alto nível de conhecimento necessário para desenvolver tais ferramentas

[16] Turtlebot - www.willowgarage.com/turtlebot
[17] Imagem retirada de [Gar]

e o inerente circuito fechado das indústrias que desenvolvem este tipo de software dificultou o amplo envolvimento que o PCL trouxe à comunidade robótica.

Exemplos de frameworks que incorporam métodos de processamento de nuvem de pontos são o PolyWorks Modeler^{TM}, desenvolvida pela empresa Innovmetric[18] e o RapidForm SDK desenvolvido pela empresa Rapidform[19].

O software PolyWorks contém ferramentas de modelação 3D e permite também processar grandes conjuntos de dados decorrentes de nuvens de pontos em modelos poligonais e superfícies NURBS. O Rapidform para além de conter ferramentas de modelação 3D, permite também a obtenção de dados provenientes de dispositivos de aquisição 3D, incluido a Kinect e o seu processamento através de alguns algoritmos incorporados no SDK.

Para além do PCL existem ainda outras bibliotecas *open source* que estão relacionadas com a percepção e processamento 3D, contendo algoritmos de filtragem, registo e segmentação, como as bibliotecas: 6DSLAM[20], MRPT[21], IVT[22], Fair[23] ou ITK[24].

Um estudo mais detalhado sobre o estado da arte deste tipo de ferramentas *open source* de processamento 3D, pode ser encontrado no artigo [BPFN11].

Por seu lado, a biblioteca PCL tem como objectivo incorporar todos os blocos de construção comuns que uma aplicação 3D necessita, lidando com os dados das nuvens de pontos adquiridos a partir de dispositivos de percepção 3D, incluindo scanners laser, câmaras stereo, câmaras RGB-D, câmaras ToF, etc.

Sendo uma ferramenta de livre uso, tem a grande vantagem de toda uma comunidade poder contribuir para o seu desenvolvimento, tendo já, num espaço de tempo muito limitado, crescido exponencialmente quer no número de utilizadores quer no seu próprio crescimento, com a incorporação de novos algoritmos. No Capítulo 3 é feita uma caracterização mais detalhada das potencialidades desta *framework*, deixando para essa parte da dissertação uma análise mais aprofundada.

[18] Innovmetric - www.innovmetric.com
[19] Rapidform - www.rapidform.com
[20] 6DSLAM - slam6d.sourceforge.net
[21] MRPT - www.mrpt.org
[22] IVT - ivt.sourceforge.net
[23] Fair - sourceforge.net/projects/openvolksbot
[24] ITK - www.itk.org

2.5 Algoritmia orientada à construção de mapas tridimensionais

A criação de mapas, tem sido um dos desafios mais importantes na história investigação da robótica e computação visual. Elfes [Elf89] foi pioneiro na área da construção de mapas ambiente, equipando os robôs de meios que lhes permitissem localizar, ou mover da sua posição posição actual até à posição de destino, seguindo uma trajectória eficiente.

2.5.1 Mapeamento

Ao longo dos anos têm sido desenvolvidas técnicas de mapeamento tridimensional através de laser range finders [LM97, TBF00, TB05, CN06, NSS+09, Hol09], câmaras ToF [MDH+09, MDH+08, PMR+08], câmaras stereo [AmFM+06, BRSM+09], câmaras monoculares [CDR+07], sensores que utilizam luz estruturada [SS03] e mesmo conjuntos de imagens desorganizadas [SSS06, FCSS09].

Mais recentemente, foram lançados no mercado sensores RGB-D de baixo custo, dos quais o mais importante é a Kinect da Microsoft. Este tipo de sensores fornecem tanto informação de cor como também mapas densos de profundidade a altas taxas de aquisição.

Com base nestes sensores, foram recentemente desenvolvidos trabalhos relacionados com técnicas de mapeamento, tendo o primeiro dos quais sido publicado por Henry *et al.* [HKH+10] em 2010. Outros trabalhos seguiram-se como os realizados por Engelhard *et al.* [EEH+11], Sueckler *et al.* [SB12], Enders *et al.* [EHE+12a]. Alguns destes sistemas, relacionados principalmente com a problemática do SLAM, foram avaliados por Endres *et al.* [EHE+12b].

No que concerne ao particular conceito de registo de nuvens de pontos, um dos algoritmos mais conhecido é o *Iteractive Closest Points* (ICP) [BM92, Zha94]. Uma avaliação ao ICP e às suas variações produzidas ao longo dos anos, pode ser consultada em [SMFF07].

Para debelar algumas das desvantagens do ICP foram produzidos alguns algoritmos no sentido de se produzir um alinhamento inicial prévio ao processamento ao ICP. Os mais importantes foram apresentados em [RBMB08] e [RBB09]. Mais recentemente foi apresentado o algoritmo *Normal Distribution Transform* (3D-NDT) [MLD07] e a sua extensão Color-NDT [HMSL08], para o registo de nuvens de pontos em grande escala.

Um dos estudos mais importantes e citados nesta área do mapeamento, foi realizado em 2002, por Sebastian Thrun [Thr02]. Aí é realizado um estudo compreensivo

da tecnologia e algoritmia associada ao mapeamento robótico, dando especial atenção ao mapeamento em ambientes *indoor*, estudo esse, que veio depois culminar com o desenvolvimento do Stanley [TMD+06].

Andreas Nüchter em [Nü09], aborda o problema do mapeamento e localização definido como SLAM para plataformas robóticas móveis nos seus 6 graus de liberdade (x, y, z, $roll$, $pitch$ e yaw). Apresenta e descreve uma grande variedade de princípios básicos e algoritmos associados aos problemas de registo de *scans* 3D num modelo global consistente e descrição semântica do mundo, bem como uma grande quantidade de resultados práticos em aplicações reais de mapas produzidos pelo Kurt3D e os sistemas de percepção nele integrados.

Um dos trabalhos mais recentes e completos produzidos na área da construção de mapas, foi realizado por Radu Bogdan Rusu na sua tese de doutoramento [Rus09]. São abordados de forma bastante exaustiva todos os prévios e novos processos e algoritmos que levam ao registo de nuvens de pontos. Mais concretamente são abordados os temas e conceitos:

- aquisição de dados através de vários tipos de dispositivos de percepção;
- representação de mapas;
- arquitectura de sistema genérica que leve ao mapeamento de ambientes *indoor* (particularmente cozinhas [RMB+08a]);
- vizinhança de pontos;
- remoção de *outliers*;
- estimação de normais das superfícies;
- descrição geométrica de pontos - *Point Feature Histograms* (PFH) [RBMB08, RMBB08] e *Fast Point Feature Histograms* (FPFH) [RBB09];
- registo de nuvens de pontos, através da análise da persistência de *features*[25] em nuvens consecutivas [RBB09];
- suavização de superfícies [RBM+07, RMB+08a].

2.5.2 Mapeamento Semântico

A percepção da geometria das estruturas do meio envolvente é um pré-requisito para a inserção de robôs em ambientes em que humanos vivem. Este ambientes

[25] designação utilizada ao longo da dissertação

tendem a ser desorganizados e altamente dinâmicos. O primeiro necessita da informação acerca das estruturas e objectos presentes, enquanto que no segundo caso é necessário processar a informação espacial adquirida em tempo real.

Um dos trabalhos mais importantes na vertente do mapeamento semântico foi o produzido por Nüchter *et al.* [NWL+05], onde é apresentado um processo de mapeamento baseado em informação semântica do meio ambiente. Este processo elaborado com recurso ao robô Kurt3D, baseia-se na obtenção de correspondências em superfícies cujos atributos são extraídos e incorporados em algoritmos de busca (*trees*) no sentido dos dados serem associados e produzir correspondências.

Na área do reconhecimento e classificação de objectos [HSD+10] e [ZSKW10] são os trabalhos mais importantes.

No que concerne a ambientes dinâmicos podem-se referir trabalhos importantes como [HLS08], [RBM+09], [OJAC11] ou [APP+12].

Para além dos conceitos já referidos, Radu Bogdan Rusu na sua tese de doutoramento [Rus09], aborda e sugere novos conceitos e algoritmos relacionados com a interpretação de cenários, tais como:

- segmentação e simplificação de conjuntos de pontos através de modelos geométricos;
- técnicas de *clustering*;
- identificação de extremidades em dados 3D;
- segmentação via *Region Growing* [RMB+08b];
- ajuste a modelos paramétricos específicos (planos) [RMB+08b, RMB+08a];
- utilização de regras heurísticas na interpretação de cenários *indoor* estáticos;
- rápida catalogação geométrica de pontos utilizando *Conditional Random Fields* (CRF) [RHBB09];
- classificação de objectos em classes [RHBB09] através da utilização de *Support Vector Machines* (SVM) [BGV92, CV95];
- descritor geométrico global Global FPFH (GFPFH) para catalogação de *clusters* [RBM+09];
- obtenção de modelos híbridos contendo coeficientes das formas e *meshes* triangulares das superfícies [MRB09, RBMB09].

Para além destes novos conceitos, são também apresentados os resultados da aplicação destes conceitos em cenários reais, tais como:

- criação de mapas de colisão em cenários que promovam replaneamento de trajectórias, em particular na execução da tarefa de limpeza de uma mesa, com a utilização do robô PR2 [RSG+09];

- identificação de portas e respectivos puxadores ou maçanetas, com recurso ao PR2 [RMCB09];

- criação de mapas semânticos através de câmaras *stereo* [BRSM+09, MRS+09, RSM+08], utilizando o robô RHex (Robotic Hexapod) [SBK01].

Este trabalho foi continuado, e publicações mais recentes relacionadas com os algoritmos apresentados podem ser encontrados em [BRJ+11], que apresenta soluções para identificar objectos distintos com a actuais capacidades do PR2 e um novo sistema hierárquico baseado em máquinas de estado concorrenciais.

Muja *et al.* [MRBL11], apresenta uma infraestrutura denominada *REcnognition INfrastruture* (REIN), que promove a rápida identificação e classificação de objectos e a sua posição com recurso ao algoritmo *Viewpoint Feature Histograms* (VFH) [RBTH10].

Em [SRKB11], é apresentado um novo algoritmo denominado *Normal Aligned Radial Features* (NARF) [SRKB10] para a extracção de *features* atendendo aos limites dos objectos.

No que diz respeito a trabalhos produzidos especificamente com câmaras RGB-D para o particular desafio de interpretação de cenas, existem poucas referências. No entanto, trabalhos muito recentes podem ser encontrados em [HHRB11], [HRF11], [BLRF11], [BRF11], [LBRF11], [RBF12], [RMP+12], [BRF12] e [SBB12].

As publicações [HRF11, BLRF11, BRF11, BRF12] dizem respeito a processos de identificação de objectos. As publicações [LBRF11] e [RBF12] estão relacionados com a classificação de objectos.

Holz *et al.* [HHRB11], promove a rápida segmentação de planos através de dados provenientes da Kinect. Rishtsfeld *et al.* [RMP+12] apresenta uma *framework* para detectar objectos 3D desconhecidos e extrair representações apropriadas para tarefas de manipulação de objectos. Struckler *et al.* [SBB12] apresenta um processo de mapeamento através da segmentação de classes de objectos.

2.6 Discussão crítica

Finda a apresentação do estado da arte em termos de hardware, software e algoritmia relacionada com o tema desta dissertação, faz-se então uma discussão crítica sobre as escolhas críticas para o projecto realizado.

Em relação à escolha do hardware, optou-se pela utilização da Kinect para a aquisição de dados tridimensionais do mundo. Um dos factores que levou a esta escolha é o de as câmaras RGB-D terem conhecido um grande desenvolvimento num curto período de tempo e de se perspectivar ainda melhorias significativas num futuro próximo. Apesar das limitações conhecidas em termos de alcance e sensibilidade a fontes de iluminação externas e algumas superfícies, o seu preço em relação a outros dispositivos como os que foram apresentados na secção 2.2, é bastante mais baixo (consultar tabela 2.1), o que torna muito apetecível a sua integração em projectos que visam a produção massiva de robôs autónomos.

Tabela 2.1: Preços de alguns dispositivos de aquisição tridimensional disponíveis no mercado.

Dispositivos	*preço (euros €)*
Velodyne HDL 64E	$\pm 58,200.00$
Velodyne HDL 32E	$\pm 23,200.00$
Sick LMS 200	$\pm 3,720.00$
SwissRanger sr4000	$\pm 3,330.00$
Hokuyo URG-04LX	$\pm 1,840.00$
Microsoft Kinect	± 85.00

Em relação ao software, a escolha da biblioteca PCL foi uma escolha óbvia. Algumas das suas grandes vantagens são a de ser um software de livre usufruto, ser muito orientada à integração em plataformas robóticas que tenham no seu sistema dispositivos de aquisição tridimensional e a de estar em constante crescimento com a inclusão de novos algoritmos. A facilidade de implementação, o apoio prestado pela comunidade e o interesse demonstrado por grandes empresas e universidades potencia o seu desenvolvimento perspectivando-se um grande potencial futuro.

Em termos de algoritmia associada ao processamento de nuvens de pontos, esta dissertação tem como base o trabalho desenvolvido para a tese de doutoramento de Radu Bogdan Rusu [Rus09], que para além de ter descritos alguns dos mais recentes desenvolvimentos nesta área, é também uma das principais pessoas responsáveis pela criação e desenvolvimento da biblioteca PCL.

Parte I

Análise do PCL

3

Introdução ao PCL

Conteúdo

Com o advento da Kinect, uma nova câmara RGB-D de baixo custo, e os esforços continuados no avanço do processamento de nuvens de pontos, a percepção 3D ganha cada vez mais importância no mundo da robótica, bem como noutras áreas como por exemplo no mundo dos jogos, medicina e plataformas multimédia. Neste capítulo pretende-se descrever a estrutura e organização do PCL[1].

3.1 O que é o PCL?

A biblioteca **Point Cloud Library** (figura 3.1[2]) é o resultado do avanço tecnológico e do grande interesse na área da percepção e modelação 3D por parte da comunidade robótica, tendo sido apresentada pela primeira vez em Abril de 2011 por Radu

[1] PCL - pointclouds.org

[2] Imagem retirada de [Liba]

Figura 3.1: Logo da biblioteca Point Cloud Library (PCL).

Bogdan Rusu e Steve Cousins na conferência ICRA2011 [RC11]. Estes são colaboradores na empresa de investigação e desenvolvimento Willow Garage, já de si muito conhecida por desenvolver o Robot Operating System[3] (ROS) e o robô pessoal PR2, e suportar a biblioteca Open Source Computer Vision[4] (OpenCV).

O PCL é um *software open-source*, licenciado pela *Berkley Software Distribuition* (BSD), utilizado para o processamento 2D e 3D de imagens e nuvens de pontos.

O seu objectivo é fornecer suporte para todos os blocos de construção comuns que uma aplicação 3D necessita, lidando com os dados das nuvens de pontos adquiridos a partir de dispositivos de percepção 3D, incluindo *scanners laser*, câmaras *stereo*, câmaras RGB-D ou câmaras *Time-of-Flight* (ToF).

Estando ainda numa fase de desenvolvimento inicial, podem-se já encontrar diversos projectos da comunidade robótica em que as suas características são aproveitadas.

Tratando-se de uma biblioteca multi-plataforma, existem distribuições disponíveis para Linux (Ubuntu, Arch, Fedora, etc.), Windows, MacOS e ainda Android. Está ainda, completamente integrada no ROS.

3.2 Estrutura e organização

Todas as classes e métodos implementados no PCL foram criados usando *templates*, ou classes e métodos genéricos, tornando-a numa biblioteca moderna, escrita em **C++**.

Como dependências (na versão 1.6), o PCL tem diversas outras bibliotecas como o *Visualization ToolKit*[5] 5.8 (VTK) [SML06], o Eigen3[6], o Boost[7] 1.49.0, o *Open Natural Interaction*[8] 1.3.2. (OpenNI), o *Open Multi-Processing*[9] (OpenMP 3.1) e o *Fast Library for Approximate Nearest Neighbors* 1.7.1 (FLANN) [ML09].

[3]ROS - www.willowgarage.com/pages/software/ros-platform
[4]OpenCV - opencv.org
[5]VTK - www.vtk.org
[6]Eigen - eigen.tuxfamily.org
[7]Boost - www.boost.org
[8]OpenNI - openni.org
[9]OpenMP - openmp.org

A maioria das operações matemáticas estão implementadas com recurso à biblioteca Eigen, que se trata de uma biblioteca *open-source* que fornece suporte a operações de álgebra linear.

As operações necessárias para a procura rápida dos vizinhos mais próximos, é fornecida pela biblioteca FLANN.

A biblioteca OpenNI permite interligação (*driver*) entre os dispositivos com base na tecnologia criada pela PrimeSense, como a Kinect, para que seja possível adquirir e interpretar os dados adquiridos pela câmara RGB-D.

A biblioteca VTK fornece a componente de visualização, garantindo suporte multi-plataforma para a visualização de nuvens de pontos 3D, superfícies volumétricas, etc. Todos os módulos e algoritmos do PCL, passam dados entre si servindo-se de apontadores partilhados gerados com recurso à biblioteca Boost, evitando assim a necessidade de voltar a copiar dados, previamente criados na aplicação.

Além disso, o PCL utiliza a biblioteca OpenMP e possibilita a integração da biblioteca *Intel Threading Building Blocks* (TBB) [Rei07], para tarefas de paralelização computacional, em processadores *multi-core*.

O PCL está estruturado de forma a incorporar uma grande variedade de algoritmos orientados ao processamento tridimensional, operando sobre uma ou colecções de dados representando nuvens de pontos, incluindo: operações de entrada e saída de dados, filtragem, detecção de pontos de interesse, estimação da geometria local e global, segmentação, registo, reconstrução de superfícies, etc..

Cada conjunto de algoritmos é definido via o uso de classes base que tentam incorporar todas as funcionalidades comuns aplicadas por todo o *pipeline*, mantendo assim as implementações dos algoritmos limpas e compactas. O interface básico para esse *pipeline* de processamento no PCL, ou seja, a forma genérica como cada classe, com respeito a um qualquer algoritmo, é codificada e organizada para posterior utilização, é descrito da seguinte forma:

1. criação de um objecto de processamento (por exemplo: de segmentação, filtragem, etc.).

2. uso do método ***setInputCloud()***, correspondente ao objecto da classe criado, em que é passada a nuvem de pontos que servirá de entrada ao algoritmo.

3. configuração de parâmetros.

4. evocação do método ***compute()*** (ou ***segment()***, ***filter()***, ***process()***, etc.) de forma a obter a saída do algoritmo implementado.

Na figura 3.2, é ilustrada uma possível estrutura para a estimação da geometria local de um ponto, exemplificando um possível *pipeline* para a obtenção desse

objectivo e como é que é efectuada a passagem de dados entre as suas classes. A sequência exemplifica o processo de estimação da geometria local em duas etapas: o objecto ***NormalEstimation*** é primeiro instanciado, sendo passada como entrada uma nuvem de pontos; o resultado do processamento é passado juntamente com a nuvem original para o objecto estimador ***FPFHEstimation***.

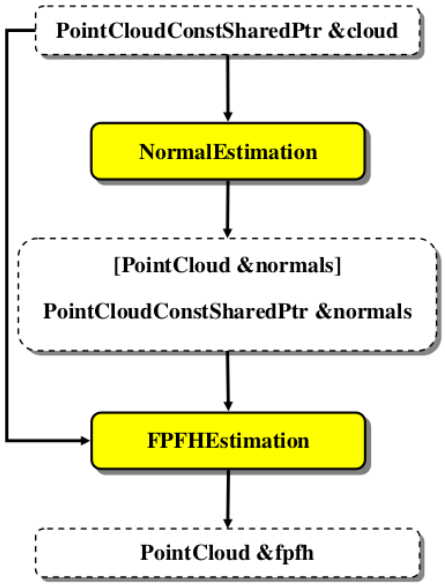


Figura 3.2: Exemplo do pipeline de implementação do PCL para a estimação de features *Fast Point Feature Histograms* (FPFH).

3.2.1 Blocos constituintes do PCL

Para conferir uma melhor organização e futuro desenvolvimento, o PCL está dividido numa série de módulos, ou pequenas bibliotecas, onde os algoritmos e classes correspondentes estão divididos por tópicos, podendo inclusive ser compilados separadamente (figura 3.3).

Esses módulos, na última versão do PCL (1.6), estão divididos da seguinte forma:

- **Módulo *io*** - contém ferramentas de leitura e escrita de dados (ficheiros do tipo *.pcd, nativos do PCL, ficheiros *.ply ou *.vtk, nativos da biblioteca VTK) e um interface genérico que permite acesso a diferentes dispositivos e respectivos *drivers*. Dos dispositivos suportados nativamente, podem-se referir as câmaras distribuídas pela PrimeSense[10], a Kinect da Microsoft[11] e a XTionPRO da Asus[12] (figura 3.4[13]).

[10]PrimeSense Sensor - www.primesense.com/en/solutions/solsensor
[11]Kinect - www.xbox.com/en-US/kinect
[12]XitionPro - http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO/
[13]Imagem retirada de [Libc]

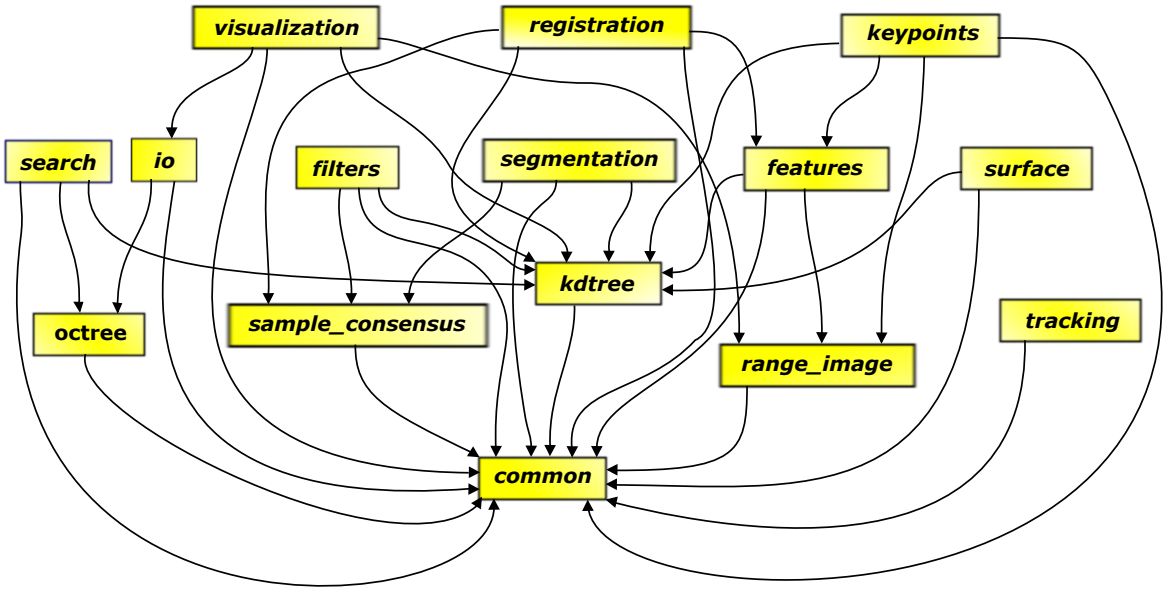


Figura 3.3: Exemplificação da organização modular do PCL e interdependência entre os diversos módulos.

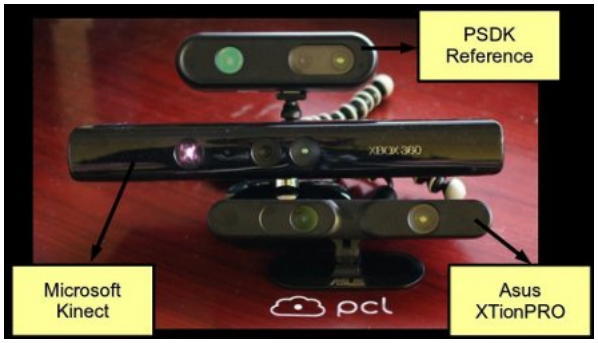


Figura 3.4: Dispositivos cuja captura de dados foi incorporada pelo PCL: sensor da PrimeSense (em cima), Kinect da Microsoft (no meio), XTionPRO da Asus (em baixo).

- **Módulo *common*** - biblioteca base do PCL, que compreende as estruturas de dados comuns e métodos usados pela maioria dos módulos integrantes do PCL. O núcleo das estruturas de dados incluem a classe **PointCloud** e uma grande variedade de tipos de estruturas de dados, usados na representação de pontos, normais das superfícies, valores de cor RGB, descritores de *features*, etc.. Contêm também várias funções que permitem a computação da distância/norma, média e covariância, ângulos, transformações geométricas, etc..

- **Módulo *visualization*** - biblioteca de visualização baseada no VTK, que fornece suporte multi-plataforma para a renderização de nuvens de pontos e superfícies tridimensionais. Encerra métodos para renderização e configuração de propriedades de visualização (cor, tamanho dos pontos, opacidade, etc),

métodos para desenhar primitivas e formas geométricas na janela de visualização (esferas, cilindros, linhas, etc) a partir de conjuntos de pontos ou de equações paramétricas e um módulo para a visualização de histogramas.

- **Módulo *search*** - compreende os métodos de procura dos vizinhos mais próximos utilizando diferentes tipo de estruturas de dados, incluindo: *kdtrees*[14] [AMN+98, SAH08] (através da biblioteca libpcl_kdtree, que por sua vez depende da biblioteca FLANN), *octrees* (através da biblioteca libpcl_octree), *brute force* e procura especializada para *datasets* organizados. Na figura 3.5[15] é exemplificada a decomposição espacial de um objecto através de uma *kdtree*.

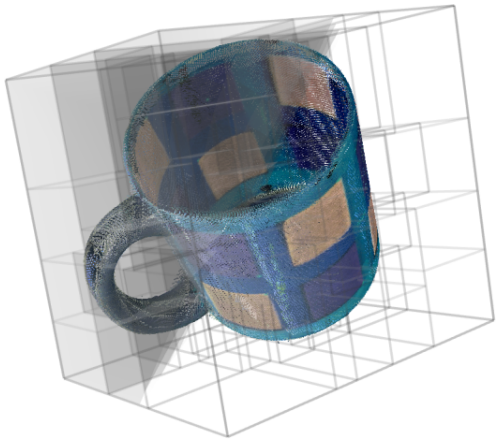


Figura 3.5: Exemplo da decomposição espacial através de uma kd-tree.

- **Módulo *octree*** - fornece métodos eficientes para a criação de estrutura de dados sob a forma de árvores hierárquicas contendo dados de nuvens de pontos, possibilitando o particionamento espacial, *downsampling* e operações de procura. A figura 3.6 ilustra as *bounding boxes* dos nodos de uma *octree*[16] no nível mais baixo da sua árvore, estando os voxeis (azul) a cercar um ponto 3D de uma superfície (vermelho).

- **Módulo *filters*** - contém mecanismos para remoção de *outliers*, redução e remoção de ruído implementando filtros como *downsampling*, remoção de *outliers*, extracção de índices de pontos, projecções de pontos, etc [Rus09].

- **Módulo *keypoints*** - contém implementações de diferentes métodos de detecção de pontos de interesse (ou *keypoints*). Esta metodologia pode ser empregue como uma etapa prévia no processo de decisão da descrição geométrica local.

  Neste módulo existem implementações de detectores de pontos de interesse por meio: de uma adaptação da ideia de extracção de pontos de interesse em

[14] consultar apêndice B
[15] Imagem retirada de [Libd]
[16] consultar apêndice C

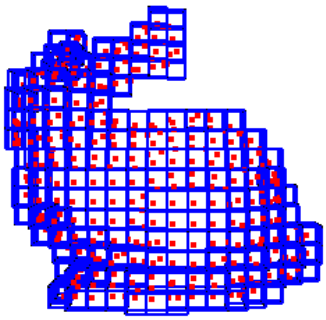


Figura 3.6: Exemplo de particionamento espacial de uma superfície através de uma *octree* - método denominado de Voxelização. Voxeis a cor azul e pontos a cor vermelha.

imagens através do método de Harris [HS88]; uma adaptação do algoritmo *Scale Invariant Feature Transform* (SIFT) [Low99, Low04b] de imagens para nuvens de pontos com texturização; do algoritmo *Uniform Sampling* [BRBB09] e detecção de pontos de interesse através do algoritmo *Normal Aligned Radial Features* (NARF) [SRKB10].

- **Módulo *features*** - contém estruturas de dados e mecanismos para estimação de *features* 3D a partir de nuvens de pontos, tais como: cálculo das normais e curvatura de superfícies [Rus09], *Spin Images* [JH98], *Integral Images* [HSD+10], estimação de invariantes de momento, descritores geométricos *Point Feature Histograms* (PFH) [RBMB08, RMBB08], *Fast Point Feature Histograms* (FPFH) [RBB09], *Signature of Histograms of Orientations* (SHOT) [TSDS10, TSdS11], NARF, SIFT, *Rotation Invariant Feature Transform* (RIFT) [LSP05], *Ensemble of Shape Functions* (ESF) [WV11], *Viewpoint Feature Histogram* (VFH) [RBTH10] e *Clustered Viewpoint Feature Histogram* (CVFH) [ABG+11].

  Na figura 3.7[17] é ilustrado o resultado da computação das normais de uma superfície, usando o espaço seleccionado na vizinhança de um ponto, normalmente referido como *k-neighborhood*.

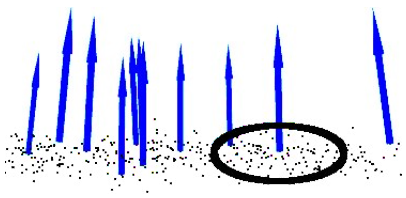


Figura 3.7: Cálculo das normais de superfície usando o espaço *k-neighborhood*.

- **Módulo *registration*** - inclui métodos geralmente conhecidos por combinar

[17] Imagem retirada de [Libb]

várias nuvens de pontos num modelo global consistente, uma técnica conhecida por registo. Este módulo inclui métodos como o *Iteractive Closest Points* (ICP) linear [BM92, Zha94], ICP não linear [Rus09], *Greedy Inicial Alignment* (GIA) [RBMB08], *Sample Consensus - Initial Alignement* (SAC-IA) [RBB09], métodos de estimação de correspondências entre nuvens, vários métodos de rejeição de correspondências, métodos de estimação da transformação, como *Singular Value Decomposition* (SVD) [AHB87], *Levenberg-Marquardt* (LM) [Fit01] e *Linear Least Squares* (LLS) [Low04a]. Estes métodos, desde os de estimação de correspondências até aos de estimação da transformação, permitem reproduzir variações do algoritmo ICP, mas o *pipeline* deste tipo de implementação ainda não está completamente estável na última versão do PCL.

Por fim, está ainda presente, a implementação de um algoritmo de detecção de *loops* denominado *Explicit Loop Closing Heuristic* (ELCH) [SNLH09].

- **Módulo *sample consensus*** - detém todos os métodos *Sample Consensus* (SAC) incluindo o *Random Sample Consensus* (RANSAC) [FB81], *Least Median of Squares* (LMEDS) [RL87], *M-Estimator Sample Consensus* (MSAC) [TZ00], *Randomized Random Sample Consensus* (RRANSAC) [MC02], *Randomized MSAC* (RMSAC) [TZ00], *Maximum LikeLihood Estimation SAmple Consensus* (MLESAC) [TZ00] e *Progressive Sample Consensus* (PROSAC) [CM05], bem como modelos paramétricos tais como linhas, planos, cilindros, esferas, etc. Estes podem ser combinados de forma a detectar modelos específicos e os seus parâmetros em nuvens de pontos. Na figura 3.8[18] é ilustrada a potencialidade deste módulo, que possibilita a detecção de diversos objectos presentes numa dada nuvem de pontos, a partir de modelos paramétricos.

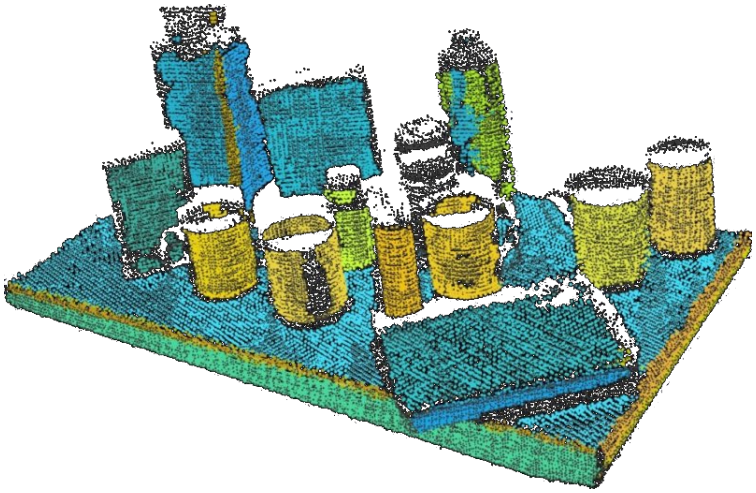
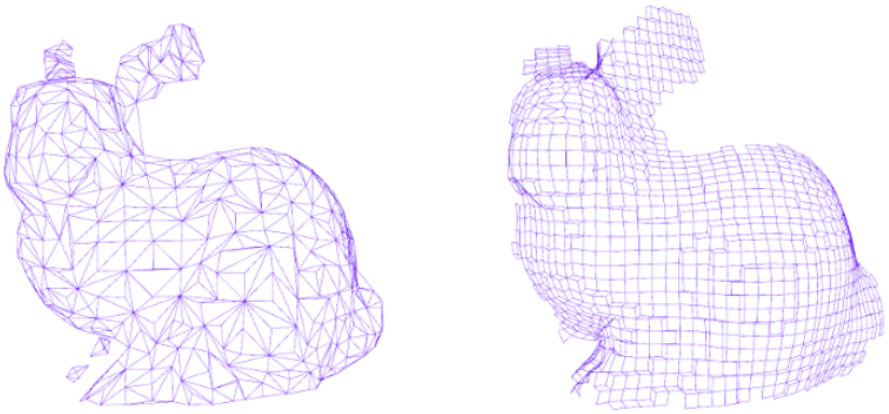


Figura 3.8: Exemplificação da detecção e segmentação de objectos presente numa nuvem de pontos (planos e cilindros).

[18]Imagem retirada de [Libe]

- **Módulo *segmentation*** - contém implementações de extracção de clusters, métodos de segmentação de modelos paramétricos específicos, métodos de ajustamento de modelos paramétricos detectados através de SAC, extracção poligonal prismática, etc.
- **Módulo *tracking*** - fornece mecanismos de *tracking*, como o filtro de partículas [PS99], ou o filtro de partículas KLD *adaptative* [Fox01].
- **Módulo *surface*** - contém técnicas de reconstrução de superfícies, *meshing*, *convex hulls*, *Moving Least Squares* (MLS) [Lev98], *Robust Moving Least Squares* (RMLS) [RBM+07], *Marching Cubes* [LC87], bem como algumas variantes da formulação original (Hoope [HDD+92] e funções radiais [CBC+01]), reconstrução volumétrica através de projecção de uma grelha [LLP+10], etc. Na figura 3.9[19] estão ilustrados dois dos métodos de reconstrução de superfícies, através de técnicas de *meshing* e reconstrução volumétrica.



(a) meshing (b) volumétrica

Figura 3.9: Exemplos de reconstrução de superfícies, recorrendo a técnicas de *meshing* e volumétrica.

3.3 Ficheiros do tipo PCD (Point Cloud Data)

O formato dos ficheiros PCD, nativo do PCL, fornece a capacidade para guardar dados inerentes às nuvens de pontos. Foi desenhado no sentido de complementar formatos de ficheiros existentes que não suportam algumas das extensões que o PCL trouxe ao processamento de nuvens de pontos, nomeadamente certos tipos de pontos associados à especificidade de alguns algoritmos, histogramas, etc..

O formato de ficheiros PCD, não é portanto, o primeiro tipo de ficheiro que suporta dados 3D provenientes de nuvens de pontos. As comunidades da computação gráfica e geométrica criaram numerosos formatos que suportam a descrição de

[19] Imagem retirada de [Libf]

polígonos arbitrários e nuvens de pontos. Alguns dos formatos mais conhecidos incluem:

- PLY (PoLYgon File Format) - formato de ficheiro que suporta dados poligonais, desenvolvido na Universidade de Standford por Greg Turk.
- STL (STereoLithography) - formato de ficheiro nativo de software stereográfico CAD, criado pela empresa 3D Systems, que descreve superfícies triangulares pelas seus três vértices e normal ($\vec{n}$).
- OBJ - formato de ficheiro contendo definições geométricas desenvolvido pela Wavefront Technologies.

Estes formatos são suportados pela biblioteca VTK.

3.3.1 Vantagens em relação a outros tipos de ficheiros

Nenhum outro formato de ficheiro mencionado anteriormente permite uma tão grande flexibilidade ou rapidez. Algumas das suas vantagens relativamente aos restantes formatos incluem:

- capacidade de armazenar e processar *datasets* organizados[20] de nuvens de pontos, conceito de extrema importância em aplicações em tempo-real, em áreas de investigação como a robótica, e realidade aumentada (*Augmented Reality* (AR)).
- os métodos *mmap/munmap* para o mapeamento em memória de ficheiros são formas eficazes de carregar e guardar dados em disco.
- capacidade de armazenar diferentes tipos de dados (todo o tipo de primitivas são suportadas: *char*, *short*, *int*, *float*, *double*) permitindo que os dados de uma nuvem de pontos sejam flexíveis e eficientes com respeito ao armazenamento e processamento. Dimensões inválidas de pontos são normalmente armazenados como tipo "Not a Number"(NaN).
- capacidade de armazenar dados de histogramas nD provenientes dos descritores geométricos de *features*, muito importante em aplicações que usem percepção/visão artificial.

[20] respeita a topologia da aquisição das nuvens de pontos

3.3.2 Estrutura dos ficheiros PCD

Como já foi mencionado anteriormente, a estrutura dos ficheiros PCD é bastante flexível, englobando um conjunto de diferentes tipos de dados. Existem vários campos definidos que deverão ser preenchidos mediante a estrutura de dados que se pretende guardar, entre eles estão:

- **Versão (*Version*)** - especifica a versão do ficheiro PCD.
- **Campos (*Fields*)** - especifica o nome de cada dimensão/campo que uma nuvem de pontos pode ter (ex: $x\ y\ z$ - posição xyz; $x\ y\ z\ rgb$ - posição + cor).
- **Tamanho (*Size*)** - especifica o tamanho de cada dimensão em *bytes* (ex: *unsigned char* - 1 *byte*).
- **Tipo (*Type*)** - especifica o tipo de cada dimensão (ex: F - representa variáveis do tipo *float*).
- **Número de elementos (*Count*)** - especifica quantos elementos cada dimensão tem. Por exemplo, dados só da posição em x têm normalmente só um elemento, mas um descritor geométrico como o VFH tem 308 elementos.
- **Largura (*Width*)** - especifica a largura de uma determinada nuvem de pontos. Pode ter dois significados: especificar o número total de pontos numa nuvem desorganizada ou especificar a largura (número total de pontos numa linha) em nuvens de pontos organizadas.
- **Altura (*Height*)** - especifica a altura de uma determinada nuvens de pontos. Pode ter dois significados: especificar o número total de linhas de uma nuvem de pontos, ou ter o valor 1 quando se trata de uma nuvem de pontos desorganizada.
- **Vista (*Viewpoint*)** - especifica o ponto de aquisição do dispositivo que adquiriu a nuvem de pontos. Esta propriedade pode potencialmente ser mais tarde utilizada para construir transformações entre diferentes sistemas de coordenadas, ou como auxílio no cálculo das normais das superfícies, que necessitam de uma orientação consistente. A informação da vista é especificada como a translação $(t_x,\ t_y,\ t_z)$ + o quaternião $(q_w\ q_x\ q_y\ q_z)$.
- **Pontos (*Points*)** - especifica o número total de pontos de uma nuvem.
- **Dados (*Data*)** - especifica o tipo de armazenamento que foi efectuado para guardar uma dada nuvem de pontos (ex: ASCII, Binário).

A ordem dos campos inseridos no ficheiro deverá ser precisamente a mesma que acima ficou ilustrado, exemplificando com a listagem seguinte (3.1).

Listagem 3.1: Exemplo de um ficheiro pcd (v.7)

```
1  # .PCD v.7 - Point Cloud Data file format
2  VERSION 0.7
3  FIELDS x y z rgba
4  SIZE 4 4 4 4
5  TYPE F F F U
6  COUNT 1 1 1 1
7  WIDTH 640
8  HEIGHT 480
9  VIEWPOINT 0 0 0 1 0 0 0
10 POINTS 307200
11 DATA ascii
12 0.68394858 -0.23533715 1.2870001 13481922
13 0.68373334 -0.23442286 1.2820001 13416648
14 0.68617529 -0.23442286 1.2820001 13350591
15 0.68593144 -0.23350859 1.2770001 13349813
16 0.68836385 -0.23350859 1.2770001 13350073
17 0.6907962 -0.23350859 1.2770001 13481405
18 0.6932286 -0.23350859 1.2770001 13612482
19 0.6929372 -0.2325943 1.2720001 13416135
20 0.69536 -0.2325943 1.2720001 13219520
21 0.69778287 -0.2325943 1.2720001 13154490
22 0.70020574 -0.2325943 1.2720001 13154751
23 0.70262861 -0.2325943 1.2720001 12958915
24 0.70283431 -0.23186287 1.268 12825280
25 0.70524955 -0.23186287 1.268 12891069
26 nan nan nan 13020348
```

4

Caracterização dos módulos e métodos do PCL

Conteúdo

Neste capítulo pretende-se caracterizar o PCL, explorando alguns dos módulos e métodos neles existentes, tendo em conta também o seu potencial interesse para o tema desta dissertação.

De notar que todos os testes descritos e caracterizações foram efectuados utilizando um portátil com: um processador Intel® Core™ 2 Duo P8700 a uma frequência de 2,53 GHz, uma placa gráfica ATI™ Mobility Radeon™ HD 3650 e 4 Gb de memória RAM. Em termos de software, foi utilizado o sistema operativo Ubuntu 12.04 LTS (*Precise Pangolin*) e aplicações desenvolvidas no ambiente de desenvolvimento integrado Qt Creator e Code::Blocks.

Os métodos são caracterizados sob a forma de tempo de processamento e, em termos de resultados finais, comparando métodos que tenham a mesma finalidade em termos qualitativos, bem como a eventual combinação com métodos de diferentes módulos no sentido de caracterizar o impacto que essa combinação tem no tempo de processamento.

Em cada caracterização, é realizada uma breve introdução dos algoritmos que compõem os métodos a caracterizar. Os resultados são apresentadas em tabelas, onde está discriminado o tempo de execução de cada teste efectuado aos diversos métodos. Com excepção de alguns testes que têm um tempo de execução muito alto, foram efectuados para cada teste 10 iterações, para que se verifique com maior exactidão o tempo de processamento. Depois da apresentação dos resultados, a caracterização de cada método é concluída com uma breve discussão de resultados.

A decisão de assim estruturar esta parte da dissertação, teve como base o interesse de uma maior clareza e encadeamento da informação para que se perceba exactamente os parâmetros e algoritmos que se estão a caracterizar.

No apêndice D é apresentado o código fonte genérico da implementação de cada método caracterizado nesta secção.

Cada nuvem de pontos capturada pela Kinect tem uma dimensão de 640x480, um total de 307.200 pontos e poderá ter $\boldsymbol{n}$ dimensões (nD) como x, y, z, $rgba$, $intensidade$, etc.

Na figura 4.1 está ilustrado um exemplo da captura de uma nuvem de pontos

pela Kinect, mostrando o seu sistema de coordenadas (eixos $\vec{x}$, $\vec{y}$ e $\vec{z}$) no visualizador para uma melhor percepção da forma de organização dos pontos. Os eixos estão referenciados no ponto de coordenadas $(0, 0, 0)$, ou seja, no ponto de aquisição da nuvem.

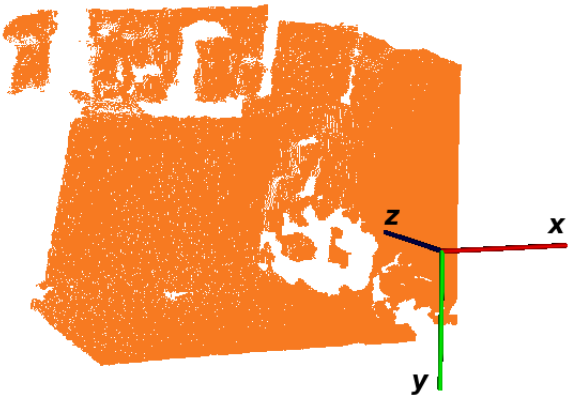


Figura 4.1: Exemplo de uma nuvem de pontos e correspondente sistema de coordenadas: $+\vec{x}$ - cor vermelha; $+\vec{y}$ - cor verde; $+\vec{z}$ - cor azul.

4.1 Módulo *io*

O módulo *io* contém ferramentas de leitura e escrita de ficheiros do tipo *.pcd, *.ply, *.vtk, *.obj e *.stl, e uma interface genérica de acesso a diferentes dispositivos e respectivos *drivers*. A classe *pcl::Grabber* fornece essa interface com os *drivers* OpenNI para a aquisição de dados da Kinect à frequência máxima de 30 Hz.

A classe ***pcl::PCDWriter***, permite a escrita de ficheiros *.pcd em formato *ASCII*, *Binary* e *Binary Compressed*. Na tabela 4.1 é mostrado o tamanho dos ficheiros quando gravados a partir de uma nuvem com 307.200 pontos nos vários formatos disponíveis. Na tabela 4.2 são mostrados os valores do tempo de processamento para a leitura e escrita de ficheiros codificados com os formatos disponíveis.

Tabela 4.1: Tamanho dos ficheiros PCD em vários tipos de codificação.

Codificação	Tamanho (Mb)
ASCII	$11,5 \pm 0,4$
Binary	$4,9$
Binary Compressed	$2,3 \pm 0,2$

Tabela 4.2: Tempo de processamento de leitura e escrita dos ficheiros PCD em vários tipos de codificação.

	Codificação	Tempo (ms)
Leitura	*ASCII*	4.006 ± 172
	Binary	383 ± 102
	Binary Compressed	396 ± 57
Escrita	*ASCII*	$9,22$
	Binary	$42,5 \pm 17,5$
	Binary Compressed	$9,22$

4.2 Módulo filters

Este módulo engloba métodos de filtragem de pontos consoante restrições numa dada dimensão (filtro *passThrough*), remoção de *outliers* e *downsampling*.

4.2.1 *PassThrough*

4.2.1.1 Fundamentos teóricos

O filtro *passThrough* permite a filtragem de pontos de uma dimensão consoante um intervalo delimitador definido pelo utilizador. Na equação 4.1, exemplifica-se a definição de um intervalo que limita a saída do filtro na dimensão z.

$$lim_{inf} \leq z \leq lim_{sup} \tag{4.1}$$

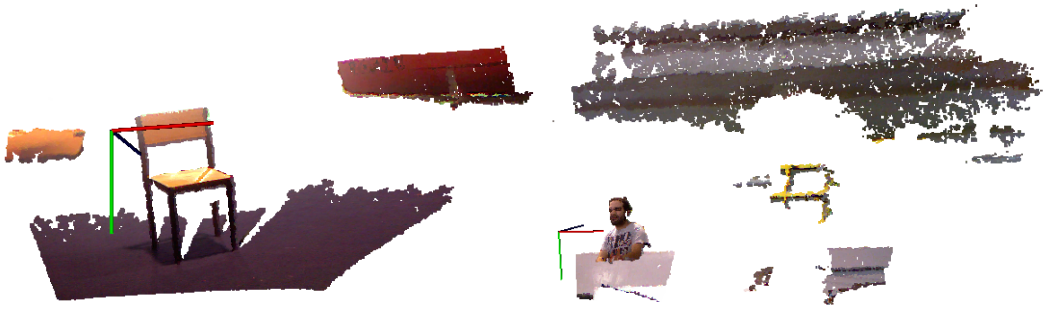
Este filtro é de especial importância quando se pretende restringir a observação ou cálculos posteriores a um espaço limitado.

4.2.1.2 Caracterização

Na figura 4.2 são mostradas as nuvens que servirão de entrada na caracterização deste método. Na figura 4.3, as mesmas nuvens, são mostradas num gradiente de intensidade com respeito à dimensão z, sendo representados a cor vermelha os pontos com menor coordenada em $\boldsymbol{z}$, e a cor azul com maior. As nuvens terão o nome de $\boldsymbol{\mathcal{P}_1}$ para a nuvem representada do lado esquerdo da figura 4.2 e $\boldsymbol{\mathcal{P}_2}$ para a do lado direito.

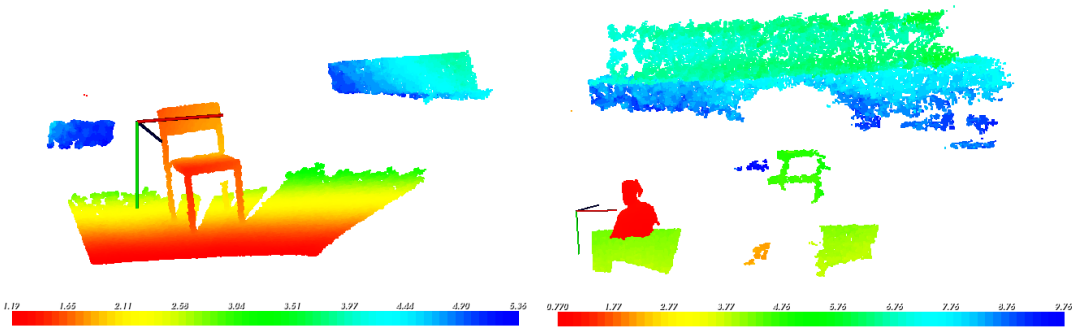
Decidiu-se caracterizar para este filtro (tabela 4.3):

- o tempo de execução do filtro mantendo ou não à sua saída a organização das nuvens, ou seja, conservando ou não todos pontos e a dimensão da nuvem original (640x480). A conservação da organização das nuvens, faz com que todos os pontos que não pertençam ao intervalo delimitado no filtro, sejam definidos como NaNs (pontos com valores de dimensão inválida). Por outro



(a) $\mathcal{P}_1$ (b) $\mathcal{P}_2$

Figura 4.2: Nuvens de pontos originais $\mathcal{P}_1$ e $\mathcal{P}_2$



(a) $\mathcal{P}_1$ (b) $\mathcal{P}_2$

Figura 4.3: Nuvens de pontos originais, representadas em intensidade na dimensão z.

lado, a não manutenção da organização da nuvem, suprime completamente da nuvem, os pontos não pertencentes ao intervalo definido.

A aplicação deste filtro pode ter um impacto significativo no processamento quando utilizado em combinação com qualquer um outro método. Isto porque, alguns dos algoritmos implementados no PCL só aceitam à entrada da sua execução nuvens organizadas. Por outro lado, o facto de se poder diminuir o número de pontos presentes numa nuvem, suprimindo os NaNs, diminui o tempo de execução do posterior processamento de outros algoritmos, que não necessitem de ter como entrada nuvens organizadas.

- a diferença de tempo de execução do filtro em nuvens com 3 dimensões (x,y e z) e com 4 dimensões (x, y, z e rgb)
- o impacto da variação dos intervalos delimitadores do filtro no tempo de execução.

Tabela 4.3: Tabela de caracterização do filtro *PassThrough*.

Condições iniciais				**saída: nuvem organizada**	**saída: nuvem desorganizada**	
Teste	***Nome***	***Dimensões***	***Intervalo (m)***	***exec. (ms)***	***nº pontos filtrados***	***exec. (ms)***
1	$\mathcal{P}_1$	x,y,z,rgb	$[0,0;5,0]$	34 ± 2	180.903	29 ± 2
2			$[1,0;4,0]$	34 ± 3	169.439	28 ± 2
3			$[1,19;3,7]$	35 ± 3	166.707	28 ± 2
4		x,y,z	$[0,0;5,0]$	29 ± 3	180.903	24 ± 2
5			$[1,0;4,0]$	29 ± 3	169.439	24 ± 2
6			$[1,19;3,7]$	30 ± 4	166.707	24 ± 2
7	$\mathcal{P}_2$	x,y,z,rgb	$[0,0;5,0]$	28 ± 3	85.543	28 ± 2
8			$[1,0;4,0]$	37 ± 1	27.745	21 ± 1
9			$[0,77;1,5]$	38 ± 3	45.582	23 ± 1
10		x,y,z	$[0,0;5,0]$	28 ± 2	85.543	24 ± 2
11			$[1,0;4,0]$	28 ± 2	27.745	19 ± 1
12			$[0,77;1,5]$	27 ± 1	45.582	20

Na figura 4.4 é mostrado o resultado da filtragem de $\mathcal{P}_1$ e $\mathcal{P}_2$, onde estão representados os pontos com coordenadas em z compreendidos nos intervalos $[1,19;3,70]$ e $[0,77;1,50]$ respectivamente.

4.2.1.3 Discussão de resultados

Da análise da tabela 4.3 pode-se observar que as maiores diferenças nos tempos de execução se verificam entre as mesmas nuvens com dimensões diferentes e entre os tempos de execução para a obtenção de nuvens organizadas ou desorganizadas.

A primeira diferença referida, deve-se ao facto do filtro ter de verificar 4 dimensões nas nuvens x,y,z e rgb, ao contrário de 3 dimensões nas nuvens x,y e z.

(a) $\mathcal{P}_1$ (teste 6) (b) $\mathcal{P}_2$ (teste 12)

Figura 4.4: Nuvens de pontos $\mathcal{P}_1$ e $\mathcal{P}_2$ filtradas com a aplicação do filtro *passthrough*.

A segunda grande diferença reside quando se pretende diferentes saídas para a obtenção de nuvens organizadas ou desorganizadas. O facto do tempo de execução ser maior quando se pretende nuvens organizadas deve-se a que o filtro, neste caso, ter de substituir os pontos com coordenadas fora do intervalo definido, para valores inválidos (NaNs).

Comparando as diferenças nos tempos de execução entre as duas nuvens com as mesmas características iniciais verificam-se ligeiras diferenças, mas não é possível retirar ilações dos resultados.

A variação do intervalo de filtragem em nuvens com as mesmas características não produz diferenças nos tempos de execução.

4.2.2 Remoção de *outliers*

As nuvens de pontos geradas a partir da Kinect podem conter erros de medida levando ao aparecimento de *outliers*. É por isso importante fazer a análise da vizinhança $\mathcal{P}^k$ de um qualquer ponto $\boldsymbol{p}$ pertencente a uma nuvem de pontos $\mathcal{P}$, antes de se estimar as suas características, como por exemplo o cálculo das normais, verificando assim se esse $\boldsymbol{p}$ é uma boa representação da superfície amostrada.

4.2.2.1 Fundamentos teóricos

Existem dois métodos implementados no PCL que permitem a remoção de *outliers* de uma nuvem de pontos:

- remoção estatística de *outliers* (***método 1*** - *Statistical Outliers Removal*), proposto em [RMB+08a]

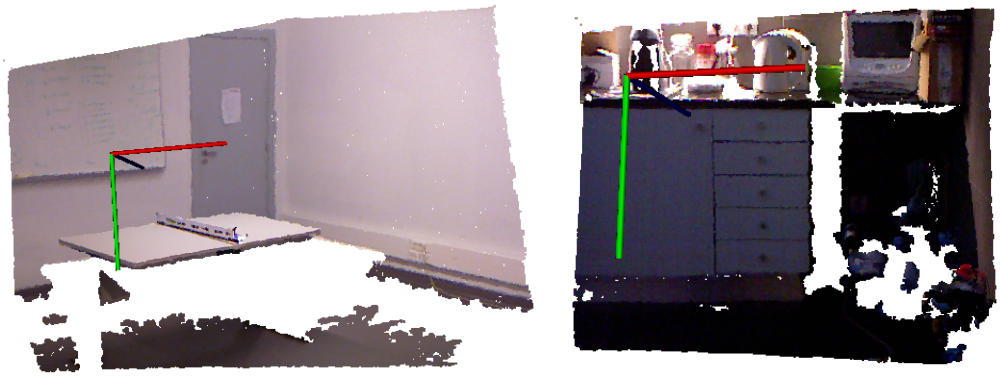
- filtro baseado no número de pontos vizinhos de um determinado ponto (***método 2*** - *Radius Outliers Removal*).

A remoção estatística de *outliers* permite a correcção de irregularidades de $\mathcal{P}$, calculando a média μ e o desvio padrão σ das distâncias para os k pontos vizinhos mais próximos de cada p, removendo os pontos que não satisfazem a condição $\mu \pm \alpha\sigma$. O valor α depende do tamanho da vizinhança ($\mathcal{P}^k$) analisada.

O segundo filtro permite a remoção de *outliers* através da análise do número de pontos vizinhos $\mathcal{P}^k$ de p que se encontrem dentro de um dado raio r de p. Se dentro desse espaço criado pela esfera de raio r existir um número mínimo de pontos (min_{pts}), definido pelo utilizador, então o ponto p que está a ser analisado, é considerado *inlier*. Por outro lado, se essa condição não se verificar, p é considerado *outlier*.

4.2.2.2 Caracterização

Nos testes, são utilizadas as nuvens de pontos representadas na figura 4.5.



(a) $\mathcal{P}_1$ (b) $\mathcal{P}_2$

Figura 4.5: Nuvens de pontos que serviram de entrada aos métodos de remoção de *outliers*.

Foram escolhidas estas nuvens porque são distintas em termos de densidade de pontos, ou seja, as superfícies amostradas em $\mathcal{P}_1$ estão mais distantes do ponto de aquisição do que em $\mathcal{P}_2$. Com isto, pretende-se analisar e caracterizar o comportamento dos métodos quando existe uma maior densidade de pontos e quando os pontos analisados têm uma profundidade maior, comportando maiores erros de medição.

Para estes métodos decidiu-se analisar:

- o tempo de processamento do método 1, com:
  - variação de k;
  - variação de α;
  - a manutenção ou não à saída, da organização de $\mathcal{P}$ introduzida na entrada do método.

  - nuvens de pontos com diferentes dimensões (x, y, z ou x, y, z, rgb).

- o tempo de processamento do método 2, com:
  - variação de $\boldsymbol{r}$;
  - variação do número mínimo de pontos $\boldsymbol{min_{pts}}$;
  - nuvens de pontos com diferentes dimensões (x,y,z ou x,y,z,rgb).

- as diferenças no tempo de processamento entre os 2 métodos;

- avaliar em termos qualitativos os resultados da filtragem.

Como nota adicional, deve-se ter em conta que o método 2 não permite, como entrada, nuvens de pontos com coordenadas inválidas (NaNs). Isto faz com que antes da filtragem se tenha de proceder à remoção desses pontos, tendo o filtro depois como entrada uma nuvem desorganizada. Pelo contrário, o método 1 permite como entrada nuvens organizadas e desorganizadas. Portanto, como termo de comparação entre os dois métodos só se podem utilizar os dados que se refiram à remoção de *outliers*, tendo os filtros como entrada nuvens desorganizadas.

Na remoção de NaNs, em $\boldsymbol{\mathcal{P}_1}$ foram removidos 84.284 pontos e mantidos 222.916 (72,56%). Em $\boldsymbol{\mathcal{P}_2}$ foram removidos 65.275 pontos e mantidos 241.925 (78,75%).

Nas tabelas 4.4 e 4.5 são mostrados os resultados dos vários testes realizados com o método 1, tendo o primeiro teste uma nuvem organizada como entrada ($\boldsymbol{\mathcal{P}_1}$ ou $\boldsymbol{\mathcal{P}_2}$) e no segundo uma nuvem de pontos com as coordenadas inválidas removidas. Na tabela 4.6 são mostrados os vários testes realizados com o método 2.

Nas figuras 4.6 e 4.7 são mostrados os resultados da remoção de *outliers* utilizando o método 1, e nas figuras 4.8 e 4.9 os resultados da filtragem aplicando o método 2. De notar que a escala dos pontos foi aumentada e as nuvens de pontos são mostradas com texturização uniforme de cor azul para uma melhor visualização dos resultados.

4.2.2.3 Discussão de resultados

No método 1, a variável $\boldsymbol{k}$ tem um impacto significativo sobre o tempo de processamento do filtro. Quanto maior for esse valor, maior é o tempo de processamento, por outro lado, se esse valor for demasiadamente pequeno, o tempo de processamento aumenta significativamente. A variável $\boldsymbol{\alpha}$ não tem um impacto significativo no tempo de processamento do filtro, no entanto, em nuvens de pontos só com dimensão x,y,z tem um melhor desempenho do que com x,y,z,rgb.

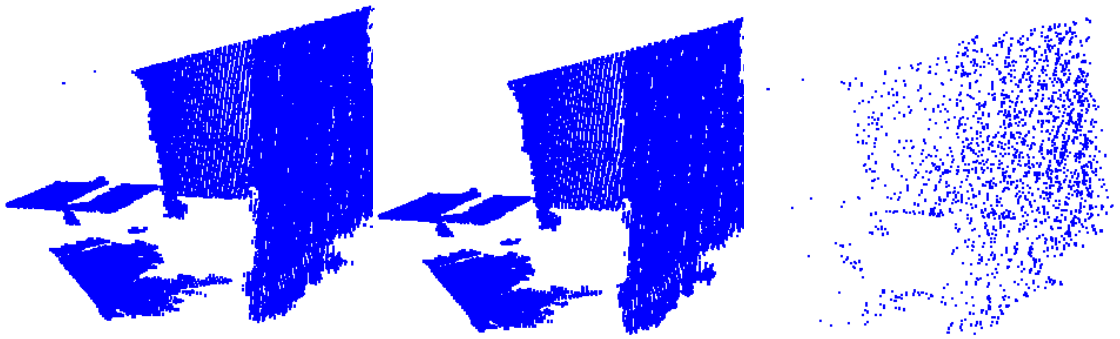
No método 2, nuvens de diferentes dimensões não têm impacto na velocidade de processamento, mantendo-se sensivelmente igual para os dois casos analisados.

Tabela 4.4: Caracterização do método 1 (entrada: nuvens de pontos organizadas) - *Statistical Outliers Removal*.

Condições iniciais					**saída: nuvem organizada**					**saída: nuvem desorganizada**
Teste	***Nome***	***Dim.***	***k***	***α***	***nº inliers***	***nº outliers***	***inliers (%)***	***outliers (%)***	***exec. (ms)***	***exec. (ms)***
1	$\mathcal{P}_1$	x,y,z,rgb	60	1	269.737	37.463	87,80	12,20	3.521 ± 22	3.611 ± 13
2			30	1	274.467	32.733	89,34	10,66	1.983 ± 20	1.986 ± 32
3			30	5	307.086	114	99,96	0,04	1.957 ± 53	1.927 ± 34
4			10	1	280.539	36.661	91,32	8,68	1.354 ± 17	1.363 ± 23
5			10	2,33	304.100	3.100	98,99	1,01	1.364 ± 24	1.363 ± 26
6			5	1	282.698	24.502	92,02	7,98	25.020 ± 492	25.011 ± 329
7		x,y,z	10	2,33	304.100	3.100	98,99	1,01	1.330 ± 13	1.296 ± 21
8			5	1	282.698	24.502	92,02	7,98	21.529 ± 345	21.643 ± 370
9	$\mathcal{P}_2$	x,y,z,rgb	60	1	278.313	28.887	90,60	9,40	3.441 ± 42	3.445 ± 62
10			30	1	278.252	28.948	90,57	9,43	1.929 ± 27	1.912 ± 13
11			30	5	307.040	160	99,95	0,05	1.927 ± 18	1.921 ± 10
12			10	1	278.995	28.205	90,82	9,18	1.459 ± 25	1.449 ± 28
13			10	2,36	304.113	3.087	99,00	1,00	1.441 ± 11	1.442 ± 9
14			5	1	282.449	24.751	91,94	8,06	22.019 ± 405	21.829 ± 293
15		x,y,z	10	2,36	304.113	3.087	99,00	1,00	1.400 ± 13	1.366 ± 18
16			5	1	282.449	24.751	91,94	8,06	19.485 ± 203	19.374 ± 356

Tabela 4.5: Caracterização do método 1 (entrada: nuvens de pontos desorganizadas) - *Statistical Outliers Removal*.

Condições iniciais						**saída**				
Teste	***Nome***	***Dim.***	***k***	***α***	***nº pontos entrada***	***nº inliers***	***nº outliers***	***inliers (%)***	***outliers (%)***	***exec. (ms)***
1	$\mathcal{P}_1$	x,y,z,rgb	30	1	222.916	200.014	22.902	89,73	10,27	1.271 ± 16
2			10	2,33	222.916	219.268	3.648	98,36	1,64	667 ± 5
3		x,y,z	10	2,33	222.916	219.268	3.648	98,36	1,64	662 ± 3



(a) $\mathcal{P}_1$ (b) $\mathcal{P}_1$ - *inliers* (c) $\mathcal{P}_1$ - *outliers*

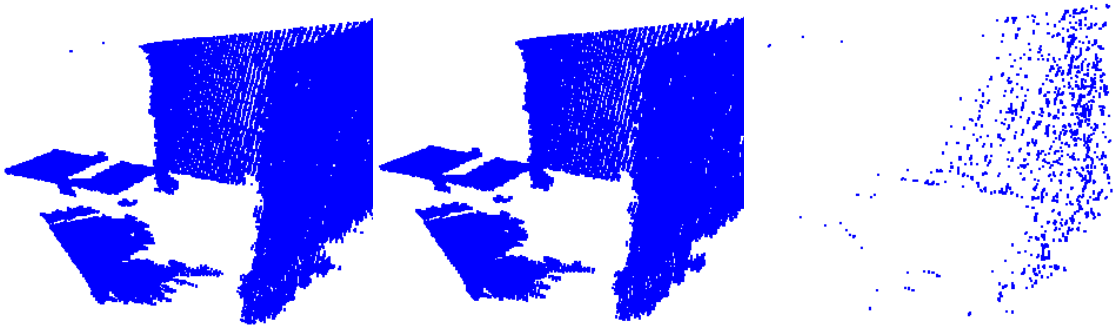
Figura 4.6: Resultado da aplicação do método 1 (*Statistical Outlier Removal*) à nuvem $\mathcal{P}_1$ - teste 5 da tabela 4.4.

Tabela 4.6: Caracterização do método 2 (entrada: nuvens de pontos desorganizadas) - *Radius Outliers Removal*.

Condições iniciais						saída				
Teste	*Nome*	*nº pontos entrada*	*Dim.*	*r (m)*	min_{pts}	*nº inliers*	*nº outliers*	*inliers (%)*	*outliers (%)*	*exec. (ms)*
1	$\mathcal{P}_1$	222916	x,y,z,rgb	0,02	30	37.625	185.291	16,88	83,12	795 ± 4
2				0,10	30	222.871	45	99,98	0,02	4.840 ± 37
3				0,05	30	215.902	7.014	96,85	3,15	1.719 ± 8
4				0,05	100	128.477	94.439	57,63	42,37	1.721 ± 13
5				0,05	5	222.734	182	99,92	0,08	1.765 ± 10
6				0,0224	4	220.676	2.240	99,00	1,00	840 ± 7
7				0,05	20	220.537	2.379	98,93	1,07	1.750 ± 15
8			x,y,z	0,10	30	222.871	45	99,98	0,02	4.840 ± 38
9	$\mathcal{P}_2$	241925	x,y,z,rgb	0,02	30	226.985	14.940	93,82	6,18	1.471 ± 23
10				0,10	30	241.915	10	99,996	0,004	17.306 ± 298
11				0,05	30	241.785	140	99,94	0,06	4.853 ± 27
12				0,05	100	239.600	2.325	99,04	0,96	4.806 ± 15
13				0,05	5	241.916	9	99,996	0,004	4.769 ± 25
14				0,02	15	239.264	2.661	98,90	1,10	1.505 ± 21
15				0,05	103	239.457	2.468	98,98	1,02	4.818 ± 19
16			x,y,z	0,10	30	241.915	10	99,996	0,004	16.991 ± 190

(a) $\mathcal{P}_2$ (b) $\mathcal{P}_2$ - *inliers* (c) $\mathcal{P}_2$ - *outliers*

Figura 4.7: Resultado da aplicação do método 1 (*Statistical Outlier Removal*) à nuvem $\mathcal{P}_2$ - teste 13 da tabela 4.4.



(a) $\mathcal{P}_1$ (b) $\mathcal{P}_1$ - *inliers* (c) $\mathcal{P}_1$ - *outliers*

Figura 4.8: Resultado da aplicação do método 2 (*Radius Outlier Removal*) à nuvem $\mathcal{P}_1$ - teste 7 da tabela 4.6.

(a) $\mathcal{P}_2$ (b) $\mathcal{P}_2$ - *inliers* (c) $\mathcal{P}_2$ - *outliers*

Figura 4.9: resultado da aplicação do método 2 (*Radius Outlier Removal*) à nuvem $\mathcal{P}_2$ - teste 14 da tabela 4.6.

Quanto maior o valor do raio (r) inserido, mais o filtro demora a processar resultados, por outro lado, a variação do valor do número mínimo de pontos que se pretendem encontrar nesse espaço não produz diferenças significativas.

Em termos de velocidade de processamento verifica-se que o método 1 tem um melhor desempenho do que o método 2, como se pode comprovar se se tiver em conta os valores das tabelas 4.5 e 4.6, nos testes em que a percentagem de *outliers* ronda os 1%.

Em termos qualitativos, o método 1 parece ser mais robusto do que o método 2. Se se analisar a distribuição dos *outliers* das nuvens de pontos $\mathcal{P}_1$ representadas nas figuras 4.6 e 4.8, verifica-se que no método 1 essa distribuição é mais uniforme. Isto deve-se ao facto de que o método 2 é muito mais susceptível à variação da densidade de pontos que diminui com o aumento da profundidade. Este facto permite concluir que, para o tipo de nuvens capturadas pela Kinect, o método 1 é o melhor para remover erros de medição de uma nuvem de pontos.

4.2.3 *Downsampling*

Na biblioteca PCL existem duas classes que abordam este filtro de diferente maneira:

- a classe *pcl::VoxelGrid <PointT>* (***método 1***);
- a classe *pcl::ApproximateVoxelGrid <PointT>* (***método 2***).

4.2.3.1 Fundamentos teóricos

Este filtro permite reduzir o número de pontos existentes (*downsampling*) numa nuvem, mantendo a estrutura da superfície amostrada através da utilização de uma técnica de decomposição espacial, denominada Voxelização. Este método cria uma

grelha de voxeis 3D[1] por todo o espaço que uma nuvem de pontos compreende, em que cada voxel tem de lado um tamanho defino pelo utilizador (*leaf size*). Seguidamente, todos os pontos presentes em cada voxel, serão aproximados em relação ao centróide ($\bar{\boldsymbol{p}}$).

O processo é executado com recurso a estruturas de dados denominadas *octrees*[2], para a execução da decomposição espacial. No entanto, em vez da nuvem resultante incluir os pontos correspondentes aos centros dos voxeis ocupados, ter-se-á uma nuvem de pontos correspondente à aproximação de todos $\boldsymbol{p_i}$ presentes em cada voxel em relação a $\bar{\boldsymbol{p}}$. Esta aproximação permite assim, uma representação mais fidedigna das superfícies amostradas.

Embora o comportamento dos dois métodos seja semelhante, existe uma diferença na sequência interna do tratamento de dados. Segundo os desenvolvedores do PCL, o método 1 remove todos os NaNs de uma nuvem de pontos antes de efectuar o *downsampling*, enquanto que no caso do método 2 esses pontos são tratados como um outro ponto qualquer, verificando-se assim a ocupação de mais voxeis.

Esta técnica é de extrema importância pois diminui o número de pontos da nuvem com as óbvias consequências em termos de peso computacional no posterior processamento de outros algoritmos.

4.2.3.2 Caracterização

Para os testes efectuados são utilizadas as nuvens $\boldsymbol{\mathcal{P}_1}$ e $\boldsymbol{\mathcal{P}_2}$ da figura 4.5.

Na caracterização, decidiu-se analisar:

- as diferenças nos tempos de processamento entre os dois métodos, levando em conta as dimensões das nuvens de entrada;
- o impacto da variação da variável *leaf-size* no tempo de processamento.

4.2.3.3 Discussão de resultados

Os resultados mostrados nas tabelas 4.7 e 4.8, e nas figuras 4.10 e 4.11, mostram que quanto maior o valor da variável *leaf-size*, menor é o número de pontos à saída do filtro. Este facto seria de esperar, visto que em cada voxel existirão mais pontos da nuvem original para aproximar a $\bar{\boldsymbol{p}}$.

No que toca à análise do impacto da dimensionalidade das nuvens, verifica-se uma diferença. Enquanto que o método 1 tem um desempenho melhor perante nuvens 4D, o método 2 tem um desempenho melhor na filtragem de nuvens 3D.

[1] um voxel pode ser visto como um cubo 3D

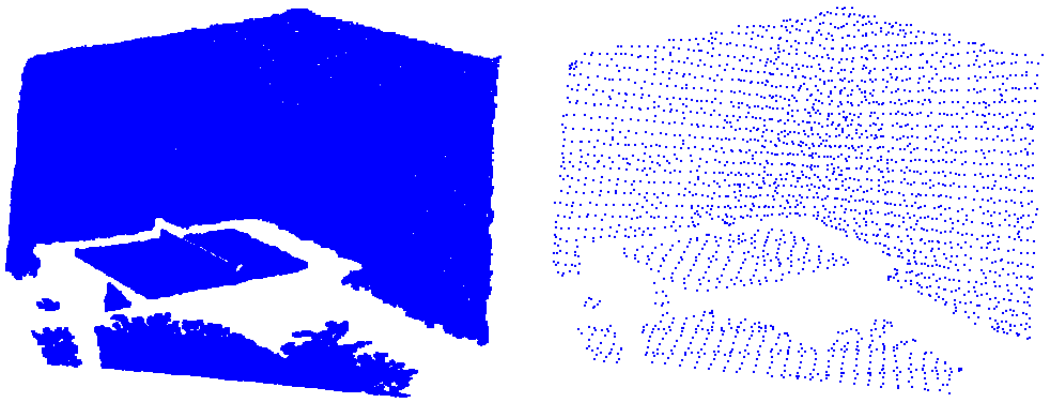
[2] esta estrutura de dados está descrita com mais pormenor no apêndice C

Tabela 4.7: Caracterização do método 1 - *Voxel Grid Downsampling*.

Condições iniciais				**saída**		
Teste	***Nome***	***Dim.***	***leafsize (m)***	***nº pontos***	$^{saida}/_{entrada}(\%)$	***exec. (ms)***
1	$\mathcal{P}_1$	x,y,z,rgb	$0,01$	127.686	$41,56$	118 ± 15
2			$0,02$	46.520	$15,14$	93 ± 11
3			$0,1$	2.758	$0,90$	74 ± 2
4		x,y,z	$0,01$	127.686	$41,56$	136 ± 6
5			$0,02$	46.520	$15,14$	128 ± 3
6			$0,1$	2.758	$0,90$	130 ± 7
7	$\mathcal{P}_2$	x,y,z,rgb	$0,01$	58.581	$19,07$	83 ± 8
8			$0,02$	17.491	$5,69$	79 ± 9
9			$0,1$	829	$0,27$	73 ± 7
10		x,y,z	$0,01$	58.581	$19,07$	161 ± 10
11			$0,02$	17.491	$5,69$	165 ± 5
12			$0,1$	829	$0,27$	149 ± 12

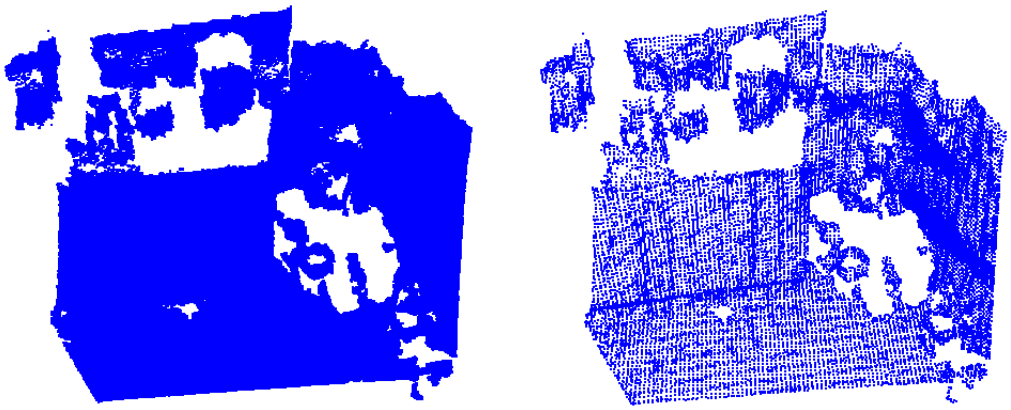
Tabela 4.8: Caracterização do método 2 - *Approximate Voxel Grid Downsampling*.

Condições iniciais				**saída**		
Teste	***Nome***	***Dim.***	***leafsize (m)***	***nº pontos***	$^{saida}/_{entrada}(\%)$	***exec. (ms)***
1	$\mathcal{P}_1$	x,y,z,rgb	$0,01$	145.985	$47,52$	165 ± 3
2			$0,02$	67.095	$21,84$	165 ± 7
3			$0,1$	4.910	$1,60$	155 ± 1
4		x,y,z	$0,01$	145.985	$47,52$	137 ± 6
5			$0,02$	67.095	$21,84$	138 ± 8
6			$0,1$	4.910	$1,60$	131 ± 12
7	$\mathcal{P}_2$	x,y,z,rgb	$0,01$	84.674	$27,56$	137 ± 8
8			$0,02$	30.256	$9,85$	135 ± 9
9			$0,1$	1.564	$0,51$	133 ± 5
10		x,y,z	$0,01$	84.674	$27,56$	115 ± 6
11			$0,02$	30.256	$9,85$	112 ± 2
12			$0,1$	1.564	$0,51$	113 ± 1



(a) $\mathcal{P}_1$ original (b) $\mathcal{P}_1$ filtrada

Figura 4.10: Resultado da aplicação do método 1 (*Voxel Grid Downsampling*) à nuvem $\mathcal{P}_1$ - parâmetros do teste 3 da tabela 4.7.



(a) $\mathcal{P}_2$ original (b) $\mathcal{P}_2$ filtrada

Figura 4.11: Resultado da aplicação do método 2 (*Approximate Voxel Grid Downsampling*) à nuvem $\mathcal{P}_2$ - parâmetros do teste 2 da tabela 4.8.

4.3 Módulo *keypoints*

Os *keypoints*, ou pontos de interesse, são pontos de uma imagem ou nuvem de pontos estáveis e que caracterizam pontos com especial relevância, podendo ser identificados com a aplicação de um critério bem definido de detecção.

Nesta secção iremos caracterizar dois algoritmos de detecção de pontos de interesse: *Scale Invariant Feature Transform* (SIFT) [Low04b] e *Harris keypoints* [HS88].

Ambos os métodos são adaptações dos algoritmos, com o mesmo nome, de extracção de pontos de interesse em imagens 2D. No entanto, os algoritmos implementados no PCL, envolvendo processamento de nuvens 3D, ainda não se encontram devidamente documentados.

4.3.1 Fundamentos teóricos

Tipicamente, o número de pontos de interesse de uma nuvem de pontos deverá ser muito menor do que o número total de pontos presentes na nuvem. Quando utilizados em combinação com métodos descritivos da geometria local, podem formar uma descrição compacta da nuvem de pontos original. A sua utilização tem também a vantagem de permitir reduzir os custos computacionais aos passos seguintes correspondentes ao registo de nuvens de pontos descritas em 4.4.2 e 4.5.

Um bom método de detecção de pontos de interesse deverá ter as seguintes propriedades:

- espaçamento entre pontos de interesse, só um pequeno subconjunto de pontos são pontos de interesse;
- repetibilidade, ou seja, pontos que sejam estáveis em duas nuvens para que possam realizar operações de registo;

- distintibilidade entre pontos de interesse, a área que envolve cada ponto de interesse deverá ter uma forma única.

4.3.1.1 Harris3D

Na documentação do PCL é referido que, no caso do algoritmo Harris é utilizado o gradiente das normais das superfícies (consultar [Libg]) em lugar do gradiente de intensidade dos píxeis de uma imagem, na matriz Hessiana da intensidade em redor de cada ponto, como normalmente é usado na computação visual.

A detecção de pontos de interesse, recorrendo à utilização do método Harris3D no PCL, pode ser descrito resumidamente da seguinte forma:

1. verifica a variação da deslocação de uma janela em várias direcções, através da soma das diferenças quadradas $\boldsymbol{E}$ (*Sum of Squared Differences* (SSD)) (em imagens 2D o deslocamento é realizado em x e y, no caso 3D acrescenta-se a dimensão z)

   (a) no caso 2D a equação de $\boldsymbol{E}$ pode ser aproximada por:

   $$E(x,y) \approx \begin{pmatrix} x & y \end{pmatrix} M \begin{pmatrix} x \\ y \end{pmatrix} \tag{4.2}$$

   onde $\boldsymbol{M}$ é igual a:

   $$M = \begin{bmatrix} \Sigma I_x I_x & \Sigma I_x I_y \\ \Sigma I_x I_y & \Sigma I_y I_y \end{bmatrix} \tag{4.3}$$

   No caso 3D, estes gradientes são substituídos por gradientes das orientações das normais de superfície.

2. Para que sejam detectados pontos de interesse, o valor da resposta $\boldsymbol{R}$ do algoritmo, é dada pela equação:

   $$R = \det(M) - k\, trace^2(M) \tag{4.4}$$

   tem de ser maior do que um dado *treshold*.

4.3.1.2 SIFT

No caso do algoritmo SIFT, é referenciado o artigo [Low04b] como base da adaptação ao processamento de nuvens de pontos (consultar [Libh]). A detecção dos pontos de interesse é realizado no espaço dimensional rgb, tal como na referência acima citada, sendo que a cada píxel está associado um ponto tridimensional, que será utilizado na posterior descrição geométrica local do ponto e da sua vizinhança.

A detecção de pontos de interesse, utilizando o método SIFT do PCL, pode ser descrita resumidamente da seguinte maneira:

1. detecção de extremos no espaço de escalas - a primeira etapa da detecção de pontos de interesse identifica localizações e escalas que podem ser repetidamente atribuídas segundo diferentes vistas do mesmo objecto.

   (a) à imagem inicial é aplicada repetidamente a convolução de filtros Gaussianos para produzir um conjunto de imagens no espaço de escalas.

   (b) as imagens convoluidas são agrupadas por oitavas, sendo que cada oitava corresponde à duplicação do desvio padrão ($\boldsymbol{\sigma}$), e $\boldsymbol{k}$ o número de escalas dentro de cada oitava.

   (c) seguidamente os pontos de interesse são identificados como sendo mínimos e máximos locais de imagens *Difference of Gaussians* (DoG) que ocorrem em múltiplas escalas.

      i. uma imagem DoG é dada por:

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma) \tag{4.5}$$

      onde $L(x, y, \sigma)$ é a convolução da imagem original $I = (x, y)$ com um *blur* Gaussiano $G(x, y, k\sigma)$ na escala $k\sigma$:

$$L(x, y, k\sigma) = G(x, y, k\sigma) \star I(x, y) \tag{4.6}$$

      Este passo é realizado comparando cada píxel das imagens DoG com os seus oito vizinhos na mesma escala e os seus nove píxeis vizinhos em cada uma das escalas vizinhas, perfazendo 26 comparações entre píxeis vizinhos. Se o valor do píxel é um máximo ou um mínimo por entre todos os píxeis comparados, então é seleccionado como um candidato a ponto de interesse.

2. localização dos pontos de interesse - a detecção de extremos no espaço de escalas produz demasiados candidatos a pontos de interesse, sendo alguns deles instáveis. Este passo descarta pontos de interesse com baixo contraste, através de uma série de Taylor de segunda ordem que analisa o desvio ($\hat{\boldsymbol{x}}$) do ponto de interesse. Este desvio representa o limite de detecção de pontos de interesse sem suficiente contraste.

4.3.2 Trabalho relacionado

Tendo em conta o facto dos métodos estarem ainda pouco documentados, podem ser mesmo assim, encontrados estudos que se servem destes métodos em tarefas de detecção de pontos de interesse, quer no processamento de nuvens de pontos, quer na área da computação gráfica.

O algoritmo SIFT foi utilizado em [HKH+10] na detecção e descrição geométrica de pontos de interesse servindo-se da dimensão rgb de nuvens de pontos adquiridas por câmaras RGB-D, no processo que leva ao seu registo. Em [HWP11] é proposta uma variação do algoritmo SIFT, Z-SIFT, utilizando-o no processo de alinhamento inicial de superfícies 3D adquiridas por *scanners* 3D, baseando-se na informação de profundidade ou dimensão z.

No que diz respeito à utilização de variantes do algoritmo Harris em dados 3D, em [SB11] é utilizado na detecção de pontos de interesse ou estruturas salientes em *meshes* 3D, tendo mostrado vantagens em operações de registo e simplificação da *mesh*.

4.3.3 Caracterização

Os parâmetros utilizados na definição do ***método 1*** (SIFT) no PCL são os seguintes:

1. o desvio padrão da escala mais pequena no espaço de escalas (σ);
2. o número de oitavas, ou duplicações de escalas a calcular (***nr-octaves***);
3. o número de escalas a calcular em cada oitava (k);
4. um *treshold* para limitar a detecção de pontos de interesse sem contraste suficiente ($\hat{x}$).

Para a definição do ***método 2*** (Harris3D) são utilizados os parâmetros:

1. o raio para a estimação das normais (***radius***);
2. o raio da esfera que será usada para determinar os vizinhos mais próximos, utilizada na detecção de pontos de interesse (***radiusSearch***);
3. o ***threshold*** para a detecção de pontos de interesse.

Para além destes parâmetros, o método 2 implementado no PCL permite 5 tipos de respostas diferentes do algoritmo:

1. Harris [HS88];
2. Noble [Nob88];

3. Tomasi [ST94];

4. Lowe [Low04b];

5. Curvature.

Estes diferentes tipos de respostas devem-se a variações do algoritmo Harris em aplicações de visão computacional, menos a resposta denominada "Curvature", que como já foi referido, não se encontra documentada, mas que se baseia na estimação da curvatura em cada ponto da nuvem. Testes preliminares, em vários tipos de cenários indicaram que uma resposta do tipo Tomasi, proposto em [ST94], daria melhores resultados práticos, como tal será essa a resposta ao algoritmo caracterizada neste estudo.

De notar também que a implementação do Harris no PCL faz uso do OpenMP para operações de paralelização de processamento.

Caracterizam-se estes métodos na nuvem de pontos teste representada na figura 4.12 em combinação com alguns algoritmos caracterizados na secção 4.2. Aplicou-se primeiro, a remoção de *outliers* através do método de análise estatística (ver 4.2.2) e depois o método de *downsampling* (ver 4.2.3) analisando-se o seu impacto nos resultados e no tempo de processamento na detecção de pontos de interesse. No caso da remoção de *outliers*, foram removidos 1% dos pontos da nuvem, definindo-os como *outliers*.

Figura 4.12: Nuvem de pontos $\mathcal{P}_1$ utilizada na caracterização do módulo *keypoints*.

4.3.4 Discussão de resultados

Nas figuras 4.13 e 4.14 podem-se observar os resultados para as melhores hipóteses encontradas, com os parâmetros definidos nas tabelas 4.9, 4.10, 4.11 e 4.12.

Em termos de tempo de processamento verifica-se que o método 2 apresenta resultados melhores do que o método 1, mas este facto torna-se irrelevante perante os melhores resultados qualitativos obtidos pelo método 1.

Tabela 4.9: Caracterização do método 1 - *SIFT*.

condições iniciais					saída	
Teste	σ	***nr-octaves***	k	$\hat{x}$	***nº keypoints***	***exec. (ms)***
1	$0,01$	3	2	$0,0$	645	6.373 ± 15
2	$0,005$	3	2	$0,0$	3.189	14.525 ± 36
3	$0,005$	10	2	$0,0$	3.259	14.997 ± 40
4	$0,005$	10	4	$0,0$	5.574	14.865 ± 59
5	$0,005$	10	8	$0,0$	14.884	18.743 ± 103
6	$0,01$	10	8	$1,5$	954	8.233 ± 23
7	$0,005$	10	8	$1,5$	2.653	18.821 ± 223
8	$0,005$	10	8	$4,0$	919	19.042 ± 321
9	$0,005$	10	8	$1,0$	3.576	18.769 ± 159
10	$0,005$	10	8	$1,4$	2.802	18.724 ± 203

Tabela 4.10: Caracterização do método 1 com utilização do algoritmo *Voxel Grid* - *SIFT*.

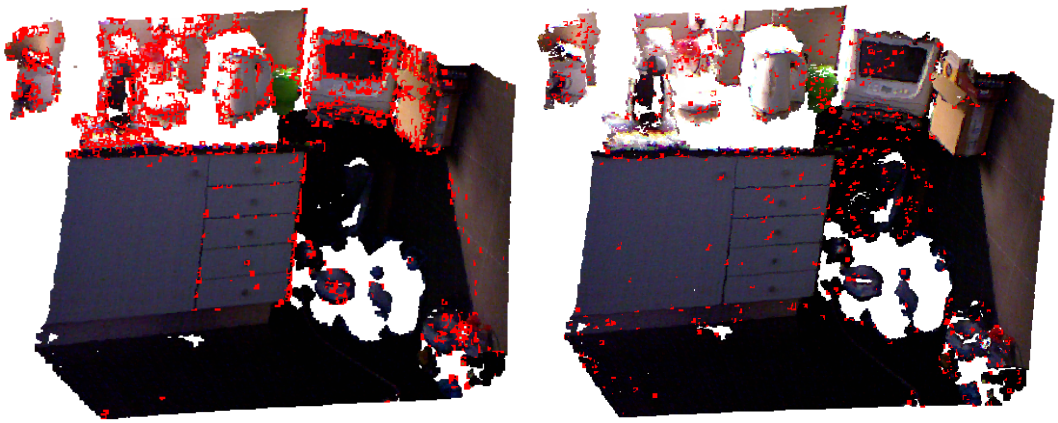
Condições iniciais							saída	
Teste	***leaf-size (m)***	***nº pontos***	σ	***nr-octaves***	k	$\hat{x}$	***nº key-points***	***exec. (ms)***
1			$0,005$	10	8	$1,4$	1.486	3.247 ± 8
2	$0,02$	16.621	$0,005$	10	8	$1,5$	1.431	3.243 ± 9
3			$0,005$	10	2	$0,0$	1.389	2.587 ± 7
4	$0,01$	56.567	$0,005$	10	8	$1,5$	2.289	9.928 ± 19
5	$0,03$	7.656	$0,005$	10	8	$1,5$	883	1.051 ± 13

Tabela 4.11: Caracterização do método 2 - *Harris3D*.

Condições iniciais				saída	
Teste	***radius (m)***	***radius Search (m)***	***thresh (m)***	***nº keypoints***	***exec. (ms)***
1	$0,01$	$0,01$	$0,1$	862	3.685 ± 23
2	$0,05$	$0,01$	$0,1$	862	3.726 ± 21
3	$0,05$	$0,03$	$0,1$	450	7.578 ± 205
4	$0,05$	$0,02$	$0,1$	641	5.178 ± 43
5	$0,05$	$0,02$	$0,05$	1.101	6.004 ± 43
6	$0,05$	$0,015$	$0,1$	845	4.352 ± 43
7	$0,05$	$0,014$	$0,1$	892	4.188 ± 43
8	$0,05$	$0,013$	$0,11$	792	4.056 ± 43

Tabela 4.12: Caracterização do método 2 com utilização do algoritmo *Voxel Grid* - *Harris3D*.

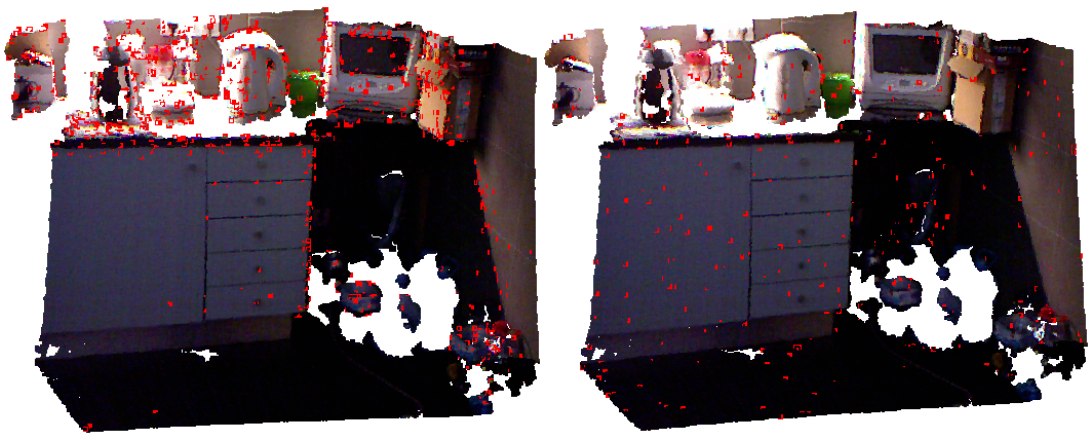
Condições iniciais						saída	
Teste	***leafsize (m)***	***nº pontos***	***radius (m)***	***radius Search (m)***	***thresh (m)***	***nº keypoints***	***exec. (ms)***
1				$0,014$	$0,1$	311	92 ± 3
2	$0,02$	16.621	$0,05$	$0,03$	$0,11$	60	123 ± 5
3				$0,013$	$0,2$	63	88 ± 4
4				$0,023$	$0,01$	1.158	84 ± 6



(a) Pontos de interesse - Sift - teste 10 da tabela 4.9

(b) Pontos de interesse Harris3D - teste 6 da tabela 4.11

Figura 4.13: Resultado da aplicação dos métodos de detecção de pontos de interesse em $\mathcal{P}_1$: pontos de interesse a cor vermelha.



(a) Pontos de interesse - Sift - teste 1 da tabela 4.10

(b) Pontos de interesse - Harris3D - teste 1 da tabela 4.12

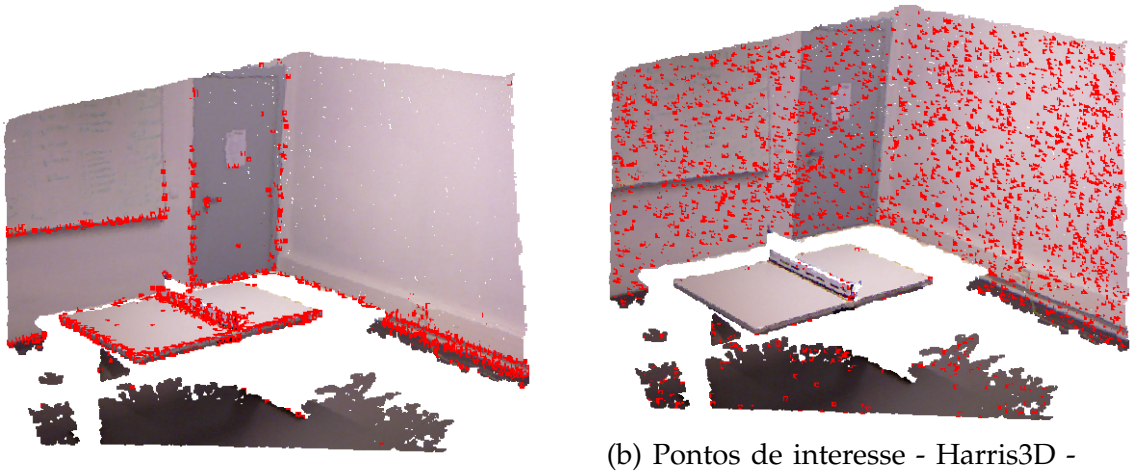
Figura 4.14: Resultado da aplicação dos métodos de detecção de pontos de interesse e de *downsampling* em $\mathcal{P}_1$: pontos de interesse a cor vermelha.

Para além dos algoritmos terem sido testados em $\mathcal{P}_1$, decidiu-se verificar o comportamento perante nuvens de pontos com outras características. Na figura 4.15 são mostrados os resultados da detecção de pontos de interesse numa nuvem de pontos ($\mathcal{P}_2$) com mais profundidade do que $\mathcal{P}_1$ e com menos zonas de contraste de cores, inserindo exactamente os mesmos parâmetros do melhor caso encontrado para $\mathcal{P}_1$.

Os resultados não poderiam ser mais esclarecedores. O método 2 obteve piores resultados, especialmente em zonas com muito ruído onde as superfícies amostradas pela Kinect têm maior profundidade e consequentemente maior erro na medição. Este facto é observável nas superfícies que representam as paredes do cenário ilustrado na figura 4.15. Pelo contrário, o método 1 obteve resultados muito mais satisfatórios, mesmo em zonas com pouco contraste de cor.

No caso em que é aplicado o método de *downsampling*, previamente à detecção de pontos de interesse, verifica-se , como seria de esperar, uma maior velocidade de execução e um menor número de pontos de interesse detectados. A figura 4.14 mostra os resultados da detecção de pontos de interesse tendo o número de pontos sido reduzido em $\pm 95\%$.

A diminuição do número de pontos de interesse detectados deve-se em grande parte, ao facto de existirem pontos de interesse muito próximos detectados na nuvem original, compreendidos no espaço de 1 voxel. Esses são aproximados para um só ponto que poderá ser um candidato na posterior detecção.



(a) Pontos de interesse - Sift - com os parâmetros do teste 10 da tabela 4.9

(b) Pontos de interesse - Harris3D - com os parâmetros teste 6 da tabela 4.11

Figura 4.15: Resultado da aplicação dos métodos de detecção de pontos de interesse em $\mathcal{P}_2$: pontos de interesse a cor vermelha.

4.4 Módulo *features*

As *Features* 3D são representações de um dado ponto tridimensional ou uma posição no espaço, descrevendo padrões geométricos baseados na informação disponível na vizinhança do ponto.

Nesta secção irão ser caracterizados os métodos de estimação das normais de superfícies (4.4.1) e descritores geométricos (4.4.2).

4.4.1 Estimação das normais de superfícies

Aqui serão caracterizados os métodos de estimação de normais ($\vec{n}_i$) de superfícies.

4.4.1.1 Fundamentos teóricos

As *features* geométricas locais, como as normais ou curvatura de superfície, são importantes propriedades para a descrição geométrica de uma superfície, pois permitem inferir a sua orientação em relação a um sistema de coordenadas. Estas propriedades são muitos usadas em aplicações ligadas à computação gráfica como, por exemplo, na determinação da orientação correcta de um plano que representa parte de uma *mesh* ou na aplicação de fontes de luz que permitem gerar sombras e outros efeitos visuais de uma determinada cena virtual.

No contexto de processamento 3D de nuvens de pontos, estas propriedades são uma base fundamental para a extracção de informação semântica desses dados, como as descritas na secção 4.7.

No PCL, existem dois métodos para estimar as normais de superfícies:

- ***método 1*** - *Normal Estimation* - baseado no algoritmo descrito em [Rus09].
  - O problema da determinação da normal de um ponto $\boldsymbol{p_i} \in \boldsymbol{\mathcal{P}}$ é aproximado pelo problema de estimação da normal ao plano tangente da vizinhança local $\boldsymbol{\mathcal{P}^k}$ de $\boldsymbol{p_i}$. A vizinhança é formada por todos os pontos que se encontrem dentro de um raio $\boldsymbol{r}$ de $\boldsymbol{p_i}$. Através de $\boldsymbol{\mathcal{P}^k}$, a normal $\boldsymbol{\vec{n}_i}$ pode ser estimada analisando os vectores próprios da matriz de covariância $\boldsymbol{\mathcal{C}_i} \in \mathbb{R}^{\mathbf{3\times 3}}$ de $\boldsymbol{\mathcal{P}^k}$, dada pela equação:

$$\mathcal{C} = \frac{1}{k}\sum_{i=1}^{k}\xi_i \cdot (p_i - \bar{p}) \cdot (p_i - \bar{p})^T, \mathcal{C} \cdot \vec{v}_j = \lambda_j \cdot \vec{v}_j, j \in \{0,1,2\} \tag{4.7}$$

    onde $\boldsymbol{\xi_i}$ representa o possível peso para $\boldsymbol{p_i}$, normalmente igual a 1;
  - se $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2$, o vector próprio $\vec{v}_0$ correspondente ao valor próprio mais pequeno analisado $\boldsymbol{\lambda_{i,0}}$ é a aproximação de $+\boldsymbol{\vec{n}} = \{n_x, n_y, n_z\}$

- ***método 2*** - *Integral Images* - baseado no algoritmo descrito em [HSD+10] tendo sido complementado e aplicado em [HHRB11] no estudo da rápida segmentação de cenas amostradas por câmaras RGB-D.

  - Nesta abordagem em vez de ser considerada a vizinhança espacial de $\boldsymbol{p_i}$, é considerada a vizinhança dos píxeis. Isto é, utiliza-se a estrutura organizada de uma $\mathcal{P}$, adquirida por uma câmara *Time-of-Flight* ou RGB-D em lugar de se procurar entre o espaço 3D que $\mathcal{P}$ compreende;
  - O princípio básico desta abordagem consiste no cálculo de dois vectores tangenciais à superfície local do ponto $\boldsymbol{p_i}$, calculando a normal através do produto externo desses vectores. No entanto, e como os dados capturados pela Kinect têm ruído e regiões onde não existe informação de profundidade, é aplicada a técnica de integração de imagens para a suavização aos vectores tangenciais calculando a média dos vectores de uma certa vizinhança.

4.4.1.2 Caracterização

Nesta análise ao cálculo de normais é utilizada a nuvem de pontos apresentada na figura 4.16 e é caracterizado:

- o tempo de processamento dos 2 métodos. De notar que no caso do método 1 tira-se partido das directivas do OpenMP;

- os resultados finais da estimação das normais em nuvens capturadas pela Kinect em termos qualitativos.

Figura 4.16: Nuvem de pontos $\mathcal{P}$ utilizada no cálculo das normais.

Tabela 4.13: Caracterização do método 1 - *Normal Estimation*.

Teste	*raio (m)*	*exec. (ms)*
1	$0,01$	1.269 ± 166
2	$0,02$	3.122 ± 203
3	$0,03$	6.056 ± 296
4	$0,04$	10.460 ± 376
5	$0,05$	15.789 ± 513
6	$0,10$	61.481 ± 3772

Tabela 4.14: Caracterização do método 2 - *Integral Images*.

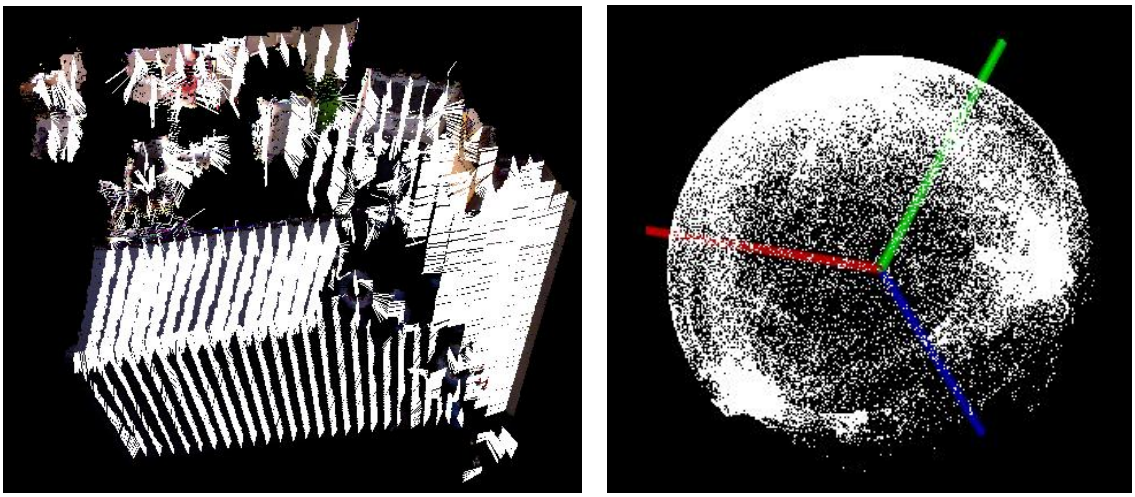
Teste	*smoothing-size*	*max-depth-change-factor*	*exec. (ms)*	*observações*
1	$10,0$	$0,02$	264 ± 30	grandes desvios nas orientações das normais
2	$15,0$	$0,02$	241 ± 11	menos desvios que o teste anterior
3	$20,0$	$0,02$	248 ± 18	em distâncias maiores existem desvios nas orientações das normais
4	$30,0$	$0,02$	233 ± 8	orientações das normais parecem mais consistentes
5	$20,0$	$0,001$	199 ± 12	maus resultados - normais calculadas de forma dispersa

4.4.1.3 Discussão de resultados

Os resultados mostrados nas tabelas 4.13 e 4.14, mostram claramente que com o método 2 obtém-se uma maior rapidez de processamento no cálculo das normais.

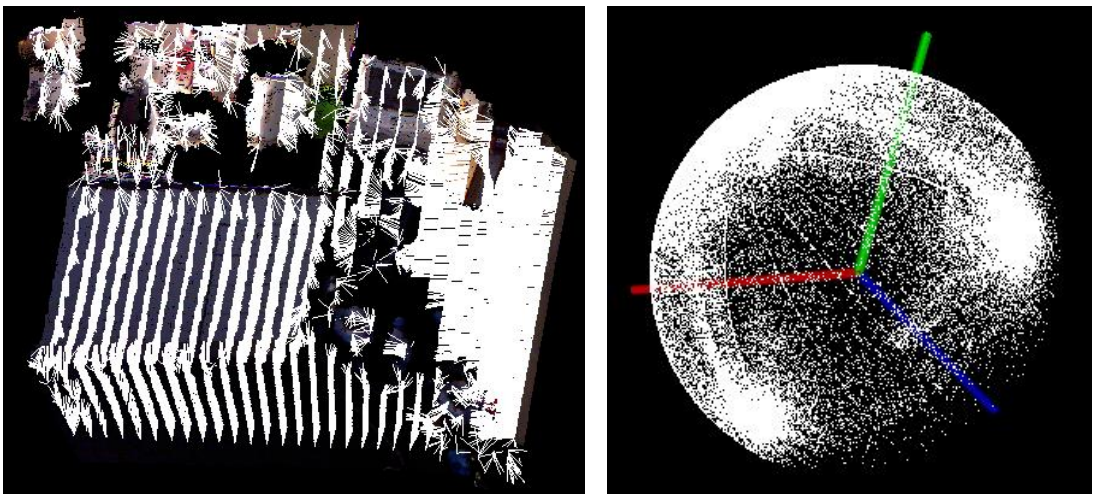
Através da análise das *Extended Gaussian Images* (EGI) [Hor84] presentes nas imagens 4.17 e 4.18, nota-se uma variação maior das orientações das normais calculadas pelo método 2, representando os desvios que este método produz tal como o reportado em [HHRB11]. Isto deve-se ao facto de que o algoritmo não consegue tratar convenientemente cantos, arestas, limites e buracos nas nuvens, como se pode verificar na figura 4.19.

No método 1, a variável $\boldsymbol{r}$ (ou $\boldsymbol{k}$) influencia significativamente o cálculo das normais. Isto acontece porque a escolha de um valor muito alto de $\boldsymbol{r}$, algumas estruturas como cantos ou arestas são suavizadas, podendo desaparecer completamente. Por outro lado, a escolha de um valor muito baixo, tem o efeito perverso de serem afectas pelo ruído presente nas nuvens. Tendo os parâmetros bem definidos, este método produz resultados melhores do que o algoritmo do método 2, mas é significativamente mais lento, sendo que para aplicações em tempo real esta propriedade é de extrema importância para, por exemplo, a rápida identificação de obstáculos no planeamento de navegação de sistemas autónomos.



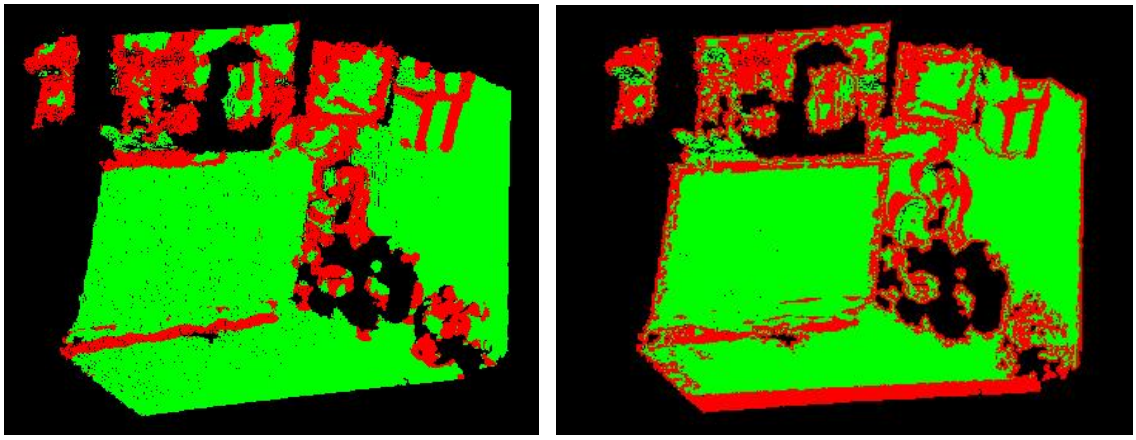
(a) Normais (b) EGI

Figura 4.17: Resultado da aplicação do método 1 no cálculo de normais, e respectiva EGI representando a distribuição das orientações das normais - teste 5 da tabela 4.13.



(a) Normais (b) EGI

Figura 4.18: Resultado da aplicação do método 2 no cálculo das normais, e respectiva EGI representando a distribuição das orientações das normais - teste 5 da tabela 4.14.



(a) método 1 (b) método 2

Figura 4.19: Curvatura das nuvens, a verde valores de curvatura inferior a 0,045, a vermelho valores de curvatura superior ou igual a 0,045.

4.4.2 Descritores geométricos locais

Nesta secção são caracterizados alguns dos descritores geométricos locais que existem no PCL e que se relacionam com o trabalho realizado nesta dissertação.

4.4.2.1 Fundamentos teóricos e trabalho relacionado

A estimação de normais e curvatura, parcialmente descrito na secção 4.4.1, são os métodos mais utilizados no processo de segmentação de cenas e na descrição geométrica local de um ponto $\boldsymbol{p}$ e da sua vizinhança no processo de registo. Essa representação embora de fácil cálculo e de pouco "peso" computacional, é um pouco básica, sendo muito sensível ao ruído inerente dos dados capturados pelos sensores. O espaço, é também, caracterizado com pouco detalhe, aproximando a geometria da vizinhança $\boldsymbol{\mathcal{P}^k}$ de $\boldsymbol{p}$ com apenas alguns valores. Como consequência deste último facto, a maioria das cenas irão conter pontos com as mesmas características.

Para debelar este problema, com especial ênfase no registo de nuvens de pontos no processo de se encontrar correspondências entre pontos, foram criados algoritmos que descrevem melhor a geometria local de um ponto e correspondente vizinhança de pontos. Os descritores FPH [RBMB08, RMBB08], FPFH [RBB09] e SHOT [TSDS10, TSdS11] são alguns dos mais recentes desenvolvimentos nesta área particular que se irá caracterizar.

4.4.2.1.1 PFH

O objectivo do descritor PFH é o de codificar as propriedades geométricas da

vizinhança de um dado ponto $\boldsymbol{p}$, generalizando as estimações das normais das superfícies e de curvatura em seu redor, utilizando histogramas de valores multi dimensionais.

Tendo dois pontos, $\boldsymbol{p}$ e $\boldsymbol{q}$, é construído um referencial fixo definido por 3 vectores unitários (u, v, w), cujo centro se encontra no ponto $\boldsymbol{p}$. Este referencial é construído seguindo o procedimento:

1. o vector $\boldsymbol{u}$ é a normal no ponto $\boldsymbol{p}$;
2. $v = u \times \frac{p-q}{d}$, onde $d = \|p - q\|_2$;
3. $w = u \times v$;

A diferença entre as normais em $p(n_p)$ e $q(n_q)$, servindo-se do referencial construído, é representado por:

1. $\alpha = \arccos(v \cdot n_p)$
2. $\phi = \arccos(u \cdot (p - q)/d)$
3. $\theta = \arctan(w \cdot n_p, u \cdot n_p)$

Os ângulos $\boldsymbol{\alpha}$, $\boldsymbol{\phi}$ e $\boldsymbol{\theta}$ são calculados para todos os pares na vizinhança do ponto $\boldsymbol{p}$. O valor de $\boldsymbol{d}$ não é guardado.

Os valores dos ângulos são armazenados num histograma de 125 posições (5^3), considerando que cada um deles pode cair numa das 5 distintas sub-divisões (*quantum bins*) e que o histograma final codifica a posição com uma combinação única de valores distintos para cada um dos ângulos.

4.4.2.1.2 FPFH

O algoritmo FPFH é uma simplificação do descritor PFH, reduzindo a complexidade computacional de $\boldsymbol{O(nk^2)}$ para $\boldsymbol{O(nk)}$. O primeiro passo do algoritmo é semelhante ao do PFH, tendo-se que calcular o histograma dos três ângulos entre um ponto $\boldsymbol{p}$ e e os seus pontos vizinhos $\boldsymbol{q_k}$. Este passo produz o *Simplified Point Feature Histogram* (SPFH). Seguidamente, para cada ponto $\boldsymbol{p}$, os valores de SPFH dos seus $\boldsymbol{k}$ vizinhos são pesados através da distância $\boldsymbol{w_i = d = \|p - q_k\|_2}$ a $\boldsymbol{p}$, para produzir o histograma final FPFH de $\boldsymbol{p}$ da seguinte maneira:

$$FPFH(p) = SPFH(p) + {}^1/_k \Sigma_{i=1}^{k} SPFH(p_k)/w_i \tag{4.8}$$

Cada um dos três ângulos é armazenado num histograma de 11 posições que serão depois concatenados num único histograma FPFH de 33 posições.

4.4.2.1.3 SHOT

O descritor SHOT baseia-se na obtenção de um referencial repetitivo, servindo-se da decomposição de valores próprios em torno de um ponto. Dado este referencial, uma grelha esférica centrada no ponto divide os pontos vizinhos, de modo a que em cada posição da grelha seja obtido um histograma pesado de normais. O descritor concatena todos estes histogramas numa assinatura final, codificando cada ponto com um total de 352 valores.

4.4.2.2 Caracterização

Os 3 métodos dependem essencialmente de 4 parâmetros:

- da nuvem de pontos de entrada no algoritmo (ex: pontos de interesse);
- da superfície que servirá como base para as operações de procura dos vizinhos mais próximos (ex: nuvem de pontos original);
- das normais previamente calculadas;
- e de um raio (*radius Search*) que será utilizado na determinação dos pontos mais próximos a serem usados na estimação das características de $\boldsymbol{p}$, ou seja, na sua descrição geométrica local.

Os pontos que serão descritos geometricamente e servirão como entrada de dados, são os pontos de interesse detectados pelo método SIFT caracterizado na secção 4.3. Para o cálculo das normais é utilizado o método de estimação de normais através da definição de um raio $\boldsymbol{r}$, caracterizado na secção 4.4.1. Este último método, embora mais lento, representa um pouco melhor as características da superfície, daí a sua utilização.

Antes da estimação das normais, detecção de pontos de interesse e estimação dos descritores geométricos são também removidos os *outliers* da nuvem, suprimindo 1% dos pontos válidos da nuvem original.

Nesta caracterização apenas se irá tomar em consideração o tempo de processamento dos três métodos, pois a análise qualitativa do desempenho dos mesmos só pode ser analisada na secção 4.5, onde se analisa todo o processo de alinhamento das nuvens de pontos.

De notar que o processamento dos descritores FPFH e SHOT é efectuado utilizando as directivas de processamento do OpenMP. Os testes são efectuados sobre a nuvem de pontos ilustrada na figura 4.16.

Tabela 4.15: Caracterização do descritor geométrico PFH.

Condições Iniciais			**saída**	
raio normais (m)	***nº keypoints***	***radius Search (m)***	***exec. PFH (min)***	***global (min)***
$0,05$	2.802	$0,10$	$22,93$	$23,61$
		$0,20$	$318,62$	$319,33$

Tabela 4.16: Caracterização do descritor geométrico PFH em combinação com o filtro Voxel Grid.

Condições iniciais				**saída**	
leaf-size	***raio normais (m)***	***nº keypoints***	***radiusSearch (m)***	***exec. PFH (ms)***	***global (ms)***
$0,02$	$0,05$	1.486	$0,10$	3.979 ± 10	10.458 ± 26
			$0,20$	54.654 ± 30	61.361 ± 302

Tabela 4.17: Caracterização do descritor geométrico FPFH.

Condições iniciais			**saída**	
raio normais (m)	***nº keypoints***	***radius Search (m)***	***exec. FPFH (ms)***	***global (ms)***
$0,05$	2.802	$0,10$	93.701 ± 3.885	102.305 ± 4.011
		$0,20$	538.710	582.470

Tabela 4.18: Caracterização do descritor geométrico FPFH em combinação com o filtro Voxel Grid.

Condições iniciais				**saída**	
leaf-size	***raio normais (m)***	***nº keypoints***	***radius Search (m)***	***exec. FPFH (ms)***	***global (ms)***
0.02	$0,05$	1.486	$0,10$	444 ± 35	6.758 ± 52
			$0,20$	1.997 ± 15	8.362 ± 5

Tabela 4.19: Caracterização do descritor geométrico SHOT.

Condições iniciais			**saída**	
raio normais (m)	***nº keypoints***	***radiusSearch (m)***	***exec. SHOT (ms)***	***global (ms)***
$0,05$	2.802	$0,10$	2.030 ± 28	40.850 ± 103
		$0,20$	7.551 ± 60	44.240 ± 130

4.4.2.3 Discussão de resultados

As tabelas 4.15, 4.16, 4.17, 4.18 e 4.19 mostram os resultados dos 3 descritores de geometria local implementados. De notar que não possível efectuar o teste do descritor SHOT em combinação com o filtro *Voxel Grid* porque a actual versão do PCL contém um *bug* na implementação deste método, que foi devidamente reportado à comunidade que desenvolve o PCL.

Tal como o reportado em [RBB09] verifica-se uma significativa diferença no tempo de processamento entre o PFH e o FPFH, sendo o último muito mais rápido a retornar todos os valores necessários dos histogramas de cada ponto de interesse analisado. Isto deve-se ao facto de o PFH necessitar de codificar um ponto com 125 valores, enquanto que o FPFH necessita apenas de 33 valores.

Um exemplo dos histogramas criados na descrição geométrica do mesmo ponto de interesse, por estes dois métodos, pode ser visualizado na figura 4.20.

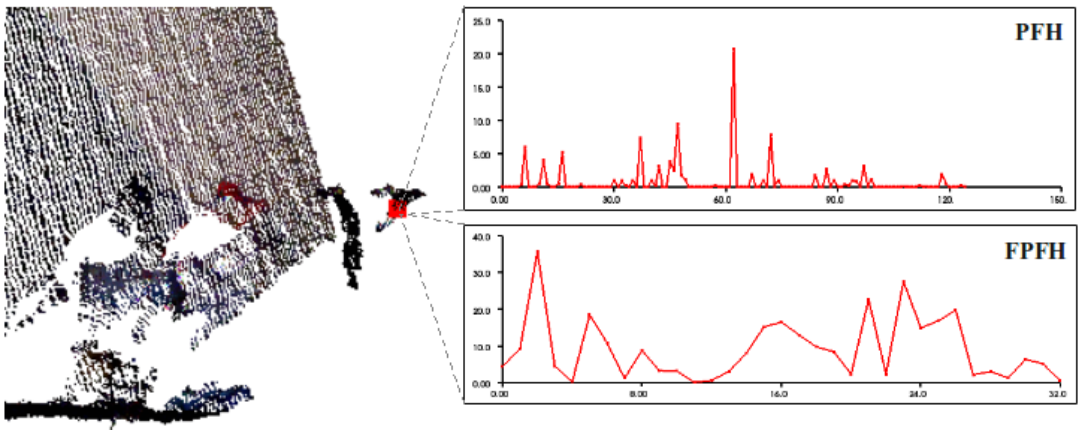


Figura 4.20: Histogramas PFH e FPFH de um ponto de interesse de $\mathcal{P}$.

Verifica-se também que, sem a utilização do filtro *Voxel Grid*, o descritor SHOT tem um desempenho computacional bastante melhor do que os dois outros descritores, mas devido a um *bug* na sua implementação não é possível executar testes mais exaustivos à sua performance na posterior caracterização dos métodos de registo de nuvens de pontos.

Com a combinação com o método *Voxel Grid* verifica-se um incremento do desempenho do descritor FPFH, o que pode indicar a possibilidade da sua utilização em operações de alinhamento de nuvens em tempo real, tendo uma melhor configuração de *hardware*.

4.5 Módulo *registration*

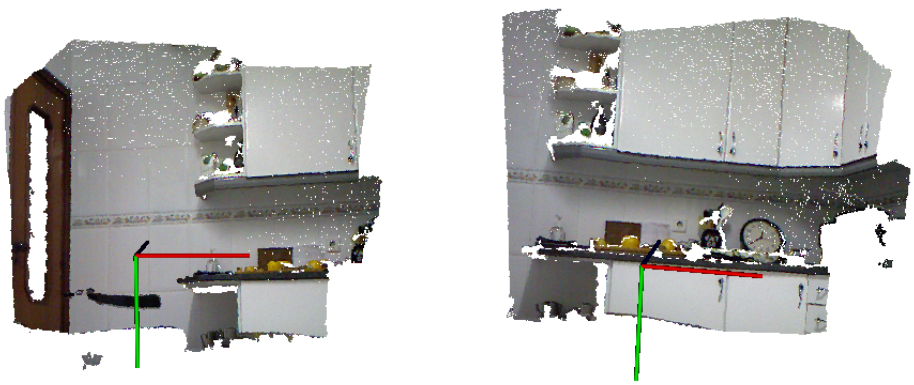
Neste módulo irão ser caracterizados os métodos que incorporam o procedimento que leva ao registo de nuvens de pontos, alinhamento inicial e refinamento da estimação das transformações geométricas entre nuvens de pontos.

Para que o registo de nuvens de pontos seja executado, é antes seguido um procedimento que inclui métodos caracterizados nas secções anteriores. Os métodos utilizados seguem a seguinte ordem:

1. remoção de *outliers*;
2. *downsampling*;
3. estimação das normais;
4. detecção de pontos de interesse;
5. descrição geométrica de cada ponto de interesse.

Este procedimento irá ser aprofundado no capítulo 5.

Nas figuras 4.21 e 4.22 são apresentadas as nuvens de pontos utilizadas nesta caracterização e o posicionamento inicial das duas nuvens no sistema de coordenadas do mundo respectivamente. Nesta caracterização $\mathcal{P}_2$ (source), representada pela cor verde na figura 4.22, é transformada para o referencial de coordenadas de $\mathcal{P}_1$ (target), representada pela cor vermelha.



(a) nuvem de pontos $\mathcal{P}_1$ (b) nuvem de pontos $\mathcal{P}_2$

Figura 4.21: Nuvens de pontos $\mathcal{P}_1$ e $\mathcal{P}_2$ que foram usadas nos testes aos procedimentos que levam ao seu registo.

4.5.1 Fundamentos teóricos e trabalho relacionado

Registo, é o nome que se dá ao processo de alinhamento de nuvens de pontos, capturadas de diferentes posições do espaço correspondendo a uma visão parcial do

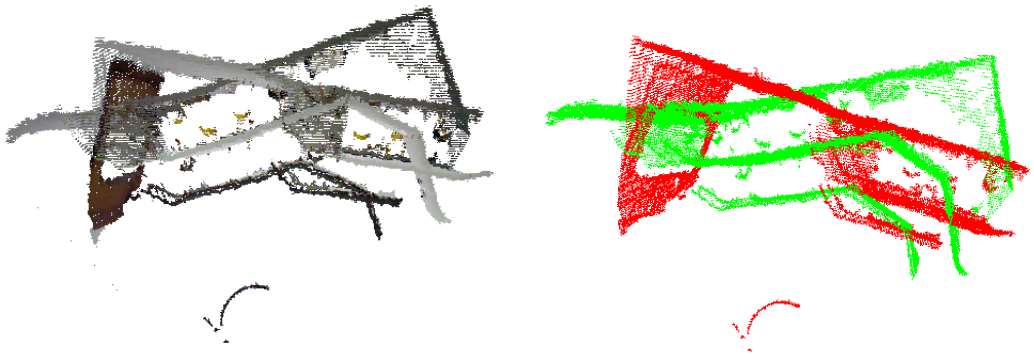


Figura 4.22: Posição original de $\mathcal{P}_1$ (vermelho) e $\mathcal{P}_2$ (verde).

mundo, num modelo global consistente. O seu objectivo é encontrar a posição e orientação de cada modelo parcial relativamente ao sistema de coordenadas do mundo fundindo-as num único sistema de coordenadas, por forma a que as áreas repetidas nos vários modelo parciais se sobreponham perfeitamente e que a informação adicional seja adicionada.

O registo de duas nuvens de pontos é conseguido através da existência de pontos correspondentes nas 2 nuvens, que servem para estimar os parâmetros da transformação rígida geométrica $\boldsymbol{T}$ de uma nuvem em relação à outra.

Para 6 DOF, essa matriz $\boldsymbol{T}$ (4.9), é composta por 3 eixos de rotação e translação, e representada por uma matriz 4x4 com coordenadas homogéneas, onde $\boldsymbol{R}$ é a matriz 3x3 de rotação e $\boldsymbol{t}$ o vector de translação.

$$T = \begin{bmatrix} & & & t_x \\ & R & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

Se a posição inicial (POS_1) de $\mathcal{P}_1$ for definida pela orientação inicial da câmara em cada eixo, com rotação inicial R_1 e translação inicial $t_1 = (x_1, y_1, z_1)^T$, então a posição inicial é dada pela matriz identidade:

$$POS_1 = I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

Os pontos de $\mathcal{P}_2$, são transformados para o sistema de coordenadas de $\mathcal{P}_1$, através da equação:

$$POS_2 = POS_1 T \tag{4.11}$$

Isto significa que a descrição geométrica de uma lista de pontos $p_i \in \mathcal{P}_1$ tem de coincidir com outra lista de pontos $p_j \in \mathcal{P}_2$, onde $\mathcal{P}_1$ e $\mathcal{P}_2$ são duas representações parciais do mundo. Os conjuntos de pontos coincidentes p_i e p_j são então, representações das mesmas superfícies amostradas. Se as duas nuvens não tiverem zonas de sobreposição, não é possível realizar este processo. De forma geral a documentação revela que essa sobreposição para a obtenção de bons resultados deverá ser de pelo menos 30%.

Um dos métodos mais populares métodos de registo é o algoritmo *Iterative Closest Point* (ICP) [BM92, Zha94]. Infelizmente o ICP sofre de algumas desvantagens, como o facto de necessitar de um grande número de iterações, trazendo um grande peso computacional até que atinja o critério de convergência. Ao longo dos anos foram apresentadas algumas soluções na tentativa de debelar esta desvantagem, processos esses que se propõem a encontrar uma estimativa grosseira da transformação rígida entre duas nuvens. Esta estimativa inicial da transformação é depois refinada pelo ICP, precisando de menos iterações, comportando um custo computacional muito menor.

De um modo geral o procedimento a ser seguido para a obtenção de uma boa transformação final deve ser:

- **Alinhamento Inicial** - encontrar a transformação inicial grosseira de uma nuvem de pontos num posição inicial arbitrária em relação às coordenadas de outra nuvem, que representa o referencial do mundo;

- **Refinamento** - refinar a solução encontrada no alinhamento inicial;

4.5.1.1 Alinhamento inicial

Duas das formulações mais recentes para a estimação inicial da transformação entre 2 nuvens podem ser encontradas em [RBMB08] e em [RBB09]. Em [RBMB08] é apresentado o método *Greedy Inicial Alignment* (GIA), mas que revela ainda uma complexidade computacional relativamente grande. Em [RBB09] é descrito um método baseado na formulação SAC denominado SAC-IA, que se irá aqui caracterizar.

No anexo A, é descrito o algoritmo RANSAC e a sua utilização nestes métodos.

Para a estimação da transformação rígida, o algoritmo **SAC-IA** segue os seguintes passos:

1. seleccionar $\boldsymbol{s}$ amostras de pontos de $\boldsymbol{\mathcal{P}_1}$, tendo em conta que a distância entre os pontos terá de ser maior do que uma distância mínima definida pelo utilizador

2. para cada um dos pontos amostrados, encontrar uma lista de pontos em $\boldsymbol{\mathcal{P}_2}$

cujos histogramas são similares aos histogramas dos pontos amostrados. Destes, um é aleatoriamente seleccionado sendo considerado a correspondência para o ponto amostrado.

3. calcular a transformação rígida definida pelos pontos amostrados e as suas correspondências.

Para além do algoritmo SAC-IA, a última versão do PCL trouxe incorporados no módulo de registo novos métodos para a estimação da transformação inicial, que englobam métodos de estimação das correspondências reciprocas de pontos que tenham relações geométricas semelhantes nas duas nuvens analisadas, métodos de rejeição de correspondências e de estimação da transformação entre correspondências. Aqui utilizaram-se dois desses métodos, dando-lhe o nome de **CE-RSAC**, podendo ser descrito segundo os seguintes passos:

1. estimação das correspondências dos pontos de interesse de duas nuvens de pontos, a *source* $\boldsymbol{\mathcal{P}_2}$ e o *target* $\boldsymbol{\mathcal{P}_1}$. Estas correspondências baseiam-se na geometria calculada pelos descritores geométricos;

2. rejeitar más correspondências anteriormente calculadas e estimar a transformação entre as correspondências boas;

   - método SAC - a rejeição das correspondências é feita utilizando o *Random Sample Consensus* para identificar *inliers* e rejeitar *outliers* baseado-se numa distância mínima $\boldsymbol{d_{min}}$ entre pontos correspondentes (*threshold*) definida pelo utilizador. O método é iterado $\boldsymbol{n}$ vezes e a melhor estimação da transformação entre os pontos correspondentes é retornada.

4.5.1.2 Refinamento

O ICP, é um método iterativo que permite encontrar a transformação rígida entre duas nuvens de pontos através da minimização métrica do erro da distância entre áreas que se sobrepõem.

Para o refinamento da transformação entre as duas nuvens, existem no PCL, duas formulações diferentes do ICP: ICP linear baseada na formulação de Besl que se serve da optimização da transformação SVD [AHB87] e o ICP não linear [Rus09] que faz uso da optimização do algoritmo Levenberg-Marquardt similar a [Fit01].

Considerando-se p_i e p_j pontos correspondentes entre duas nuvens, a minimização do erro das distâncias entre eles, no caso do ICP linear, é dada pela função de erro 4.12 e a medida da distância por 4.13.

$$\min \sum_{i=1}^{n} dist\left(R \cdot p_j + t, p_i\right) \tag{4.12}$$

$$\sum_{i=1}^{n} \|R \cdot p_j + t - p_i\|^2 \tag{4.13}$$

No ICP não linear a função de erro utiliza a aproximação da medida da distância entre pontos e superfícies [PLH04] e a medida da distância definida em 4.13, é transformada para:

$$\sum_{i=1}^{n} \left\|(R.p_i + T - p_j) \cdot \vec{n}_{p_j}\right\| \tag{4.14}$$

onde $\vec{n}_{p_j}$ é a normal da superfície no ponto p_j.

4.5.2 Caracterização do alinhamento inicial

Tabela 4.20: Caracterização do método SAC-IA.

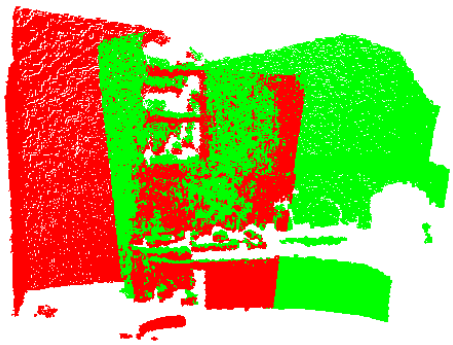
Condições iniciais											**saída**
Teste	***leaf-size (m)***	***nº pontos $\mathcal{P}_1$***	***nº pontos $\mathcal{P}_2$***	***raio normais (m)***	***nº keypts $\mathcal{P}_1$***	***nº keypts $\mathcal{P}_2$***	***raio features (m)***	***min sample dist (m)***	***th-reshold (m)***	***nº iter.***	***exec. (ms)***
1	0,01	58.447	59.246	0,1	2.472	3.333	0,35	0,2	0,05	1.000	10.348 ± 204
2	0,02	17.793	17.606	0,1	1.632	1.987	0,35	0,1	0,05	1.000	5.068 ± 179
3	0,03	8.123	8.133	0,1	960	1.179	0,35	0,2	0,05	1.000	2.354 ± 98
4								0,1	0,05	1.000	5.105 ± 165
5								0,1	0,01	1.000	2.474 ± 80
6								0,1	0,01	100	248 ± 12

Tabela 4.21: Caracterização do método CE-RSAC.

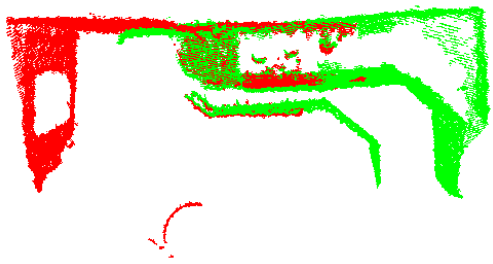
Condições iniciais									**saída**		
Teste	***leaf-size (m)***	***nº pontos $\mathcal{P}_1$***	***nº pontos $\mathcal{P}_2$***	***raio normais (m)***	***nº keypts $\mathcal{P}_1$***	***nº keypts $\mathcal{P}_2$***	***raio features (m)***	***th-reshold (m)***	***nº corresp. estima-das***	***nº de corresp. boas***	***exec. (ms)***
1	0,01	58.447	59.246	0,07	2.472	3.333	0.35	0,5	275	215	201 ± 5
2				0,1				0,05		134	207 ± 4
3	0,02	17.793	17.606	0,07	1.632	1.987	0,35	0,04	194	141	231 ± 6
4				0,1				0,1	194	119	99 ± 2
5				0,08				0,1	178	103	121 ± 3
6				0,1				0,01	194	13	244 ± 9
7				0,1				0,05	194	94	113 ± 2
8	0,03	8.123	8.133	0,1	960	1.179	0,40	0,1	108	84	53 ± 1
9							0,35	0,02	116	17	49 ± 2
10								0,07	116	58	42 ± 1
11	0,04	4.649	4.678	0,1	719	821	0,35	0,05	73	16	28

Na figura 4.23 é mostrado o resultado final do alinhamento usando o método SAC-IA com os parâmetros do teste 5 definidos na tabela 4.20.

Nas figuras 4.24 e 4.25 são mostrados os resultados do teste 10 (tabela 4.21), para o método CE-RSAC. São ilustradas as correspondências estimadas, as correspondências ditas boas que foram usadas para estimar a transformação de $\mathcal{P}_2$, e o resultado final da aplicação da transformada estimada para alinhar $\mathcal{P}_2$ com $\mathcal{P}_1$.

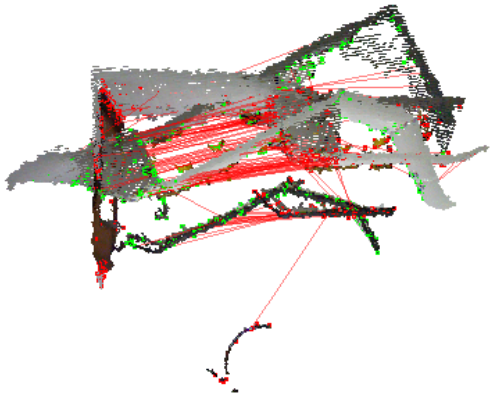


(a) $\mathcal{P}_1$ e $\mathcal{P}_2$ alinhadas - vista de frente

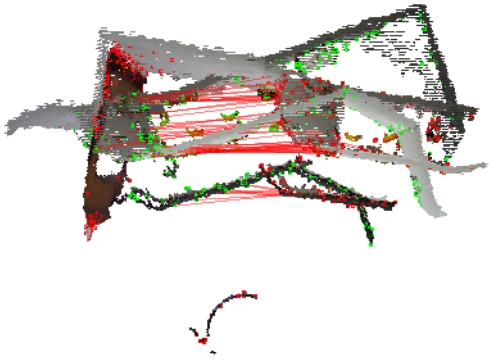


(b) $\mathcal{P}_1$ e $\mathcal{P}_2$ alinhadas - vista de cima

Figura 4.23: Visualização de $\mathcal{P}_1$ (target- a vermelho) e $\mathcal{P}_2$ (source - a verde) alinhadas pelo método SAC-IA.

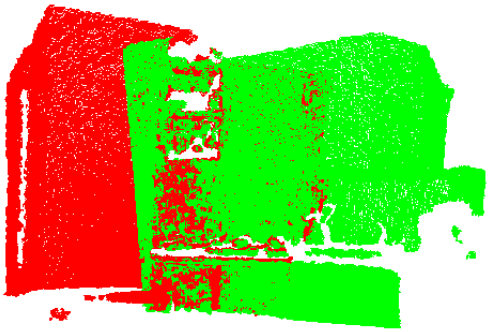


(a) Correspondências estimadas

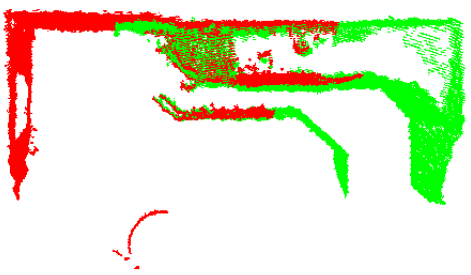


(b) Correspondências boas

Figura 4.24: Visualização das correspondências entre as duas nuvens (linhas vermelhas) e pontos de interesse das duas nuvens: source (verde) e target (vermelho) - método CE-RSAC.



(a) $\mathcal{P}_1$ e $\mathcal{P}_2$ alinhadas - vista de frente



(b) $\mathcal{P}_1$ e $\mathcal{P}_2$ alinhadas - vista de cima

Figura 4.25: Visualização de $\mathcal{P}_1$ (target- a vermelho) e $\mathcal{P}_2$ (source - a verde) alinhadas pelo método CE-RSAC.

4.5.2.1 Discussão de resultados

As tabelas 4.20 e 4.21 apresentam os dados de alguns dos melhores resultados conseguidos dos extensos testes efectuados.

Verificou-se que a diminuição do número de pontos das nuvens através do uso do *downsampling* não influência negativamente a qualidade dos resultados finais, tendo sim, uma influencia muito grande na velocidade de processamento em todo o processo que leva ao alinhamento das nuvens.

De facto, a única desvantagem do uso do filtro *downsampling* neste processo é de que quando mais espaço cada voxel ocupa, menor é o número de pontos da nuvem à saída do filtro. Este facto, leva a que seja menor o nível de detalhe da nuvem analisada, o que dificulta a definição de todos os parâmetros que levam ao registo. Com isto conclui-se, que se deverá atingir um equilibro entre a perda de detalhe da nuvem e a velocidade de processamento pretendida, tendo em conta que os valores definidos como óptimos para um cenário podem divergir um pouco noutro tipo de cenário.

Uma boa estimação das normais e especialmente uma boa descrição geométrica dos pontos de interesse tem uma influência muito grande na estimação das correspondências entre pontos. Através da realização de vários testes determinou-se, para este cenário, que para a estimação das normais deveria ser utilizado um raio de 0,1 metros e 0,35 metros para o raio que descreve geometricamente um ponto. Se para esta última variável for definido um valor um pouco mais baixo ou um pouco mais alto, verifica-se a existência de mais falsos positivos na estimação das correspondências, produzindo piores resultados no processo de alinhamento.

Para o método SAC-IA verificou-se que para que se obtenham bons resultados deverão, neste caso, realizar-se 1.000 iterações, com uma distância mínima entre *samples* de 0,2 metros e uma distância máxima entre correspondências de 0,05 metros.

No que toca à variável do método CE-RSAC para o maior erro permitido entre pontos correspondentes determinou-se, perante múltiplos testes efectuados que um valor entre 0,05 e 0,07 metros, é um valor aceitável para que o método consiga convergir para um bom resultado. Um valor menor, produz menos correspondências boas entre pontos, o que pode levar noutros cenários a que o método não encontre uma boa ou até nenhuma solução. Um valor mais alto leva a que mais correspondências sejam consideradas boas, o que em cenários com poucos pontos de interesse detectados pode ser importante, mas leva também a piores resultados finais no alinhamento, como se verificou em alguns dos testes.

Em termos qualitativos os dois métodos produzem boas estimações da transformação inicial, mas o método CE-RSAC é mais eficiente, necessitando em geral, de

menos tempo de processamento para produzir bons resultados.

4.5.3 Caracterização do refinamento - ICP

As nuvens de pontos que serviram como entrada aos algoritmos, são as que resultam depois de serem removidos os *outliers* e pontos inválidos, pois estes métodos não aceitam como entrada, pontos com valores inválidos. Como parâmetros os dois algoritmos necessitam da definição de: a distância máxima entre dois pontos correspondentes; o *threshold* para a definição de *outliers* no ciclo interno de rejeição de *outliers* RANSAC; a variável epsilon (ϵ) que representa a transformação mais pequena permitida até que o algoritmo convirja (critério de convergência); e o número máximo de iterações que o método ICP poderá executar, mesmo que não atinja a convergência.

Tabela 4.22: Caracterização do método ICP linear.

Condições Iniciais					**saída**	
teste	***max corresp. dist. (m)***	***outlier rej. thresh (m)***	***ϵ (epsilon)***	***Iterações ICP***	***Fitness score***	***exec. (ms)***
1	0,01	0,01	0,00001	100	0,118968	470.765
2	0,1	0,04	0,00000001	1.000	0,12608	5.551.260

Tabela 4.23: Caracterização do método ICP não linear.

Condições Inciais					**saída**	
teste	***max corresp. dist. (m)***	***outlier rej. thresh (m)***	***ϵ (epsilon)***	***Iterações ICP***	***Fitness score***	***exec. (ms)***
1	0,1	0,01	0,00001	100	0,133164	61.006
2	0,2	0,07	0,00000001	500	0,12608	103.235
3	0,1	0,07	0,00000001	500	0,12608	204.429
4	0,1	0,05	0,00000001	1.000	0,124213	159.031
5	0,1	0,04	0,00000001	1.000	0,125104	127.333

4.5.3.1 Discussão de resultados

O ICP tem um grande peso computacional. Para se provar que os procedimentos seguidos nas secções anteriores, nomeadamente os métodos de alinhamento inicial (SAC-IA e CE-RSAC), são necessários para que o registo de nuvens de pontos seja o mais rápido possível, testou-se a realização do registo com recurso apenas ao ICP, ou seja, sem nenhum tratamento prévio à nuvem a não ser a remoção de *outliers*. Neste teste e para que fossem atingidos resultados satisfatórios, o tempo de execução foi de sensivelmente 3 horas. Este tempo de execução prova de sobremaneira,

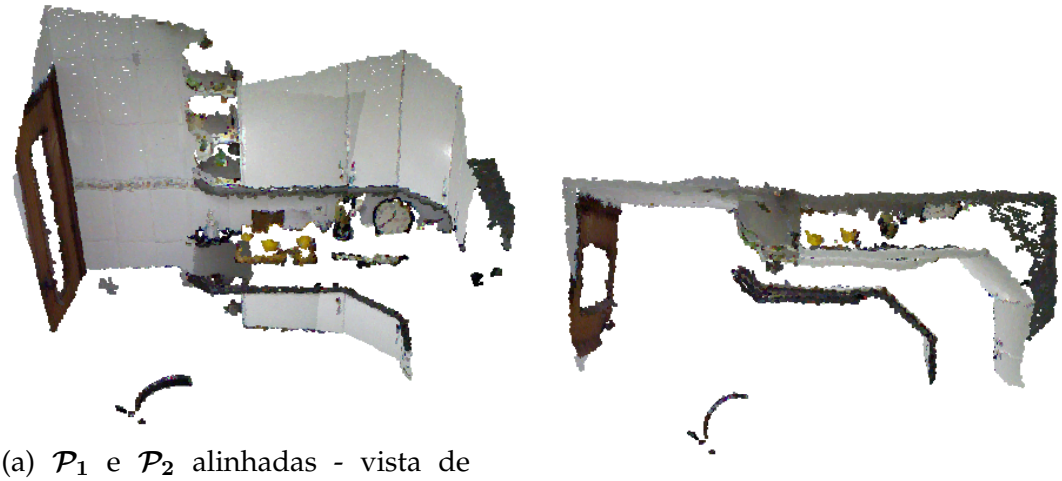
que os procedimentos utilizados, promovem um significativo aumento do desempenho computacional.

Em termos qualitativos dos resultados finais, os dois algoritmos têm comportamentos semelhantes como se pode verificar nas figuras 4.26 e 4.27, onde estão mostrados os resultados do teste 2 do ICP linear (tabela 4.22) e do teste 5 do ICP não linear (tabela 4.23).

Em termos de velocidade de processamento, observa-se, que o ICP linear tem um comportamento bastante pior do que o ICP não linear, como se pode concluir dos tempos de execução indicados nas tabelas 4.22 e 4.23.

Como nota, deve-se referir que a avaliação da qualidade dos resultados da refinação é difícil, devido à natureza inerentemente ruidosa das nuvens de pontos produzidas pela Kinect. Uma das métricas que se podem usar para a verificação da convergência, é a soma euclidiana dos quadrados das distâncias (SSD) das correspondências entre a *source* e o *target*, representada nas tabelas pela variável *fitness score*.

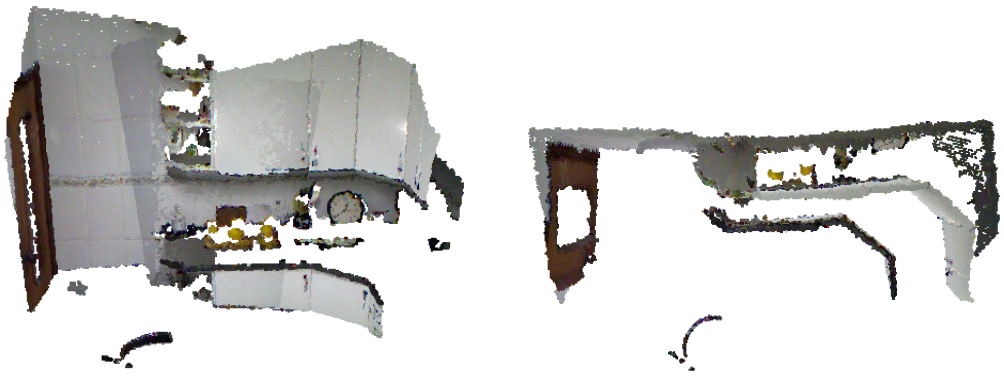
Embora esta métrica possa ser utilizada como uma constatação da convergência entre as duas nuvens de pontos, a validação final terá de ser sempre feita visualmente, pois o facto de que a soma dos quadrados das distâncias entre correspondências tenha produzido um valor aceitável, nem sempre corresponde a um bom alinhamento. Isto deve-se ao facto de que numa das iterações o método tenha ficado preso num mínimo local, que não corresponde à melhor estimação possível da transformação.



(a) $\mathcal{P}_1$ e $\mathcal{P}_2$ alinhadas - vista de frente

(b) $\mathcal{P}_1$ e $\mathcal{P}_2$ alinhadas - vista de cima

Figura 4.26: Visualização de $\mathcal{P}_1$ e $\mathcal{P}_2$, tendo o seu alinhamento sido refinado pelo método ICP linear.



(a) $\mathcal{P}_1$ e $\mathcal{P}_2$ alinhadas - vista de frente

(b) $\mathcal{P}_1$ e $\mathcal{P}_2$ alinhadas - vista de cima

Figura 4.27: Visualização de $\mathcal{P}_1$ e $\mathcal{P}_2$, tendo o seu alinhamento sido refinado pelo método ICP não linear.

4.6 Módulo *surface*

Este módulo contém métodos de reconstrução de superfícies, como RMLS, *Marching Cubes*, etc. Nesta secção apenas se irá caracterizar o método RMLS. Os outros algoritmos presentes neste módulo, dizem respeito a formulações de reconstrução das nuvens de pontos em *meshes* ou volumes.

4.6.1 Robust Moving Least Squares

Este método é baseado no algoritmo *Robust Moving Least Squares* (RMLS) proposto em [RBM+07] (descrito mais ao pormenor em [Rus09]), tratando-se de uma técnica de *resampling* que pode realizar filtragem de ruído e suavização de superfícies.

4.6.1.1 Fundamentos teóricos

Durante o processo de aquisição de dados é normal existirem erros de medida ou ruído. As superfícies amostradas terão então *outliers* e buracos resultantes de áreas amostradas que não retornaram qualquer medida de profundidade, mais, depois do processo de registo de nuvens (ver secção 4.5), o modelo global obtido pode ter regiões sobrepostas. Para além dos *outliers* detectados pelos métodos descritos na secção 4.2.2, existem ainda pequenos erros de medida fazendo com que a superfície pareça espessa. Estas anomalias podem ser minimizadas recorrendo ao algoritmo RMLS.

O algoritmo RMLS pode ser resumido da seguinte forma:

1. As coordenadas dos pontos de uma nuvem $\mathcal{P}$ são normalizadas.

2. Uma primeira estimativa é calculada através da aproximação de $\mathcal{P}$ a um conjunto de pontos $\mathcal{Q}^3$ que se situam na vizinhança dos pontos originais, representando uma superfície.

3. Uma vizinhança local é seleccionada para cada ponto $q_k \in \overline{\mathcal{Q}^3}$ a projectar, utilizando um número fixo de vizinhos ou um raio fixo ($\boldsymbol{r}$).

4. Um plano local de referência é seleccionado para a transformação das coordenadas.

5. O segundo passo do procedimento do MLS é o do ajuste polinomial, onde o sistema de coordenadas xyz é transformado para o sistema de coordenadas uvn, que se situa no plano de referência. Cada ponto $\boldsymbol{q}$ da primeira estimativa $\mathcal{Q}$ é ajustado por uma função polinomial bivariada.

4.6.1.2 Caracterização

As nuvens de pontos usadas para este teste são as ilustradas na figura 4.5.

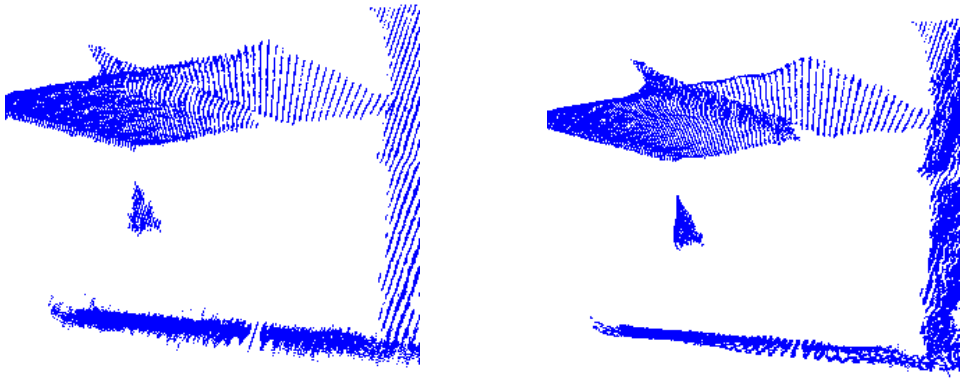
Tabela 4.24: Caracterização do método RMLS

Condições Iniciais		**saída**	
teste	***raio (m)***	$\boldsymbol{\mathcal{P}_1}$ ***exec. (ms)***	$\boldsymbol{\mathcal{P}_2}$ ***exec. (ms)***
1	$0,2$	200.937	807.161
2	$0,1$	46.415	186.188 ± 4.510
3	$0,05$	10.964	45.735 ± 1.324
4	$0,02$	3.010	9.372 ± 203

4.6.1.3 Discussão de resultados

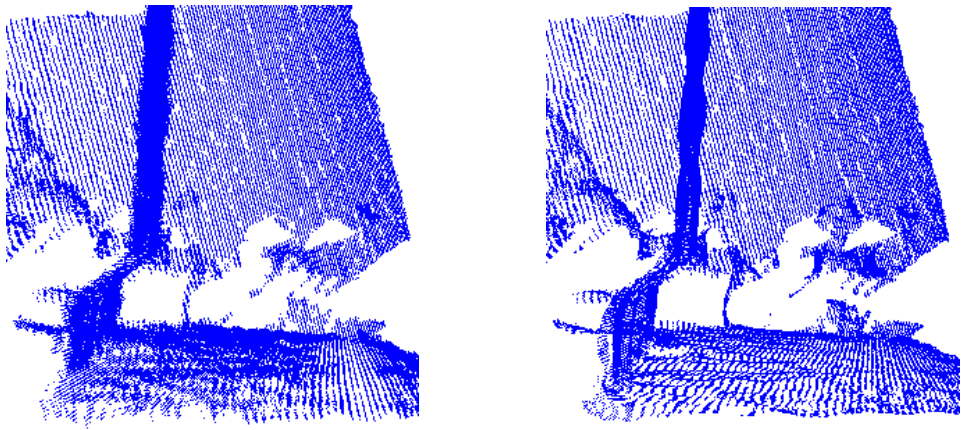
Nas figuras 4.28 e 4.29 é mostrado o resultado da aplicação deste algoritmo em $\boldsymbol{\mathcal{P}_1}$ e $\boldsymbol{\mathcal{P}_2}$, onde claramente se vêem as diferenças, especialmente na espessura dos planos amostrados. O ajuste das superfícies a polinómios de segundo grau produzido por este algoritmo parece dar bons resultados, retirando algum do erro inerente às nuvens de pontos capturadas pela Kinect. A diferença no tempo de processamento pode ser explicada pela densidade de pontos ser maior em $\boldsymbol{\mathcal{P}_2}$ do que em $\boldsymbol{\mathcal{P}_1}$.

De notar, que a implementação deste método no PCL encerra ainda alguns problemas. A informação proveniente da dimensão rgb, não é convenientemente incorporada na nuvem proveniente da saída do método. Foram tentadas soluções que permitissem a resolução deste problema, nomeadamente a cópia dos valores de rgb de cada ponto da nuvem introduzida à entrada do método para a nuvem produzida à sua saída. No entanto, a solução não é a ideal pois alguns dos índices dos pontos durante o processamento são modificados, dificultando a concatenação dos valores no ponto correspondente da nuvem de saída.



(a) vista parcial de $\mathcal{P}_1$ original (b) $\mathcal{P}_1$ com aplicação do RMLS

Figura 4.28: Visualização parcial da aplicação do método RMLS em $\mathcal{P}_1$ com os parâmetros do teste 2.



(a) vista parcial de $\mathcal{P}_2$ original (b) $\mathcal{P}_2$ com aplicação do RMLS

Figura 4.29: Visualização parcial da aplicação do método RMLS em $\mathcal{P}_2$ com os parâmetros do teste 4.

4.7 Módulo *segmentation*

Nesta secção irão ser caracterizados alguns dos métodos implementados no PCL incluindo:

- segmentação de formas paramétricas, planos e cilindros;
- simplificação de modelos geométricos;
- extracção de *clusters*.

Estes métodos permitem que o meio ambiente em que um robô se situa seja interpretado, dando informações relevantes para fins de navegação e manipulação de objectos.

4.7.1 Segmentação de formas paramétricas

A formulação dos métodos apresentados nesta secção baseia-se na descrição efectuada em [Rus09]. Nos testes efectuados é utilizada a nuvem de pontos representada na figura 4.30.

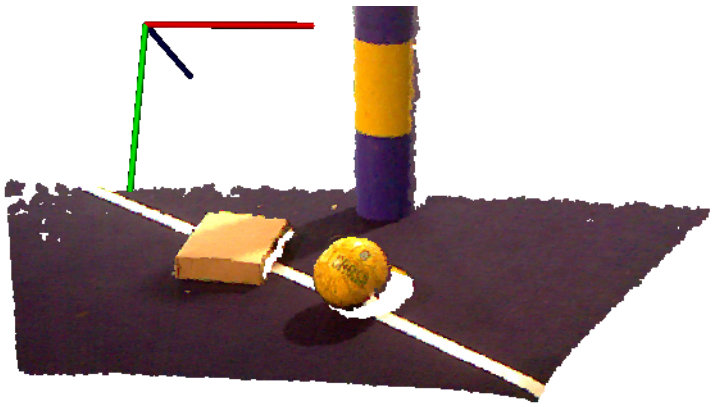


Figura 4.30: Nuvem de pontos $\mathcal{P}$ usada para os testes de segmentação de formas paramétricas.

4.7.1.1 Fundamentos teóricos

Num ambiente não estruturado, o problema de segmentação de superfícies é bastante complexo. Assim sendo, uma das técnicas mais populares para atingir o objectivo de segmentar e simplificar as superfícies amostradas é a da substituição de conjuntos de pontos por primitivas geométricas 3D.

Uma primitiva geométrica é um qualquer modelo básico 3D que possa aproximar os dados amostrados, dentro de um certo erro delimitador. Exemplos deste tipo de modelos, muito usados na computação gráfica, são: linhas, círculos, planos, cilindros, esferas, cones, toróides, etc..

4.7.1.2 Segmentação de planos

Existem dois métodos para segmentar formas paramétricas no PCL, uma depende apenas da nuvem de pontos de entrada e de um parâmetro de *threshold*. Esse *threshold* define que todos os pontos que se encontrem a uma certa distância do modelo paramétrico estimado serão considerados *inliers*, ou seja, farão parte dos pontos extraídos pela segmentação.

O outro método de segmentação depende, não só da variável *threshold*, como também das normais previamente calculadas.

Neste caso, de segmentação de planos, os coeficientes do modelo paramétrico a serem calculados, deverão obedecer à equação 4.15.

$$ax + by + cz + d = 0 \tag{4.15}$$

Tabela 4.25: Segmentação de planos.

Condições Iniciais		**saída**				
teste	***threshold (m)***	***a***	***b***	***c***	***d***	***exec. (ms)***
1	$0,01$	$-0,00168707$	$-0,828996$	$-0,559252$	$1,12305$	93 ± 2
2	$0,02$	$0,00302122$	$-0,830988$	$-0,556282$	$1,12112$	48 ± 1
3	$0,03$	$0,00310209$	$-0,831299$	$-0,555817$	$1,11967$	46 ± 2
4	$0,035$	$0,00315851$	$-0,831092$	$-0,556126$	$1,12007$	44 ± 1

Tabela 4.26: Segmentação de planos com recurso às normais.

entrada			***saída***				
teste	***threshold (m)***	***peso normais***	***a***	***b***	***c***	***d***	***exec. (ms)***
1	$0,01$	$0,1$	$0,00260336$	$-0,832591$	$-0,553882$	$1,11639$	98.055 ± 2.400
2	$0,02$	$0,1$	$0,00261402$	$-0,839513$	$-0,558481$	$1,12372$	54.272 ± 1.896
3	$0,03$	$0,1$	$0,00280287$	$-0,831422$	$-0,555634$	$1,11937$	54.776 ± 1.770
4	$0,035$	$0,1$	$0,00287923$	$-0,831522$	$-0,555485$	$1,11916$	54.141 ± 1.945
5	$0,07$	$0,1$	$0,00430894$	$-0,830844$	$-0,556495$	$1,12024$	36.537 ± 1.291
6	$0,03$	$0,2$	$0,00278073$	$-0,82999$	$-0,557772$	$1,12246$	13.562 ± 309

4.7.1.3 Segmentação de cilindros

Para determinar modelos cilíndricos o algoritmo terá de calcular 7 coeficientes, que são: valores de um dado ponto no seu eixo; a direcção dos eixos; e o raio. Os três primeiros valores dizem respeito às coordenadas (x, y e z) do ponto central do modelo cilíndrico, os três valores seguintes à orientação dos eixos x, y e z, e o último valor ao raio em metros do cilindro. Só é possível a determinação da forma cilíndrica procedendo à estimação das normais, antes da execução do método de segmentação.

(a) Plano segmentado (b) Negativo da segmentação

Figura 4.31: Resultado positivo e negativo da segmentação do plano existente em $\mathcal{P}$ com os parâmetros do teste 3 da tabela 4.25.

(a) Plano segmentado (b) Negativo da segmentação

Figura 4.32: Resultado positivo e negativo da segmentação do plano existente em $\mathcal{P}$ com recurso às normais estimadas e os parâmetros do teste 3 da tabela 4.26.

Tabela 4.27: Segmentação de cilindros

Condições iniciais			**saída**							
testes	*thresh (m)*	*peso normais*	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*exec. (ms)*
1	$0,01$	$0,1$	$0,224082$	$-1,47491$	$1,28687$	$-0,015253$	$0,841214$	$0,540487$	$0,11017$	140.790
2	$0,02$	$0,1$	$0,288736$	$-5,20087$	$-1,15069$	$-0,0156834$	$0,038629$	$0,544477$	$0,10385$	142.593
3	$0,03$	$0,1$	$0,250396$	$-2,86768$	$0,354226$	$-0,0178463$	$0,837655$	$0,545908$	$0,101976$	141.159

Figura 4.33: Resultado da segmentação do cilindro.

4.7.1.4 Discussão de resultados

Em termos de qualidade de resultados, no«caso da segmentação de planos, o método que inclui as normais produz melhores resultados, isto porque as normais têm um peso significativo nessa estimação, prevenindo a inclusão de pontos pertencentes a outros objectos ou conjuntos de pontos (figuras 4.31 e 4.32).

No outro método, em que só é incluída a nuvem de pontos original, apenas é considerado o *treshold* definido pelo utilizador. Este facto, faz com que à saída sejam incluídos pontos pertencentes ao plano mas também pontos pertencentes à bola, à caixa e ao cilindro. No entanto, com este método, tem-se um menor tempo de processamento relativamente ao anterior (tabelas 4.25 e 4.26), o que o torna mais apetecível para a execução de rápida segmentação.

No caso da segmentação de cilindros (secção 4.7.1.3) é necessário proceder à estimação das normais antes da respectiva extracção do modelo. Constatou-se que o método tem um tempo de processamento elevado, mas no entanto, e no que concerne à qualidade do resultado o seu comportamento é bastante satisfatório, tendo inclusivamente detectado o modelo cilíndrico de uma nuvem de pontos em que só tinha informação de uma parte do objecto, estando a outra parte ocluída aquando da captura (figura 4.33).

4.7.2 Simplificação de modelos geométricos

Os métodos que serão caracterizados aqui, ajuste (*fit*) de planos ao seu modelo paramétrico e a definição de um polígono 2D que o represente de forma simples, fazem parte de algumas das formulações feitas em [Rus09], com o objectivo de simplificar a representação de modelos geométricos com especial ênfase nos modelos de planos.

Os dois métodos fazem, na realidade, parte de outros módulos do PCL (*filters* e *surface*), mas como a sua apresentação faz mais lógica no seguimento dos conceitos descritos nesta secção, decidiu-se caracteriza-los aqui.

Para os testes foi utilizado o plano extraído de $\mathcal{P}$ e representado na figura 4.32.

Para projectar os pontos no seu modelo paramétricos, são apenas necessários os coeficientes estimados aquando da segmentação do plano. Para a definição de um polígono 2D que representa os limites do plano segmentado, é necessário definir o parâmetro α. Este parâmetro determina o tamanho máximo de um vértice ao centro do polígono, quanto mais pequeno esse valor é, mais detalhes terá o polígono.

4.7.2.1 Discussão de resultados

Como se pode constatar da figura 4.34, os resultados da projecção dos pontos no modelo paramétrico de um plano foi bem sucedida, com uma velocidade de processamento bastante aceitável (tabela 4.28).

Tabela 4.28: Projecção de pontos com utilização de modelos paramétricos.

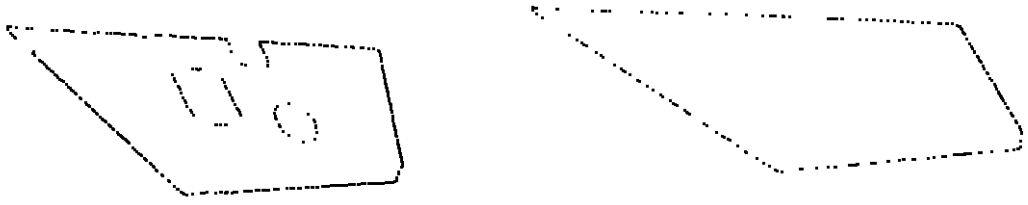
Condições iniciais					**saída**
modelo	***a***	***b***	***c***	***d***	***exec. (ms)***
plano	$0,00280287$	$-0,831432$	$-0,555634$	$1,11937$	12 ± 1

Tabela 4.29: Representação de um plano por um polígono 2D - *concave hull*.

Condições iniciais		**saída**	
teste	α	***nº pontos***	***exec. (ms)***
1	$0,1$	564	1.879
2	$0,5$	263	1.857
3	1	191	1.844

(a) Plano segmentado (b) Pontos do plano projectados

Figura 4.34: Nuvem de pontos do plano extraído através da segmentação, pontos projectados através do modelo paramétrico estimado durante o processo de segmentação e o *concave hull* do plano.



(a) *Concave hull* - teste 1 (b) *Concave hull* - teste 2

Figura 4.35: Polígono 2D que representa o plano segmentado.

Já os resultados da simplificação do plano num polígono 2D tem um tempo de processamento alto (tabelas 4.29) para aplicações em tempo real, mas a simplificação de superfícies planas em polígonos 2D, reduzirá substancialmente o processamento, nomeadamente em operações de navegação de um robô, que em vez de ter de calcular a sua distância em relação a todos os pontos, apenas terá de calcular a sua distância em relação ao plano previamente interpretado como tal.

4.7.3 Extracção de *clusters*

Nesta secção pretende-se caracterizar o método de extracção de conjuntos de pontos da nuvem $\mathcal{P}$, correspondentes a diferentes *clusters*.

4.7.3.1 Fundamentos teóricos

Os métodos de *clustering* têm como objectivo dividir uma nuvem de pontos $\mathcal{P}$ em parcelas mais pequenas $\mathcal{C}_k$. Baseando-se na descrição de [Rus09], este método, permite a distinção entre conjuntos de pontos representando diferentes *clusters*, ou seja, $\mathcal{C}_i = \{p_i \in \mathcal{P}\}$ difere de $\mathcal{C}_j = \{p_j \in \mathcal{P}\}$ se:

$$\min \|p_i - p_j\|_2 \geq d_{th} \tag{4.16}$$

onde d_{th} é o limite mínimo da distância entre *clusters*. A equação 4.16 estabelece que se a distância mínima entre dois conjuntos de pontos $p_i \in \mathcal{P}$ e $p_j \in \mathcal{P}$ for maior do que uma dada distância, então os pontos em p_i são definidos como pertencentes a $\mathcal{C}_i$ e os pontos em p_j definidos como pertencentes a $\mathcal{C}_j$.

4.7.3.2 Caracterização

Para os testes, recorreu-se à nuvem de pontos representada em 4.30. É aplicado um dos métodos descritos na secção 4.7.1.2 para extrair o plano correspondente ao solo.

Para além do parâmetro da distância mínima entre *clusters* anteriormente descrito, este método permite a definição de dois outros parâmetros. O primeiro diz respeito à definição de um número mínimo de pontos que um *cluster* deve compreender (*min-pts-per-cluster*), o segundo é o ângulo máximo (θ_{max}) que as normais dos pontos incluídos num conjunto, podem fazer entre si para que os pontos associados sejam considerados parte integrante do *cluster*.

4.7.3.3 Discussão de resultados

Na figura 4.36 pode-se observar os resultados obtidos através dos parâmetros definidos para o teste 3 ilustrado na tabela 4.30. Nessa figura são mostrados os três *clusters*, correspondentes a 3 conjuntos diferentes de pontos, extraídos de $\mathcal{P}$.

Tabela 4.30: Extracção de *clusters*.

Condições iniciais				**saída**				
teste	***d_{th} (m)***	***θ_{max} (graus)***	***min-pts-per-cluster***	***nº clusters***	***nº pts cilindro***	***nº pts caixa***	***nº pts bola***	***exec. (ms)***
1		78		3	9.094	7.944	3.744	1.372 ± 74
2	$0,05$	90	3.000	3	11.094	7.556	4.395	1.336 ± 94
3		180		3	11.603	7.956	4.971	1.321 ± 98

(a) $\mathcal{P}$

(b) *cluster* 1 - cilindro

(c) *cluster* 2 - caixa

(d) *cluster* 3 - bola

Figura 4.36: Nuvem de pontos utilizada nos testes, e os respectivos 3 *clusters* extraídos através dos parâmetros do teste 3.

Em termos de tempo de processamento, a implementação apresenta bons resultados.

Em termos qualitativos, os resultados finais da extracção dependem muito do limite mínimo da distância entre *clusters* e do número mínimo de pontos que deverão estar presentes em cada conjunto de pontos para que esse conjunto seja considerado um *cluster*.

Se a distância mínima for definida com um valor demasiadamente alto, 2 ou mais conjuntos de pontos, que nos testes produzidos foram considerados *clusters* ou conjuntos de pontos com menos de 3000 pontos, poderão ser concatenados no mesmo conjunto e produzir falsos *clusters* ou objectos. Por outro lado, se for definido um valor muito baixo para o número mínimo de pontos por *cluster*, conjuntos de pontos que não definem um objecto, mas sim parte dele, poderão ser considerados *clusters*.

No que concerne à variável $\boldsymbol{\theta_{max}}$, a variação do seu valor pode ter ao mesmo tempo boas e más consequências. Na figura 4.36, pode-se observar que definindo esta variável com 180^o, todos os pontos que representam a bola são extraídos. O mesmo não aconteceu nos outros 2 testes ilustrados na tabela 4.30, onde só foram extraídos metade ou mesmo menos de metade dos pontos que representam a bola. O mesmo resultado foi observado no teste 1 em que parte dos pontos pertencentes ao cilindro não foram considerados como parte integrante do *cluster*. Por outro lado, no teste 3, em que foi definido um ângulo de 180^o, foram considerados parte integrante dos *clusters*, pontos que não correspondem aos objectos observados, mas sim ao plano do chão no qual os objectos se encontravam assentes.

Parte II

Mapeamento - Aplicações

5

Mapeamento com recurso a câmaras RGB-D

Conteúdo

Neste capítulo irá ser apresentada a arquitectura do sistema de mapeamento e respectivos resultados da implementação do mesmo.

5.1 Sistema de Mapeamento

O desenho de um sistema de mapeamento é uma tarefa complexa que envolve imensas variáveis, como o tipo de sensor que se pretende utilizar na captura de dados tridimensionais, a dimensionalidade dos dados adquiridos, o tipo de plataforma robótica em que se pretende integrar o sistema, o tipo de tarefas que se pretende que essa plataforma efectue ou as limitações de hardware. Tendo em conta estas condicionantes, um bom sistema de mapeamento deverá ter os seguintes pré-requisitos:

1. **Robustez** - o sistema proposto deverá ser robusto, no sentido de que o seu bom funcionamento seja independente: do tipo de cenário, do tipo de sensor utilizado na aquisição de dados, dos erros de medida associados aos dados adquiridos por cada um desses sensores, bem como das próprias limitações dos algoritmos utilizados.

2. **Eficiência** - as tarefas que envolvem todos os procedimentos do alinhamento de nuvens de pontos, capturadas de diferentes posições no espaço num modelo global consistente, têm normalmente um custo computacional elevado. Todos os algoritmos deverão, por isso, ser implementados e integrados, tendo em conta as limitações de hardware existentes na plataforma robótica em que se pretende incorporar o sistema.

3. **Modularidade** - o sistema deverá ser modular e aberto, ou seja, cada algoritmo implementado em cada módulo ou bloco do sistema poderá ser substituído por outro sem problemas, tendo uma diferente ou mais eficiente implementação.

5.1.1 Arquitectura de Sistema

A figura 5.1 ilustra a proposta da arquitectura do sistema de mapeamento, apresentando todos os módulos a compõem.

O *design* desta arquitectura teve como principal foco o mapeamento de cenários *indoor* com utilização da Kinect. Mesmo assim, este facto não inviabiliza a utilização da mesma estrutura em sistemas de mapeamento orientados a cenários *outdoor*, bem como nuvens de pontos adquiridas a partir de outro tipo de dispositivos, como os apresentados em 2.2

Todos os módulos associados à arquitectura tiveram como base a caracterização realizada no capítulo 4.

O primeiro passo após a aquisição de uma nuvem de pontos, é o de remover os *outliers*, através da análise estatística da vizinhança de um ponto $p \in \mathcal{P}$. As nuvens de pontos provenientes desta computação são armazenadas, para posterior utilização no refinamento.

Seguidamente, é aplicado o método de *downsampling*, reduzindo assim o número de pontos da nuvem, tendo em atenção de que a nuvem de pontos deverá manter alguma da definição das superfícies amostradas para que o alinhamento inicial tenha bons resultados.

No passo seguinte são calculadas as normais das superfícies amostradas, cada normal é calculada através de todos os pontos que se encontrem dentro de um dado raio $\boldsymbol{r}$.

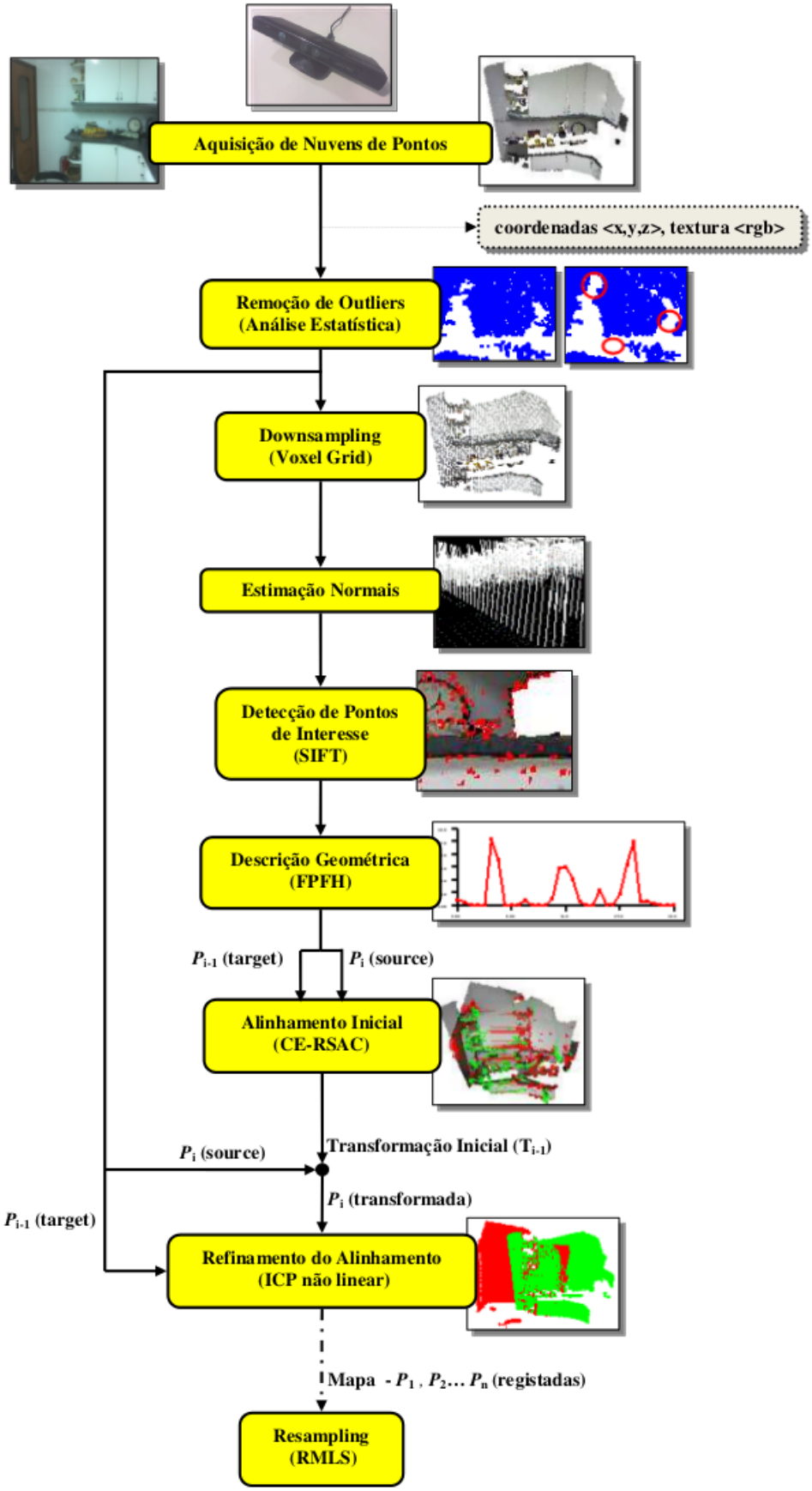


Figura 5.1: Arquitectura do sistema de registo de nuvens de pontos capturadas por câmaras RGB-D.

Os pontos de interesse são depois extraídos e descritos geometricamente, através do método SIFT e FPFH respectivamente. Seguidamente, é calculada a transformação inicial, ou "grosseira", entre duas nuvens de pontos adquiridas consecutivamente, através do método CE-RSAC. Como entrada o método recebe as nuvens de pontos que contêm os pontos de interesse e os descritores geométricos, correspondentes às duas nuvens de pontos que se pretendem alinhar. Assim sendo, $\mathcal{P}_2$ (*source*) é transformada para o sistema de coordenadas de $\mathcal{P}_1$ (*target*).

O alinhamento inicial é depois refinado pelo método ICP, recebendo como entrada as nuvens de pontos originais, depois se lhes serem removidos os *outliers*.

O registo de $\mathcal{P}_n$ nuvens de pontos é realizado iterativamente com base nas transformações geométricas (6 DOF) em pares de nuvens de pontos.

A primeira nuvem $\mathcal{P}_1$ de um dado *dataset* é definida pela orientação inicial da câmara em cada eixo, ou seja pela matriz identidade (consultar secção 4.5.1).

Se se considerar um sequência finita de *frames* $[frame_1; frame_N]$ e POS_k a posição de cada nuvem, para $k \in [1; N]$, a transformação de uma qualquer nuvem $\mathcal{P}_k$ pode ser determinada através da combinação de todas as transformações estimadas até esse ciclo de registo:

$$POS_k = POS_1 \prod_{i=1}^{k-1} T_i \tag{5.1}$$

Como o produto de matrizes não tem a propriedade comutativa, é essencial que as matrizes sejam multiplicadas na ordem correcta. Por exemplo, o procedimento seguido para se transformar $\mathcal{P}_4$ no referencial de $\mathcal{P}_1$, é dado por:

$$POS_4 = POS_1 T_1 T_2 T_3 \tag{5.2}$$

Outro método de alinhamento, seria o de concatenar constantemente as nuvens registadas, e alinhar sequencialmente, em cada ciclo do processo, a nuvem definida como *source* com o modelo global, que contém todas as nuvens previamente registadas. Este método, embora produza bons resultados não foi utilizado neste caso, pois tem custos computacionais bastante maiores e necessita de mais espaço de memória disponível para registar o mesmo número de nuvens, não trazendo vantagens significativas em termos qualitativos.

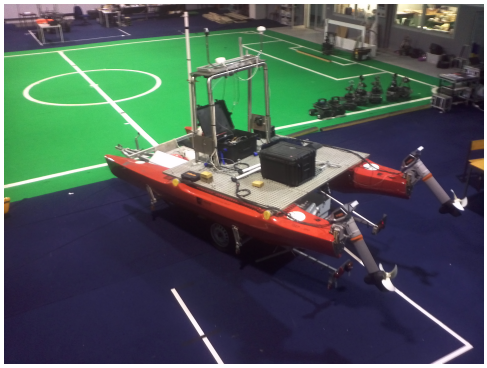
Depois de todas as nuvens estarem registadas no mesmo modelo global, procede-se à suavização das superfícies amostradas através do método RMLS, para que pequenos erros de medida ou ruído e áreas sobrepostas provenientes do processo de registo sejam minimizados.

5.2 Aplicação em cenários reais

Nesta secção são apresentados os resultados da aplicação da arquitectura proposta em **6** cenários diferentes ilustrados na figura 5.2.

De notar que todos os testes foram efectuados com a mesma máquina descrita no capítulo 4.

A escolha dos cenários teve como preocupação serem representativos, de uma forma geral, de possíveis ambientes *indoor* que os robôs com este sistema integrado, possam encontrar. Diferentes tipos de iluminação, cor, dimensões do ambiente a mapear, objectos de diferentes tamanhos e detalhe são algumas das variáveis que se pretendiam diversificar.



(a) Cenário **1** - barco.

(b) Cenário **2** - sala ferramentas.

(c) Cenário **3** - entrada lsa.

(d) Cenário **4** - corredor lsa.

(e) Cenário **5** - sala de aula.

(f) Cenário **6** - cozinha lsa.

Figura 5.2: Cenários teste.

5.2.1 Resultados experimentais

No cenário 1 foram registadas 21 nuvens de pontos, no cenário 2, 28 nuvens de pontos, no cenário 3, 40 nuvens, no cenário 4, 41 nuvens, no cenário 5, 28 nuvens e por último no cenário 6 foram registadas 25 nuvens de pontos.

Na tabela 5.1, estão representados os valores dos parâmetros utilizados para cada processo de registo de nuvens nos diferentes cenários. Estes foram os valores óptimos encontrados para cada *dataset*, para os quais os resultados do registo foram os melhores encontrados. O tempo de processamento teve um peso muito pequeno na escolha dos parâmetros, sendo apenas comparado entre valores de parâmetros que produzissem os mesmos resultados qualitativos no registo das nuvens.

Tabela 5.1: Parâmetros utilizados no processo de registo dos vários cenários.

	Outliers		*Down-sample*	Nor-mais	*Keypoints*				*Features*	Alinha-mento Inicial	Refinamento			
Cenário	k	α	*leafsize (m)*	*raio (m)*	σ	*nr-octaves*	k	$\hat{x}$	*raio (m)*	*threshold (m)*	*max corresp. dist. (m)*	*th-reshold (m)*	ϵ	*max iterações*
1	10	2,36	0,015	0,07	0,005	10	8	1,0	0,35	0,05	0,1	0,05	0,00000001	1.000
2	10	2,67	0,03	0,1	0,005	10	8	0,2	0,60	0,04	0,1	0,05	0,00000001	1.000
3	10	2,82	0,03	0,1	0,005	10	8	0,6	0,55	0,04	0,1	0,04	0,00000001	1.000
4	10	2,68	0,03	0,1	0,005	10	8	0,0	0,60	0,05	0,1	0,04	0,00000001	1.000
5	10	2.54	0,03	0,07	0,005	10	8	0,4	0,60	0,05	0,1	0,05	0,00000001	1.000
6	10	2,42	0,03	0,1	0,005	10	8	0,0	0,65	0,05	0,1	0,02	0,00000001	1.000

De notar que, em alguns *datasets*, foi utilizado o filtro *passThrough*, para que fossem suprimidos os pontos cujas coordenadas em z não pertencessem ao intervalo $[0,0;5,0]$. Esta filtragem foi efectuada, devido aos erros associados à captura de dados através da Kinect, que aumenta muito devido à distância das superfícies amostradas fora desse intervalo, o que tem algumas consequências nocivas no registo de nuvens. Este procedimento, aumenta o tempo de processamento em cada nuvem em ± 28 ms.

Na remoção de *outliers*, foram escolhidos valores que suprimissem 1% dos pontos, na maioria das nuvens de cada *dataset*. Na realidade, e como os valores são definidos de forma geral em cada conjunto de nuvens a registar, a percentagem de *outliers* detectados e removidos das nuvens de pontos situa-se entre 0,5% e 2%.

5.2.1.1 Caracterização do tempo de processamento de cada módulo do processo de registo

Nas figuras 5.3, 5.4, 5.5, 5.6, 5.7, 5.8 e 5.9, estão representados os gráficos que descrevem o tempo de processamento de cada módulo sobre cada nuvem representativa dos vários cenários mapeados.

Nas tabelas 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 e 5.8 estão discriminados: o tempo de processamento médio ($\hat{x}$) e desvio padrão (σ) de cada módulo no respectivo cenário.

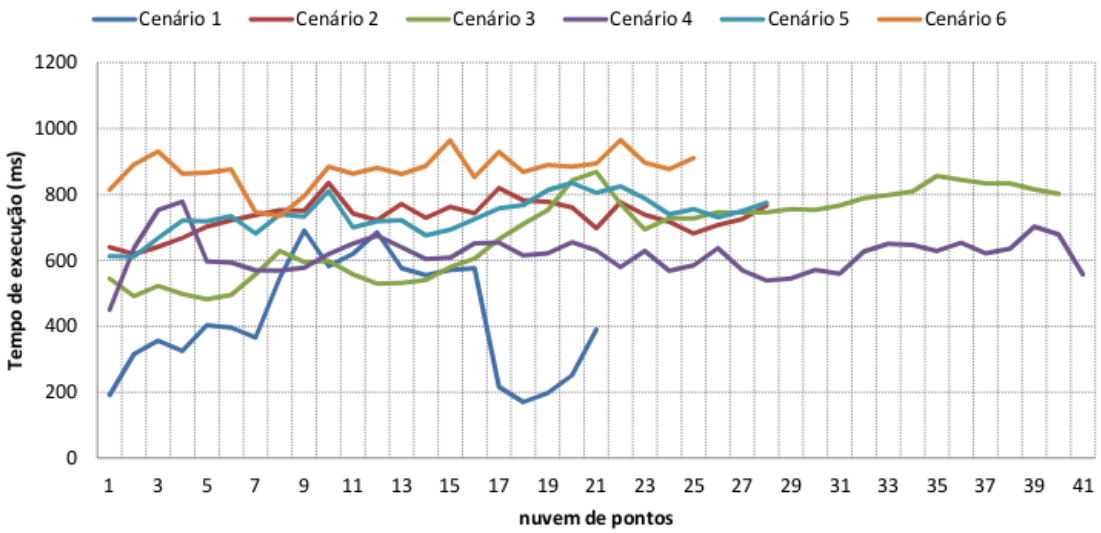


Figura 5.3: Gráfico do tempo de processamento da remoção de *outliers*.

Tabela 5.2: Caracterização da execução da remoção de *outliers*.

Cenário	$\bar{x}$ (ms)	σ (ms)
1	$427,67$	$170,04$
2	$731,57$	$50,76$
3	$684,95$	$124,48$
4	$617,78$	$57,36$
5	$735,93$	$56,65$
6	$872,88$	$54,85$

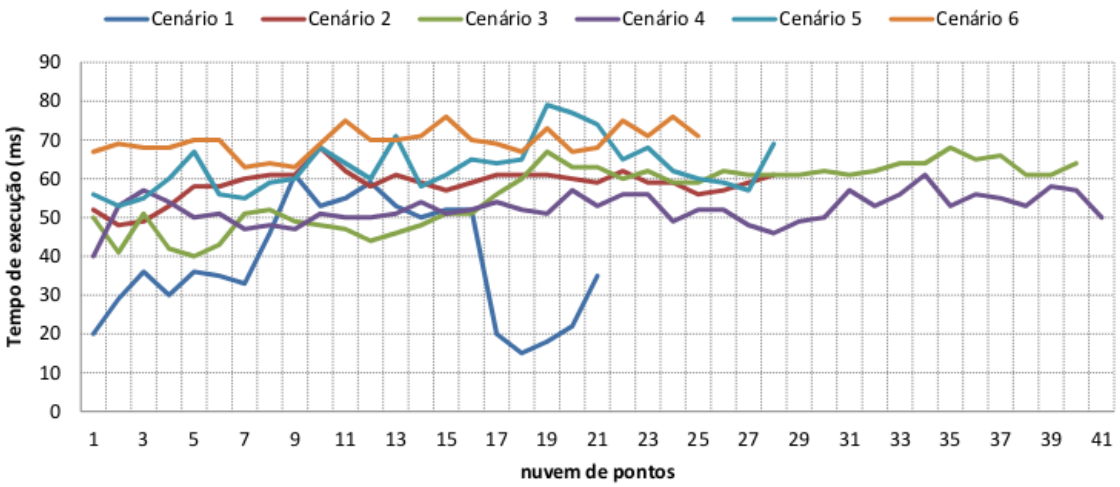


Figura 5.4: Gráfico do tempo de processamento do *downsample*.

Tabela 5.3: Caracterização da execução do downsample.

Cenário	$\bar{x}$ (ms)	σ (ms)
1	38, 57	14, 72
2	58, 54	4, 08
3	56, 15	8, 13
4	56, 20	3, 91
5	63, 11	6, 70
6	69, 80	3, 57

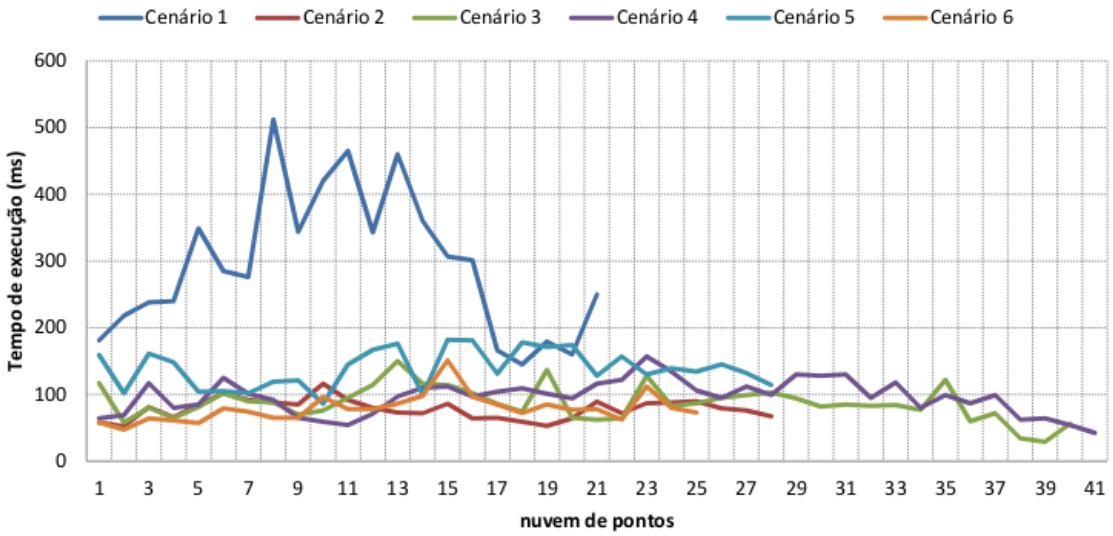


Figura 5.5: Gráfico do tempo de processamento da estimação de normais.

Tabela 5.4: Caracterização da execução da estimação de normais.

Cenário	$\bar{x}$ (ms)	σ (ms)
1	265, 19	107, 09
2	77, 86	15, 04
3	86, 98	25, 57
4	95, 98	25, 75
5	138, 93	28, 62
6	79, 12	21, 08

Tabela 5.5: Caracterização da execução da detecção de pontos de interesse.

Cenário	$\bar{x}$ (ms)	σ (ms)
1	3496, 24	784, 45
2	1451, 61	287, 02
3	1610, 20	527, 77
4	1306, 66	285, 64
5	2993, 32	625, 08
6	1067, 76	291, 81

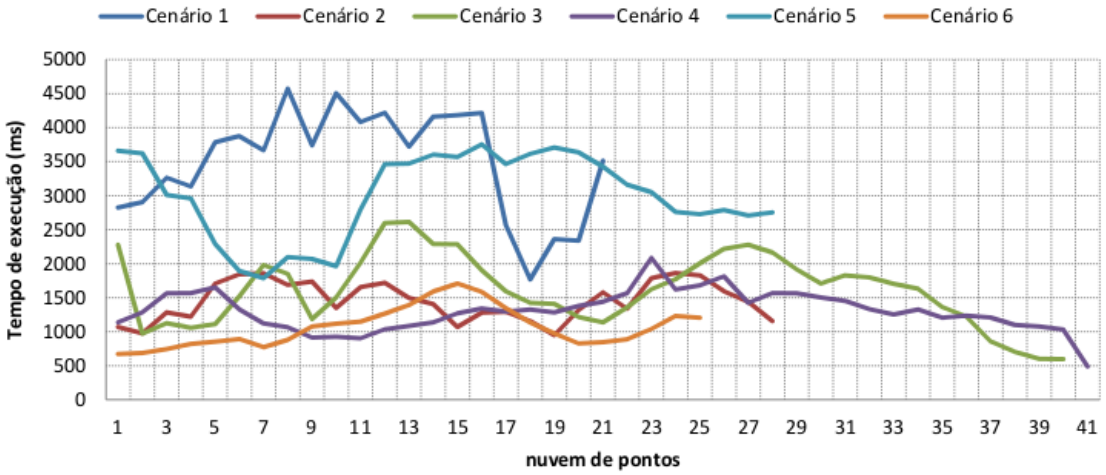


Figura 5.6: Gráfico do tempo de processamento da detecção de pontos de interesse.

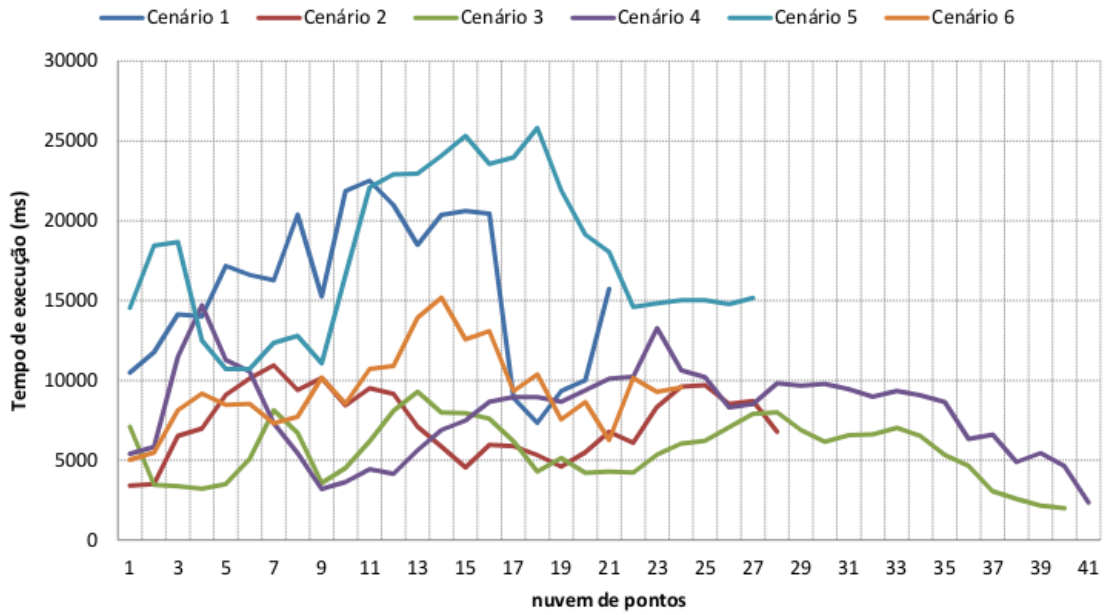


Figura 5.7: Gráfico do tempo de processamento da descrição geométrica local.

Tabela 5.6: Caracterização da execução da descrição geométrica local.

Cenário	$\bar{x}$ (ms)	σ (ms)
1	15840	4745, 77
2	7375, 32	2129, 89
3	5605, 55	1897, 65
4	8007, 58	2754, 24
5	17626, 54	4777, 93
6	9268, 28	2560, 71

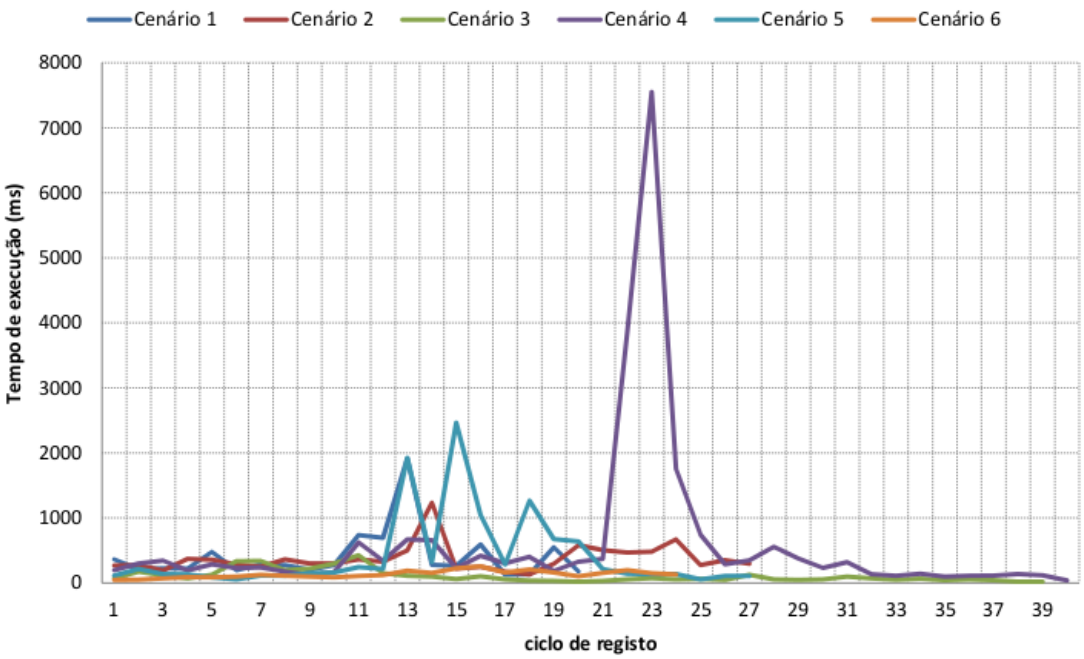


Figura 5.8: Gráfico do tempo de processamento do alinhamento inicial.

Tabela 5.7: Caracterização da execução do alinhamento inicial.

Cenário	$\bar{x}$ (ms)	σ (ms)
1	407, 85	397, 73
2	372, 00	212, 69
3	104, 18	96, 54
4	594, 48	1291, 63
5	415, 44	595, 84
6	131, 54	54, 51

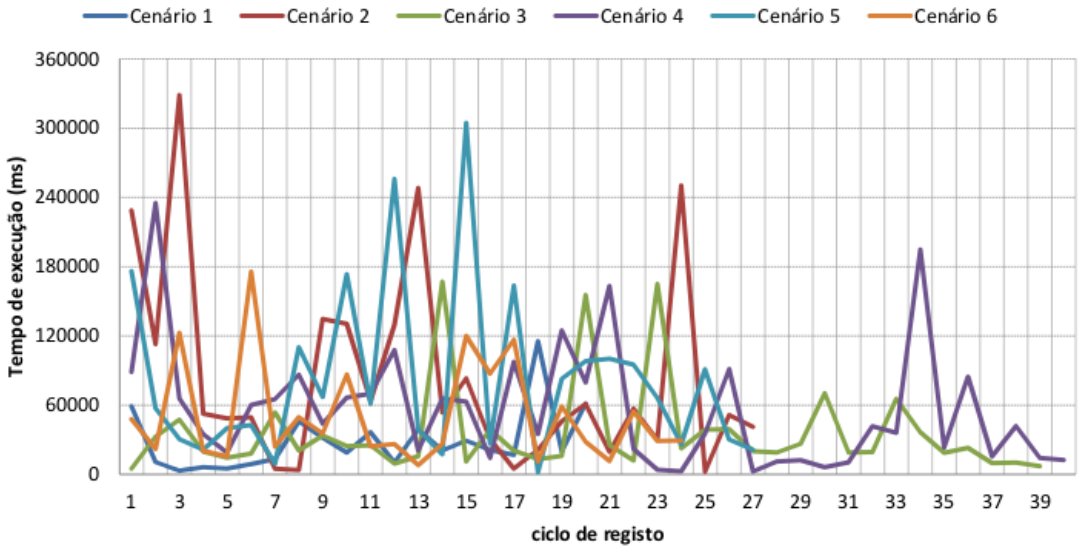


Figura 5.9: Gráfico do tempo de processamento do refinamento do registo de nuvens.

Tabela 5.8: Caracterização da execução do refinamento do registo de nuvens.

Cenário	$\bar{x}$ (ms)	σ (ms)
1	$28649,40$	$26561,20$
2	$84766,74$	$86147,42$
3	$35639,74$	$39978,93$
4	$56742,23$	$52693,10$
5	$81856,04$	$75161,21$
6	$51332,08$	$44140,39$

5.2.1.2 Caracterização global do processo de registo

A caracterização dos resultados é expressa na tabela 5.9 e no gráfico da figura 5.10.

Na tabela 5.9 estão discriminados: o tempo de processamento médio de cada ciclo de registo ($\bar{x}$) e respectivo desvio padrão (σ); o tempo total de todo o processo de registo; e o número total de pontos do modelo global obtido.

O gráfico da figura 5.10 apresenta a evolução dos tempos de execução de cada ciclo de registo, nos diversos cenários onde o sistema foi testado.

O tempo do primeiro ciclo de registo engloba os tempos de execução desde a remoção de *outliers* até à descrição geométrica das nuvens consideradas como *source* e *target* nesse ciclo, como também os tempos de execução do alinhamento inicial e refinamento entre as duas nuvens.

Nos ciclos seguintes, somente são considerados os tempos de execução desde a remoção de *outliers* até à descrição geométrica da *source*, bem como o procedimento de alinhamento inicial e refinamento. Isto acontece, porque os procedimentos até à descrição geométrica da nuvem considerada como *target* em cada ciclo de registo, foram executados nos respectivos ciclos anteriores.

Tabela 5.9: Caracterização do processo de registo em cada cenário.

Cenário	$\bar{x}$ (ms)	σ (ms)	Tempo Total (min)	N^o total de pontos
1	$50.000,75$	$25.878,50$	$16,67$	2.732.083
2	173.628	$91.447,60$	$44,63$	6.086.540
3	$45.430,05$	$40.571,21$	$28,77$	8.244.025
4	$70.886,53$	$63.070,17$	$47,26$	7.266.262
5	$116.037,07$	$113.160,26$	$52,22$	5.546.820
6	$64.542,04$	$45.103,82$	$25,82$	6.212.531

A tabela 5.10 e o gráfico apresentado na figura 5.11, relacionam o peso que cada módulo tem na execução da solução implementada para o registo de nuvens de pontos.

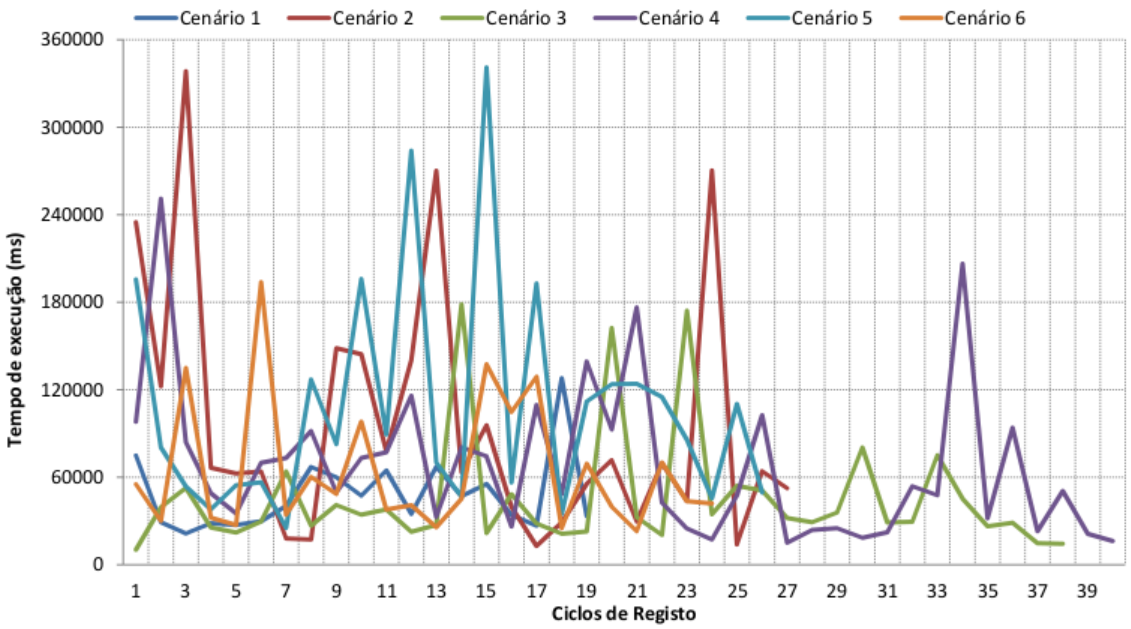


Figura 5.10: Gráfico do tempo de processamento dos ciclos de registo.

Tabela 5.10: Peso médio de cada módulo do processo de registo de todos os cenários testados.

módulo	$\bar{x}$ **(ms)**	**Peso (%)**
Remoção de outliers	$678,4629$	$0,96$
Downsample	$56,3599$	$0,08$
Estimação das normais	$129,0078$	$0,18$
Detecção de pontos de interesse	$1987,631$	$2,83$
Descrição geométrica	$10620,57$	$15,11$
Alinhamento inicial	$337,5818$	$0,48$
Refinamento	$56497,7$	$80,36$

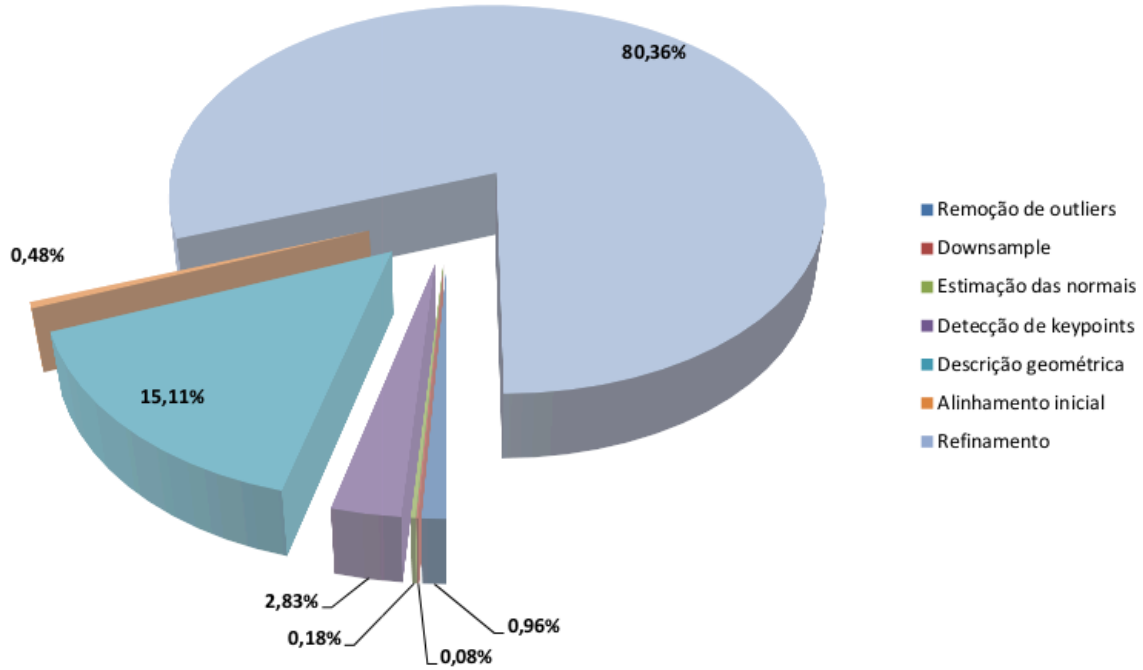


Figura 5.11: Gráfico circular do peso percentual médio que cada módulo tem, no processo de registo dos vários cenários.

5.2.1.3 Caracterização da aplicação do *resampling* dos mapas criados

Na tabela 5.11 estão discriminados os parâmetros utilizados na aplicação do método RMLS aos 6 mapas criados pelo processo de registo.

Tabela 5.11: Caracterização da aplicação do método RMLS aos mapas criados.

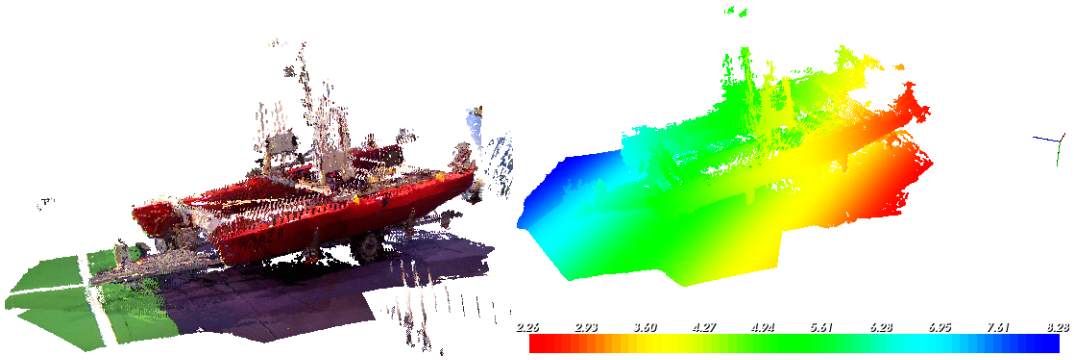
Cenário	**raio (m)**	Tempo (minutos)
1	0.05	30, 95
2	0.05	79
3	0.05	155, 9
4	0.05	30, 95
5	0.05	32, 88
6	0.05	116, 04

5.2.1.4 Mapas resultantes do processo de mapeamento implementado

Nas figuras 5.12, 5.13, 5.14, 5.15, 5.16 e 5.17 estão ilustrados os resultados práticos da implementação da arquitectura de sistema proposto.

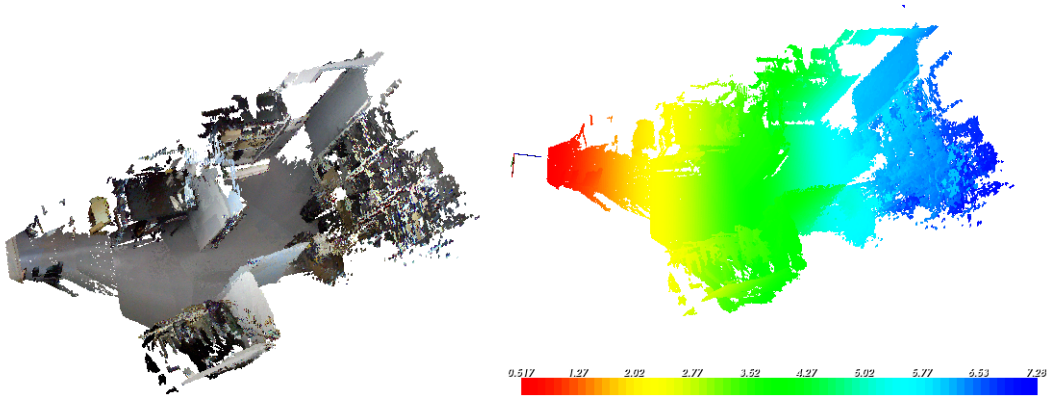
5.2.2 Discussão de resultados

Foi conseguido efectuar com sucesso o registo de nuvens de pontos num único modelo global, através da implementação da arquitectura de sistema idealizada para a resolução do problema associado ao registo de nuvens de pontos. Os resultados



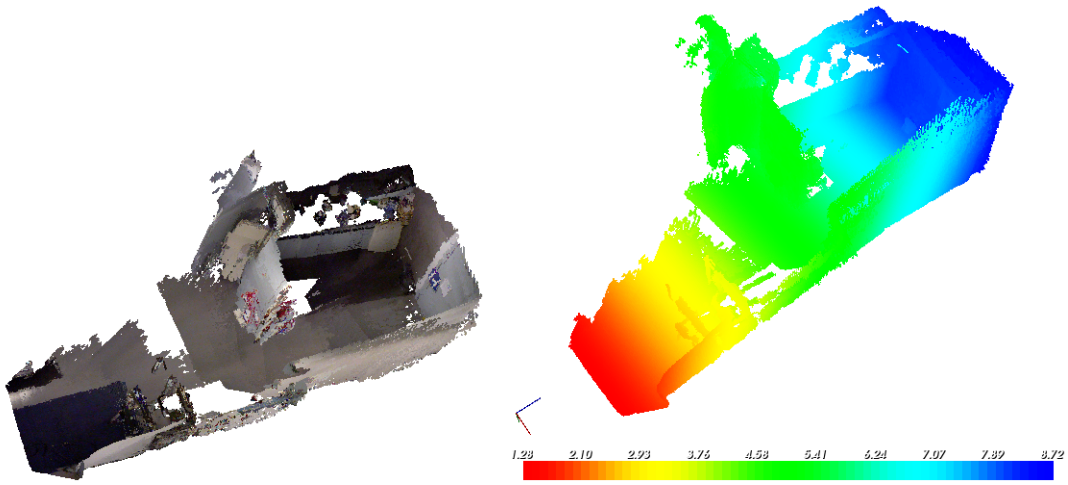
(a) (b)

Figura 5.12: Resultante do mapeamento do cenário 1 - 21 nuvens de pontos.



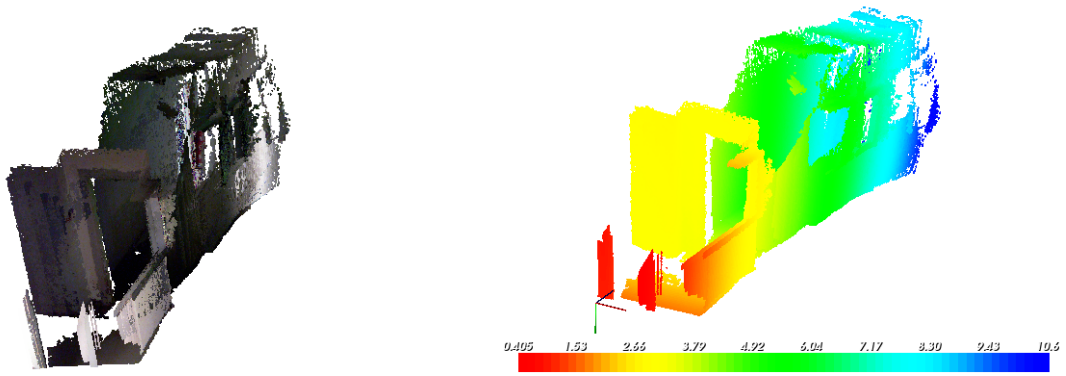
(a) (b)

Figura 5.13: Resultante do mapeamento do cenário 2 - 28 nuvens de pontos.



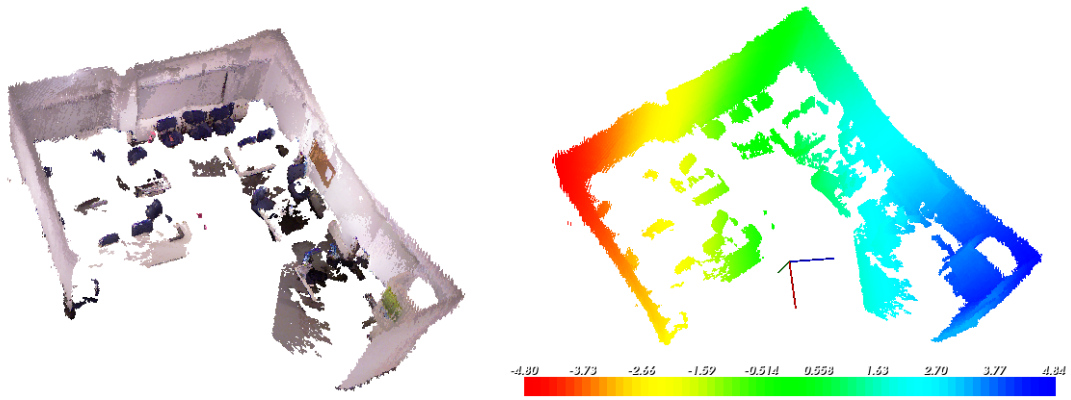
(a) (b)

Figura 5.14: Resultante do mapeamento do cenário 3 - 40 nuvens de pontos.



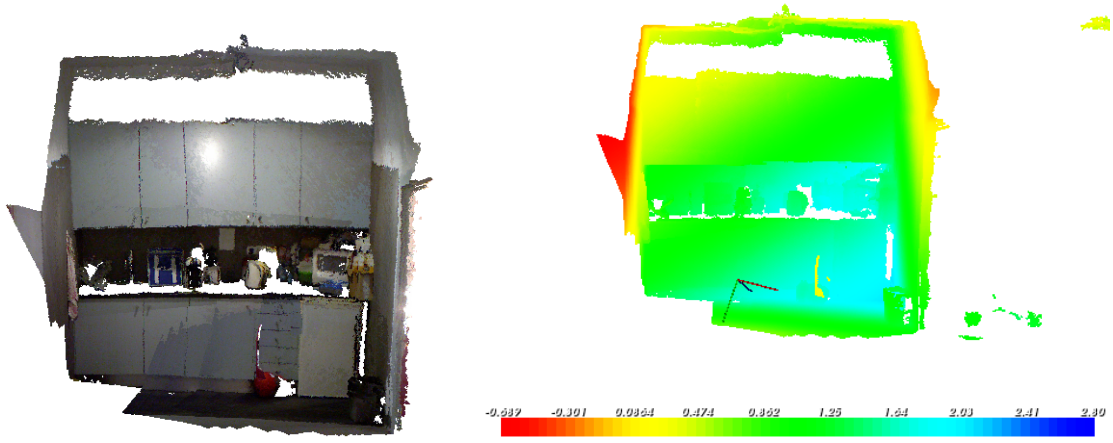
(a) (b)

Figura 5.15: Resultante do mapeamento do cenário 4 - 41 nuvens de pontos.



(a) (b)

Figura 5.16: Resultante do mapeamento do cenário 5 - 28 nuvens de pontos.



(a) (b)

Figura 5.17: Resultante do mapeamento do cenário 6 - 25 nuvens de pontos.

obtidos em termos da qualitativos do processo de mapeamento foram bastante satisfatórios, tendo em conta que só é utilizada a informação proveniente da Kinect.

Todo o procedimento desde a remoção de *outliers* até ao refinamento da estimação da transformação entre duas nuvens consecutivas, levou em média 1m20s a ser executado. Este tempo, embora ainda longe de ser ideal, no que concerne à integração da arquitectura em sistemas que exigem execução em tempo real, denota um bom valor inicial, tendo em conta a grande quantidade de dados processada.

Pese embora os bons resultados, o procedimento compreende ainda alguns problemas.

Uma boa estimação inicial é obtida se forem encontradas boas correspondências entre pontos de interesse detectados nos pares de nuvens. Os erros de medição, inerentes da aquisição de dados pela Kinect, influencia de sobremaneira estes resultados, pois a correspondência entre pontos de interesse, é efectuada através da descrição geométrica da sua vizinhança. Verificaram-se, portanto, muitas dificuldades na obtenção de boas correspondências nos casos em que as superfícies amostradas estavam a uma maior distância, o que trouxe uma maior dificuldade na obtenção de boas descrições geométricas e como consequência pontos que deveriam ser correspondentes, são descartados, ou mesmo em alguns casos produzidas algumas falsas correspondências, com óbvios efeitos negativos.

Quando a estimação da transformação inicial retornada pelo método CE-RSAC não é tão boa como deveria ser, o ICP tem mais dificuldade em encontrar a melhor transformação possível. Isso é verificável, pelos picos no tempo de processamento observados no gráfico 5.9.

Estes picos no aumento do tempo de processamento, devem-se em parte, à definição de parâmetros gerais para um dado cenário, em que para a maioria dos pares de nuvens analisadas parece resultar bastante bem, mas que em outros pares, o mesmo não se verifica.

Mesmo em casos em que foi rapidamente encontrada a estimação da transformação pelo método do ICP, verificou-se que em muitos casos, essa não era de facto a melhor solução. Isto ocorre porque, como já foi referido anteriormente na secção 4.5.1, o ICP ter a desvantagem de poder ficar retido num mínimo local, tendo consequências perversas no processo de registo. Este facto encerrou uma das maiores dificuldades encontradas na definição de parâmetros gerais para cada cenário mapeado.

As consequências dos problemas referidos, pese embora a grande panóplia de testes efectuados, no sentido de se produzir os melhores resultados possíveis, podem ser visualizados, por exemplo, nos mapas criados dos cenários 3 e 4. Nesses cenários, alguns erros na estimação da melhor transformação possível foram propagados às nuvens seguintes, fazendo com que, por exemplo, superfícies planas por

natureza muito regulares, tenham alguma curvatura.

6

Conclusões e Trabalho Futuro

Conteúdo

Neste capítulo faz-se o balanço dos objectivos cumpridos, avaliam-se os resultados obtidos e enquadra-se o trabalho futuro a realizar no contexto desta dissertação.

6.1 Conclusões

O trabalho realizado no contexto desta dissertação consistiu na avaliação da biblioteca PCL, para a construção de mapas multi dimensionais e interpretação dos mesmos, a partir de nuvens de pontos capturados pela Kinect, em ambientes *indoor*, não estruturados. No seguimento desta avaliação, foi desenhada uma arquitectura de sistema associada ao registo de nuvens de pontos num mapa global, representativo do mundo observado. Essa arquitectura foi depois testada em cenários reais, permitindo a discretização das superfícies amostradas.

O estudo dos módulos constituintes da biblioteca PCL e da documentação da

algoritmia associada aos métodos presentes na sua estrutura, permitiu identificar os métodos que melhor se enquadram na resolução dos problemas referidos.

Verificou-se que o PCL oferece imensas ferramentas que efectuam processamento sobre nuvens de pontos. Estas ferramentas estão em constante mutação e desenvolvimento, sendo que a cada lançamento de uma nova versão estável do PCL, são normalmente introduzidas algumas modificações na implementação dos métodos já existentes em versões anteriores, bem como, integradas implementações de novos métodos e módulos. Desde o início do trabalho realizado nesta dissertação, foram lançadas pelo menos três novas versões da biblioteca, o que veio adicionar uma dificuldade acrescida ao trabalho produzido.

Do estudo efectuado, os métodos escolhidos foram avaliados, caracterizados e comparados com métodos concorrentes no seu objectivo final. Para realizar este trabalho de avaliação dos métodos foram criadas com sucesso simples aplicações, desenvolvidas com recurso às actuais ferramentas disponíveis na versão 1.6 do PCL e aos ambientes integrados de desenvolvimento *Code::Blocks* e *Qt Creator*. Isto permitiu gerar resultados da velocidade de execução dos diferentes métodos e a visualização em 3D dos efeitos da aplicação de cada método.

Desta caracterização, efectuada no capítulo 4, foi possível inferir com sucesso os métodos que melhor desempenho computacional e resultados qualitativos oferecem.

Da caracterização efectuada ao filtro *passThrough* constata-se que a sua aplicação a uma qualquer nuvem de pontos permite extrair rapidamente um intervalo do espaço, correspondente a zonas de interesse de nuvens de pontos. O adopção deste método pode ser vantajoso, especialmente em nuvens de pontos nas quais existam superfícies amostradas a alguma distância, suprimindo pontos que encerram maiores erros de medição.

Da comparação dos dois métodos de remoção de *outliers*, *Statistical Outliers Removal* e *Radius Outliers Removal*, verificou-se que o que o primeiro detêm um melhor comportamento quer em termos de desempenho computacional quer nos resultados qualitativos, sendo mais robusto à variação da densidade de pontos.

Em relação às duas formulações de *downsampling*, *Voxel Grid* e *Approximate Voxel Grid*, observou-se um melhor comportamento do *Voxel Grid* em nuvens 4D. A utilização deste filtro tem também a vantagem de remover complemente pontos inválidos da nuvem, ao contrário do *Approximate Voxel Grid* que os mantém na nuvem produzida à sua saída.

No que respeita à detecção de pontos de interesse, foram comparados dois métodos: o SIFT e o Harris3D. Em termos de tempo de execução o Harris3D apresenta melhores resultados. Este facto, deve-se em parte à implementação com recurso ao OpenMP, o que denota uma vantagem em relação à execução do SIFT. No entanto,

verificaram-se resultados na detecção de pontos de interesse bastante mais consistentes em termos qualitativos com a utilização do método SIFT.

A caracterização do módulo *features* encerrou avaliações a métodos de estimação das normais das superfícies e de descritores geométricos locais. Na caracterização da estimação das normais foram comparados: o método de estimação de normais com base num dado raio r definido pelo utilizador e o método *Integral Images*. Observou-se que com o primeiro, o desempenho computacional é bastante maior mas produz melhores resultados qualitativos. Por seu lado o método *Integral Images*, tem um desempenho bastante melhor mas produz resultados piores do que o método anterior. No entanto, este poderá ser utilizado para tarefas de rápida segmentação de cenas, como as efectuadas na secção 4.7.1.

A comparação dos métodos PFH, FPFH e SHOT para a descrição geométrica de pontos, permitiu inferir o melhor desempenho do método SHOT quando utilizadas nuvens de pontos organizadas. No entanto, a sua utilização foi preterida perante o FPFH, na arquitectura de sistema proposta, pois sofre ainda de falhas de implementação na sua integração no PCL. Também não foi possível testar a sua contribuição no processamento posterior, nomeadamente no processo de registo de nuvens. De notar, no entanto que o FPFH, para além de resultados de desempenho aceitáveis, denotou uma boa contribuição nos resultados obtidos no processo de registo.

Para a estimação da transformação inicial entre nuvens de pontos consecutivas, foram comparados os métodos SAC-IA e CE-RSAC, os quais reproduzem resultados qualitativos semelhantes. No entanto, o método CE-RSAC denota um tempo de processamento geralmente melhor.

No refinamento da estimação da transformação entre nuvens, o ICP não linear, apresentou resultados em termos de tempo de processamento, bastante melhores do que o ICP linear. Em termos qualitativos os resultados são semelhantes.

A caracterização do método RMLS para execução de tarefas de suavização das superfícies amostradas, permitiu inferir a sua utilidade na remoção de erros inerentes às nuvens de pontos capturadas pela Kinect. No entanto, o tempo de execução deste método é ainda elevado.

Na segmentação de formas paramétricas verificou-se que com a utilização das normais das superfícies previamente calculadas, os resultados qualitativos são bastante melhores do que sem a sua utilização, embora em termos de execução isso denote um aumento significativo no tempo de processamento.

No que concerne à simplificação de conjuntos de nuvens de pontos representativos de planos segmentados e extracção de *clusters*, os resultados mostram um bom desempenho computacional e encerram ferramentas bastante úteis para a simplificação de nuvens de pontos e possível identificação de objectos decorrentes das superfícies amostradas.

Verificou-se que, com a configuração de *hardware* em que foram realizados os testes, não é ainda possível integrar a grande maioria dos métodos avaliados, em aplicações robóticas que executam tarefas em tempo real. No entanto, com uma melhor configuração de *hardware*, especialmente a nível de processador, a velocidade de execução seria drasticamente diminuída nos métodos que utilizam computação paralela na sua execução. Mesmo assim, os tempos de execução são aceitáveis, quando relacionados com a grande magnitude de dados processada e a quantidade de cálculos que cada algoritmo se propõe a efectuar. Este facto, aliado à qualidade dos resultados da aplicação dos métodos no processamento de nuvens de pontos, permite atestar a viabilidade da biblioteca PCL para a resolução dos problemas associados à construção de mapas, e sua interpretação.

A implementação da arquitectura de sistema proposta para o registo sistemático de nuvens de pontos num modelo global, obteve resultados satisfatórios nos testes efectuados nos diversos cenários, especialmente na qualidade dos resultados finais, tendo em conta que só é utilizada a informação proveniente das nuvens de pontos capturadas pela Kinect.

Pese embora, todo o processo efectuado para que uma boa estimação da transformação inicial fosse encontrada, a refinação da mesma pelo método do ICP ainda encerra um peso computacional significativo. No entanto, a estimação da transformação de uma nuvem em relação a outra demora em média 1 minuto e 10 segundos a ser encontrada.

Este comportamento não permite ainda aplicabilidade em tarefas executadas em tempo real, mas a maturação da arquitectura poderá levar a isso mesmo, com a substituição dos métodos utilizados por outros com melhor comportamento e a exploração das potencialidades dos GPUs.

Algumas das modificações que poderão ser efectuadas e potenciais algoritmos de interesse para debelar os problemas encontrados são apontados na descrição do trabalho futuro a ser realizado.

Como contribuição final do trabalho realizado nesta dissertação, aponta-se o facto de que foi construída uma base de conhecimento e aplicativos que permitirão num futuro próximo a integração do *hardware* e *software* desenvolvido em plataformas robóticas desenvolvidas no contexto do Laboratório de Sistemas Autónomos do ISEP.

6.2 Trabalho Futuro

A avaliação realizada aos módulos e métodos que compõem a biblioteca PCL, permitem a possibilidade de futura integração numa grande panóplia de plataformas

robóticas, cujas tarefas para as quais foram desenhadas para executar, variam enormemente.

O trabalho realizado no âmbito desta dissertação é apenas um primeiro passo na definição e maturação do protótipo da arquitectura de sistema apresentada.

Neste contexto, os próximos passos a serem dados, deverão ser no sentido de construir mapas semânticos, ou seja, mapas em que as superfícies amostrados e os objectos que elas representam, sejam reconhecidas e classificadas. Para a identificação de soluções que promovam a resolução deste problema, poderão ser utilizados os métodos de segmentação de formas paramétricas e extracção de *clusters*, caracterizados na secção 4.7.

Nesta interpretação dos mapas criados, poderão ser ainda explorados os trabalhos realizados em: [RMB+08b] e [Rus09] em especial o algoritmo *region growing*; [KAJS11] na interpretação e catalogação de objectos presentes num determinado cenário; o trabalho realizado em [RMP+12] para detecção de objectos desconhecidos para tarefas de manipulação de objectos.

Deverá também ser incorporada uma forma de detecção e identificação de objectos dinâmicos normalmente presentes em ambientes de construção humana, por natureza muito dinâmicos.

Este problema pode ser solucionado, através da verificação do espaço ocupado pelas superfícies descritizadas nas nuvens de pontos desde que estas estejam referenciadas no mesmo referencial, ou seja, perfeitamente registadas. A utilização de estruturas de dados como as *octrees* pode facilitar essa verificação do espaço, conferindo a diferença entre os voxeis ocupados entre as várias nuvens.

Seguidamente serão descritos alguns dos melhoramentos na qualidade dos resultados e desempenho que poderão ser efectuados na arquitectura de sistema apresentada para o processo de registo de nuvens de pontos.

Alguns dos algoritmos e métodos abaixo referenciados estarão disponíveis nas próximas versões estáveis do PCL, daí serem referidos como potenciais focos de interesse no que diz respeito ao trabalho que futuramente será desenvolvido.

6.2.1 Melhoramento na qualidade dos resultados

O lançamento no mercado da Kinect 2.0 em 2013, irá trazer maior precisão na amostragem de superfícies, o que levará a um melhoramento dos resultados da generalidade dos algoritmos aqui apresentados, e na percepção do ambiente que rodeia um robô no qual este sistema esteja implementado. A maior precisão dos dados adquiridos, poderá ter também, um impacto significativo nos robôs, cujas tarefas envolvam uma maior precisão dos dados amostrados, como por exemplo: tarefas de manipulação de objectos.

No que diz respeito ao registo de nuvens de pontos poderão ser explorados os trabalhos realizados em [GSGB07], [SNLH09], [GKS+10] e [EHE+12a]. O primeiro, segundo e terceiro artigos apresentam algoritmos de detecção de *loops* no processo de registo de nuvens de pontos, permitindo por diferentes métodos, alguma correcção. O quarto trabalho citado apresenta uma nova abordagem ao problema de localização e mapeamento simultâneo (SLAM) em 6-DOF e detecção de *loops*, recorrendo à utilização de câmaras RGB-D para construir mapas voxelizados de grande precisão.

Estas quatro abordagens de detecção de *loops*, poderão contribuir para melhores resultados na solução proposta do registo de nuvens de pontos adquiridas pela Kinect, corrigindo algum do erro associado às transformações entre as nuvens na construção do modelo global.

No trabalho realizado, o único sensor usado foi a Kinect. A possibilidade da utilização da informação proveniente de outros sensores como um sistema de navegação inercial (INS), ou de dados provenientes da hodometria de um robô, poderá ser uma forma de obter melhores resultados. A informação recolhida dos vários sensores poderá ser combinada através de um filtro de Kalman, produzindo melhores resultados, especialmente em casos em que a transformação produzida pelo método de registo aqui apresentado não é muito confiável.

6.2.2 Melhoramento no desempenho

O avanço da tecnologia na vertente do hardware, como a evolução do processadores e o aumento da capacidade de memória e armazenamento dos computadores, irá também dar um forte contributo para o melhoramento na capacidade de processamento e armazenamento de nuvens de pontos.

A biblioteca PCL irá concerteza sofrer uma grande evolução, com o melhoramento da performance dos algoritmos já incluídos na sua estrutura, inclusão e desenvolvimento de novos algoritmos, que irão diminuir o tempo de processamento e melhorar a capacidade interpretativa dos dados adquiridos.

Um problema relevante, encontrado nos resultados do processo de registo de nuvens (ver secção 5.1), foi o grande peso computacional que o ICP comporta. Deverão por isso, ser encontrados métodos alternativos a este algoritmo, que satisfaçam a mesma exigência na qualidade de resultados. Um desses métodos alternativos poderá basear-se nos trabalhos realizados para os artigos [MLD07, HMSL08], correspondentes à descrição dos algoritmos *Normal Distribution Transform* (3D-NDT) e Color-NDT.

Nos módulos que dizem respeito à detecção de pontos de interesse e descrição geométrica dos mesmos, poderá ser integrada a implementação do algoritmo SIFT

em linguagem CUDA [Bjo]. A extracção de pontos de interesse e a sua descrição poderão ser processados em $15-20$ ms, o que denota um significativo melhoramento no desempenho.

Poderão também ser explorados os trabalhos [CLSF10] e [EHE+12a]. O primeiro, apresenta um descritor geométrico de nome *Binary Robust Independent Elementary Features* (BRIEF), que pode ser uma interessante alternativa ao descritor FPFH, tendo potencial para melhorar o desempenho deste passo do processo de registo. O segundo trabalho referido, para além de um método de detecção de *loops*, contém também novos desenvolvimentos na detecção e descrição de *features* recorrendo a implementações do algoritmo SIFT e do algoritmo *Speeded Up Robust Feature* (SURF). Este último algoritmo, poderá trazer melhorias na performance na extracção de pontos de interesse.

O uso generalizado de buscas de vizinhança poderá ser acelerada com a construção de *kdtrees* no GPU. No artigo [ZHWG08] é apresentado um algoritmo, implementado em linguagem CUDA, que atinge desempenho em tempo real, explorando a arquitectura dos GPU na construção de *kdtrees*.

De futuro, a arquitectura de sistema aqui proposta, poderá ser implementada utilizando as capacidades de processamento dos GPUs. Nesse sentido, está neste momento a ser desenvolvida pelos colaboradores do PCL, uma *framework* denominada "*KinFu*", programada em linguagem CUDA, nativa dos GPUs da Nvidia. Esta nova *framework* irá incluir alguns dos métodos presentes no PCL, bem como novos algoritmos de reconstrução tridimensional que estão neste momento a serem desenvolvidos.

6.2.3 Outras abordagens

O projecto *KinectFusion*, que a *Microsoft Research* desenvolveu, apresenta uma solução para construir um mundo sem ser necessária qualquer detecção de pontos de interesse ou descritores geométricos. Os algoritmos desenvolvidos e implementados sobre GPUs conseguem atingir níveis de desempenho muito altos, permitindo a construção de um modelo global em tempo real[1]. Os artigos [IKH+11, NIH+11], descrevem como a construção de mapas precisos pode ser efectuada. Esta solução parece ser bastante promissora na construção de futuras aplicações, orientadas não só ao mapeamento, como também em tarefas de segmentação e reconhecimento de objectos.

Existem também abordagens recentes que utilizam *octrees* em sistemas robóticos que realizam tarefas de mapeamento. Exemplos da utilização deste tipo de estruturas, podem ser encontrados em recentes publicações como: [WHB+10] e [WHH+11].

[1] KinectFusion: http://www.youtube.com/watch?v=quGhaggn3cQ

Os mapas criados são apresentados por representações volumétricas como as apresentadas em C.2.

A utilização deste tipo de estruturas de dados na decomposição espacial, para além de fornecer rápido acesso à localização de pontos e à sua vizinhança, são também populares no contexto de detecção de colisões. Neste contexto, a estimação das distâncias em vez de serem calculadas em relação a cada ponto, pode ser desempenhada através da utilização da técnica de *raycasting* em relação aos voxeis abrangidos por $\mathcal{P}$, para encontrar porções de espaço vazias ou ocluidas.

Uma grande vantagem das *octrees* é o serem de fácil actualização e suportarem a rápida inserção e supressão de pontos.

De facto, o interesse da sua utilização na construção de mapas tem aumentado nos últimos tempos, tendo sido inclusive, financiado este ano pela empresa *Urban Robotics*[2] um *code sprint*[3], para a criação de um novo módulo para a biblioteca PCL (pcl_outofcore). Este novo módulo tem como objectivo integrar um sistema de criação de enormes mapas 3D geoespaciais, cobrindo centenas de quilómetros quadrados.

[2]Urban Robotics - www.urbanrobotics.net
[3]PCL Urban Robotics Code Sprint Blog - www.pointclouds.org/blog/urcs

Bibliografia

[ABG+11] Aitor Aldoma, Nico Blodow, David Gossow, Suat Gedikli, Radu Bogdan Rusu, Markus Vincze, e Gary Bradski. CAD-Model Recognition and 6 DOF Pose Estimation using 3D Cues. In *Proceedings of the International Conference on Computer Vision (ICCV), workshop on 3D Representation and Recognition*, Barcelona, Spain, Novembro 7 2011.

[AHB87] K. S. Arun, T. S. Huang, e S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700, Maio 1987.

[AmFM+06] A. Akbarzadeh, J. m. Frahm, P. Mordohai, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nistér, e M. Pollefeys. Towards urban 3d reconstruction from video. In *3DPVT*, pág. 1–8, 2006.

[AMN+98] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, e Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, Novembro 1998.

[APP+12] Alexey Abramov, Karl Pauwels, Jeremie Papon, Florentin Wörgötter, e Babette Dellen. Real-time segmentation of stereo videos on a portable system with a mobile gpu. *IEEE Trans. Circuits Syst. Video Techn.*, 22(9):1292–1305, 2012.

[BGV92] Bernhard E. Boser, Isabelle M. Guyon, e Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pág. 144–152, New York, NY, USA, 1992.

[Bjo] M. Bjorkman. A cuda implementation of sift. Computer Vision and Active Perception Lab. Último acesso: Out. 2012 [Online]. url: http://www.csc.kth.se/~celle/.

[BLRF11] L. Bo, K. Lai, X. Ren, e D. Fox. Object recognition with hierarchical kernel descriptors. In *IEEE International Conference on Computer Vision and Pattern Recognition*, Junho 2011.

[BM92] P.J. Besl e N.D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:239–256, 1992.

[BPFN11] Sebastian Blumenthal, Erwin Prassler, Jan Fischer, e Walter Nowak. Towards identification of best practice algorithms in 3d perception and modeling. In *ICRA*, pág. 3554–3561. IEEE, 2011.

[BRBB09] Morten Rufus Blas, Radu Bogdan Rusu, Mogens Blanke, e Michael Beetz. Fault-tolerant 3d mapping with application to an orchard robot. In *The 7th IFAC International Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS)*, Barcelona, Spain, Junho 2009.

[BRF11] Liefeng Bo, Xiaofeng Ren, e Dieter Fox. Depth kernel descriptors for object recognition. In *IROS*, pág. 821–826. IEEE, Setembro 2011.

[BRF12] L. Bo, X. Ren, e D. Fox. Unsupervised Feature Learning for RGB-D Based Object Recognition. In *ISER*, Junho 2012.

[BRJ+11] John Bohren, Radu Bogdan Rusu, Gil Jones, Eitan Marder-Eppstein, Caroline Pantofaru, Melonee Wise, Lorenz Moesenlechner, Wim Meeussen, e Stefan Holzer. Towards Autonomous Robotic Butlers: Lessons Learned with the PR2. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, Maio 9-13 2011.

[BRSM+09] Radu Bogdan Rusu, Aravind Sundaresan, Benoit Morisset, Kris Hauser, Motilal Agrawal, Jean-Claude Latombe, e Michael Beetz. Leaving flatland: Efficient real-time three-dimensional perception and motion planning. *J. Field Robot.*, 26(10):841–862, Outubro 2009.

[CBC+01] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, e T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pág. 67–76, New York, NY, USA, 2001.

[CDR+07] Laura Clemente, Andrew Davison, Ian Reid, José Neira, e Juan Domingo Tardós. Mapping large loops with a single hand-held camera. In *Proc. Robotics: Science and Systems Conference*, Junho 2007.

[CLSF10] Michael Calonder, Vincent Lepetit, Christoph Strecha, e Pascal Fua. Brief: binary robust independent elementary features. In *Proceedings of the 11th European conference on Computer vision: Part IV*, ECCV'10, pág. 778–792, Berlin, Heidelberg, 2010. Springer-Verlag.

[CM05] Ondrej Chum e Jiri Matas. Matching with prosac "progressive sample consensus. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pág. 220–226, Washington, DC, USA, 2005. IEEE Computer Society.

[CN06] David M. Cole e Paul M. Newman. Using laser range data for 3d slam in outdoor environments. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, Maio 15-19, 2006, Orlando, Florida, USA*, pág. 1556–1563. IEEE, 2006.

[Col] Super Colossal. Último acesso: Out. 2012 [Online]. url: `http://supercolossal.ch/2008/07/15/house-of-cards/`.

[CV95] Corinna Cortes e Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, Setembro 1995.

[EEH+11] N. Engelhard, F. Endres, J. Hess, J. Sturm, e W. Burgard. Real-time 3d visual slam with a hand-held camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, Vasteras, Sweden, April 2011.

[EHE+12a] F. Endres, J. Hess, N. Engelhard, J. Sturm, e W. Burgard. 6D visual SLAM for RGB-D sensors. *at - Automatisierungstechnik*, 60:270–278, 2012.

[EHE+12b] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, e W. Burgard. An evaluation of the RGB-D SLAM system. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, St. Paul, MA, USA, Maio 2012.

[Elf89] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, Junho 1989.

[FB81] Martin A. Fischler e Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, Junho 1981.

[FCSS09] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, e Richard Szeliski. Reconstructing building interiors from images. In *ICCV*, pág. 80–87. IEEE, 2009.

[Fit01] A. W. Fitzgibbon. Robust registration of 2D and 3D point sets. In *British Machine Vision Conference*, pág. 662–670, 2001.

[Fox01] D. Fox. Kld-sampling: Adaptive particle filters. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

[Gar] Willow Garage. Overview. Último acesso: Out. 2012 [Online]. url: `http://www.willowgarage.com/turtlebot`.

[Gen11] Jason Geng. Three-dimensional image acquisition; three-dimensional sensing. *Adv. Opt. Photon.*, 3(2):128–160, Junho 2011.

[GKS+10] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, e C. Hertzberg. Hierarchical optimization on manifolds for online 2d and 3d mapping. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pág. 273 –278, maio 2010.

[GSGB07] Giorgio Grisetti, Cyrill Stachniss, Slawomir Grzonka, e Wolfram Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *In Proc. of Robotics: Science and Systems (RSS)*, 2007.

[HDD+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, e Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '92, pág. 71–78, New York, NY, USA, 1992. ACM.

[HHRB11] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, e Sven Behnke. Real-time plane segmentation using rgb-d cameras. In Thomas Röfer, Norbert Michael Mayer, Jesus Savage, e Uluc Saranli, editores, *RoboCup*, volume 7416 of *Lecture Notes in Computer Science*, pág. 306–317. Springer, 2011.

[Hiz] Hizook. Velodyne hdl-32e: A new high-end laser rangefinder. Último acesso: Out. 2012 [Online]. url: http://www.hizook.com/blog/2010/08/24/velodyne-hdl-32e-new-high-end-laser-rangefinder.

[HKH+10] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, e Dieter Fox. Rgbd mapping: Using depth cameras for dense 3d modeling of indoor environments. In *In RGB-D: Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS*, 2010.

[HLS08] Dirk Holz, Christopher Lörken, e Hartmut Surmann. Continuous 3D Sensing for Navigation and SLAM in Cluttered and Dynamic Environments. In *Proceedings of the International Conference on Information Fusion (FUSION)*, pág. 1469–1475, Cologne, Germany, Junho/Julho 2008.

[HMSL08] B. Huhle, M. Magnusson, W. Strasser, e A.J. Lilienthal. Registration of colored 3d point clouds with a kernel-based extension to the normal distributions transform. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pág. 4025 –4030, may 2008.

[Hol09] D. Holz. *Autonomous exploration and inspection*. Technical report. University of Applied Sciences Bonn-Rhein-Sieg, Department of Computer Science, 2009.

[Hor84] Berthold K. P. Horn. Extended gaussian images. *Proceedings of the IEEE*, 72(2):1671–1686, 1984.

[HRF11] Evan Herbst, Xiaofeng Ren, e Dieter Fox. Rgb-d object discovery via multi-scene analysis. In *IROS*, pág. 4850–4856. IEEE, 2011.

[HS88] C. Harris e M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pág. 147–151, 1988.

[HSD+10] Dirk Holz, Ruwen Schnabel, David Droeschel, Jörg Stückler, e Sven Behnke. Towards semantic scene analysis with time-of-flight cameras. In *RobuCup'10*, pág. 121–132, 2010.

[HWP11] Lulu He, Sen Wang, e Thrasyvoulos N. Pappas. 3d surface registration using z-sift. In Benoît Macq e Peter Schelkens, editores, *ICIP*, pág. 1985–1988. IEEE, 2011.

[IKH+11] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, e Andrew Fitzgibbon. Kinectfusion:

real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pág. 559–568, New York, NY, USA, 2011. ACM.

[JH98] Andrew Edie Johnson e Martial Hebert. Surface matching for object recognition in complex three-dimensional scenes. *Image Vision Comput.*, 16(9-10):635–651, 1998.

[KAJS11] Hema Swetha Koppula, Abhishek Anand, Thorsten Joachims, e Ashutosh Saxena. Semantic labeling of 3d point clouds for indoor scenes. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, e Kilian Q. Weinberger, editores, *NIPS*, pág. 244–252, 2011.

[Kit] KitGuru. Microsoft release windows kinect sdk. Último acesso: Out. 2012 [Online]. url: http://www.kitguru.net/software/carl/microsoft-release-windows-kinect-sdk/.

[KTCS09] Laurent Kneip, Fabien Tâche, Gilles Caprari, e Roland Siegwart. Characterization of the compact hokuyo urg-04lx 2d laser range scanner. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ICRA'09, pág. 2522–2529, Piscataway, NJ, USA, 2009. IEEE Press.

[LBRF11] Kevin Lai, Liefeng Bo, Xiaofeng Ren, e Dieter Fox. Sparse distance learning for object recognition combining rgb and depth information. In *ICRA*, pág. 4007–4013, 2011.

[LC87] William E. Lorensen e Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *COMPUTER GRAPHICS*, 21(4):163–169, 1987.

[Lev98] David Levin. The approximation power of moving least-squares. *Math. Comput.*, 67(224):1517–1531, Outubro 1998.

[Liba] Point Cloud Library. Downloads. Último acesso: Out. 2012 [Online]. url: http://pointclouds.org/downloads/.

[Libb] Point Cloud Library. Module features. Último acesso: Out. 2012 [Online]. url: http://docs.pointclouds.org/1.6.0/group__features.html.

[Libc] Point Cloud Library. Module io. Último acesso: Out. 2012 [Online]. url: http://docs.pointclouds.org/1.6.0/group__io.html.

[Libd] Point Cloud Library. Module kdtree. Último acesso: Out. 2012 [Online]. url: http://docs.pointclouds.org/1.6.0/group__kdtree.html.

[Libe] Point Cloud Library. Module sample_consensus. Último acesso: Out. 2012 [Online]. url: http://docs.pointclouds.org/1.6.0/group__sample__consensus.html.

[Libf] Point Cloud Library. Module surface. Último acesso: Out. 2012 [Online]. url: http://docs.pointclouds.org/1.6.0/group__surface.html.

[Libg] Point Cloud Library. pcl::harriskeypoint3d<pointint, pointoutt, normalt> class template reference. Último acesso: Out. 2012 [Online]. url: http://docs.pointclouds.org/1.6.0/classpcl_1_1_harris_keypoint3_d.html.

[Libh] Point Cloud Library. pcl::siftkeypoint<pointint, pointoutt> class template reference. Último acesso: Out. 2012 [Online]. url: http://docs.pointclouds.org/1.6.0/classpcl_1_1_s_i_f_t_keypoint.html.

[LLP+10] Ruosi Li, Lu Liu, Ly Phan, Sasakthi S. Abeysinghe, Cindy Grimm, e Tao Ju. Polygonizing extremal surfaces with manifold guarantees. In Gershon Elber, Anath Fischer, John Keyser, e Myung-Soo Kim, editores, *Symposium on Solid and Physical Modeling*, pág. 189–194. ACM, 2010.

[LM97] F. Lu e E. Milios. Globally consistent range scan alignment for environment mapping. *AUTONOMOUS ROBOTS*, 4:333–349, 1997.

[Low99] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pág. 1150–1157, Washington, DC, USA, 1999. IEEE Computer Society.

[Low04a] Kok-Lim Low. Linear least-squares optimization for point-to-plane icp surface registration. Relatório Técnico TR04-004, University of North Carolina, Fevereiro 2004.

[Low04b] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Novembro 2004.

[LSP05] Svetlana Lazebnik, Cordelia Schmid, e Jean Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1265–1278, 2005.

[MC02] Jiri Matas e Ondrej Chum. Randomized ransac with t(d, d) test. In Paul L. Rosin e A. David Marshall, editores, *BMVC*. British Machine Vision Association, 2002.

[MDH+08] Stefan May, David Droeschel, Dirk Holz, Christoph Wiesen, e Stefan Fuchs. 3D Pose Estimation and Mapping with Time-of-Flight Cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on 3D Mapping*, Nice, France, Outubro 2008.

[MDH+09] Stefan May, David Droeschel, Dirk Holz, Stefan Fuchs, Ezio Malis, Andreas Nüchter, e Joachim Hertzberg. Three-dimensional mapping with time-of-flight cameras. *J. Field Robot.*, 26(11):934–965, Novembro 2009.

[Mea82] D. Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, Junho 1982.

[ML09] Marius Muja e David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pág. 331–340, 2009.

[MLD07] Martin Magnusson, Achim J. Lilienthal, e Tom Duckett. Scan registration for autonomous mining vehicles using 3d-ndt. *J. Field Robotics*, 24(10):803–827, 2007.

[MRB09] Zoltan Csaba Marton, Radu Bogdan Rusu, e Michael Beetz. On fast surface reconstruction methods for large and noisy datasets. In *The IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, Maio 2009.

[MRBL11] Marius Muja, Radu Bogdan Rusu, Gary Bradski, e David Lowe. REIN - A Fast, Robust, Scalable REcognition Infrastructure. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, Maio 9-13 2011.

[MRS+09] Benoit Morisset, Radu Bogdan Rusu, Aravind Sundaresan, Kris Hauser, Motilal Agrawal, Jean-Claude Latombe, e Michael Beetz. Leaving Flatland: Toward Real-Time 3D Navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, Maio 12-17 2009.

[NIH+11] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, e Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pág. 127–136, Washington, DC, USA, 2011. IEEE Computer Society.

[Nob88] J. Alison Noble. Finding corners. *Image Vision Comput.*, 6(2):121–128, 1988.

[NSS+09] Paul Newman, Gabe Sibley, Mike Smith, Mark Cummins, Alastair Harrison, Chris Mei, Ingmar Posner, Robbie Shade, Derik Schroeter, Liz Murphy, Winston Churchill, Dave Cole, e Ian Reid. Navigating, recognizing and describing urban spaces with vision and lasers. *Int. J. Rob. Res.*, 28(11-12):1406–1433, Novembro 2009.

[NWL+05] Andreas Nüchter, Oliver Wulf, Kai Lingemann, Joachim Hertzberg, Bernado Wagner, e Hartmut Surmann. 3d mapping with semantic knowledge. In *IN ROBOCUP INTERNATIONAL SYMPOSIUM*, pág. 335–346, 2005.

[NYT] NYTimes. Your very own 400,000 robot. Último acesso: Out. 2012 [Online]. url: `http://bits.blogs.nytimes.com/2010/09/08/your-very-own-400000-robot`.

[Nü09] Andreas Nüchter. *3D Robotic Mapping: The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom*, volume 24. Springer Publishing Company, Incorporated, 1st edition, 2009.

[oCS] Institute of Computer Science. Project kurt3d. Último acesso: Out. 2012 [Online]. url: `http://www.inf.uos.de/kbs/KURT3D.html`.

[OJAC11] Agustín Alberto Ortega Jiménez e Juan Andrade-Cetto. Segmentation of dynamic objects from laser data. 2011.

[PLH04] Helmut Pottmann, Stefan Leopoldseder, e Michael Hofer. Registration without icp. *Comput. Vis. Image Underst.*, 95(1):54–71, Julho 2004.

[PMR+08] A. Prusak, O. Melnychuk, H. Roth, I. Schiller, e R. Koch. Pose estimation and map building with a time-of-flight camera for robot navigation. *Int. J. Intell. Syst. Technol. Appl.*, 5(3/4):355–364, Novembro 2008.

[PS99] Michael K Pitt e Neil Shephard. Filtering via simulation: auxiliary particle filters. Economics Papers 1997-W13, Economics Group, Nuffield College, University of Oxford, 1999.

[RBB09] Radu Bogdan Rusu, Nico Blodow, e Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, Maio 12-17 2009.

[RBF12] Xiaofeng Ren, Liefeng Bo, e Dieter Fox. Rgb-(d) scene labeling: Features and algorithms. In *CVPR*, pág. 2759–2766, 2012.

[RBM+07] Radu Bogdan Rusu, Nico Blodow, Zoltan-Csaba Marton, Alina Soos, e Michael Beetz. Towards 3d object maps for autonomous household robots. In *IROS*, pág. 3191–3198, 2007.

[RBM+09] Radu Bogdan Rusu, Jan Bandouch, Franziska Meier, Irfan Essa, e Michael Beetz. Human Action Recognition using Global Point Feature Histograms and Action Shapes. *Advanced Robotics journal, Robotics Society of Japan (RSJ)*, 2009.

[RBMB08] R.B. Rusu, N. Blodow, Z.C. Marton, e M. Beetz. Aligning point cloud views using persistent feature histograms. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pág. 3384 –3391, Setembro 2008.

[RBMB09] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, e Michael Beetz. Close-range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Human Environments. In *Proceedings of the 22nd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, Outubro 11-15 2009.

[RBTH10] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, e John Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, Outubro 2010.

[RC11] Radu Bogdan Rusu e Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, Maio 9-13 2011.

[Rei07] J. Reinders. *Intel threading building blocks : outfitting C++ for multicore processor parallelism*. O'Reilly, 2007.

[RHBB09] Radu Bogdan Rusu, Andreas Holzbach, Nico Blodow, e Michael Beetz. Fast Geometric Point Labeling using Conditional Random Fields. In *Proceedings of the 22nd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, Outubro 11-15 2009.

[RL87] P. J. Rousseeuw e A. M. Leroy. *Robust regression and outlier detection*. John Wiley & Sons, Inc., New York, NY, USA, 1987.

[RMB+08a] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, e Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robot. Auton. Syst.*, 56(11):927–941, Novembro 2008.

[RMB+08b] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Emanuel Dolha, e Michael Beetz. Functional object mapping of kitchen environments. In *IROS*, pág. 3525–3532, 2008.

[RMBB08] R.B. Rusu, Z.C. Marton, N. Blodow, e M. Beetz. Learning informative point classes for the acquisition of object model maps. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pág. 643 –650, Dezembro 2008.

[RMCB09] Radu Bogdan Rusu, Wim Meeussen, Sachin Chitta, e Michael Beetz. Laser-based Perception for Door and Handle Identification. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, Munich, Germany, Junho 22-26 2009. Best paper award.

[RMP+12] Andreas Richtsfeld, Thomas Mörwald, Johann Prankl, Jonathan Balzer, Michael Zillich, e Markus Vincze. Towards scene understanding - object segmentation using rgbd-images. In *Proceedings of the 2012 Computer Vision Winter Workshop (CVWW)*, Mala Nedelja, Slovenia, Fevereiro 2012.

[Rob] RobotShop. Hokuyo urg-04lx scanning laser rangefinder. Último acesso: Out. 2012 [Online]. url: `http://www.robotshop.com/ca/hokuyo-urg-04lx-laser-rangefinder-2.html`.

[RSG+09] Radu Bogdan Rusu, Ioan Alexandru Sucan, Brian Gerkey, Sachin Chitta, Michael Beetz, e Lydia E. Kavraki. Real-time Perception-Guided Motion Planning for a Personal Robot. In *Proceedings of the 22nd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, Outubro 11-15 2009.

[RSM+08] Radu Bogdan Rusu, Aravind Sundaresan, Benoit Morisset, Motilal Agrawal, e Michael Beetz. Leaving Flatland: Realtime 3D Stereo Semantic Reconstruction. In *Proceedings of the International Conference on Intelligent Robotics and Applications (ICIRA)*, Wuhan, China, Outubro 15-17 2008.

[Rus09] Radu Bogdan Rusu. Semantic 3d object maps for everyday manipulation in human living environments. phd, Tecnische Universitatet Muenchen, Munich, Germany, Outubro 2009.

[SAH08] Chanop Silpa-Anan e Richard Hartley. Optimised kd-trees for fast image descriptor matching. In *CVPR*. IEEE Computer Society, 2008.

[SB11] Ivan Sipiran e Benjamin Bustos. Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes. *The Visual Computer*, 27:963–976, 2011.

[SB12] Joerg Stueckler e Sven Behnke. Robust real-time registration of rgb-d images using multi-resolution surfel representations. *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, pág. 1 –4, Maio 2012.

[SBB12] Jörg Srückler, Nenad Biresev, e Sven Behnke. Semantic mapping using object-class segmentation of rgb-d images. In *Proceedings of IEEE/RSJ International Conference on Robots and Systems (IROS)*, Vilamoura, Portugal, Outubro 2012. IEEE.

[SBK01] Uluc Saranli, Martin Buehler, e Daniel E. Koditschek. Rhex: A simple and highly mobile hexapod robot. *International Journal of Robotics Research*, 20:616–631, 2001.

[SMFF07] Joaquim Salvi, Carles Matabosch, David Fofi, e Josep Forest. A review of recent range image registration methods with accuracy evaluation. *Image Vision Comput.*, 25(5):578–596, Maio 2007.

[SML06] W. Schroeder, K. Martin, e B. Lorensen. *The Visualization Toolkit: An Object-oriented Approach to 3D Graphics*. Kitware, 2006.

[SNLH09] Jochen Sprickerhof, Andreas Nüchter, Kai Lingemann, e Joachim Hertzberg. An explicit loop closing technique for 6d slam. pág. 229–234, Mlini/Dubrovnik, Croatia, Setembro 2009. 4th European Conf. Mobile Robots (ECMR-09).

[SRKB10] Bastian Steder, Radu Bogdan Rusu, Kurt Konolige, e Wolfram Burgard. Narf: 3d range image features for object recognition. In *Workshop*

on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS, Taipei, Taiwan, Outubro 8, 2010.

[SRKB11] Bastian Steder, Radu Bogdan Rusu, Kurt Konolige, e Wolfram Burgard. Point Feature Extraction on 3D Range Scans Taking into Account Object Boundaries. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, Maio 9-13 2011.

[SS03] Daniel Scharstein e Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the 2003 IEEE computer society conference on Computer vision and pattern recognition*, CVPR'03, pág. 195–202, Washington, DC, USA, 2003. IEEE Computer Society.

[SSS06] Noah Snavely, Steven M. Seitz, e Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, Julho 2006.

[ST94] Jianbo Shi e Carlo Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pág. 593 – 600, 1994.

[TB05] Rudolph Triebel e Wolfram Burgard. Improving simultaneous mapping and localization in 3d using global constraints. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 3*, AAAI'05, pág. 1330–1335. AAAI Press, 2005.

[TBF00] Sebastian Thrun, Wolfram Burgard, e Dieter Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. pág. 321–328, 2000.

[Tec] Alces Technology. Alces technology: 3d depth capture and structured light. Último acesso: Out. 2012 [Online]. url: `http://main.alcestech.com/2012/01/3d-depth-capture-and-sensing.html`.

[Thr02] Sebastian Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[TMD+06] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek,

C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, e P. Mahoney. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006.

[TSDS10] Federico Tombari, Samuele Salti, e Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Proceedings of the 11th European conference on computer vision conference on Computer vision: Part III*, ECCV'10, pág. 356–369, Berlin, Heidelberg, 2010. Springer-Verlag.

[TSdS11] Federico Tombari, Samuele Salti, e Luigi di Stefano. A combined texture-shape descriptor for enhanced 3d feature matching. In Benoît Macq e Peter Schelkens, editores, *ICIP*, pág. 809–812. IEEE, 2011.

[TZ00] P. H. S. Torr e A. Zisserman. Mlesac: a new robust estimator with application to estimating image geometry. *Comput. Vis. Image Underst.*, 78(1):138–156, Abril 2000.

[Vena] VentureBeatProfiles. Mesa imaging - company information on mesa imaging. Último acesso: Out. 2012 [Online]. url: http://venturebeatprofiles.com/company/profile/mesa-imaging.

[Venb] Gadget Venue. Microsoft kinect teardown. Último acesso: Out. 2012 [Online]. url: http://www.gadgetvenue.com/microsoft-kinect-teardown-08093500/.

[WHB+10] Kai M. Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, e Wolfram Burgard. Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *In Proc. of the ICRA 2010 workshop*, 2010.

[WHH+11] Kai M. Wurm, Daniel Hennes, Dirk Holz, Radu Bogdan Rusu, Cyrill Stachniss, Kurt Konolige, e Wolfram Burgard. Hierarchies of octrees for efficient 3d mapping. In *IROS*, pág. 4249–4255. IEEE, 2011.

[Wik] Wikipedia. k-d tree. Último acesso: Out. 2012 [Online]. url: http://en.wikipedia.org/wiki/File:3dtree.png.

[wll] wll. Último acesso: Out. 2012 [Online]. url: http://www.wll.kr/.

[WV11] Walter Wohlkinger e Markus Vincze. Ensemble of shape functions for 3d object classification. In *Proc. of the 2011 IEEE International Conference on Robotics and Biomemetics*, pág. 2987–2992. IEEE, Dezembro

2011. talk: IEEE International Conference on Robotics and Biomimetics (IEEE-ROBIO), Phuket Island, Thailand; 2011-12-07 – 2011-12-11.

[Zha94] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput. Vision*, 13(2):119–152, Outubro 1994.

[ZHWG08] Kun Zhou, Qiming Hou, Rui Wang, e Baining Guo. Real-time kd-tree construction on graphics hardware. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pág. 126:1–126:11, New York, NY, USA, 2008. ACM.

[ZSKW10] Leon Ziegler, Frederic Siepmann, Marco Kortkamp, e Sven Wachsmuth. Towards an informed search behavior for domestic robots. *SIMPAR 2010 Workshop on Domestic Service Robots in the Real World*, 2010.

A

RANSAC

O *RANdom SAmple Consensus* (RANSAC) [FB81] é um método iterativo, amplamente conhecido na computação visual, que permite a estimação de parâmetros de uma transformação de um dado *dataset*. Para que esses parâmetros sejam satisfeitos, é geralmente utilizado o método do mínimos quadrados. No entanto, quando se processa pontos com ruído associado, não é provável que esta solução retorne resultados considerados óptimos. Daí, a ideia será a de encontrar parâmetros que são válidos para a maioria dos pontos, um consenso, descartando pontos com ruído.

Em cada iteração, são seleccionadas aleatoriamente um pequeno número de *samples*, representando uma hipótese que define o modelo. Este modelo é então avaliado por todo o *dataset* com uma dada função de erro. Este processo é repetido diversas vezes, através da selecção de novos *samples* em cada iteração, mantendo a melhor transformação encontrada.

Depois de efectuar o procedimento um número fixo de vezes, o algoritmo garante a convergência para a melhor transformação, ou seja, a transformação com o menor erro associado.

O princípio é estabelecido da seguinte maneira: se $\boldsymbol{\epsilon}$ é a probabilidade de ser seleccionado um *sample* que retorna uma má estimativa (*outlier*), então $\mathbf{1} - \boldsymbol{\epsilon}$ é a probabilidade de que seja seleccionado pelo menos um bom *sample* (*inlier*). Isto significa que a probabilidade de serem seleccionados $\boldsymbol{s}$ bons *samples* seja, $(\mathbf{1} - \boldsymbol{\epsilon})^{\boldsymbol{s}}$.

Em $\boldsymbol{k}$ iterações, a probabilidade de fracassar torna-se $(\mathbf{1} - (\mathbf{1} - \boldsymbol{\epsilon})^{\boldsymbol{s}})^{\boldsymbol{k}}$. Se $\boldsymbol{p}$ é a probabilidade de sucesso desejada, então:

$$1 - p = (1 - (1 - \epsilon)^s)^k \Rightarrow k = \frac{\log(1 - p)}{\log(1 - (1 - \epsilon)^s)} \tag{A.1}$$

B

Kdtrees

Na ciência computacional, uma *kdtree* é uma estrutura de dados (árvore binária) que serve para efectuar operações de decomposição espacial. No PCL, este tipo de estruturas são largamente utilizadas em métodos que necessitam de pesquisas de vizinhança.

No primeiro nível (raiz) da árvore, o espaço é divido em duas metades por um hiperplano ortogonal a um dada dimensão escolhida (ex: x). Geralmente esta divisão é efectuada com a média da dimensão com o maior variância no conjunto dos dados. Por exemplo, se a dimensão x é escolhida, todos os pontos do espaço com um valor menor que x_i irão aparecer do lado esquerdo da árvore e os valores maiores irão aparecer do lado direito.

Cada uma das duas sub-divisões do espaço é recursivamente dividida em nodos, da mesma forma até que ser criada um árvore binária completamente balanceada. Na base da árvore, cada nodo irá corresponder a um único ponto do conjunto de dados.

Em operações de busca, uma *kdtree* tem um peso computacional de $O(logN)$, onde N é o número de pontos no conjunto de dados.

Na figura B.1[1] está representada a decomposição espacial efectuada na construção de uma kd-tree. A primeira divisão (vermelho) divide raiz (branco) em duas partes. Cada uma destas partes é depois dividida em outras duas células (verde), sendo cada uma destas divididas em mais duas outras células (azul).

[1] Imagem retirada de [Wik]

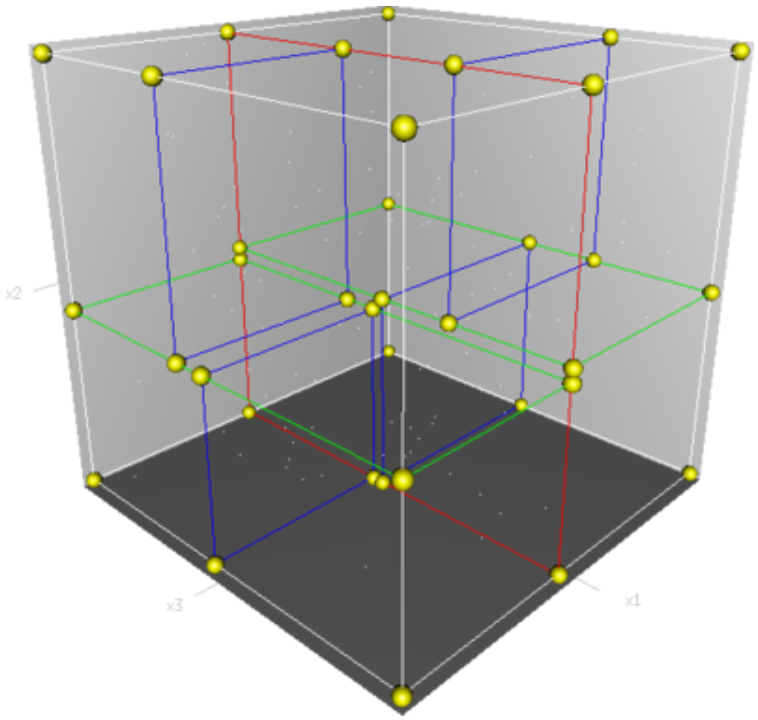


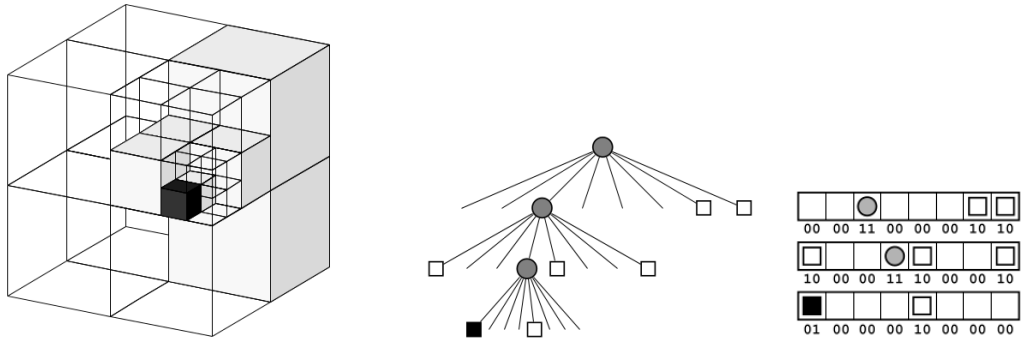
Figura B.1: Exemplo da decomposição espacial efectuada na construção de uma kd-tree tridimensional.

C

Octrees

A utilização de *octrees* na área da computação gráfica, foi originalmente proposto por Donald Meagher [Mea82] em 1982.

Uma *octree*, é uma estrutura de dados hierárquica utilizada no particionamento de um espaço tridimensional. Cada nodo de uma octree representa o espaço contido num volume cúbico, normalmente denominado como voxel. Este volume é subdividido recursivamente em oito sub-volumes até que o tamanho mínimo do voxel é atingido, como é ilustrado na figura C.1[1]. O tamanho mínimo do voxel é determinado pela resolução da *octree* [WHB+10].



(a) (b) (c)

Figura C.1: Exemplo do particionamento do espaço realizado por uma *octree*, com armazenamento de células vazias (branco sombreado) e ocupadas (preto) (a); a correspondente representação da árvore (b); e o fluxo de dados para o armazenamento compacto num ficheiro (c). A estrutura completa da *octree* apresentada pode ser armazenada recorrendo a apenas 6 bytes, 2 bits por cada sub-nodo de cada nodo.

[1] imagens retiradas de [WHB+10]

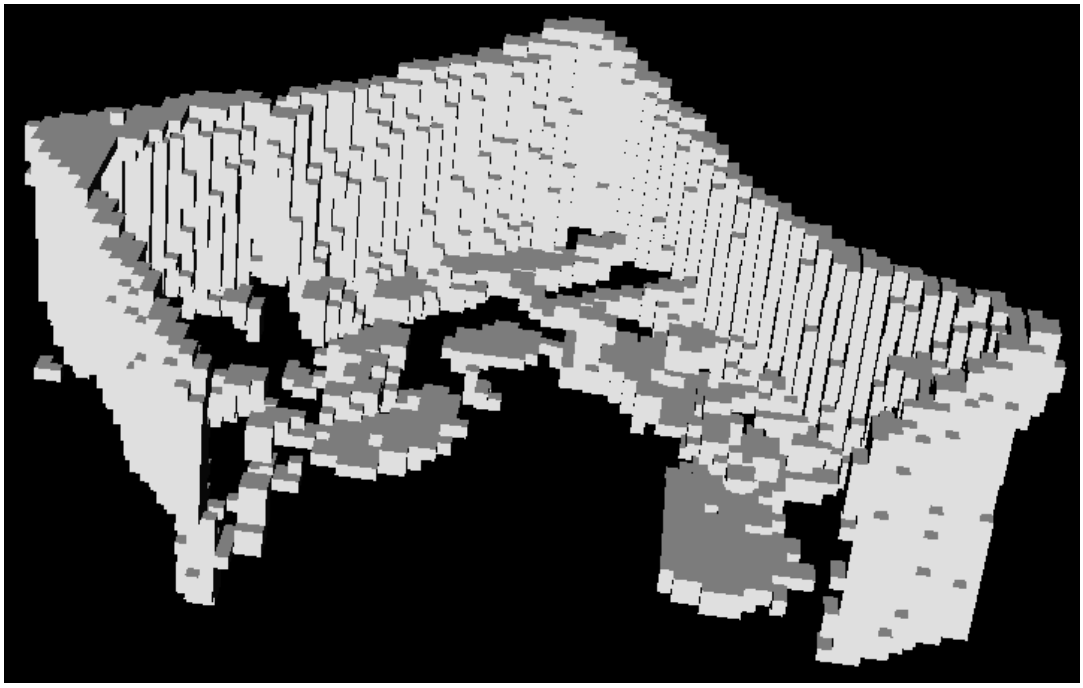


Figura C.2: Exemplo de uma nuvem de pontos voxelizada com recurso a *octrees*; *leaf size*: 0,1280m; 8054 voxeis.

D

Código genérico da implementação dos métodos avaliados do PCL

D.1 Módulo Filters

Listagem D.1: Código genérico do filtro passThrough.

```
1       CloudPtr passCloud (new Cloud);
2
3       pcl::PassThrough<PointT> *pass = new pcl::PassThrough<PointT>;
4       pass->setInputCloud (input);
5       pass->setKeepOrganized (is_organized);
6       pass->setFilterFieldName ("z");
7       pass->setFilterLimits (limit_min, limit_max);
8
9       pass->filter (*passCloud);
```

Listagem D.2: Código genérico do filtro Statistical Outlier Removal.

```
1  CloudPtr inliersCloud (new Cloud);
2
3  pcl::StatisticalOutlierRemoval<PointT> *sor = new
       pcl::StatisticalOutlierRemoval<PointT>;
4  sor->setInputCloud (input);
5  sor->setKeepOrganized (is_organized);
6  sor->setMeanK (nr_k);
7  sor->setStddevMulThresh (stddev_multi);
8
9  sor->filter (*inliersCloud);
```

Listagem D.3: Código genérico do filtro Radius Outlier Removal.

```
1  CloudPtr inliersCloud (new Cloud);
2
3  pcl::RadiusOutlierRemoval<PointT> *radOutRem = new
       pcl::RadiusOutlierRemoval<PointT>;
4  radOutRem->setInputCloud (input);
5  radOutRem->setKeepOrganized (is_organized);
6  radOutRem->setRadiusSearch (radius);
7
8  radOutRem->filter (*inliersCloud);
```

Listagem D.4: Código genérico dos métodos Voxel Grid e Approximate Voxel Grid.

```
1  CloudPtr output (new Cloud);
2  Eigen::Vector4f &leaf-size
3
4  pcl::VoxelGrid<PointT> *grid = new pcl::VoxelGrid<PointT>;   // no
       caso do filtro Approximate Voxel Grid a inicialização da class é
       pcl::ApproximateVoxelGrid<PointT> *grid = new pcl::ApproximateVoxelGrid<PointT>;
5
6  grid->setInputCloud (input);
7  grid->setLeafSize (leaf-size);
8
9  grid->filter (*output);
```

D.2 Módulo Keypoints

Listagem D.5: Código genérico do detector SIFT.

```
1   PointCloudXYZI::Ptr keypts (new PointCloudXYZI);
2   typename pcl::search::KdTree<PointT>::Ptr tree (new
        pcl::search::KdTree<PointT> ());
3
4   typename pcl::SIFTKeypoint<PointT, PointXYZIT> *siftkey = new
        pcl::SIFTKeypoint<PointT, PointXYZIT>;
5
6     siftkey->setSearchMethod (tree);
7     siftkey->setScales (min_scale, nr_octaves,
          nr_scales_per_octave);
8     siftkey->setMinimumContrast (min_constrast);
9     siftkey->setInputCloud (input);
10    siftkey->setSearchSurface (input);
11
12    siftkey->compute (*keypts);
```

Listagem D.6: Código genérico do detector HARRIS3D.

```
1   PointCloudXYZI::Ptr keypts (new PointCloudXYZI);
2
3     typename pcl::HarrisKeypoint3D<PointT, PointXYZIT, NormalT>
          *harris3D = new pcl::HarrisKeypoint3D<PointT, PointXYZIT,
          NormalT>;
4
5   harris3D->setInputCloud (input);
6     harris3D->setRadius (radius);
7     harris3D->setRadiusSearch (radiusSearch);
8     harris3D->setThreshold (thresh);
9     harris3D->setNumberOfThreads(nr_threads);
10  harris3D->setMethod (pcl::HarrisKeypoint3D<PointT,
        PointXYZIT>::TOMASI);
11
12    harris3D->compute (*keypts);
```

D.3 Módulo Features

D.3.1 Estimação de normais

Listagem D.7: Código genérico da estimação de normais.

```
1   Normals::Ptr output (new Normals);
2   typename pcl::search::KdTree<PointT>::Ptr tree (new
        pcl::search::KdTree<PointT> ());
3
4   pcl::NormalEstimationOMP<PointT, NormalT> *ne = new
        pcl::NormalEstimationOMP<PointT, NormalT>;
5
6     ne->setNumberOfThreads (nr_threads);
7     ne->setInputCloud (input);
8     ne->setSearchMethod (tree);
9     ne->setRadiusSearch (radius);
10
11    ne->compute (*output);
```

Listagem D.8: Código genérico da estimação de normais pelo método Integral Images.

```
1   Normals::Ptr output (new Normals);
2
3   pcl::IntegralImageNormalEstimation<PointT, NormalT> *iine = new
        pcl::IntegralImageNormalEstimation<PointT, NormalT>;
4
5     iine->setNormalEstimationMethod (iine->COVARIANCE_MATRIX);
6
7     iine->setInputCloud (input);
8     iine->setMaxDepthChangeFactor (max_depth_change_factor);
9     iine->setDepthDependentSmoothing (true);
10    iine->setNormalSmoothingSize (normal_smoothing_size);
11
12    iine->compute (*output);
```

D.3.2 Descritores geométricos locais

Listagem D.9: Código genérico da estimação da geometria local pelo método PFH.

```
1   PointCloudPFHSig125::Ptr features (new PointCloudPFHSig125);
2   typename pcl::search::KdTree<PointT>::Ptr tree (new
        pcl::search::KdTree<PointT> ());
3
4     pcl::PFHEstimation<PointT, NormalT, PFHSig125T> *ldpfh = new
          pcl::PFHEstimation<PointT, NormalT, PFHSig125T>;
5
6     ldpfh->setSearchSurface (input);
7     ldpfh->setInputCloud (keypoints);
8     ldpfh->setInputNormals (normals);
9     ldpfh->setSearchMethod (tree);
10    ldpfh->setRadiusSearch (radiusSearch);
11
12    ldpfh->compute (*features);
```

Listagem D.10: Código genérico da estimação da geometria local pelo método FPFH.

```
1   PointCloudFPFHSig33::Ptr features (new PointCloudFPFHSig33);
2   typename pcl::search::KdTree<PointT>::Ptr tree (new
        pcl::search::KdTree<PointT> ());
3
4     pcl::FPFHEstimationOMP<PointT, NormalT, FPFHSig33T> *ldfpfh =
          new pcl::FPFHEstimationOMP<PointT, NormalT, FPFHSig33T>;
5
6     ldfpfh->setNumberOfThreads (nr_threads);
7     ldfpfh->setSearchSurface (input);
8     ldfpfh->setInputCloud (keypoints);
9   ldfpfh->setInputNormals (normals);
10  ldfpfh->setSearchMethod (tree);
11
12  ldfpfh->setRadiusSearch (radiusSearch);
13
14    ldfpfh->compute (*features);
```

Listagem D.11: Código genérico da estimação da geometria local pelo método SHOT.

```
1     PointCloudSHOT::Ptr features (new PointCloudSHOT);
2     typename pcl::search::KdTree<PointT>::Ptr tree (new
          pcl::search::KdTree<PointT> ());
3
4       pcl::SHOTEstimationOMP<PointT, NormalT, ShotT> *ldshot = new
            pcl::SHOTEstimationOMP<PointT, NormalT, ShotT>;
5
6     ldshot->setSearchSurface (input);
7       ldshot->setInputCloud (keypoints);
8       ldshot->setInputNormals (normals);
9       ldshot->setRadiusSearch (radiusSearch);
10      ldshot->setNumberOfThreads (nr_threads);
11
12      ldshot->compute (*features);
```

D.4 Módulo Registration

Listagem D.12: Código genérico da estimação da transformação inicial pelo método SAC-IA.

```
1   Eigen::Matrix4f tflaSac = Eigen::Matrix4f::Identity ();
2
3   pcl::SampleConsensusInitialAlignment<PointT, PointT, FPFHSig33T>
        sac_ia;
4
5   sac_ia.setMaxCorrespondenceDistance (max_corresp_dist);
6     sac_ia.setMaximumIterations (nr_iterations);
7     sac_ia.setInputCloud (source);
8     sac_ia.setSourceFeatures (source-descriptors);
9     sac_ia.setInputTarget (target);
10    sac_ia.setTargetFeatures (target-descriptors);
11
12    sac_ia.align (source);
13    tflaSac = sac_ia.getFinalTransformation();
```

Listagem D.13: Código genérico da estimação da transformação inicial pelo método CE-RSAC.

```
1  pcl::CorrespondencesPtr correspondences (new pcl::Correspondences);
2  Eigen::Matrix4f tfRejSAC = Eigen::Matrix4f::Identity ();
3
4  pcl::registration::CorrespondenceEstimation<FPFHSig33T, FPFHSig33T>
       corr_est;
5
6  corr_est.setInputCloud (source-descriptors);
7    corr_est.setInputTarget (target-descriptors);
8    corr_est.determineReciprocalCorrespondences (*correspondences);
9
10 pcl::registration::CorrespondenceRejectorSampleConsensus<PointXYZIT>
       corr_rej_sac;
11   corr_rej_sac.setInputCloud (source-keypts);
12   corr_rej_sac.setTargetCloud (target-keypts);
13
14   corr_rej_sac.setInlierThreshold (max_dist);
15   corr_rej_sac.setMaxIterations (max_iter);
16   corr_rej_sac.setInputCorrespondences (correspondences);
17
18   corr_rej_sac.getCorrespondences (*correspondences);
19 tfRejSAC = corr_rej_sac.getBestTransformation();
```

Listagem D.14: Código genérico do refinamento do registo pelo ICP Linear e ICP Não Linear.

```
1    Eigen::Matrix4f tfICP = Eigen::Matrix4f::Identity ();
2
3    pcl::IterativeClosestPoint<PointT, PointT> icp; // ou
         pcl::IterativeClosestPointNonLinear<PointT, PointT> icp;
4    icp.setMaxCorrespondenceDistance (max_correspondence_distance);
5      icp.setRANSACOutlierRejectionThreshold
           (outlier_rejection_threshold);
6      icp.setTransformationEpsilon (transformation_epsilon);
7      icp.setMaximumIterations (max_iterations);
8
9      icp.setInputCloud (source);
10     icp.setInputTarget (target);
11
12     icp.align (*source);
13     tfICP = icp.getFinalTransformation ();
```

D.5 Módulo Surface

Listagem D.15: Código genérico do RMLS.

```
1    CloudPtr resampledCloud (new Cloud);
2
3      typename pcl::search::KdTree<PointT>::Ptr tree (new
           pcl::search::KdTree<PointT>);
4      pcl::MovingLeastSquaresOMP<PointT, PointT> *mls = new
           pcl::MovingLeastSquaresOMP<PointT, PointT>;
5
6      mls->setInputCloud (input);
7      mls->setSearchMethod (tree);
8      mls->setNumberOfThreads(nr_threads);
9      mls->setSearchRadius (radius);
10     mls->setPolynomialFit (true);
11     mls->setPolynomialOrder (2);
12     mls->setPointDensity (60000 * int (radius));
13
14     mls->setUpsamplingRadius (0.025);
15     mls->setUpsamplingStepSize (0.015);
16     mls->setDilationIterations (2);
17     mls->setDilationVoxelSize (0.01f);
18
19     mls->process (*resampledCloud);
```

D.6 Módulo Segmentation

Listagem D.16: Código genérico da segmentação de formas paramétricas.

```
1  CloudPtr output (new Cloud);
2  pcl::PointIndices::Ptr inliers (new pcl::PointIndices);
3
4  pcl::SACSegmentationFromNormals<PointT, NormalT> seg;
5
6  seg.setOptimizeCoefficients(true);
7  seg.setModelType (pcl::SACMODEL_NORMAL_PLANE);
8    seg.setMethodType (pcl::SAC_RANSAC);
9  seg.setDistanceThreshold (threshold);
10   seg.setNormalDistanceWeight (distance_weight);
11   seg.setMaxIterations (max_iteration);
12   seg.setInputCloud (input);
13   seg.setInputNormals (normals);
14
15   seg.segment (*inliers, *coeffs);
```

Listagem D.17: Código genérico da extracção de clusters.

```
1  std::vector<pcl::PointIndices> cluster_indices;
2
3  boost::shared_ptr<pcl::KdTree<PointT> > tree (new
       pcl::KdTreeFLANN<PointT>);
4  tree->setInputCloud(input);
5
6  unsigned int max_pts_per_cluster=(std::numeric_limits<int>::max)
       ();
7
8  pcl::extractEuclideanClusters (*input, *normals, tolerance, tree,
       cluster_indices, eps_angle, min_pts_per_cluster,
       max_pts_per_cluster);
```