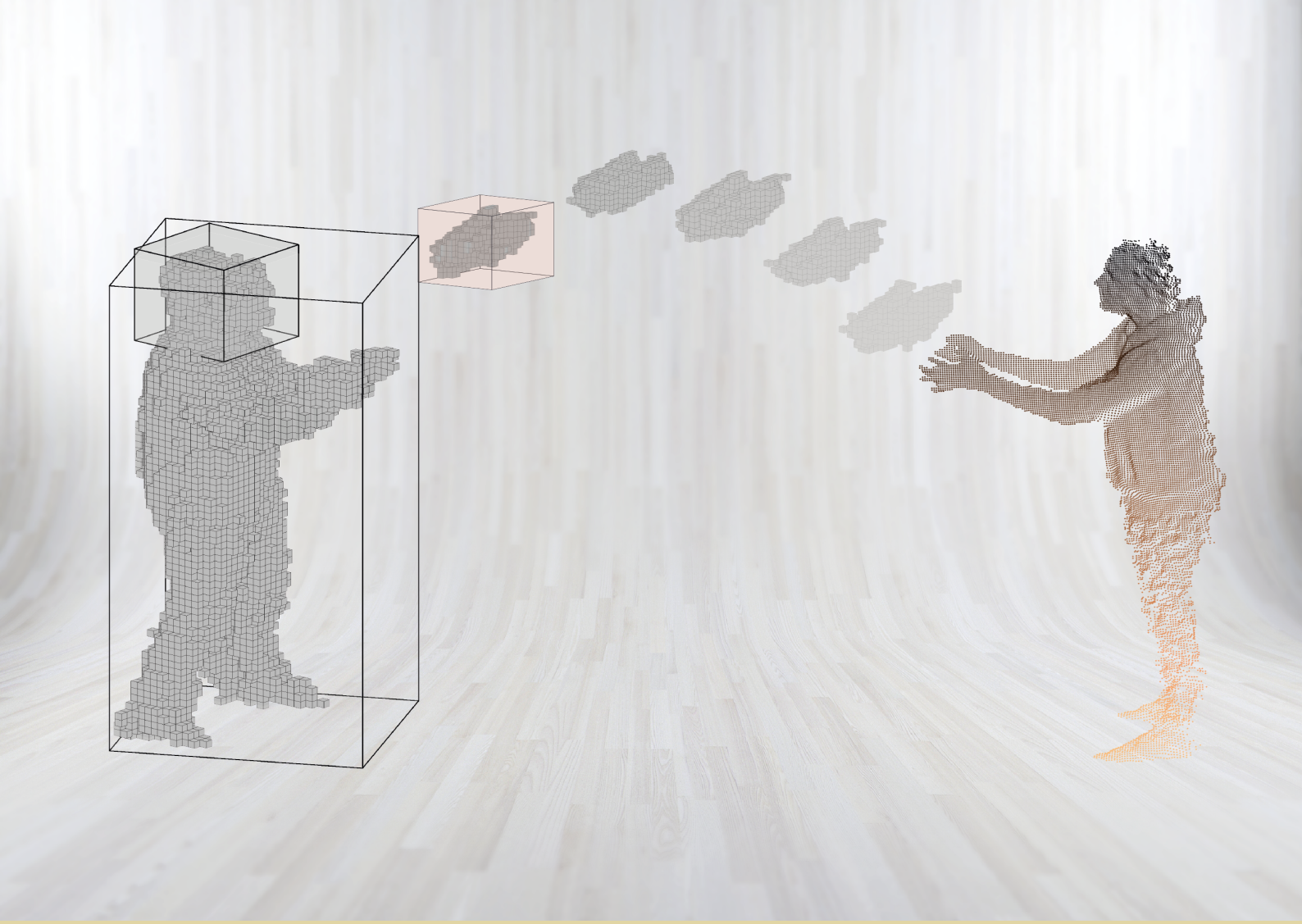


Dylan Jordão Bicho

Deteção e Seguimento de Objetos utilizando Grelhas de Ocupação para Aplicações em Realidade Aumentada

Dissertação submetida para a satisfação parcial dos requisitos do grau de

Mestre em Engenharia Eletrotécnica e de Computadores

Setembro de 2018

· U C ·

Universidade de Coimbra

FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Deteção e Seguimento de Objetos utilizando Grelhas de Ocupação para Aplicações em Realidade Aumentada

Dylan Jordão Bicho

Setembro de 2018

Deteção e Seguimento de Objetos utilizando Grelhas de Ocupação para Aplicações em Realidade Aumentada

Orientador:

Professor Paulo Peixoto

Júri:

Professor Paulo Jorge Carvalho Menezes

Professor Rui Pedro Duarte Cortesão

Dissertação submetida para a satisfação parcial dos requisitos do grau de Mestre em Engenharia Eletrotécnica e de Computadores

Setembro de 2018

Agradecimentos

Apesar da tese de mestrado ser um trabalho individual, existiram diversos contributos importantes para a conclusão da mesma, que não podem, nem devem deixar de ser realçados. Expresso, de seguida, os meus sinceros agradecimentos:

Ao Professor Paulo Peixoto por todo o seu apoio prestado não só ao longo da tese, que ativamente me orientou e acompanhou, como também durante todo o percurso académico.

Ao Pedro Girão, um muito obrigado pela presença constante durante todos estes meses, onde sempre procurou me guiar pacientemente para a direção certa através do seu espírito crítico.

Ao Luís Garrote que, ao longo das inúmeras "noitadas" passadas no departamento, transmitiu-me os seus conhecimentos científicos, dando valiosas contribuições para este trabalho.

Aos meus amigos de curso, em especial, João, Eduardo, Miguel, que me acompanharam nos bons e maus momentos deste percurso.

Ao meu primo Cristian, que apesar da sua distância geográfica, esteve sempre presente e me apoiou incondicionalmente, fazendo-me acreditar em mim próprio.

Ao meu pai e a minha irmã por todo o apoio, compreensão, paciência, carinho e por fazerem de mim, o que sou hoje.

Por fim, mas não menos importante, à Raquel, pela força transmitida, pelo amor demonstrado em todo o caminho percorrido em conjunto, pelo encorajamento e acima de tudo pela sua paciência.

A todos o meu sincero e profundo **muito obrigado**!

Resumo

Ao longo dos últimos anos, os sistemas de Realidade Virtual e Realidade Aumentada têm vindo a ser desenvolvidos com o intuito de fornecer ao utilizador uma experiência totalmente imersiva através de uma estimulação sensorial artificial, trazendo inúmeros benefícios em várias áreas desde a saúde à educação. Contudo, estes sistemas encontram-se ainda limitados por diversos fatores: uma representação não realista da cena, falta de personalização e flexibilidade, viabilidade financeira, desconforto físico e psicológico dos utilizadores causando experiências nauseantes, entre outros. Estes também exigem que o utilizador se desloque num espaço vazio ou muito limitado pois não recriam o ambiente físico em que o utilizador se move num ambiente virtual com uma relação um-para-um (tanto nos movimentos efetuados como na interação com objetos presentes).

No entanto, o desenvolvimento de tecnologias no domínio dos microprocessadores e do processamento gráfico, bem como o aparecimento de sensores de captura de informação tridimensional de baixo custo, mais eficientes para o mapeamento de cenários reais, tais como as Microsoft® Kinect v2, têm vindo a tornar estes sistemas mais viáveis financeiramente. Nestes sistemas, uma boa representação tridimensional do ambiente é uma tarefa essencial, pois o seguimento de objetos e dos utilizadores é uma das componentes chave deste processo. Um pré-processamento da informação sensorial extraída dos sensores facilita este processo. Aplicando técnicas de seguimento a um objeto, é possível estimar a sua localização e a sua velocidade bem como prever futuros estados do mesmo.

Nesta tese é proposto um sistema modular para a representação de um cenário tridimensional através de grelhas de ocupação recorrendo à informação sensorial de quatro Microsoft® Kinects v2. Este processo pode ser dividido essencialmente em três módulos: primeiro, é feita a captura dos dados sensoriais das câmaras sendo posteriormente aplicadas técnicas para filtrar o ruído existente e para remover a informação relativa ao plano de fundo do cenário; depois, são aplicadas técnicas para a segmentação da nuvem de pontos construída; e finalmente, são aplicados filtros Bayesianos (tanto filtros de partículas, como filtros de

Kalman) para o seguimento de todos os objetos e da cabeça de todos os utilizadores presentes no cenário. Deste modo, é obtida uma estimativa para a localização e para a velocidade instantânea dos objetos, bem como uma estimativa da sua próxima localização.

O processo supramencionado foi sujeito a uma série de testes realizados em situações particularmente exigentes, sendo os resultados qualitativos obtidos apresentados neste documento. Os resultados demonstram que o sistema proposto é capaz de realizar o seguimento de qualquer objeto presente na cena, estando este limitado porém no caso de ocorrer uma interação com um objeto dinâmico. Relativamente ao módulo de seguimento de cabeça, este demonstrou ser robusto e aplicável em tempo real.

Palavras Chave

Sensores RGB-D, Grelhas de Ocupação 3D, Segmentação, Filtro de Kalman, Filtro de Partículas.

Abstract

Over recent years, Virtual Reality and Augmented Reality systems have been developed with the mission of giving a user a completely immersive experience through artificial stimulation of the user's senses, and have brought countless benefits in several areas such as health and education. However, these systems are still limited by different challenging factors: absence of a realistic representation of the real world, lack of customization and flexibility, financial viability, physical and psychological discomfort causing nauseating effects, among others. These systems also demand that the user moves in an empty or very limited space, given that they do not recreate the physical environment where the user is moving into the virtual representation in a one-to-one nature (both in user movements as well as objects that are present).

In spite of these facts, the technological advances in the domains of microprocessors and graphical processing, as well as the upcoming low-cost and efficient 3D data capturing sensors such as the Microsoft® Kinect v2 (useful for real scenario reconstructions), have made such systems more financially viable. In these systems, a good 3D representation of the environment is key to success, seeing that object and user tracking is one of the most important steps of this process. A pre-processing step of the extracted sensory data makes this process easier. Applying tracking techniques to the present objects, it is possible to know the location and velocity of any given object, as well as estimate its future states.

In this thesis a modular system is presented for the accurate representation of a real-world scenario as 3D occupancy grids using sensory data from four Microsoft® Kinect v2. This process can be divided into three essential modules: first, sensory data is captured from the cameras and image processing techniques are applied to filter out noise and information related to the background of the environment; next, 3D segmentation techniques are applied to the constructed point clouds; finally, Bayesian filters (both particle filters as well as Kalman filters) are applied to track all the objects in the scene, as well as the heads of all the users. In this way, an estimation of all relevant objects and users location and

instantaneous velocity, as well as their next location, is obtained.

The aforementioned process was subject to a series of particularly challenging tests, with results of these qualitative tests presented in this document. The results show that the proposed system is capable of correctly tracking any object present in the scene (being however limited by a possible interaction between dynamic objects). In addition, the user head tracking module showed to be robust and deployable in a real-time application.

Keywords

RGB-D Sensors, 3D Occupancy Grids, Segmentation, Kalman Filter, Particle Filters.

"Nothing is permanent in this world, not even our troubles."

— Charlie Chaplin

Contents

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

2.5D Duas Dimensões e Meia.

2D Bidimensional.

3D Tridimensional.

AR Realidade Aumentada.

BB *Bounding Box*.

CAKF *Constant Acceleration Kalman Filter*.

CH *Convex Hull*.

DBSCAN *Density-Based Algorithm for Discovering Clusters*.

FoV *Field of View* - Campo de visão.

FPS *Frames per Second* - Imagens por Segundo.

ICP *Iterative Closest Point*.

IR Infravermelho.

KF Filtro de Kalman - *Kalman Filter*.

MAR *Minimum Area Rectangle* - Retângulo de Área Mínima.

NN *Nearest Neighbors* - Vizinhos mais próximos.

OMBB *Oriented Minimum Bounding Box*.

PC USB *Peripheral Component Universal Serial Bus*.

PCD *Point Cloud* - Nuvem de Pontos.

PF Filtro de Partículas - *Particle Filter*.

RBNN *Radially Bounded Nearest Neighbor*.

RGB-D *Red Green Blue - Depth*.

SCC Sistema de Coordenadas da Câmara.

SCI Sistema de Coordenadas da Imagem.

SCM Sistema de Coordenadas do Mundo.

SL *Structured Light* - Luz Estruturada.

ToF *Time-of-Flight* - Tempo de Voo.

VR Realidade Virtual.

1 Introdução

A presente dissertação enquadra-se no domínio dos sistemas de perceção para o contexto da realidade virtual, aumentada e mista, mais concretamente no domínio da reconstrução, segmentação, seguimento e interação do utilizador com o cenário. O trabalho desenvolvido encontra-se inserido no projeto *Human Tracking and Perception in Dynamic Immersive Rooms* (HTPDIR). Neste projeto são utilizados 4 sensores *Red Green Blue - Depth* (RGB-D) para capturar as imagens de profundidade do cenário no qual o ambiente se encontra fisicamente presente. O objetivo principal desta dissertação é o de fazer a construção de um mundo virtual, tendo em conta o ambiente real do cenário, recorrendo para tal a sensores que façam a captura de informação da cena. Será processada também a informação relativa à forma e a posição de todos os objetos presentes no cenário, bem como a posição da cabeça de cada utilizador.

1.1 Contexto

Os sistemas de realidade imersiva (virtual, aumentada e mista) estão constantemente a ser desenvolvidos e têm atraído muito interesse nos últimos anos [1]. Sendo um novo paradigma da interface do utilizador, oferece grandes benefícios e é agora, alvo de inúmeras aplicações, desde a saúde (p. ex. na investigação no domínio da neurociência [2]) ao entretenimento. O primeiro conceito para uma Realidade Aumentada (AR) foi apresentada em 1965, por Ivan Sutherland: "fazer com que esse mundo (virtual) pareça real, soe real, seja sentido de forma real, e responda realisticamente às ações do visualizador" [3].

No entanto, a grande maioria desses sistemas requer uma elevada capacidade computacional, sendo estes constituídos por dispositivos e máquinas tecnologicamente avançadas, tornado-os significativamente dispendiosos. A capacidade computacional tem sido uma forte restrição nestes sistemas, sendo que apenas nos últimos anos foi superada através de um enorme desenvolvimento na tecnologia dos microprocessadores e no processamento gráfico.

Estes desenvolvimentos têm vindo a trazer ao mercado dispositivos com uma capacidade computacional elevada [4].

No caso particular dos sistemas de realidade aumentada e mista e para garantir uma melhor imersão dos utilizadores é necessário um processo robusto de *virtualização*, onde os seus estímulos sensoriais são manipulados, i.e., é necessário um mapeamento do ambiente real no mundo virtual, para que o utilizador, quando esteja no ambiente virtual, se sinta dentro desse mesmo ambiente, dando-lhe a hipótese de interagir com os elementos que o constituem. As dimensões destes elementos deverão corresponder a objetos de dimensões idênticas no mundo real sendo necessária uma representação espacial precisa, sendo este passo crucial para alcançar uma experiência de realidade virtual coerente.

Para se obter a informação relativa à forma e à localização dos diversos objetos no mundo real, são necessárias então técnicas de mapeamento, manipulação e interpretação de dados (como deteção e seguimento de objetos).

1.2 Objetivos

O projeto HTPDIR supramencionado visa desenvolver um sistema de deteção e seguimento de objetos utilizando grelhas de ocupação num ambiente virtualizado, para aplicações de AR. Este deverá apresentar caraterísticas tais como baixo custo, intuitividade e eficiência de processamento de dados recorrendo:

- à fusão da informação sensorial obtida a partir de múltiplos sensores RGB-D;
- à implementação de técnicas para a deteção de objetos a partir da grelha Tridimensional (3D) de informação;
- à implementação de um módulo de seguimento de objetos através da aplicação de filtros de Kalman;
- ao seguimento da posição da cabeça de todos os utilizadores presentes no cenário, utilizando filtros de partículas.

Para além dos requisitos acima apresentados, também será feita uma avaliação dos métodos propostos.

1.3 Visão Geral do Sistema Proposto

O problema proposto será estruturado modularmente, de modo a ser possível identificar e seguir uma coerência lógica ao longo de todo o processo.

O sistema proposto é composto por duas etapas: a primeira consiste num procedimento de calibração que é essencial sempre que o sistema é instalado num ambiente diferente; a segunda corresponde a todo o processo principal desde a aquisição sincronizada de dados a partir de 4 sensores RGB-D, até à segmentação e ao seguimento de objetos presentes no ambiente.

Na Figura 1.1 é apresentado um diagrama conceptual de todo o sistema proposto nesta dissertação:

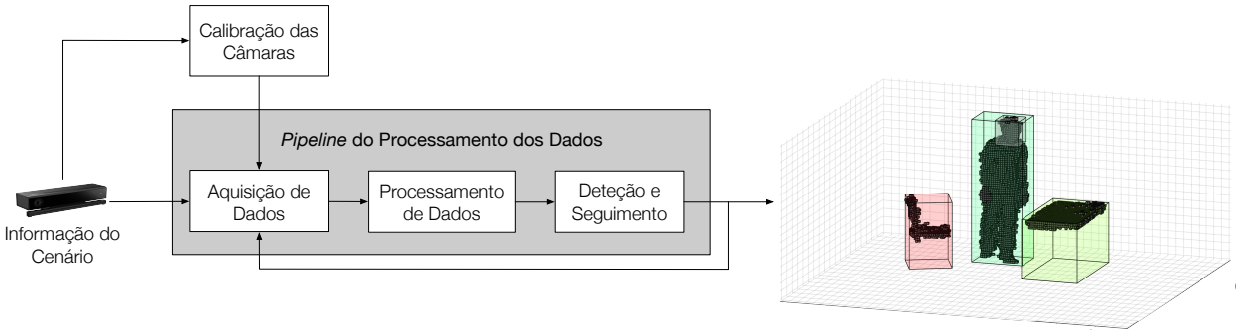


Figura 1.1: Visão conceptual da arquitetura do sistema proposto.

1.4 Principais Contribuições

A presente dissertação propõe um método do qual resultam três contribuições principais:

- Representação sob forma de grelhas de ocupação 3D da fusão da informação sensorial relevante capturada por múltiplos sensores RGB-D;
- Segmentação da informação obtida no domínio 3D, para a deteção de todos os objetos no cenário, recorrendo a métodos de segmentação eficientes;
- Seguimento de todos os objetos presentes na cena, recorrendo à aplicação de Filtro de Kalman - *Kalman Filter* (KF)s. É também feito o seguimento da cabeça de todos os utilizadores, através de um método proposto onde é aplicado um Filtro de Partículas - *Particle Filter* (PF)s para esse efeito.

1.5 Organização

O presente documento está organizado em cinco capítulos. Inicialmente, no presente capítulo, o problema a resolver é contextualizado e é brevemente apresentado o fluxo de processamento no sistema proposto.

No Capítulo 2, é feito um breve estudo sobre as diversas tipologias de sensores de captura de informação de profundidade. Posteriormente, são apresentados vários métodos descritos na literatura para permitir o mapeamento 3D de ambientes. Também são apresentadas algumas das técnicas mais comuns para a segmentação de objetos no domínio 3D, bem como os métodos normalmente utilizados para o seguimento desses objetos ao longo do tempo.

O Capítulo 3 descreve detalhadamente todo o sistema proposto, desde a extração da informação por parte dos sensores Kinect até aos módulos de seguimento.

Posteriormente é feita uma análise dos resultados experimentais dos módulos desenvolvidos no Capítulo 4. Os conjuntos de capturas usados para a avaliação dos métodos serão descritos e uma ligação *Uniform Resource Locator* (URL) da respetiva captura será fornecida.

No Capítulo 5 as conclusões extraídas do presente trabalho serão apresentadas. Serão também sugeridas possíveis melhorias ou extensões ao trabalho desenvolvido.

Todas as imagens apresentadas neste documento estão na forma vetorial.

Qualquer figura que não seja visível na versão impressa poderá ser visualizada com melhor definição no formato digital.

2 Estado da Arte

2.1 Sensores de captura

A construção de um cenário virtual que mapeie um cenário real cujas informações espaciais (sob a forma de profundidade), entre outras, são detetadas a partir de um ou vários sensores. A reconstrução 3D de ambientes reais tem sido objeto de estudo nos últimos anos. Parte da atração por este tipo de sistemas de reconstrução espacial é o reconhecimento do valor que estes podem potencialmente trazer aos sistemas de Realidade Virtual (VR) ou AR. Para fazer a captura da informação espacial dos ambientes a reconstruir, é possível utilizar diversos sensores de várias tipologias, dentro das quais se podem considerar: sensores de cor (câmaras RGB) em configurações mono ou estéreo; sensores que recorrem à tecnologia de *Structured Light* - Luz Estruturada (SL); sensores *Time-of-Flight* - Tempo de Voo (ToF); sensores RGB-D.

Considerando a breve comparação das várias tecnologias de sensores de captura que existem atualmente no mercado, apresentada no Apêndice A.1 e o fato do sistema pretendido dever ser de baixo custo, os sensores Kinect v2 desenvolvidos pela Microsoft®, demonstraram ser a escolha mais recomendada tendo em conta os requisitos pretendidos.

2.1.1 Microsoft Kinect v2

O sensor Kinect v2 adquire imagens de profundidade, Infravermelho (IR) e RGB a uma taxa de captura de 30 *Frames per Second* - Imagens por Segundo (FPS). Desde a sua apresentação, múltiplas áreas de investigação mostraram interesse na sua utilização (devido à combinação da informação visual e de profundidade a um preço bastante reduzido), desde a robótica [5, 6] à engenharia biomédica [7, 8] e à visão por computador [9, 10].

Após o lançamento da segunda geração da Kinect em julho de 2014, não tardaram a aparecer estudos relativos ao desempenho da mesma para a modelação 3D de curto alcance [11, 12], onde foi analisada a influência da temperatura do sensor no processo de medida, a

influência dos materiais e das cores, o desempenho em ambientes externos, etc.

Sendo a informação de profundidade processada recorrendo à tecnologia ToF (o cálculo da distância é obtido através do conhecimento da velocidade da luz, medindo o tempo de voo de um sinal de ótico, desde a sua emissão até a sua receção pelo sensor, para cada ponto da imagem), há problemas a ter em conta durante a operação destes sensores. Para além do ruído ser dependente da luz refletida nas superfícies ou no sensor, um dos principais problemas associado a este tipo de sensor é o fato dos mesmos sofrerem da interferência multi-caminho *(multipath interference)*. Este fenómeno, deve-se aos raios de luz enviados para cada píxel individual que, podendo refletir de várias formas, podem fazer com que um específico píxel receba fotões originalmente enviados para outros píxeis. Freedman et al. apresentou um algoritmo de nome *Sparse Reflections Analysis* [13], que procura resolver este problema em tempo real e para interferências gerais, incluindo 3 ou mais caminhos.

Com base nos motivos supramencionados, e tendo em conta que é possível mitigar grande parte do efeito multi-caminho através da colocação e orientação dos sensores, a Kinect v2 foi o sensor escolhido para a aquisição da informação espacial do cenário.

2.2 Mapeamento de ambientes

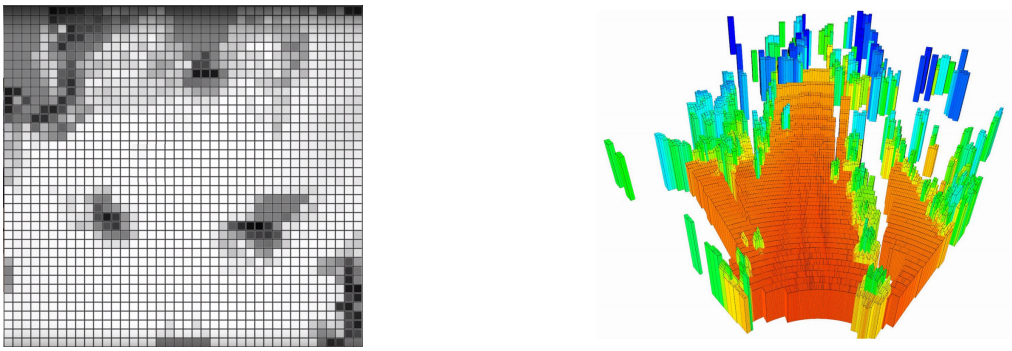
Os métodos para mapear a estrutura 3D de ambientes internos foram estudados inicialmente no contexto da navegação robótica. Foram sugeridos dois paradigmas para o mapeamento de ambientes internos [14]: topológicos e representação baseada em grelhas. Os métodos baseados em grelhas produzem mapas métricos e a sua complexidade é superior, o que pode dificultar a sua utilização em situações em que a área a mapear tenha grandes dimensões. Por outro lado, mapas topológicos podem ser mais eficientes, contudo a sua aplicabilidade é mais comum na área da navegação robótica (permite um planeamento rápido do percurso a ser efetuado), tendo pouca expressão na representação 3D para ambientes de AR.

As abordagens baseadas em grelhas, como as propostas por Moravec [15] entre outras, representam o ambiente através de grelhas igualmente espaçadas. Cada célula na grelha representa a presença de um objeto nessa área ou volume correspondente e, possivelmente, a respetiva probabilidade de ocupação.

2.2.1 Mapa de grelhas de ocupação 2D e 2.5D

A representação do mundo através da construção de um mapa Bidimensional (2D) (Figura 2.1a), através de grelhas de ocupação discretas, foi originalmente proposto por Elfes [16] e Movarec [15] e desde então, a sua utilização passou a ser frequente em sistemas de navegação robótica.

Posteriormente, os mapas de elevação (mapa Duas Dimensões e Meia (2.5D)) foram introduzidos para o mapeamento de superfícies não-planas. Este método é similar ao tradicional mapa 2D com a vantagem de fornecer a altura de cada célula (representativa de um obstáculo ou da própria irregularidade no terreno). Uma das primeiras aplicações de um mapa de elevação, foi a prototipagem de um rover planetário para uma missão de exploração em Marte [17]. Recentemente, foi proposta uma representação de ambientes urbanos baseada em grelhas 2.5D polares [18] (Figura 2.1b) devido à natureza do sensor laser de captura, que de forma rotacional procede à captura dos respetivos pontos.



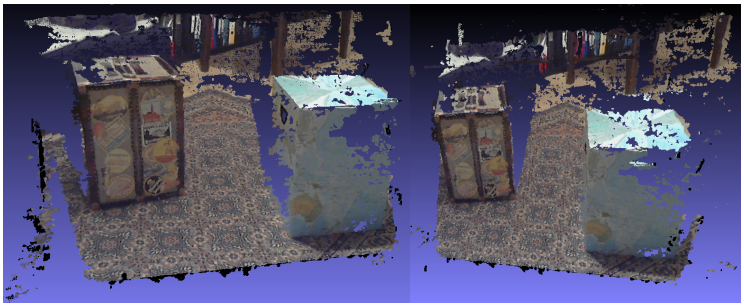
(a) Grelha 2D (imagem retirada de [19]). **(b)** Grelha Polar 2.5D (imagem retirada de [18]).

Figura 2.1: Exemplos de mapeamentos recorrendo a grelhas 2D e a mapas de elevação.

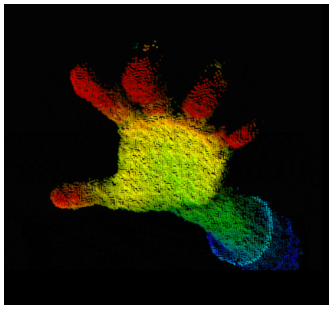
Contudo, a contribuição da sua implementação para sistemas de AR é limitada pois tipicamente requer-se um maior detalhe na descrição da cena.

2.2.2 Nuvem de pontos

Uma forma comum da representação de uma cena 3D é através de uma *Point Cloud* - Nuvem de Pontos (PCD). Uma PCD é, um conjunto de pontos representado num sistema de coordenadas 3D (geralmente definidos por x, y e z), que representa as superfícies externas dos objetos físicos presentes no *Field of View* - Campo de visão (FoV) do sensor. Alguns exemplos da representação da cena através de uma PCD obtida através de duas tecnologias distintas, são apresentadas nas Figuras 2.2a e 2.2b.



(a) PCD obtida através de um sistema de visão *stereo* (imagem retirada de [20]).



(b) PCD obtida através de um sensor ToF (imagem retirada de [21]).

Figura 2.2: Exemplos da representação de um cenário através de uma PCD.

2.2.3 Mapa de grelhas de ocupação 3D

Uma forma alternativa de representar informação volumétrica é utilizando uma grelha 3D discreta e regular de *voxels*[1] (Figura 2.3), usando uma grelha cartesiana como plano de referência. O processo de conversão de um objeto geométrico (p. ex. representado por uma PCD) num conjunto de *voxels* que melhor representa o objeto é chamado de voxelização.

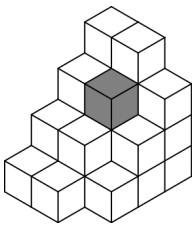


Figura 2.3: Conjunto de *voxels* empilhados com um único voxel em destaque (imagem retirada de [22]).

Alternativamente, outras estruturas de dados foram desenvolvidas e novos formatos têm sido usados para armazenar e manipular o conjunto de dados 3D, tais como a decomposição de células em *octrees* [24], *r-trees* [25] ou em *octomaps* [26] (baseado nas *octrees*, Figura 2.4). Estes métodos de armazenamento representam um papel importante pois, para mapas de grandes dimensões ou de alta resolução, a alocação de toda a grelha 3D requer muita memória e estes métodos procuram mitigar o seu uso através da alocação eficiente dos dados em estruturas ou árvores.

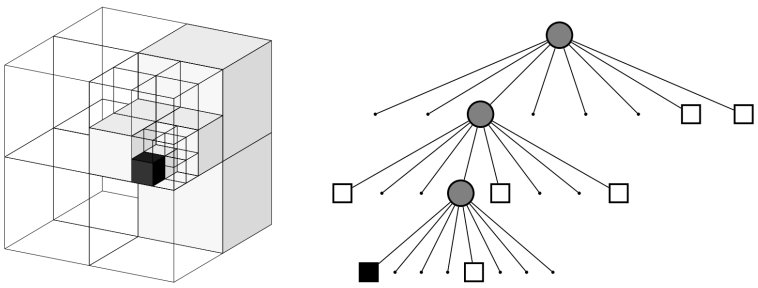


Figura 2.4: Exemplo de um armazenamento por *octree*, células vazias (branco sombreado) e ocupadas (preto). O modelo volumétrico é mostrado à esquerda e a correspondente árvore à direita. (imagem retirada de [27]).

[1]Como descrito por Kaufman et al. [23], um *voxel* é a unidade cúbica de volume centrado no ponto da respetiva grelha.

2.3 Técnicas de Segmentação

Os algoritmos de *clustering* desempenham um papel importante em vários domínios desde a análise estatística de dados [28], ao reconhecimento de padrões [29] e à compressão de dados [30]. Estes, procuram classificar individualmente cada amostra de um conjunto de dados, separando-os em distintos grupos (*clusters*) de tal forma que, amostras que apresentem características semelhantes se encontrem no mesmo grupo. Formalmente, a estrutura dos *clusters* é representada como um conjunto de subconjuntos $C = C_1, \ldots, C_k$ em S, de tal forma que: $S = \bigcup_{i=1}^{k} C_i$ e $C_i \cap C_j = \varnothing \; \forall \; i \neq j$. Assim, qualquer instância em S pertence a um e um só subconjunto. Os métodos de *clustering* vistos de uma perspetiva de alto-nível, podem ser divididos em [31]:

1. *Técnicas Hierárquicas:* dividem iterativamente o conjunto de dados em subconjuntos usando uma representação baseada em árvores (dendrogramas);
2. *Técnicas de Partição:* divide o conjunto de dados em subconjuntos não sobrepostos.

A sua aplicação em grandes quantidades de dados, com as resultantes de representações 3D em grelha ou numa PCD, requer um grau mínimo do conhecimento sobre a cena (não só para a determinação de parâmetros de entrada do algoritmo, como também para manter a sua implementação computacionalmente eficiente).

Os métodos baseados em técnicas hierárquicas não constituem uma boa solução para a combinação destes requisitos, pois esta abordagem tem uma complexidade de tempo de $\mathcal{O}(n^3)$ e $\mathcal{O}(n^2)$ para a memória (onde n representa o número de amostras), o que torna a sua implementação muito lenta até para conjuntos de dados de dimensão média [32].

2.3.1 Técnicas de Partição (ou Não-Hierárquicas)

Os métodos de segmentação baseados em centroides[2], representam os *clusters* através de um conjunto de centroides, que podem não corresponder necessariamente a um ponto do conjunto de dados. Quando o número de *clusters* é fixo e conhecido, o método *K-means* [33] transforma o problema de partição numa questão de otimização: este encontra os centroides e atribui todos os pontos ao centroide mais próximo, minimizando assim as distâncias quadráticas médias aos *clusters*.

[2]Um centroide é geralmente representado por um ponto que corresponde à média aritmética de um grupo de pontos.

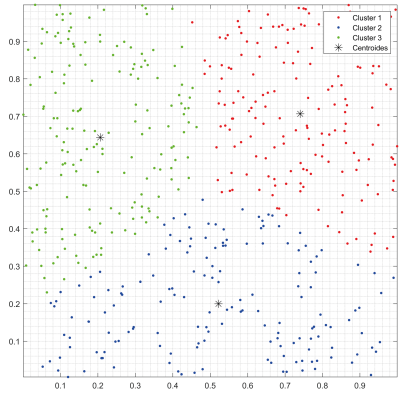


Figura 2.5: Exemplo da aplicação do algoritmo *K-means*.

Contudo, os resultados obtidos através do método podem não ser repetíveis, tratando-se de um método inconsistente. Isto deve-se à escolha aleatória inicial dos centroides que pode resultar em distintos *clusters* para diferentes execuções do algoritmo. No entanto, existentes variantes deste algoritmo que procuram superar este problema. Uma das variantes é o *K-means++* [34] que adiciona uma etapa inicial para uma escolha menos arbitrária dos centroides na inicialização. Lee et al. [35] aplicou esta variante para a criação de *clusters* geográficos de fotos, com base na informação de latitude e longitude anexadas às fotos. Recentemente, Rukmi et al. [36] aplicou o método para a deteção de relações entre investigadores com base nas publicações realizadas.

No entanto, o fato de ser necessário ter conhecimento prévio do cenário para a escolha do número de *clusters* a identificar e, dos métodos baseados em centroides não lidarem com ruído, identificando sempre cada ponto ao centroide mais próximo (como patente na Figura 2.5) forçou a procura de outros métodos alternativos.

Como alternativa, o conhecido algoritmo *Density-Based Algorithm for Discovering Clusters* (DBSCAN) [37] procura resolver as questões acima identificadas de forma mais eficiente. A abordagem utilizada baseia-se na noção de densidade para descobrir *clusters* de formas arbitrárias, sendo eficiente para conjuntos de dados espaciais de grande dimensão. O algoritmo procura por *clusters* através da pesquisa na vizinhança ϵ (raio máximo da vizinhança) de cada ponto no conjunto de dados e verifica se este contém mais que um mínimo de pontos $minPts$ (onde *epsilon* e $minPts$ são dois parâmetros a ser inicialmente definidos), caso esta condição não se verifique, este será definido como ruído.

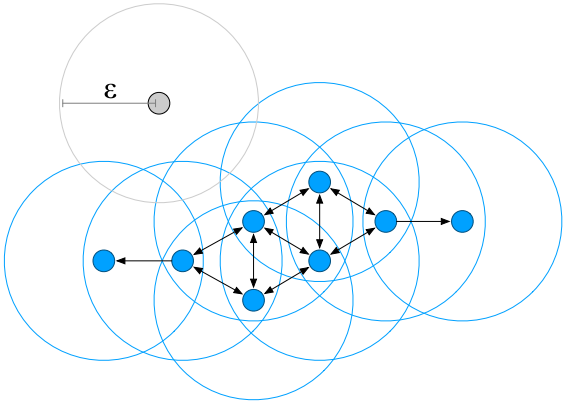


Figura 2.6: Exemplo da implementação do método DBSCAN com $minPts = 4$ (pontos a azul representam a identificação de um *cluster*, e a cinzento encontra-se a informação que, no final do algoritmo, não pertence a nenhum *cluster* e é considerada como ruído).

Os parâmetros de entrada ϵ e $minPts$ são fortemente dependentes dos dados. É evidente que se houver conhecimento prévio sobre o cenário da aplicação, nomeadamente no que diz respeito aos objetos de interesse que estão presentes, essa informação pode ser útil para ajustar ambos os parâmetros. Contudo, caso não haja conhecimento prévio, Sander et al. [38] sugerem inicializar o valor com o dobro da dimensionalidade do conjunto de dados, i.e, $minPts = 2 \cdot dim$ (e para conjuntos de dados, que contenham muito ruído associado, aumentar $minPts$ poderá melhorar os resultados).

Para mais informações relativamente aos métodos de *clustering* existentes, [39] apresenta de forma detalhada uma série de técnicas hierárquicas e de partição possíveis de ser utilizadas no contexto da segmentação.

2.4 Técnicas de Seguimento

As técnicas baseadas em filtros Bayesianos [40], são uma ferramenta probabilística eficaz para a estimação do estado de um sistema dinâmico quer a partir de observações ruidosas, como através da fusão de múltiplos dados sensoriais de distintos sensores. Na estimação de uma localização 3D, o estado pode ser a localização de pessoa/objeto, podendo este ser definido apenas como uma posição 3D, ou através de um vetor mais detalhado incluindo ângulos *pitch-roll-yaw* (ângulos relativamente aos principais eixos de inércia), bem como velocidades lineares e rotacionais.

Os filtros Bayesianos representam o estado no tempo t pela variável x_t. A cada momento no tempo, a distribuição probabilística em x_t, chamada de *belief* - confiança, $bel\,(x_t)$, representa a incerteza. Assumindo que a informação extraída do sensor consiste numa sequência temporal indexada de observações $z_1,\ z_2,\ \ldots,\ z_t$. A confiança $bel\,(x_t)$, é então definida por:

$$bel\,(x_t) = p\,(x_t | z_1,\ z_2,\ \ldots,\ z_t) \tag{2.1}$$

i.e., $bel\,(x_t)$ é condicionada a toda a informação disponível do sensor no tempo t face ao seu estado x_t. Sempre que o (ou um) sensor apresenta uma nova observação z_t, o filtro prevê o estado de acordo com,

$$bel^{-}(x_t) \leftarrow \int p\,(x_t \mid x_{t-1})\, bel\,(x_{t-1})\, dx_{t-1} \tag{2.2}$$

Posteriormente, este corrige a predição estimada usando a nova observação do sensor,

$$bel(x_t) \leftarrow \alpha_t p\,(z_t | x_t)\, bel^{-}(x_t) \tag{2.3}$$

onde α_t, é uma constante de normalização.

2.4.1 Filtros de Kalman

Os KF [41] são uma variante dos filtros Bayesianos, vastamente utilizados em sistemas de seguimento críticos [42, 43]. Estes são estimadores ótimos que assumem que a incerteza inicial é gaussiana e que, tanto o modelo de observação como a dinâmica do sistema são funções lineares do estado. Visto que muitos dos sistemas não são estritamente lineares, normalmente é aplicada uma extensão do KF, que lineariza o sistema usando expressões de primeira ordem da série de Taylor.

A principal vantagem do KF é a sua eficiência computacional, que advém de operações matriciais eficientes tanto na fase de predição (Equação 2.2), como na de correção (Equação 2.3). No entanto, o filtro é limitado e a sua aplicação é recomendada apenas caso a incerteza do estado não seja muito elevada, devido a este representar apenas distribuições unimodais (distribuições normais).

2.4.2 Filtro de Partículas

Desde a sua introdução, os PF [44] têm apresentado um papel importante em vários problemas relativos à estimação de estados incluindo, seguimento de pessoas [43, 45], navegação robótica [43] e deteção de falhas em sistemas [46, 47].

O PF é uma implementação não-paramétrica alternativa aos filtros Bayesianos, onde representa a probabilidade posterior $bel(x_t)$ através de um conjunto de amostras aleatórias extraídas desta probabilidade posterior, tornando-se numa ferramenta mais robusta e flexível, nomeadamente para sistemas altamente não-lineares, embora relativamente ao KF (ou à sua extensão) apresente uma menor eficiência computacional.

As amostras da distribuição posterior são chamadas de partículas e são denotadas por

$$\mathcal{X}_t := \{x_t{}^{[1]}, x_t{}^{[2]}, ..., x_t{}^{[M]}\} \tag{2.4}$$

onde M é o número de partículas do conjunto.

Para cada partícula $x_t^{[m]}$ (com $1 \leq m \leq M$) há um fator de importância não-negativo normalizado, $w_t^{[m]}$, dessa mesma partícula. Fatores de importância são usados para incorporar a medida z_t em $\mathcal{X}_t$, i.e, é a probabilidade da medida z_t perante a partícula $x_t^{[m]}$, $w_t^{[m]} = p(z_t \mid x_t^{[m]})$. Interpretando $w_t^{[m]}$ como o peso da partícula, o conjunto de pesos

representa, aproximadamente, $bel(x_t)$,

$$bel(x_t) \approx \left\{ \langle x_t^{(m)}, w_t^{(m)} \rangle \mid m = 1, \ldots, M \rangle \right\} \tag{2.5}$$

3 Método Proposto

3.1 Configuração do Sistema

A disposição das câmaras no cenário foi feita tendo por base a necessidade de criar o maior espaço possível usando 4 sensores Kinect v2. Tendo como conhecimento que a distância máxima da leitura de profundidade é de 4.5 m, um espaço de 4 $m \times 4$ m será o indicado para um reconstrução correta do cenário. Para minimizar possíveis interferências entre as câmaras, estas foram colocados a 2.2 m no topo de um perfil de alumínio, orientada para baixo por um ângulo $\simeq 30^{o}$.

Um diagrama de baixo-nível, representativo do *pipeline* implementado bem como cada módulo individual no processo é apresentado na Figura 3.1.

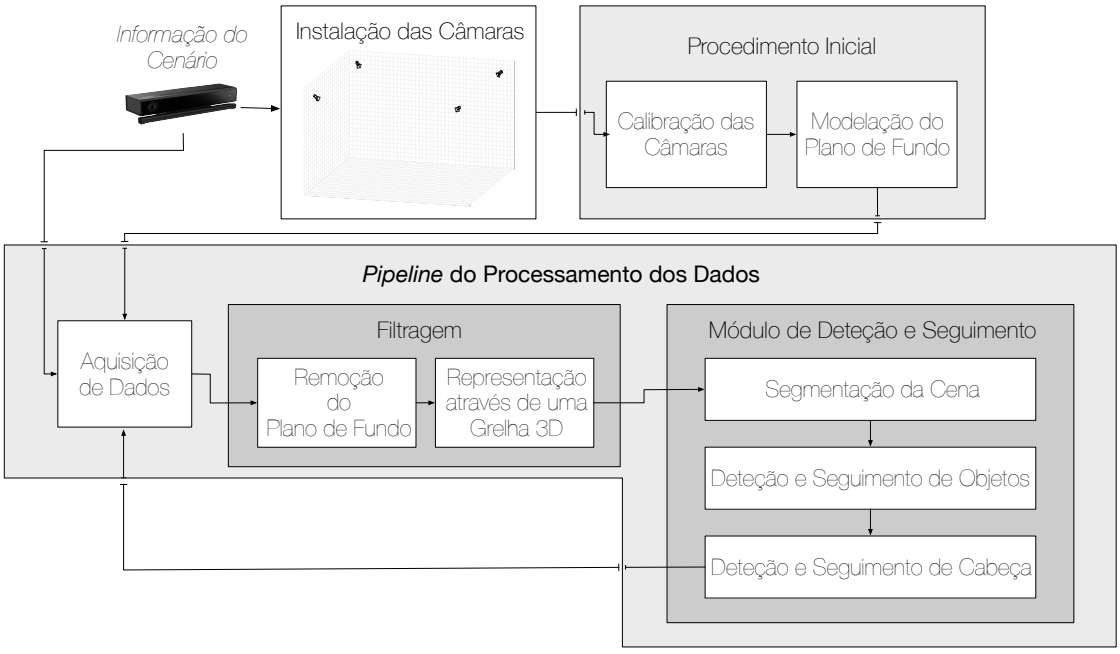


Figura 3.1: Diagrama de baixo-nível representativo do método proposto.

3.2 Aquisição de Dados

A recolha de informação por parte da câmara Kinect, para os fins do projeto, é feita apenas usando o sensor IR, capturando assim tanto a imagem IR como o mapa de profundidade. Para a captura das imagens foi desenvolvida uma plataforma em C++, onde foram utilizadas drivers *open-source* para a Kinect fornecidas pela biblioteca libfreenect2 [48]. Esta escolha deveu-se ao facto de das *drivers* oficiais fornecidas pela Microsoft® apenas suportarem a utilização de 1 sensor Kinect por computador [49]. Sendo desejado trabalhar com 4 câmaras a uma taxa de captura de 30 FPS, há um conjunto de considerações que se devem ter em conta: ao nível do hardware, será necessário usar múltiplas *Peripheral Component Universal Serial Bus* (PC USB) 3.0 Expansion Cards, visto que cada uma das câmaras usará $\sim 50\%$ da largura de banda; outro fator que se revelou importante foi a perda da qualidade do sinal USB que se verificou ocorrer a partir dos ~ 4.57 metros [50]. Por este motivo, foi necessário ligar os sensores mais afastados do computador a extensões com a funcionalidade de *Active Boosting*, que replicam o sinal enviado pelo sensor.

Como demonstrado por Oliver Wasenmüller, existe uma correlação forte entre a precisão das medições de profundidade e a temperatura do sensor [51]. Desta forma, é recomendado que haja um pré-aquecimento da Kinect de, pelo menos, 25 minutos antes de iniciar qualquer processo de captura.

3.3 Calibração das Câmaras

Face à necessidade de converter os mapas de profundidade retornados pelos sensores Kinect numa PCD num sistema de referência comum, o Sistema de Coordenadas do Mundo (SCM), é necessário proceder à calibração dos parâmetros da câmara. Nestes parâmetros incluem-se os intrínsecos, que se referem às características inerentes dos componentes da câmaras, e os extrínsecos, que identificam a localização e a orientação na cena.

Deste modo, o presente módulo de calibração das câmaras visa estimar os parâmetros intrínsecos, extrínsecos e a distorção da lente da câmara. Ambos os módulos de calibração usam um padrão de calibração axadrezado, com um tamanho do quadrado conhecido, onde os cantos internos são usados para a estimação dos parâmetros da câmara. Para tal foram criados dois tabuleiros (Apêndice B.1) impressos em papel fotográfico para evitar a ondulação causada pela injeção da tinta:

- um tabuleiro, de menores dimensões, para uma calibração dos parâmetros intrínsecos;
- outro, de dimensões superiores, indicado para a calibração dos parâmetros extrínsecos.

3.3.1 Calibração dos Parâmetros Intrínsecos

A implementação adotada para o processo de calibração dos parâmetros intrínsecos baseia-se principalmente no método proposto por Z. Zhang [52], presente na *toolbox* "Camera Calibrator" do MATLAB®.

Para uma melhor estimação dos parâmetros intrínsecos, capturou-se individualmente para cada sensor 250 imagens do padrão (Apêndice B.1a), variando a orientação 3D do mesmo e o seu posicionamento, procurando percorrer todo o FoV da câmara, principalmente próximo das arestas e nos cantos da imagem, a fim de se obter uma melhor estimação dos coeficientes de distorção da lente.

Com o intuito de verificar a precisão da calibração, analisou-se o erro de reprojeção através da aplicação dos parâmetros obtidos para cada imagem (Apêndice C.1). Esse estudo permite identificar as imagens que contribuem negativamente para a calibração, procedendo-se seguidamente, à eliminação das mesmas e a uma posterior recalibração.

3.3.2 Calibração dos Parâmetros Extrínsecos

Os parâmetros extrínsecos são compostos por uma matriz rotacional R_w^i e uma de translação T_w^i, que denotam a transformação do SCM para o Sistema de Coordenadas da Câmara (SCC) i, com $i \in \{1, \ldots, 4\}$. Tendo em conta que as imagens IR e os mapas de profundidade são extraídas do mesmo sensor, não se requer uma calibração *stereo* da câmara. O procedimento adotado para a calibração dos parâmetros extrínsecos foi desenvolvido por Bouguet [53] e foi utilizada a implementação presente na *toolbox "Camera Calibration"* para o MATLAB®. Foi ainda aplicado um método de *Iterative Closest Point* (ICP) para minimizar a diferença entre a reprojeção dos pontos do xadrez e os pontos do xadrez no mundo. Para uma reprojeção mais robusta dos pontos, foi calculada a média em cada píxel das 300 imagens capturadas. A leitura de profundidade dos cantos internos do xadrez foi obtida através da média ponderada dos píxeis adjacentes.

3.4 Modelação do Ambiente Vazio

Devido à localização e à orientação dos sensores, os mapas de profundidade por eles capturados irão conter na sua maioria pontos que fazem parte do chão. Desta forma, é importante realizar uma segmentação desses pontos, por duas razões:

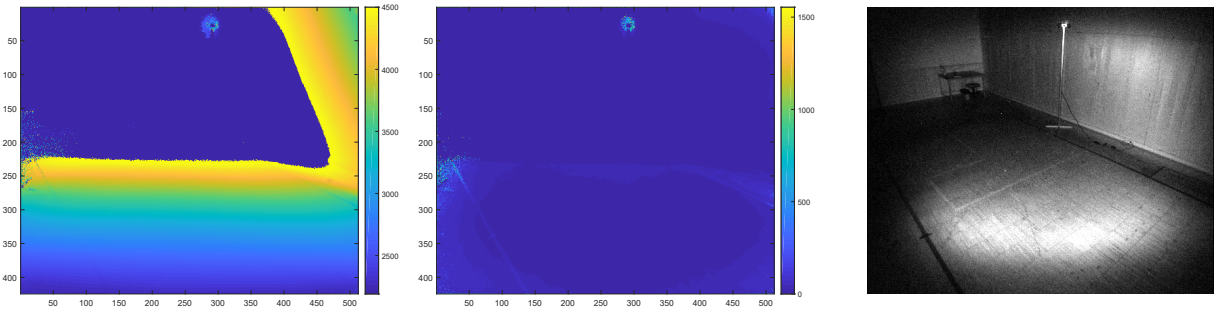
- a necessidade da redução da carga computacional relacionada com o processamento dos dados (menos pontos, menor carga computacional);
- a informação relativa ao plano do chão é prejudicial ao processamento dos dados, sendo novamente importante a sua remoção.

Esta análise[1] do plano do chão deve ser efetuada em cada uma das câmaras, sempre que haja uma nova instalação no cenário desejado.

A análise consiste no estudo da média, μ, e do desvio padrão, σ, de cada píxel ao longo de múltiplas imagens, como representado na Equação 3.1. São ignoradas leituras de profundidade inferiores a 0.5 m (que representam, por si só, uma leitura de profundidade inválida [54]). Uma ilustração deste procedimento pode ser encontrada nas Figuras 3.2a e 3.2b.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}, \text{ onde } \mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.1}$$

Na equação 3.1, N representa o número de imagens consideradas e x representa as medidas de profundidade do mesmo píxel ao longo de múltiplas imagens.



(a) Média das leituras de profundidade ao longo de 1000 capturas.

(b) Desvio padrão das leituras de profundidade ao longo de 1000 capturas.

(c) Imagem IR demonstrativa do FoV.

Figura 3.2: Análise realizada para a remoção do chão na projeção dos pontos no mundo (cenário vazio).

Assim, o método para a remoção dos pontos relativos à superfície do chão baseia-se nas características de uma distribuição normal para a qual 99.73% da informação presente se encontra entre $\mu - 3\sigma$ e $\mu + 3\sigma$. Tendo em conta a informação estatística obtida, é possível aplicar este conhecimento no SCC, para a deteção de *outliers*[2] no modelo do plano de fundo estudado. Logo, a subtração do plano de fundo é feita projetando apenas, para o SCM, os píxeis que apresentem uma leitura de profundidade inferior a $\mu - 3\sigma$ de acordo com o modelo construído.

[1] O pseudocódigo referente ao estudo do plano de fundo, encontra-se presente no Apêndice D.

[2] Estatisticamente, um *outlier* é um ponto observado que se encontra afastado das restantes observações.

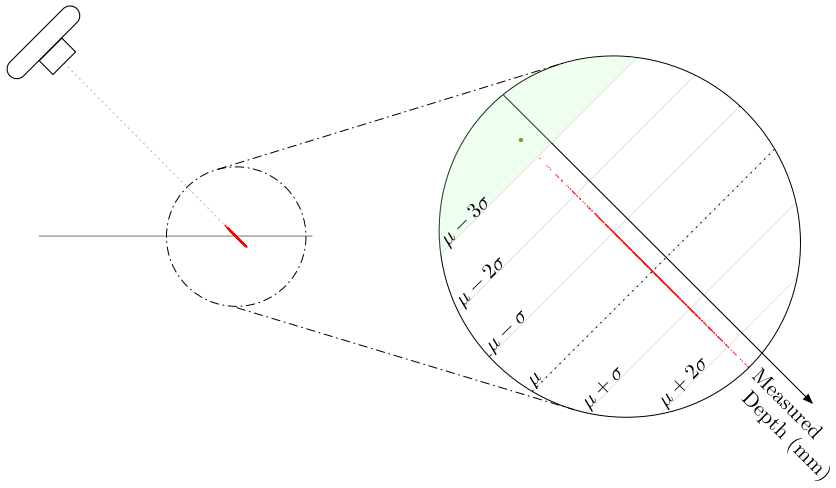


Figura 3.3: Exemplo da projeção de um píxel ao longo de múltiplas imagens, com o estudo da média e do desvio padrão associado (a região verde conterá apenas 0.13% da informação, usada para a deteção de *outliers*).

3.5 Criação da Nuvem de Pontos

As medições de profundidade capturas por uma câmara ToF são severamente influenciadas por ruído aleatório [55] (composto principalmente por reflexões imprevistas em materiais com índices de refletividade altos, e do fenómeno de mistura dos sinais IR enviados pelas várias câmaras), produzindo leituras de profundidade incorretas. Com o intuito de mitigar esse ruído, foi aplicado um filtro de mediana na imagem.

Obtidos os parâmetros intrínsecos e extrínsecos de cada câmara, e tendo sido feita a modelação do plano de fundo para cada uma das 4 câmaras, é feita de seguida a conversão das imagens de profundidade capturadas por cada câmara para uma PCD 3D no SCM. O processo[3] pode ser sumariado da seguinte forma:

1. Remoção da distorção presente na imagem (tanto a componente radial como a tangencial) da lente, utilizando os parâmetros intrínsecos previamente determinados;
2. Aplicação de um filtro de mediana usando uma vizinhança 3×3.
3. Transformação do Sistema de Coordenadas da Imagem (SCI) expresso por (x, y), para o SCC expresso por (x_c, y_c, z_c), recorrendo ao conhecimento do centro ótico, c_x e c_y, e da distância focal, f_x e f_y, obtidos no processo de calibração dos parâmetros intrínsecos:

$$x_c = (x - c_x) \cdot \frac{d}{f_x} \quad , \quad y_c = (y - c_y) \cdot \frac{d}{f_y} \quad , \quad z_c = d \tag{3.2}$$

[3] O pseudocódigo referente à projeção dos mapas de profundidade para o SCM, encontra-se presente no Apêndice E.

onde d representa a leitura de profundidade do píxel (x, y).

4. Transformação do SCC i, $i \in \{1, \ldots, 4\}$ para o SCM expresso em (X, Y, Z), usando a matriz rotacional, R_w^i, e de translação, t_w^i, obtidos no processo de calibração dos parâmetros extrínsecos:

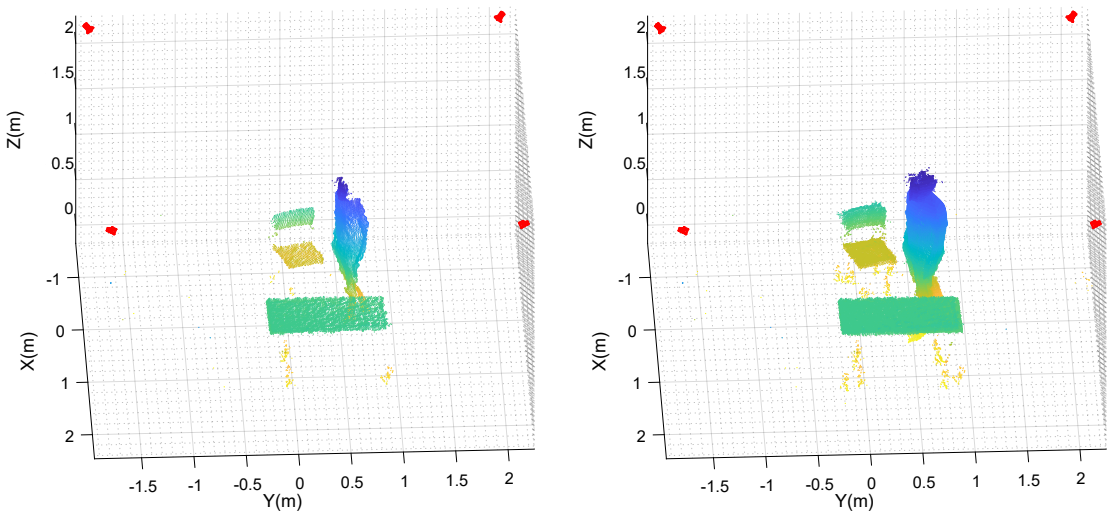
$$[X, Y, Z]_{w_i} = \left([x_c, y_c, z_c]_{c_i} - t_w^i\right) \times {R_w^i}^T, i \in \{1, \ldots, 4\} \ ^4 \tag{3.3}$$

   (a) A remoção dos pontos relativos ao plano de fundo é feita ainda no SCC, através das matrizes da média e do desvio padrão obtidas na fase de pré-processamento. Neste sentido, o píxel $p(y, x)$ apenas é projetado para o mundo se $p(y, x) < \mu_{(y,x)} - 3 \cdot \sigma_{(y,x)}$, i.e., apenas é projetado para o mundo caso a profundidade medida seja inferior ao limite imposto para a medição ser considerada como não pertencendo ao plano de fundo.

3.5.1 Fusão da Informação

Tal como referido na Secção 3.2, a informação adquirida de cada sensor é feita simultaneamente a uma taxa de captura de 30 FPS. Assim cada captura é composta por 4 imagens de profundidade e, consequentemente, 4 PCDs. Posteriormente, todas as PCDs são combinadas numa única representação, através da Equação 3.4.

$$[X, Y, Z]_M = \bigcup_{i=1}^{4} [X, Y, Z]_{w_i} \tag{3.4}$$



(a) Obtida a partir de um dos sensores. **(b)** Obtida pela união dos quatro sensores.

Figura 3.4: PCD no SCM.

[4] Como R é ortogonal, $R^{-1} = R^T$.

3.6 Procedimento de *Downsampling* Usando uma Grelha de *Voxels*

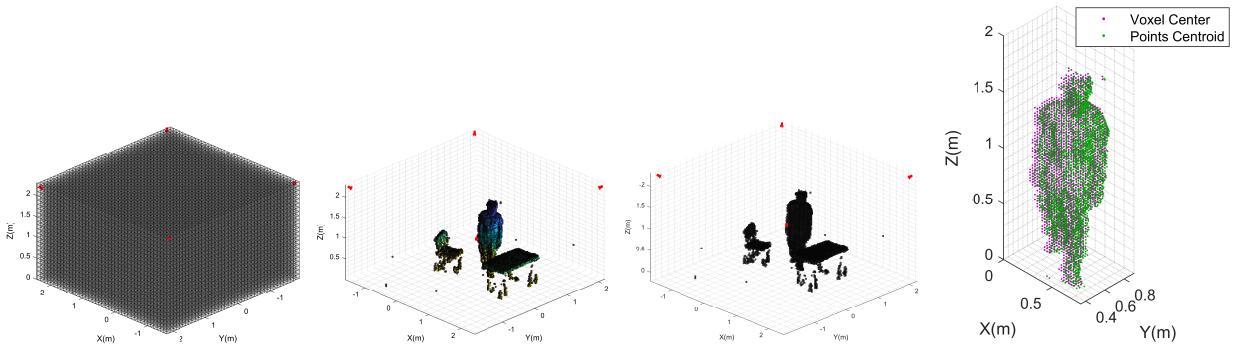
Os dados relativos à PCD podem conter muita informação redundante que trará carga computacional aos futuros módulos de processamento. Deste modo, optou-se por aplicar um processo de *downsampling* usando uma abordagem baseada numa grelha de *voxels* [56] para a redução da densidade de pontos.

O processo da voxelização discretiza o espaço em pequenos elementos em grelha, no qual cada grelha contém informação acerca do espaço que representa. Apesar dos mapas de grelhas requisitarem uma grande quantidade de memória para ambientes grandes dimensões, os cenários alvo são relativamente pequenos, *indoor*, e delimitados pelas posições das 4 câmaras (Figura 3.5a). O método opera fundamentalmente em duas etapas:

1. A PCD é dividida em *voxels*, recorrendo-se a métodos de indexação através do conhecimento das dimensões da grelha 3D construída e da dimensão de cada *voxel*;

2. Para cada *voxel* considerado ocupado, será gerado apenas um ponto, calculado através da média de todos os pontos internos; esta abordagem é computacionalmente mais exigente do que usar apenas o centroide do respetivo *voxel*, porém representa a superfície associada com mais precisão (Figura 3.5d).

Para os efeitos desta dissertação, a dimensão de cada *voxel*, V_s, optada para a discretização do mundo foi de 0.025 m (valor estipulado face aos requisitos impostos no projeto HTPDIR).



(a) Grelha de *voxels* geral do mapa. **(b)** PCD com respetiva grelha de *voxels*. **(c)** Grelha de *voxels* da PCD presente em **(b)**. **(d)** Comparação gráfica entre a grelha de *voxels* e o ponto real associado a cada *voxel* individual.

Figura 3.5: Exemplo da implementação de uma grelha voxelizada.

3.7 Segmentação

Obtida a representação 3D do cenário num dado momento de captura serão aplicadas técnicas de *clustering* de modo a segmentar a informação relevante para o processamento do cenário.

Face ao discutido no Estado da Arte, o método optado para o módulo de segmentação foi o DBSCAN. No contexto deste caso prático, os sub-conjuntos (*clusters*) irão representar objetos de interesse (pessoas ou objetos passivos, como mesas e cadeiras).

O algoritmo pode ser descrito pelos seguintes passos:

1. Escolha arbitrária de um ponto p não verificado:
   (a) Se o p já tem um *cluster* associado, voltar ao $Passo$ 1;
   (b) Cálculo da vizinhança (S) de p, usando o raio ϵ:
      i. Se p tiver menos de $minPts$ vizinhos (incluindo o ponto p), será identificado como ruído. Voltar ao $Passo$ 1.
   (c) Criar um novo *cluster* (c) e associar esse *cluster* ao ponto p;
   (d) Para cada ponto q em S:
      i. Se q está identificado como ruído, q passa a pertencer ao *cluster* c;
      ii. Se q é ruído ou já está associado a um *cluster*, voltar ao $Passo$ 1. d);
      iii. Associar o *cluster* c, ao ponto q e calcular a sua vizinhança N, usando o raio ϵ;
      iv. Se q tiver pelo menos $minPts$ vizinhos (incluindo o ponto q): $S = S \cup N$.

No caso deste trabalho, em particular, visto que os dados sofreram um *downsampling* baseado numa grelha voxelizada, considerou-se $\epsilon = \sqrt{2 \cdot V_s^2}$, significando que todos os *voxels* adjacentes ao query point p são tidos em conta.

Analisando os passos do algoritmo acima, verifica-se que todos os pontos do conjunto de dados, realizam uma pesquisa dos *Nearest Neighbors* - Vizinhos mais próximos (NN) num raio de ϵ (ora no passo 1b, ora em 1(d)iii), tornando-se uma tarefa computacionalmente pesada, com uma complexidade no tempo de $\mathcal{O}(n \cdot \log n)$, onde grande parte do tempo de execução é passado no algoritmo de pesquisa dos NN. Através da procura de novas alternativas, foi possível encontrar outro método mais eficiente de *clustering* também baseado no método DBSCAN, designado como *Radially Bounded Nearest Neighbor* (RBNN) [57]. A principal

vantagem do método RBNN é o fato de este não realizar uma pesquisa dos NN para cada instância. Tal particularidade do algoritmo demonstra ser exatamente a solução desejada para resolver o problema acima mencionado. O algoritmo pode ser descrito brevemente pelos seguintes passos:

1. Percorrer todos os pontos presentes na PCD:
   (a) Se o ponto atual já foi atribuído a um *cluster*, passar para o próximo ponto;
   (b) Para o ponto atual:
      i. Procurar todos os vizinhos do ponto atual dentro da distância ϵ;
      ii. Se algum desses vizinhos já estiver associado a um *cluster*, associar o ponto atual e todos os vizinhos que não tenham nenhum *cluster* associado a esse mesmo *cluster*;
      iii. Se o ponto atual foi associado a um *cluster* e se houver vizinhos que estejam associados a diferentes *clusters*, unir todos esses *clusters*.
2. Para todos os *clusters* criados:
   (a) Calcular a densidade de pontos presentes no *cluster* atual;
   (b) Se essa densidade for menor que $minPts$, remover o respetivo *cluster*.

Com a aplicação desta metodologia, o algoritmo de *clustering* passa a apresentar um tempo computacional mais eficiente através da associação dos NN a *clusters* (passo 1(b)ii) sem que estes, consequentemente, tenham que fazer uma pesquisa dos seus vizinhos (passo 1a). Deste modo, a complexidade computacional passa a ser aproximadamente $\mathcal{O}\left(\frac{n}{k_{average}} \log n\right)$, onde $k_{average}$ representa o número médio de NN em todas as pesquisas realizadas em ϵ. Torna-se evidente que, ϵ deve ser escolhido o mais elevado possível de tal forma que o algoritmo retorne a mesma segmentação, maximizando o número de vizinhos visitados por pesquisa. Contudo é importante ter em conta que, independentemente do valor atribuído a ϵ, em certos casos é inevitável que ocorram os chamados fenómenos de sub ou sobre-segmentação, cujo tópico será discutido mais à frente na Secção 3.8.4.

3.8 Módulo de Seguimento de Objetos

Antes de se proceder à aplicação de técnicas de seguimento é necessário determinar a localização de cada *cluster* no mundo. Para tal será construída uma *Bounding Box* (BB)

orientada, que irá identificar a presença do *cluster* no mundo e consequentemente o seu centroide que representará a posição do respetivo *cluster*.

3.8.1 Construção das *Bounding Boxes* dos Objetos

Para o cálculo da *Oriented Minimum Bounding Box* (OMBB), é necessário encontrar o *Minimum Area Rectangle* - Retângulo de Área Mínima (MAR) que contenha todos os pontos projetados para o plano XY do dado *cluster*. Para tal, considera-se que uma das arestas do *Convex Hull* (CH)[5] pertence a uma das arestas do MAR encontrado. Para o seu cálculo, foram considerados e implementados dois algoritmos: Gift Wrapping [58], pela sua simplicidade, e Graham Scan [59], pela sua eficiência computacional.

O processo para o cálculo da OMBB pode ser visto de seguida:

1. Calcular o CH dos pontos projetados para o plano XY;
2. Para cada aresta do CH:
   (a) Calcular a orientação da aresta;
   (b) Rodar os pontos do CH usando a orientação calculada, para consequentemente calcular a área do retângulo criado com os menores/maiores pontos de ambas as coordenadas;
   (c) Guardar a orientação que corresponda à mínima área encontrada.
3. Retornar o retângulo correspondente à mínima área encontrada; e as alturas do ponto mais baixo e mais alto de todo o conjunto de pontos.

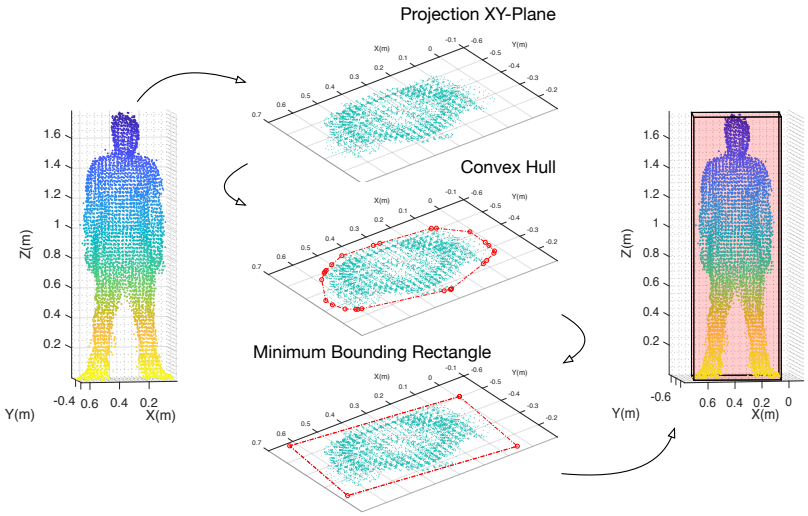


Figura 3.6: Representação gráfica do processo realizado para a construção da BB de um *cluster*.

[5]O *Convex Hull* de um conjunto de dados é o menor polígono convexo que contém todos os pontos do mesmo.

3.8.2 Filtro de Kalman

A partir da BB dos múltiplos objetos identificados na cena obtiveram-se também os respetivos centroides, sendo estes usados como observação para o processo de seguimento como referido anteriormente. Estando perante um sistema linear, um *Constant Acceleration Kalman Filter* (CAKF) [41] é inicializado sempre que um objecto aparece no cenário, sendo esta ferramenta usada para prever a futura localização de cada objeto e, assim, ajudar a associação de vários objetos físicos com as suas respetivas trajetórias.

Considerando o vetor de estados,

$$x_k = [\ x_k\ y_k\ z_k\ \dot{x_k}\ \dot{y_k}\ \dot{z_k}\ \ddot{x_k}\ \ddot{y_k}\ \ddot{z_k}\ ]^T \tag{3.5}$$

e sabendo que a identificação do objeto no mundo é feita através de um ponto, o centroide do objeto está localizado nas coordenadas $\{x,\ y,\ z\}$, enquanto que $\{\dot{x},\ \dot{y},\ \dot{z}\}$ e $\{\ddot{x},\ \ddot{y},\ \ddot{z}\}$ representam a velocidade e a aceleração do objeto para cada coordenada, respetivamente. O vetor de estados inicial, para cada *cluster* é definido como,

$$x_0 = [\ c_x\ c_y\ c_z\ 0\ 0\ 0\ 0\ 0\ -g]^T \tag{3.6}$$

onde, $\{c_x,\ c_y,\ c_z\}$ representa a posição do centroide da BB do novo objeto identificado; e g a aceleração gravítica, i.e., tem em conta a aceleração causada pela força da gravidade em z.

Pretendendo-se realizar o seguimento de um objeto, é necessário fazê-lo tendo presente a equação da cinemática,

$$x = x_0 + v_0 t + \frac{1}{2} a t^2 \tag{3.7}$$

desta forma, a matriz de transição de estados A torna-se,

$$A = \begin{bmatrix} 1 & 0 & 0 & d_t & 0 & 0 & \frac{1}{2}{d_t}^2 & 0 & 0 \\ 0 & 1 & 0 & 0 & d_t & 0 & 0 & \frac{1}{2}{d_t}^2 & 0 \\ 0 & 0 & 1 & 0 & 0 & d_t & 0 & 0 & \frac{1}{2}{d_t}^2 \\ 0 & 0 & 0 & 1 & 0 & 0 & d_t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & d_t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & d_t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

onde d_t representa o período da amostragem, inicializada neste caso pelo inverso da taxa de captura definida.

Como a medida é feita diretamente pela posição do centroide, a matriz de medição M, foi definida como

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.9}$$

A matriz de covariância de ruído do processo Q, corresponde consequentemente à incerteza esperada nas equações de transição de estados definidas como, por exemplo, a presença de forças externas (e.g. vento) que podem influenciar a movimentação normal de um objeto em queda livre. Tendo em conta que este sistema será indicado para ambientes *indoor*, a incerteza será relativamente baixa, logo Q foi definida como

$$Q = 0.1 \times I_3 \tag{3.10}$$

onde I_3, representa uma matriz identidade 3×3.

A matriz de covariância do erro do estado inicial, P_0, foi inicializada da seguinte forma

$$P_0 = \begin{bmatrix} 0.025 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.025 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.025 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \tag{3.11}$$

O erro associado ao estado inicial da posição foi definido como 0.025 m, equivalente a V_s, que representa a discretização considerada durante o processo de *downsampling* através da abordagem da grelha de *voxels*. Para a velocidade, considerou-se 5 m/s pois não é expectável num cenário de 4 $m \times 4$ m que objetos tenham uma velocidade superior a essa constante. Quanto ao erro associado com o vetor de estados da aceleração, assumiu-se um valor de 10 m/s^2 para compensar o fato de se ter considerado a aceleração gravítica, no qual, em objetos estáticos, a sua inicialização estará incorreta.

Para finalizar, procedeu-se a um processo iterativo para inicializar a matriz de covariância do ruído da medição R, através da análise dos vetores de estados previstos para objetos supostamente estáticos, como cadeiras e bancos, procurando estabilizar a sua velocidade para valores relativamente baixos, um segundo após a criação do filtro, obtendo-se consequentemente,

$$R = 0.125 \times I_3 \tag{3.12}$$

3.8.3 Processo de Atualização do Filtro de Kalman

Cada CAKF associado a um objeto físico presente na cena, pode encontrar-se em dois estados:

- Ativo, caso no momento de captura atual exista uma observação que esteja associada a esse mesmo objeto;
- Perdido, caso não se encontre nenhum objeto físico que represente esse mesmo objeto, não existindo assim uma observação para a processo de atualização do estado.

O processo para a atualização dos KFs encontra-se detalhado nos Apêndices F.1 e G.1. Havendo então a necessidade de atualizar os KFs associados a cada objeto, os KFs que se encontram no estado perdido serão atualizados antes dos que estão no estado ativo. Esta preferência prende-se com a necessidade de tentar voltar a estabelecer a associação entre um objeto na cena com o KF que lhe estava associado, quando ocorrem fenómenos de junção de *clusters* ou oclusões. Caso esta ocorrência não seja duradoura, o KF relativo a esse dado objeto terá a possibilidade de a ele se associar novamente, sendo atualizado com a observação da sua nova localização; caso a interação seja mais prolongada, e no final de um número predeterminado de capturas sucessivas, a tentativa de reassociação será infrutífera e o KF será removido. A procura desse objeto associado ao KF segue os dois critérios seguintes:

1. Verificação da distância euclidiana d entre a predição $\hat{x_t}$ obtida pelo KF e o centroide da BB do *cluster* obtido no processo de segmentação do momento atual t. Caso $d \leq d_{th}$ (onde d_{th} é um parâmetro de entrada a definir que representa a distância máxima de procura considerada):
    - Encontra-se possivelmente presente o objeto associado ao KF e é então feita a verificação do segundo critério;
    - Caso contrário, nenhuma observação está disponível dentro da região, logo a observação é feita através da predição retornada pelo KF.
2. Cada KF tem associado a si uma janela temporal w, de N amostras, na qual é armazenada a densidade de pontos ao longo do tempo. Sendo p_c a densidade de pontos do respetivo *cluster* em análise, o objeto físico associado a esse cluster corresponde ao do KF se, a condição 3.13 se confirmar:

$$\mu_w - 5 \cdot \sigma_w \leq p_c \leq \mu_w + 5 \cdot \sigma_w \tag{3.13}$$

onde μ_w e σ_w, representam a média e o desvio padrão da janela temporal w.

Caso os dois critérios acima mencionados se confirmem, o KF que anteriormente perdeu o seguimento ao seu respetivo objeto físico é agora atualizado com a observação obtida através do centroide da BB do *cluster* em questão e o seu estado passará para ativo.

Ainda relativamente aos KFs que se encontram perdidos, caso não seja encontrada uma observação válida durante N momentos de captura, procede-se à eliminação do mesmo.

Relativamente aos KFs ativos, o fluxo de processamento segue uma metodologia idêntica:

1. Para cada KF ativo i:

(a) Percorrer todos os *clusters* ainda não associados aos seus respetivos KFs:

i. Caso, $d \leq d_{th}$ e a condição 3.13 se verifique, o KF i é atualizado com a nova observação.

(b) Caso nenhum *cluster* cumpra ambos os requisitos, a observação do KF i será feita a partir da sua predição e o seu estado passará para perdido.

2. Para todo o *cluster* j que não foi associado a nenhum KF:

(a) Criar um KF cuja observação inicial é dada pela centroide da BB do *cluster* j.

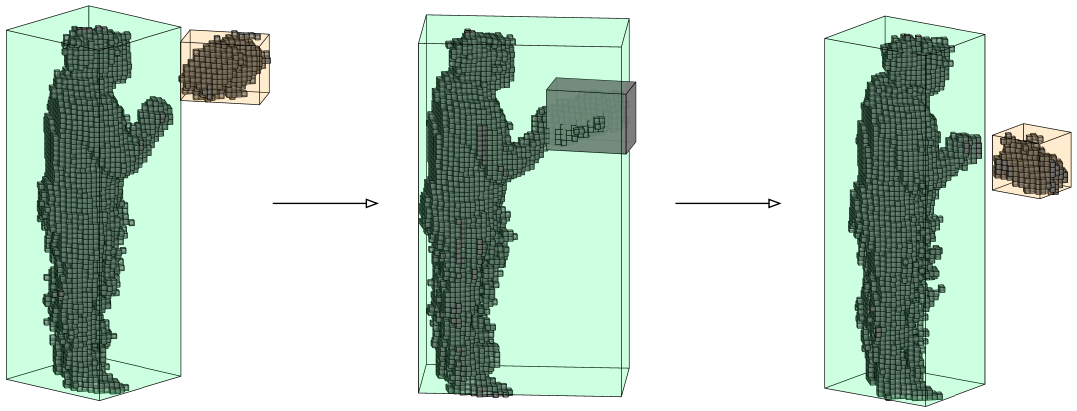
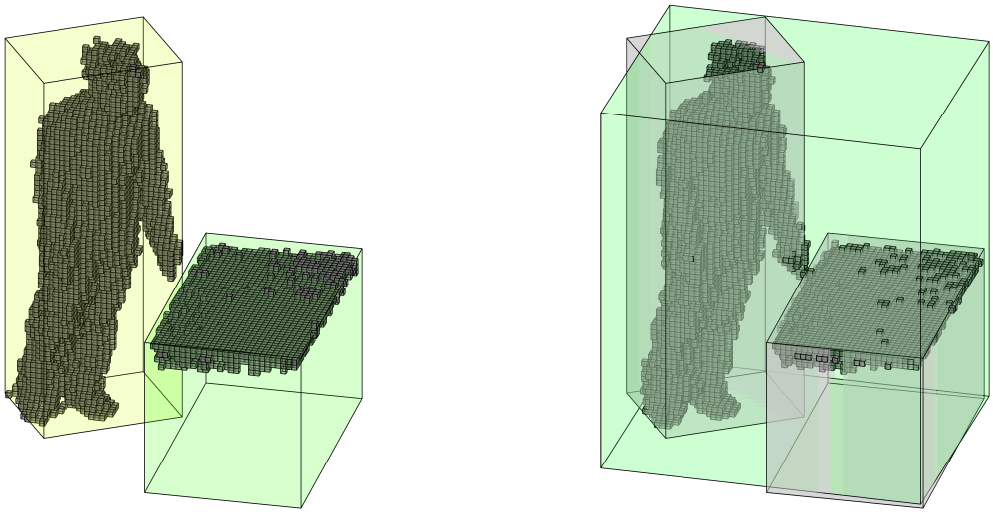


Figura 3.7: Exemplo da presença do processo de seguimento. A laranja encontra-se uma caixa que foi arremessada contra o utilizador (representada a verde). No momento presente na segunda imagem, verifica-se a interação entre a pessoa e a caixa, perdendo-se o seguimento da caixa (representado através da BB a cinza, representativa do KF perdido e que se encontra a prever o seu deslocamento). Após um toque subtil na caixa, no último momento de captura, houve a reassociação do cluster relativo à caixa ao KF que se encontrava perdido.

3.8.4 Sobre-segmentação

Como referido na Secção 3.7, é inevitável a ocorrência de sobre-segmentação em certos casos tais como o contacto entre dois objetos distintos, a interação de uma pessoa com um objeto, etc. A integração do módulo de seguimento de objetos no sistema permite detetar uma segmentação incorreta de forma direta, i.e., com o conhecimento de cada objeto a deteção de uma sobre-segmentação fará com que não haja uma leitura possível para o processo de atualização do estado do objeto, como demonstrado na Figura 3.8.

Fazendo a verificação da presença do número de KFs perdidos dentro do novo *cluster* detetado é possível verificar os respetivos objetos que se encontram em contacto. Para lidar com estas situações há dois tipos de histórico que foram tidos em conta:



(a) Momento imediatamente anterior à ocorrência do fenómeno de sobre-segmentação.

(b) Ocorrendo a sobre-segmentação, não há observação que corresponda à pessoa ou à mesa; assim, a atualização será feita através da predição do filtro (representado através das BB a cinza).

Figura 3.8: Exemplo da presença de sobre-segmentação.

- Para um KF associado a uma pessoa, será sempre mantida a informação acerca dos *voxels* correspondentes ao momento de captura anterior $t-1$;
- Para um KF associado a um objeto estático (verificado através da comparação do módulo da estimação do vetor de estados do KF, $||\dot{x}_k||$, com uma velocidade previamente definida v_{th}, representativa de um objeto estático), procede-se à reconstrução desse mesmo objeto através da união dos *voxels* ao longo dos vários momentos de captura em que o objeto se mantenha estático.

Posteriormente, para cada situação de sobre-segmentação o seguinte processamento é realizado:

1. Deteção dos objetos conhecidos k_i (associados a um KF perdido) que se encontram em situação de sobre-segmentação;
2. Procurar por objetos estáticos em k_i e realizar a interseção entre todos os *voxels* que se encontram em situação de sobre-segmentação (v_m) e os modelos construídos desses mesmos objetos e proceder à remoção dos *voxels* resultantes da interseção em v_m;
3. Caso esteja presente uma sobre-segmentação entre um objecto dinâmico e um *cluster* referente a um utilizador, será feita a fusão dos mesmos e mantida a informação referente apenas ao KF da pessoa;

4. Caso duas ou mais pessoas se encontrem numa situação de sobre-segmentação, é necessária uma segmentação que traduza já com um grau de exatidão suficiente as respetivas pessoas para o processo de seguimento da cabeça (detalhado posteriormente na Secção 3.9). Deste modo, um processamento adicional é tido em conta:

   (a) Para cada pessoa i que se encontra em situação de sobre-segmentação:

      i. Intersetar todos os *voxels* restantes de v_m com os *voxels* da pessoa i no momento de captura anterior $t-1$, M^i. Para os *voxels* resultantes da interseção, proceder à etiquetagem dos mesmos com o identificador da pessoa i. Remover de v_m os *voxels* obtidos através da interseção;

      ii. Calcular a matriz de covariância S_i dos *voxels* do *cluster* M_i.

   (b) Para cada *voxel* j de v_m:

      i. Calcular a distância de Mahalanobis, d_M, relativamente a cada um dos *clusters* M^i:

$$d_M^i(x) = \sqrt{(x-\mu_i)^T S_i^{-1} (x-\mu_i)} \tag{3.14}$$

      onde μ representa a média aritmética dos *voxels* do *cluster* M_i e x o *voxel* j.

      ii. Etiquetar o *voxel* j com o identificador do *cluster* referente à menor distância d_M.

3.9 Módulo de Seguimento da Cabeça

O módulo de seguimento da cabeça apresenta um papel muito importante neste sistema pois, no contexto prático deste trabalho, enquanto a localização da cabeça no mundo será extraída deste processo, a sua orientação irá ser obtida através do acelerómetro presente nos óculos VR a serem utilizados, criando assim uma perspetiva correta do mundo do ponto de vista do utilizador.

Para o módulo de seguimento de objetos, a observação é feita usando o centroide da BB; contudo, para o seguimento da cabeça, o centro de massa dos *voxels* do *cluster* associado à pessoa terá uma melhor correlação com o movimento da cabeça. Contudo, estamos perante uma correlação positiva fraca, i.e., uma translação de (x, y, z) no centro de massa não indica uma translação de igual valor para a cabeça, formando assim um sistema altamente não-linear. Assim, tendo em vista a inviabilidade da aplicação de outro KF para o seguimento da cabeça, foi implementado um PF pela sua flexibilidade (dado permitir lidar com tais sistemas através da discretização do problema em diversas partículas). Cada uma destas partículas define um possível estado do modelo e através da aglomeração de um número suficiente de partículas é possível obter uma predição eficaz do estado do sistema (com a desvantagem de ser significativamente mais pesado computacionalmente).

3.9.1 Filtro de Partículas

O módulo desenvolvido para o seguimento da cabeça recorrendo a um PF é apresentado na Figura 3.9.

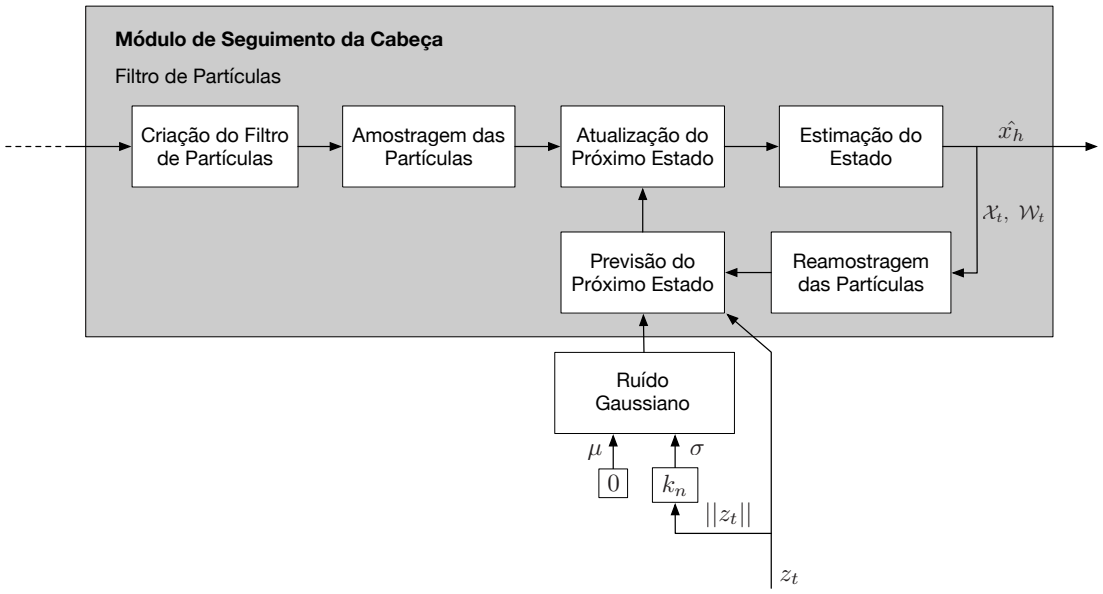


Figura 3.9: Módulo desenvolvido para o seguimento de cabeça, através da aplicação de um PF.

Amostragem de Partículas

A amostragem inicial do conjunto de partículas tem em consideração que, na fase de inicialização do PF, a pessoa se encontra de cabeça erguida. Caso não fosse feita esta consideração, a amostragem seria realizada de forma aleatória dentro da respetiva BB da pessoa, o que exigiria um maior número de partículas para cobrir todo esse volume de forma densa, levando a uma maior carga computacional e à possibilidade de gerar falsos positivos. Com esta consideração, a área onde será amostrada aleatoriamente o conjunto de partículas, $\mathcal{X}_0$, é então limitada pela BB e por uma altura, $z \in [h_{max} - h_z; h_{max}]$, sendo h_{max} a altura da BB e h_z uma constante que representa a altura da cabeça ($\geq 0.30\ m$, pela análise feita dos diversos conjuntos de dados extraídos e do estudo realizado em [60]). Todas as partículas são inicializadas com o mesmo peso, $w_0^{[m]} = \frac{1}{M}$, onde M representa o números de partículas do conjunto de partículas $\mathcal{X}$.

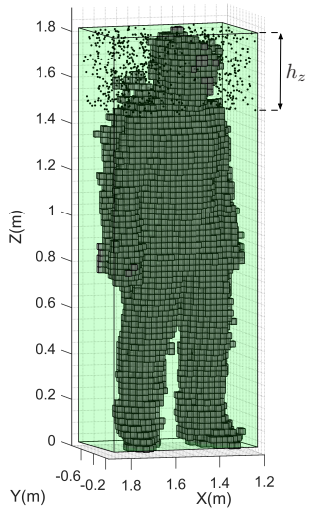


Figura 3.10: Amostragem inicial das partículas (cada ponto preto na região h_z representa uma única partícula).

Atualização do Próximo Estado

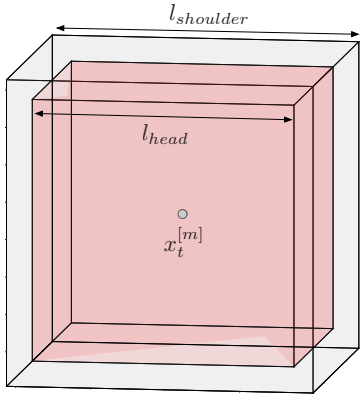


Figura 3.11: Apresentação gráfica das regiões B_{head} e $B_{shoulder}$.

Para a atualização dos pesos $\mathcal{W}_t$ é feita uma análise sequencial em torno de cada partícula individual. Essa análise é dividida essencialmente em 2 regiões centradas no estado da partícula m:

- um cubo B_{head} de dimensões l_{head}, que teoricamente procura analisar a presença da cabeça (apresentado a vermelho na Figura 3.11).
- uma região $B_{shoulder}$, que contorna o cubo B_{head} de espessura $l_{shoulder} - l_{head}$, onde se analisa a presença dos ombros (apresentado em cinza na Figura 3.11).

Para a discretização atual da grelha de *voxels*, os valores usados para l_{head} e $l_{shoulder}$ foram $0.35\ m$ e $0.41\ m$, respetivamente, tendo em conta o valor empírico seleccionado para a altura da cabeça, como previamente mencionado.

Cada componente da análise da informação presente em cada uma das regiões atua como uma contribuição para a atualização dos pesos $\mathcal{W}_t$. Será detalhado de seguida o processo para obtenção dos mesmos, inspirado nas funções de transferência presentes na lógica difusa:

Para o cubo B_{head} :

1. **Verificação da densidade de pontos**

   A primeira análise corresponde à verificação da densidade de pontos presente em B_{head}, usando para tal uma função sigmóide dada pela Equação 3.15) para o cálculo do peso, $f_{head}^{[1]}$. Aqui, $f_{head}^{[1]}$ é relativo à densidade de pontos que representa a presença da cabeça:

$$f_{head}^{[1]}(x, a, c) = \frac{1}{1 + e^{-a(x-c)}} \tag{3.15}$$

   onde x, representa o número de pontos presentes na região B_{head} da respetiva partícula $x_t^{[m]}$ e (a, c) são parâmetros de entrada a definir.

   Através da análise dos pontos pertencentes à cabeça ao longo de diversos conjuntos de dados, considerando a obstrução do FoV de duas câmaras, verificou-se uma densidade de pontos na região da cabeça de aproximadamente 450 pontos. Assim, foram atribuídos aos parâmetros de entrada da função a e c os valores .1 e 400, respetivamente (Figura 3.12).

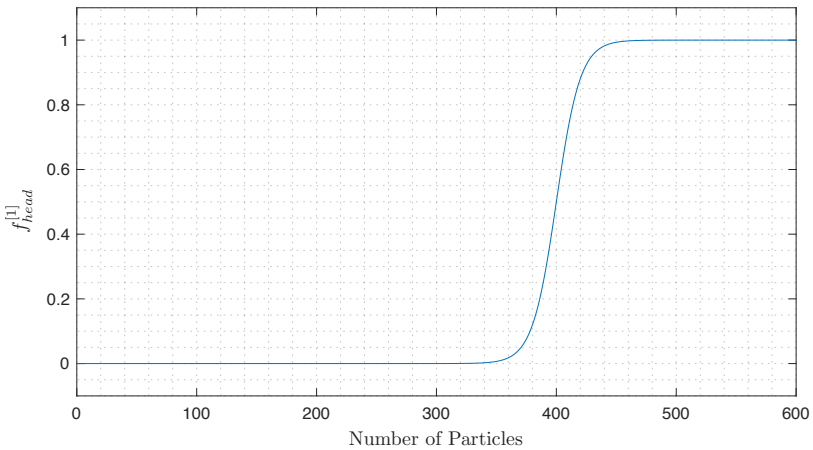
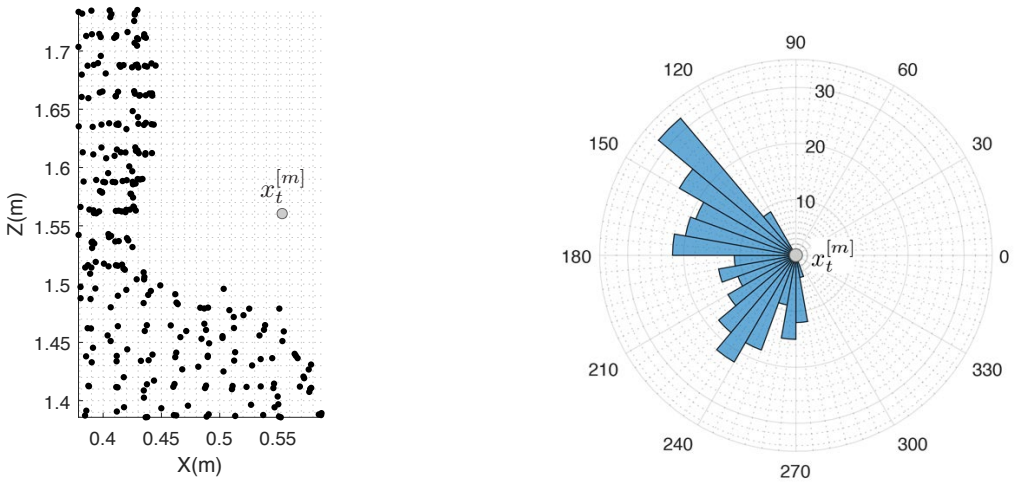


Figura 3.12: Função sigmóide dada pela Equação 3.15, com $a = 0.1$ e $c = 400$.

2. **Presença de pontos em torno de $x_t^{[m]}$**

Neste passo é feita uma análise da presença de pontos em torno de $x_t^{[m]}$. Para tal, é criado um histograma em coordenadas polares através da separação dos ângulos criados entre todos os pontos presentes na região B_{head} e $x_t^{[m]}$, em N regiões radiais igualmente espaçadas, para as perspetivas XZ e YZ.



(a) Grelha de pontos.

(b) Histograma em coordenadas polares, com $N = 36$.

Figura 3.13: Exemplo do histograma polar obtido para uma dada grelha de pontos presente na região B_{head} da partícula $x_t^{[m]}$.

Sendo n_{xz} e n_{yz} os números de *bins* ocupados dos histogramas polares para as perspetivas XZ e YZ, respetivamente, e tendo em conta que um ponto representativo do centro da cabeça se encontra naturalmente rodeado de pontos, o peso deste passo, $f_{head}^{[2]}$, é retornado novamente através da aplicação da Equação 3.15, com $a = 0.1$ e $c = 270$, onde se teve em conta uma possível obstrução de uma das faces da cara.

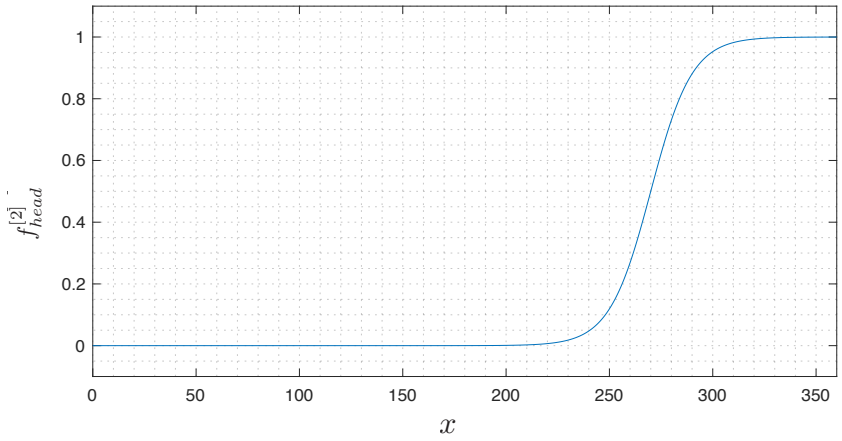


Figura 3.14: Função sigmóide dada pela Equação 3.15, com $a = 0.1$, $c = 270$ e $x = \min(n_{xz}, n_{yz}) \cdot \frac{N}{360}$.

Para a região $B_{shoulder}$:

1. **Perfil existente entre a cabeça e os ombros**

Processada a região relativa à cabeça, será analisada a presença dos ombros na região $B_{shoulder}$. A chave para a deteção da cabeça advém, para esta implementação, da relação existente entre a posição do centro da cabeça e os ombros (Figura 3.16a), que cria aproximadamente um ângulo de 100º entre si. Salienta-se o facto desta relação ter sido determinada através da verificação da mesma ao longo de vários testes experimentais realizados com várias pessoas. De forma a detetar este perfil, é determinado novamente um histograma polar, centrado na partícula $x_t^{[m]}$ face aos pontos pertencentes à região $B_{shoulder}$, obtendo posteriormente o número de *bins* consecutivamente ocupados para a perspetiva XZ, n_{xz} e para a perspetiva YZ, n_{yz}. Consecutivamente, o peso para cada perspetiva é retornado através de uma curva gaussiana (Figura 3.15) dada pela Equação 3.16,

$$f(x, \sigma, c) = e^{\frac{-(x-c)^2}{2\sigma^2}} \tag{3.16}$$

onde $\sigma = 30$ e $c = 100$. O parâmetro c define a relação supremencionada entre o centro da cabeça e os ombros e σ define o ângulo através da qual a região $c \pm \sigma$ representa um peso igual ou superior a 0.67.

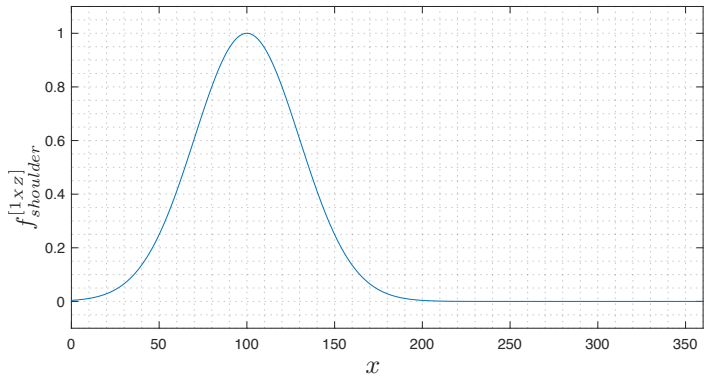
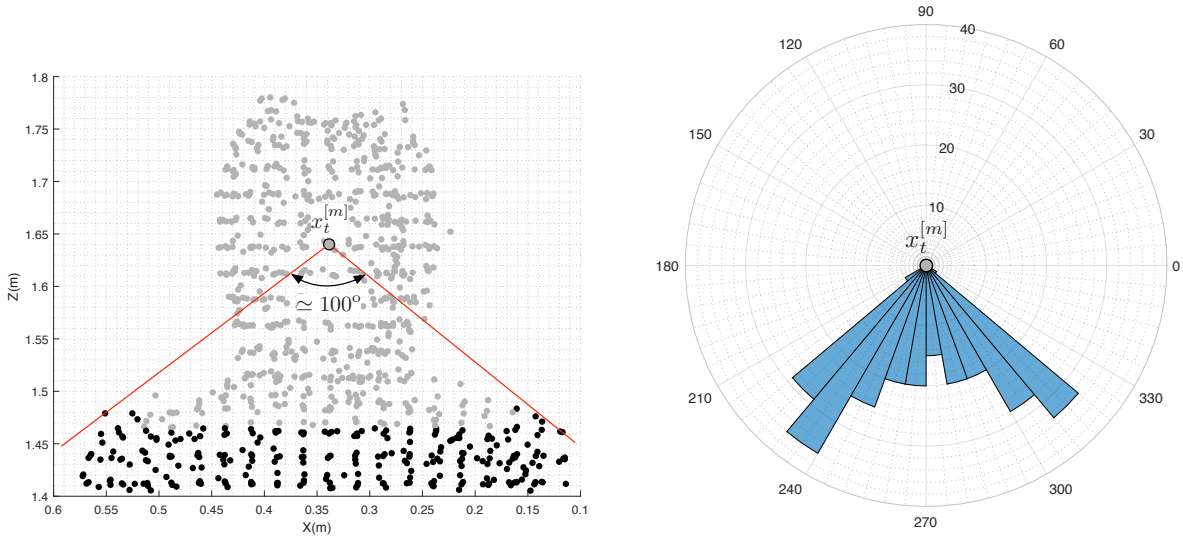


Figura 3.15: Curva gaussiana dada pela Equação (3.16), com $\sigma = 30$ e $c = 100$ e $x = n_{xz} \cdot \frac{N}{360}$, para a perspetiva XZ.

O peso final para a presente análise, $f_{shoulder}^{[1]}$, é atribuído pelo maior peso obtido nas duas perspetivas, $f_{shoulder}^{[1_{XZ}]}$ e $f_{shoulder}^{[1_{YZ}]}$, i.e., $f_{shoulder}^{[1]} = max\left(f_{shoulder}^{[1_{XZ}]}, f_{shoulder}^{[1_{YZ}]}\right)$.



(a) Grelha de pontos presente em ambas as regiões; em destaque, encontram-se os pontos presentes na região $B_{shoulder}$.

(b) Histograma polar obtido para os pontos em $B_{shoulder}$.

Figura 3.16: Análise dos pontos presentes na região $B_{shoulder}$, através da construção de um histograma polar, para uma dada partícula $x_t^{[m]}$.

Obtidas as 3 contribuições acima detalhadas, serão agregados todos os pesos obtidos. A atribuição do peso $w_t^{[m]}$ de cada partícula $x_t^{[m]}$ é realizada por uma função aplicada extensivamente no domínio da lógica difusa, dada pela Equação 3.17,

$$w_t^{[m]}(x,a,b) = \begin{cases} 0 & , \quad x \leq a \\ 2\left(\dfrac{x-a}{b-a}\right)^2 & , \quad a \leq x \leq \dfrac{a+b}{2} \\ 1-2\left(\dfrac{x-b}{b-a}\right)^2 & , \quad \dfrac{a+b}{2} \leq x \leq b \\ 1 & , \quad x \geq b \end{cases} \tag{3.17}$$

Através de um processo iterativo foi procurado verificar o comportamento do PF em situações propensas à presença de falsos positivos. Através deste estudo, verificou-se que uma partícula apenas deve contribuir para a estimação do estado se a soma das três contribuições acima calculadas for superior a 2.5. Desta forma, foram atribuídos aos parâmetros a e b, os valores 2.5 e 3, respetivamente.

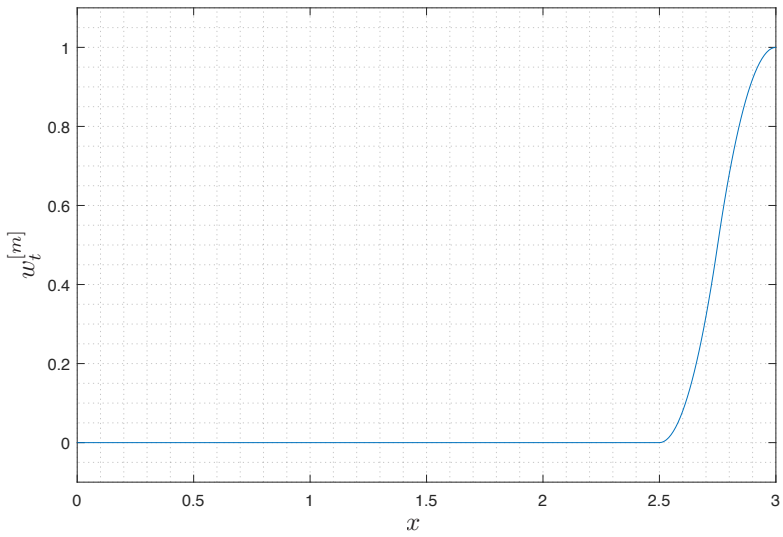


Figura 3.17: Curva dada pela Equação (3.17), com $a = 2.5$, $b = 3$ e $x = f_{shoulder}^{[1]} + \sum_{i=1}^{2} f_{head}^{[i]}$.

Estimação do Estado

O conjunto de todas as partículas $\mathcal{X}_t$ é usado para a estimação do estado $\hat{x_h}$ no instante de tempo t através do cálculo da média ponderada entre os estados das partículas $\mathcal{X}_t$ e os respetivos pesos normalizados $\mathcal{W}_t$,

$$\hat{x_h} = \sum_{m=1}^{M} x_t^{[m]} \cdot w_t^{[m]} \tag{3.18}$$

onde M representa o número de partículas do conjunto $\mathcal{X}_t$.

Reamostragem das Partículas

O processo de reamostragem procura providenciar áreas de confiança para a predição através da redução da variância do estimador. Controlar essa variação, ou erro, do PF é essencial para qualquer implementação prática. A estratégia usada é conhecida por *Low Variance Sampling* [61], onde a seleção envolve um processo sequencial estocástico. O algoritmo calcula apenas um número aleatório, $r \in [0; M^{-1}[$, onde M é um número de amostras a serem escolhidas, selecionando amostras de acordo com este número, mas ainda com uma probabilidade proporcional ao peso da amostra. Este processo é ilustrado na Figura 3.18, onde a partícula referente a r é selecionada tal como as restantes partículas obtidas pela adição repetitiva da constante M^{-1} a r, i.e, as partículas correspondentes a $u = r + (m-1) \cdot M^{-1}$, onde $m = 1, \ldots, M$.

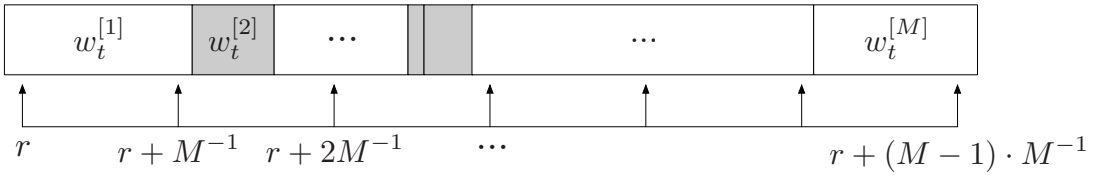


Figura 3.18: Princípio do procedimento de reamostragem pelo método *Low Variance Sampling* [61].

As vantagens desta abordagem são as seguintes:

- eficiência computacional, apresentando uma complexidade de $\mathcal{O}(M)$: o tempo computacional é essencial ao usar um PF e uma implementação eficiente do processo de reamostragem tem uma grande importância numa implementação prática.
- o método cobre o espaço de amostragem de uma forma mais sistemática comparativamente a *Independent Random Samplers* (*Stratified*, *Variable Probability*) [62].
- se todas as amostras tiverem os mesmos fatores importância, o conjunto de amostras resultante $\bar{\mathcal{X}}_t$ equivale a $\mathcal{X}_t$, significando que nenhuma amostra é perdida.

Predição

Sendo C_t o centro de massa do conjunto de *voxels* do *cluster* associado à pessoa no momento de captura atual, a medição z_t usada para o procedimento da predição é dada por $(C_t - C_{t-1})$. A predição para cada partícula individual é então dada pela Equação 3.19,

$$x_t^{[m]} = x_{t-1}^{[m]} + \left[z_t + \mathcal{N}\left(\mu, \sigma^2\right)\right] \quad \forall \quad m \in [1, ..., M] \tag{3.19}$$

A medição z_t tem a si associada um ruído gaussiano dado pela distribuição normal $\mathcal{N}(\mu, \sigma^2)$, $\mu = 0$ e $\sigma^2 = 0.05 + k_n \cdot ||z_t||$. Um ruído constante de variância 0.05 m (valor determinado empiricamente) é considerado e k_n representa uma constante de proporcionalidade relativa ao módulo da medição. Através de uma nova análise iterativa do comportamento do filtro, k_n foi definido como 0.5.

4 Avaliação Experimental dos Métodos Propostos

O presente Capítulo é dedicado à avaliação do método proposto. Para avaliar o funcionamento do sistema, foi desenvolvido um banco de testes em que cada módulo do *pipeline* é avaliado, sendo apresentados os resultados referentes a cada um destes testes modulares.

Os testes foram realizados utilizando MATLAB® R2018a instalado num computador com 16 GB de RAM de 3200 MHz, Intel Core i7-8700k 3.7 Ghz no Windows 10 de 64 bits. É importante ter em conta que se trata de uma programação em MATLAB® , e que apesar de se recorrer à vetorização[1] ao longo do projeto, uma conversão do código para uma plataforma em C++ poderia trazer uma melhoria no desempenho entre 1,000% a 100,000% [63] tornando-o apto para uma aplicação em tempo real.

4.1 Calibração da câmara

Os parâmetros intrínsecos estimados durante o processo de calibração para o sensor IR de uma das câmaras Kinect presentes no sistema são apresentados na Tabela 4.1.

Tabela 4.1: Parâmetros intrínsecos obtidos através do procedimento de calibração, com os respetivos desvios padrões.

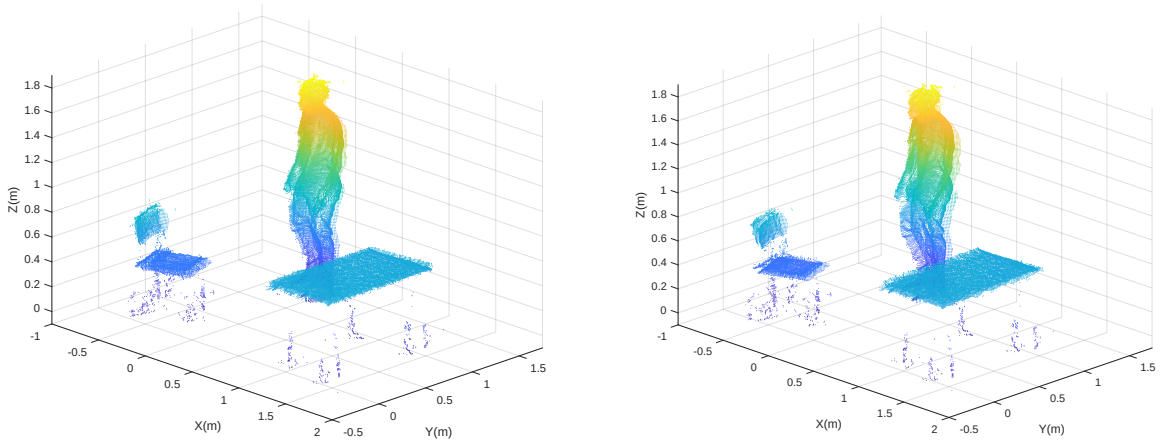
			Valor (píxeis)	Desvio padrão
Comprimento focal x			361.6891	0.2083
Comprimento focal y			361.1894	0.2014
Ponto principal x			257.2890	0.1373
Ponto principal y			208.5984	0.1689
Parâmetros de distorção	Radial	K_1	0.1084	0.0013
		K_2	-0.3179	0.0042
		K_3	0.1395	0.0048
	Tangencial	P_1	7.4369×10^{-5}	1.2478×10^{-4}
		P_2	0.0016	8.1979×10^{-5}

[1]Processo de reestruturação de código à base de ciclos e orientado a escalares de modo a beneficiar do melhor desempenho do MATLAB® em operações matriciais e vetoriais.

Os resultados apresentados na Tabela 4.1 são os esperados, e comparáveis aos demonstrados em [11, 64].

Para testar a qualidade da calibração, foram realizadas várias capturas para avaliar o método proposto. O primeiro conjunto de dados extraído foi utilizado para avaliar a qualidade da calibração em termos dos parâmetros extrínsecos.

Através da análise da Figura 4.1 é possível verificar que o processo suplementar da aplicação do método ICP não resultou numa melhoria no processo de calibração; apesar de haver uma diminuição da diferença entre a reprojecção dos pontos do xadrez da imagem e os pontos reais do xadrez no mundo. O mapeamento para o SCM de um cenário cujos dados de interesse não se encontram no centro do FoV dos sensores é gravemente afetado por este processo suplementar, como pode ser verificável na Figura 4.1b. Este desalinhamento deve-se às incertezas inerentes ao mapa de profundidade retornado pelos sensores que, para além do erro associado à medida de profundidade, as mesmas apresentam um maior desvio padrão (e consequentemente, uma menor precisão) nos cantos dessas imagens (cujo efeito pode ser visto na Figura 3.2b).

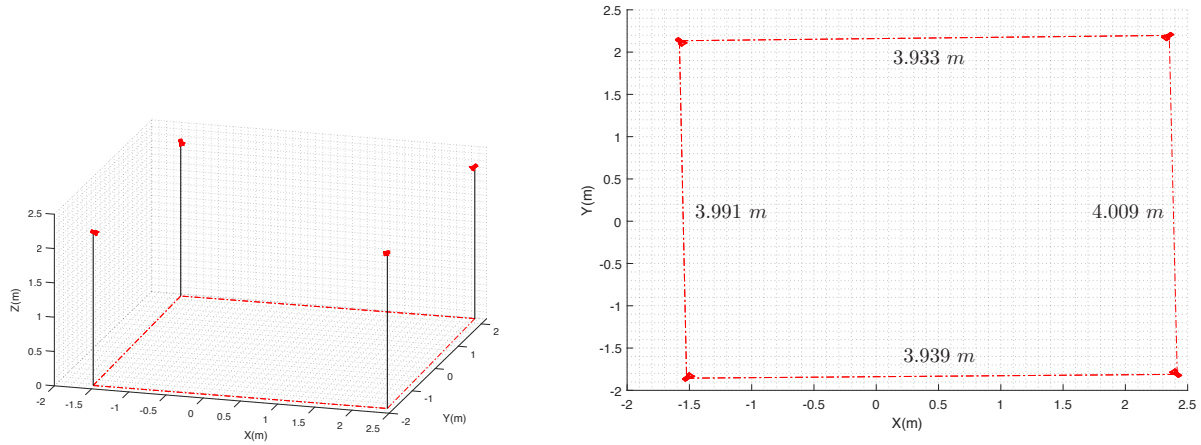


(a) Sem a aplicação do método ICP.

(b) Com a aplicação do método ICP.

Figura 4.1: Mapeamento do mapa de profundidade para o SCM.

A Figura 4.2b apresenta a projeção da localização das câmaras resultante do processo de calibração. Olhando para o mapeamento da posição das câmaras, é possível verificar que as mesmas se encontram localizadas como esperado pois, tendo em conta que a disposição espacial real dos sensores imposta pelos requisitos do projeto HTPDIR tem o intuito de criar um cenário quadrangular de 4 $m \times 4$ m, a calibração dos parâmetros extrínsecos retornou uma configuração muito próxima à real, apresentando uma área aproximada de $4\,m \times 3.936\,m$ como demonstrado na Figura 4.2b.



(a) Representação 3D da localização das câmaras.

(b) Análise das distâncias entre câmaras.

Figura 4.2: Projeção da localização das câmaras para o SCM.

4.2 Pré-processamento e filtragem

De forma a avaliar o desempenho do sistema proposto, foram capturados vários conjuntos de dados:

A [2] Cenário composto por uma cadeira e uma mesa. Um utilizador entra no cenário, move-se e acaba por sair do mesmo;

B [3] Dois utilizadores entram no cenário, cumprimentado-se três vezes, voltando a sair do mesmo;

C [4] Dois utilizadores encontram-se inicialmente no cenário, sendo que um deles tem em sua posse uma caixa. Ao longo da captura, estes arremessam repetitivamente a caixa entre si;

D [5] Uma parede de 0.8 m de altura em forma de L encontra-se presente no meio do cenário. Posteriormente, duas caixas de dimensões diferentes são lançadas contra essa parede;

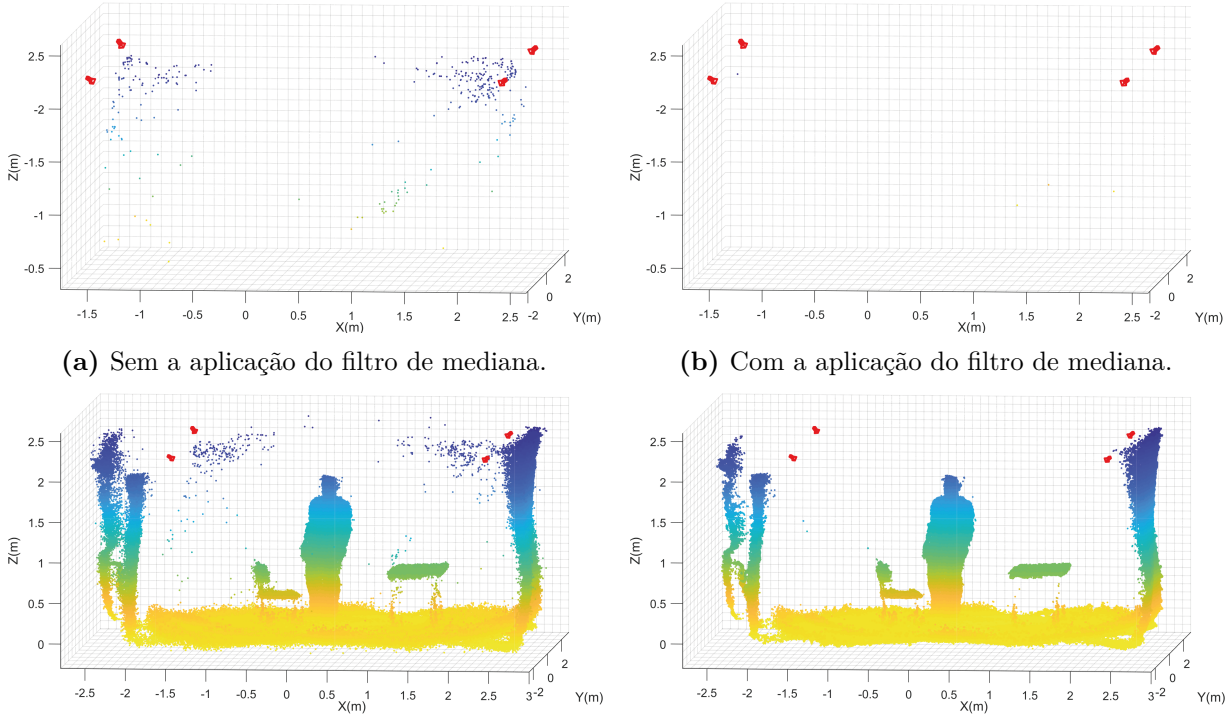
E [6] Um utilizador encontra-se presente no cenário no qual se vai deslocando repentinamente de um lado para o outro, bem como adoptando posições horizontais (p. ex. flexão de braços) ;

[2] `https://goo.gl/TTpssR`
[3] `https://goo.gl/hncP5G`
[4] `https://goo.gl/4Kks9k`
[5] `https://goo.gl/aZffaV`
[6] `https://goo.gl/RbV9S1`

F [7] Dois utilizadores entram num cenário vazio e deslocam-se no mesmo aleatoriamente sem qualquer interação entre eles;

G [8] Um utilizador encontra-se presente no centro do cenário, sem qualquer objeto presente no mesmo. Este realiza movimentações suaves do tronco e da cabeça sem alterar a sua posição no espaço.

Foi efetuada uma avaliação qualitativa, tanto do resultado da aplicação de um filtro de mediana no SCI, como da remoção do pontos relativos ao plano de fundo do cenário. Em primeiro lugar, foram projetados os mapas de profundidade das 4 câmaras para o mundo de um cenário vazio (como é demonstrado anteriormente no Secção 3.5), extraíndo os primeiros 0.3 m de altura, afim de evitar as leituras referentes ao chão. Posteriormente, foi testado também o comportamento deste módulo para um dado momento de captura do conjunto de dados A.



(a) Sem a aplicação do filtro de mediana.

(b) Com a aplicação do filtro de mediana.

(c) Sem a aplicação do filtro de mediana.

(d) Com a aplicação do filtro de mediana.

Figura 4.3: Exemplo comparativo da aplicação do filtro de mediana no SCI. 4.3a e 4.3b referem-se ao cenário vazio; 4.3c e 4.3d a um momento de captura extraído do conjunto de dados A.

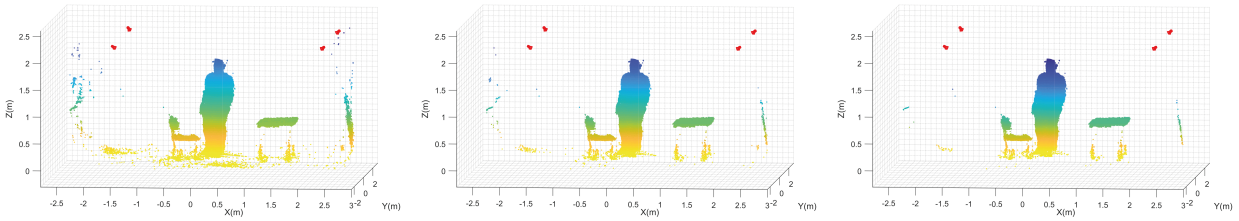
Os resultados deste teste são apresentados na Figura 4.3. É possível verificar uma diminuição abrupta do ruído existente próximo das câmaras com a aplicação do filtro de

[7] `https://goo.gl/VcMAqs`
[8] `https://goo.gl/GBnb3Q`

mediana. Este demonstra também ser eficaz na remoção dos *flying pixels*[9] referidos por Sarbolandi et al. [65] que ocorrem especialmente nas bordas dos objetos.

Foi testado ainda a influência da filtragem num conjunto de dados com 1000 momentos de captura referentes a um cenário vazio, e foi verificado que eram mantidos 497.83 ± 86.15 pontos de ruído sem a aplicação do filtro, tendo este número sido reduzido para 23.80 ± 19.08 com a aplicação do filtro, tendo-se alcançado uma redução média de 95.22% do ruído.

Relativamente à remoção do plano de fundo, esta depende de uma constante de proporcionalidade k_σ, relativa ao desvio padrão estudado do modelo. Assim, foi feito o estudo da regra empírica[10] através da variação da constante k_σ.



(a) $\sigma = 3$. **(b)** $\sigma = 5$. **(c)** $\sigma = 7$.

Figura 4.4: Análise da regra empírica para a distribuição normal obtida para o plano de fundo.

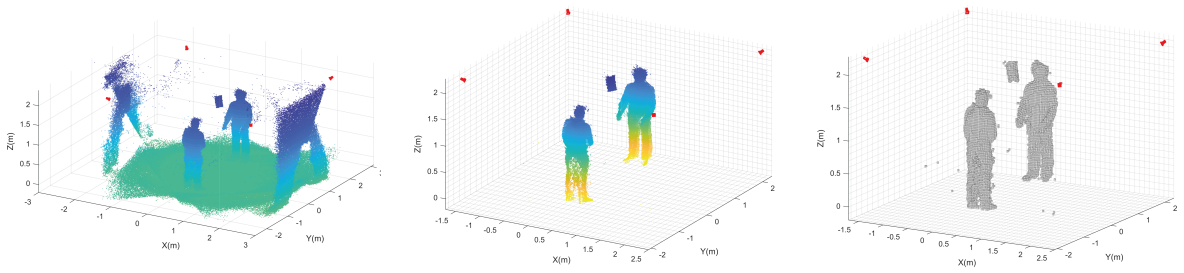
A Figura 4.4 mostra o resultado desta avaliação. Olhando para a Figura 4.4a verifica-se que apesar de teoricamente $k_\sigma = 3$ levar a uma remoção de 99.73% dos pontos referentes ao plano de fundo, ainda é evidente a presença de muitos pontos que fazem parte do plano do chão. Assim, definindo $k_\sigma = 7$, constata-se pela Figura 4.4c uma remoção mais eficaz dos pontos relativos ao chão, mantendo-se ainda presentes alguns pontos fora da área quadrangular criada pelas 4 câmaras que serão filtrados na passagem para a grelha de *voxels*. É importante também salientar que o aumento de k_σ poderá levar a uma eliminação de pontos não pertences ao chão mas que se encontram próximos a este. A grelha de *voxels* resultante DA remoção de pontos do plano do chão apresentada na Figura 4.5b é apresentada na Figura 4.5c.

4.3 Segmentação

Para avaliar a qualidade do processo de segmentação, foram selecionados quatro dos conjuntos de dados de teste acima descritos (A, B, C e D). Os resultados qualitativos obtidos

[9] Píxeis resultantes de um sinal sobreposto causado pela reflexão da luz em diferentes profundidades levando a valores de distância incorretos.

[10] Abreviação usada para lembrar a percentagem de valores que se encontram dentro de uma faixa em torno da média em uma distribuição normal com um desvio padrão específico.



(a) Mapeamento do mapa de profundidade sem qualquer pré-processamento realizado.

(b) Mapeamento do mapa de profundidade com a remoção do plano de fundo e a aplicação do filtro de mediana.

(c) Construção da grelha de *voxels* para a PCD em **(b)**.

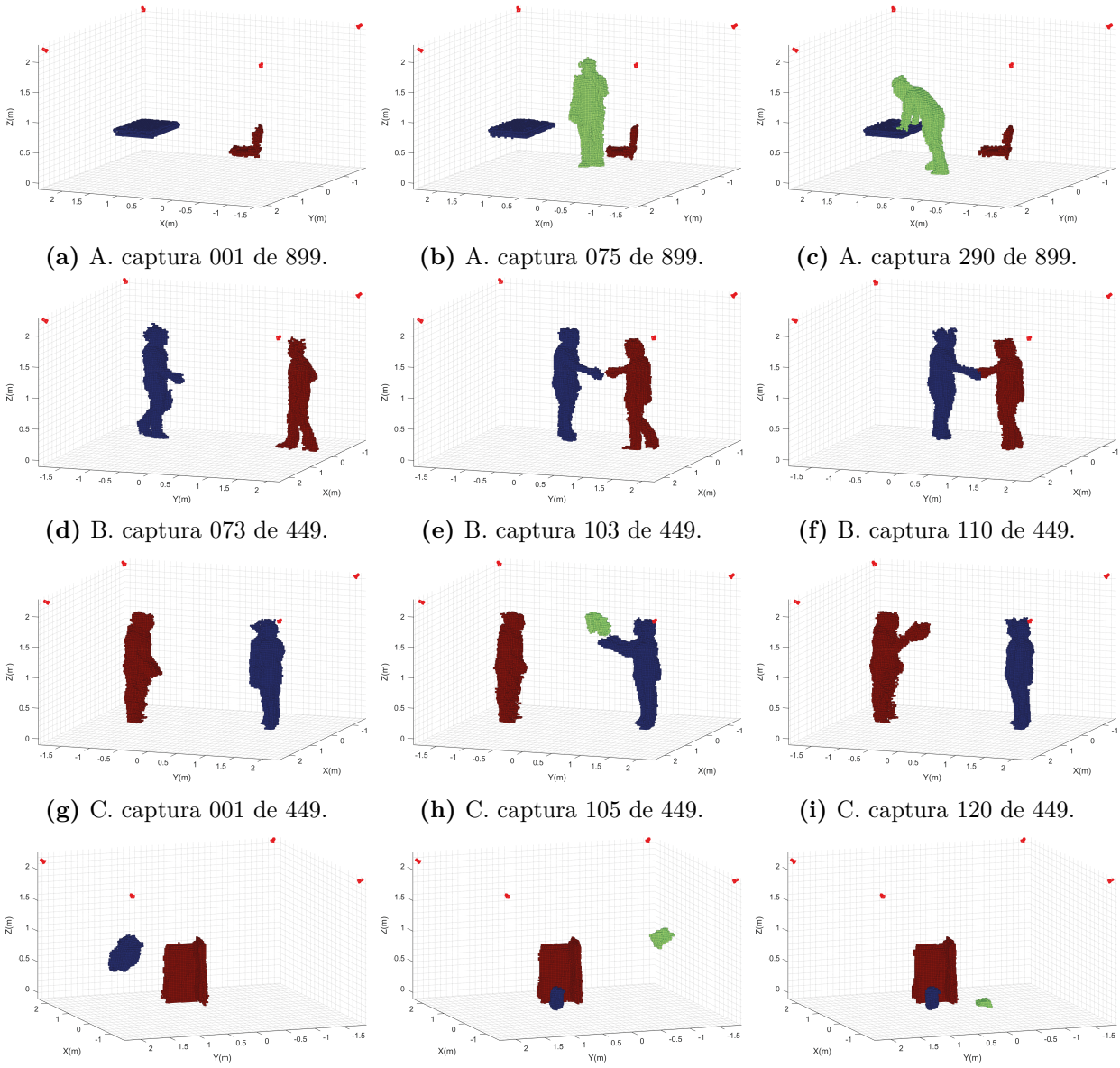
Figura 4.5: Resultados obtidos pelo sistema em diferentes etapas do processo.

para cada conjunto de dados podem ser observados na Figura 4.6, onde cada linha representa um conjunto de dados de A a D e cada coluna um momento de captura distinto. O parâmetro ϵ foi definido como demonstrado no método proposto, i.e., $\epsilon = \sqrt{2 \cdot V_s^2}$, onde $V_s = 0.025\ m$. Relativamente ao parâmetro $minPts$ este, foi definido como 70 através de um processo iterativo que procurou remover todo o *cluster* associado a ruído mantendo contudo, a informação relativa a objetos pequenos.

O resultado da segmentação para o conjunto de dados A apresentado na primeira linha da Figura 4.6, demonstra corretamente os dois objetos e o utilizador. A ausência dos suportes tanto da mesa como da cadeira é um resultado normal neste tipo de sistemas devido a estes serem normalmente metálicos com uma elevada taxa de reflexão, bem como pela própria espessura fina dos suportes, acabando por não haver muitas leituras válidas nestas zonas. Na Figura 4.6c, onde são identificados corretamente tanto a pessoa como a mesa, é o resultado do processo suplementar da sobre-segmentação através do algoritmo RBNN. Esta identificação apenas é possível neste sistema dado ter havido a identificação prévia do estado do movimento do objeto a azul (ou, nesta caso, da falta dele, dado ter sido identificado como estático).

Para o conjunto de dados B, a identificação correta das duas pessoas na Figura 4.6f é resultado do processo suplementar para o tratamento de dados presente na sobre-segmentação. Contudo, caso o primeiro momento de captura correspondesse à Figura 4.6f, o sistema não estaria apto para retornar uma segmentação correta e o resultado seria apenas um único *cluster*.

Relativamente às Figuras 4.6g, 4.6h, os algoritmos adotados para o processo de segmentação identificam facilmente os 3 *clusters* uma vez que não há interação entre eles devido à sua natureza radial. Contudo, na Figura 4.6i, a correta segmentação durante a interação com a caixa requereria um processamento mais complexo e desafiante para um sistema em



(a) A. captura 001 de 899. **(b)** A. captura 075 de 899. **(c)** A. captura 290 de 899.

(d) B. captura 073 de 449. **(e)** B. captura 103 de 449. **(f)** B. captura 110 de 449.

(g) C. captura 001 de 449. **(h)** C. captura 105 de 449. **(i)** C. captura 120 de 449.

(j) D. captura 180 de 299. **(k)** D. captura 222 de 299. **(l)** D. captura 250 de 299.

Figura 4.6: Resultados obtidos pelo módulo de segmentação para diversos conjuntos de dados.

tempo real.

No conjunto de dados D, uma vez que existe conhecimento prévio acerca da parede estática presente no cenário, qualquer interação resultará numa correta segmentação pela extração da mesma através do seu modelo construído ao longo do tempo.

Devido a limitações de tempo e a impraticabilidade de construir manualmente uma grelha de referência, não foi possível analisar quantitativamente o método proposto.

4.4 Seguimento da Cabeça

Como explanado anteriormente, o módulo de deteção de seguimento da cabeça recorre a um PF. Como tal, a utilização de um modelo com um número de partículas demasiado elevado pode ser computacionalmente muito pesado e, consequentemente, inviável para um sistema em tempo real. Desta forma, pretendeu-se estudar a contribuição do incremento de número de partículas bem como comprovar a hipótese de que com menos partículas (oferecendo um ganho linear em termos computacionais) o desempenho do sistema não é significativamente comprometido.

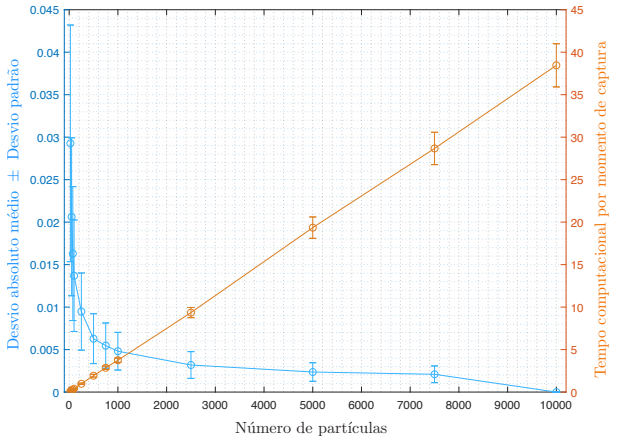
Para tal, considerou-se como referência um filtro com $10,000$ partículas (valor bastante dispendioso computacionalmente) sabendo que este valor irá permitir representar de forma bastante densa qualquer área resultante da distribuição gaussiana associada à incerteza de medição no processo de predição. Foi então estudado o desvio absoluto médio e os respetivos desvios padrões da posição retornada para vários valores do número de partículas (25; 50; 100; 250; ... ; 7,500) relativamente à referência imposta para dois conjuntos de dados distintos: G, pouco desafiante para a deteção e seguimento da cabeça, representado nas Figuras 4.7a e 4.7b; e E, desafiante devido aos movimentos bruscos realizados ao longo deste, representado nas Figuras 4.7c e 4.7d.

Através da análise das Figuras 4.7a e 4.7b constata-se uma melhoria de 0.0117 m ao passar de 100 partículas para 7,500 partículas, tendo como consequência um aumento médio no tempo computacional de 28.1 s por momento de captura. Este valor é sinónimo de um aumento de tempo computacional próximo de 7,025%, o que vai de encontro ao o valor teórico expectável (7,500%, resultante da complexidade temporal do algoritmo ser $\mathcal{O}(n)$, onde n representa o número de partículas).

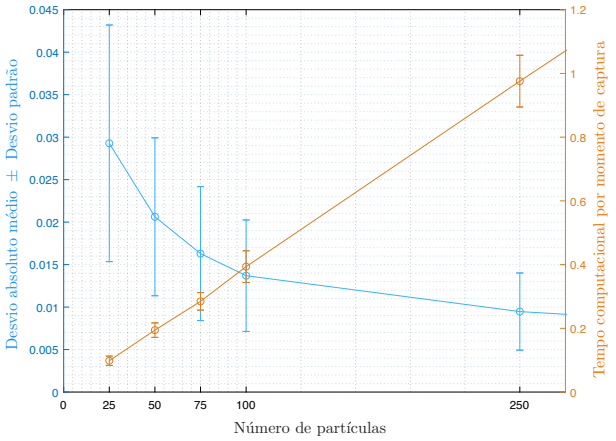
Nas Figuras 4.7c e 4.7d verifica-se que o seguimento da cabeça não foi realizado com sucesso para o conjunto de dados E através do uso de 25 e 75 partículas. Este comportamento é expectável para um número baixo de partículas visto tratar-se de um cenário desafiante para o PF: os movimentos bruscos ao longo do tempo traduzem-se num erro gaussiano aplicado à observação substancialmente elevado. Como tal, um número baixo de partículas é incapaz de preencher satisfatoriamente o presente volume de incerteza.

Desta forma, tendo em conta os motivos supramencionados, um PF constituído por 100 partículas é uma escolha eficiente, não comprometendo significativamente o desempenho do sistema.

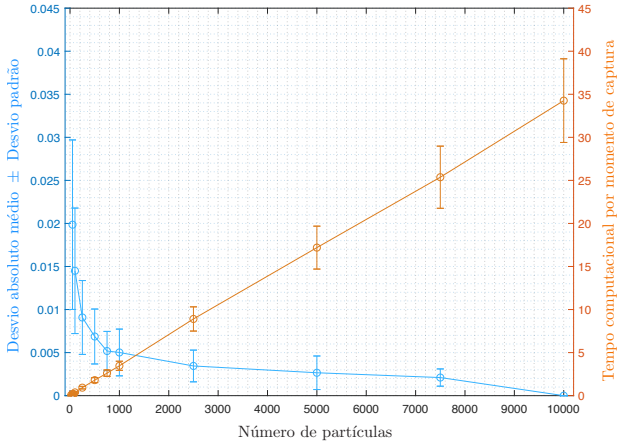
Consecutivamente, o módulo foi configurado de modo a utilizar 100 partículas e foi nova-



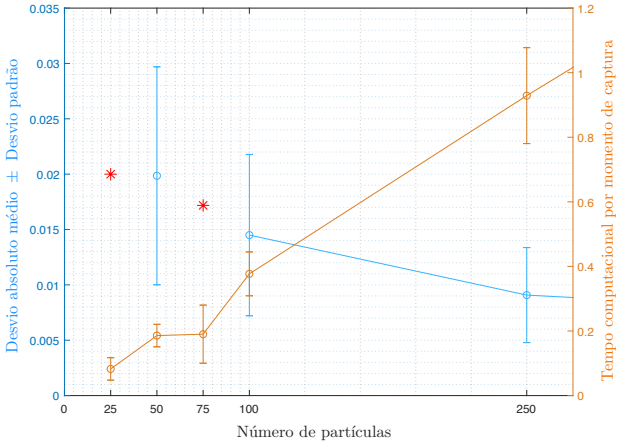
(a) Valores obtidos para o conjunto de dados G.



(b) Apresentação detalhada do gráfico (a) para um menor número de partículas.



(c) Valores obtidos para o conjunto de dados E.

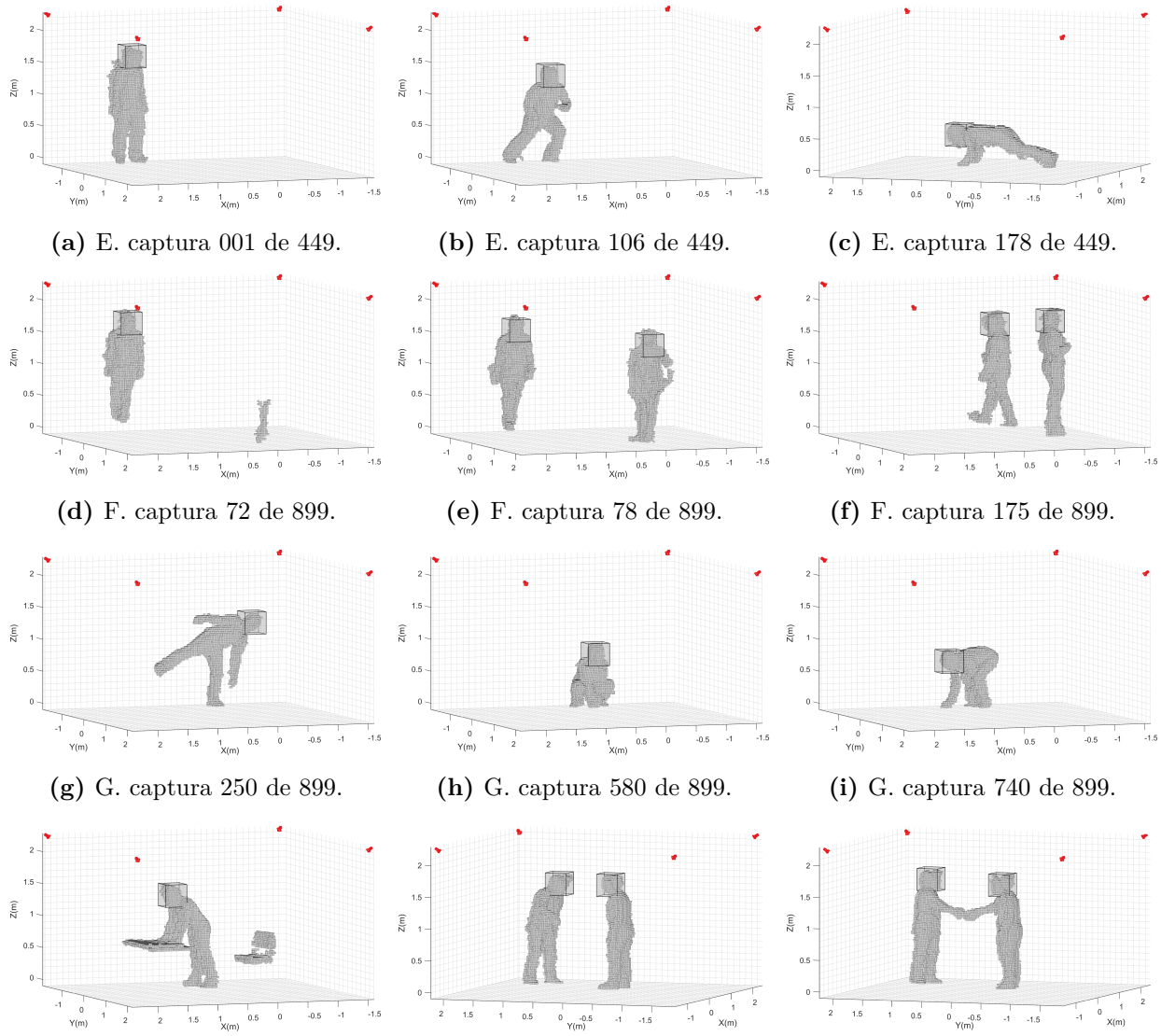


(d) Apresentação detalhada do gráfico (c) para um menor número de partículas.

Figura 4.7: Representação gráfica do desvio absoluto médio e desvios padrões para diferentes valores do número de partículas: a azul, representa-se o desvio absoluto médio e desvios padrões da posição da cabeça retornada pelo filtro de partículas para distintos números de partículas tomando como referência a posição para 10,000 partículas; a vermelho, são também apresentados os respetivos tempos de processamento por captura ao longo dos conjuntos de dados E e G.

mente aplicado a alguns dos conjuntos de dados anteriormente descritos (A, B, E, F, G). Os resultados qualitativos obtidos para diferentes momentos de captura nos respetivos conjuntos de dados podem ser observados na Figura 4.8.

De uma forma geral, através da análise da Figura 4.8, verifica-se que o sistema demonstra um comportamento satisfatório para uma variedade de casos, incluindo algumas situações extremas como as apresentadas nas Figuras 4.8c, 4.8g e 4.8i. As Figuras 4.8a e 4.8e representam as capturas onde o PF foi inicializado. Apesar de na Figura 4.8a a estimativa estar correta, na Figura 4.8e é evidente, em ambas as pessoas, uma discrepância entre a posição estimada e a real. Este comportamento é expectável na amostragem inicial das partículas pois o volume definido neste processo é relativamente grande, como demonstrado na Figura 3.10, resultando numa estimação inicial menos precisa.



(a) E. captura 001 de 449. **(b)** E. captura 106 de 449. **(c)** E. captura 178 de 449.

(d) F. captura 72 de 899. **(e)** F. captura 78 de 899. **(f)** F. captura 175 de 899.

(g) G. captura 250 de 899. **(h)** G. captura 580 de 899. **(i)** G. captura 740 de 899.

(j) A. captura 284 de 899. **(k)** B. captura 151 de 499. **(l)** B. captura 183 de 499.

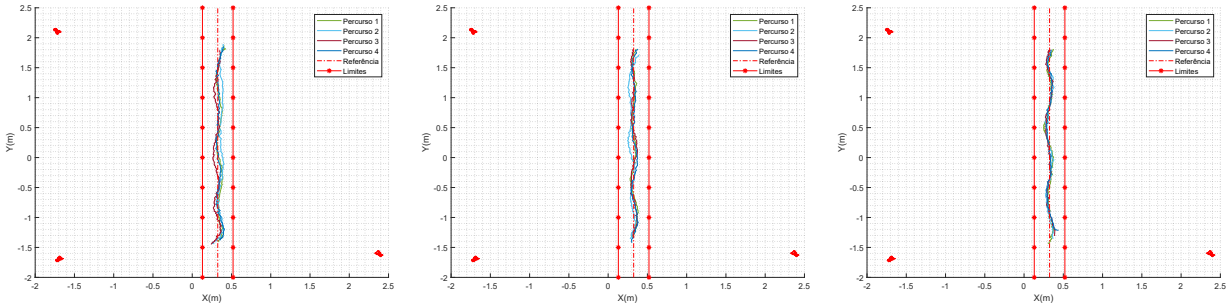
Figura 4.8: Resultados obtidos pelo módulo de seguimento de cabeças do sistema para diversos conjuntos de dados. A posição da cabeça é representada por um cubo de lado 0.35 m centrado na respetiva posição.

Para além desta avaliação qualitativa, foi pensada também uma avaliação quantitativa para o sistema proposto, tendo sido integrado um sistema inercial que efetuava a aquisição sensorial de um dispositivo móvel junto à cabeça cuja comunicação foi feita recorrendo ao *software* disponibilizado pelo MATLAB [66]. No entanto, os resultados obtidos por parte do sensor não foram satisfatórios, i.e., as medições retornadas pelo acelerómetro do dispositivo móvel tinham uma incerteza demasiado elevado para uma construção precisa de uma referência no SCM.

Para tentar obter um método que, apesar de ser novamente qualitativo, de melhor modo avalie o desempenho deste módulo, foi feita uma avaliação relativa a duas referências: a primeira corresponde a uma linha reta delimitada por uma região de 39 cm de largura

fisicamente imposta (por duas cordas colocadas paralelamente); a segunda corresponde a dois percursos com o formato de um quadrado (de lados 2.5 m e 2.3 m, respetivamente).

Na Figura 4.9 encontram-se representados os percursos realizados por 3 sujeitos, sendo que os respetivos desvios absolutos médios e desvios padrões associados a cada percurso de cada sujeito se encontram sumariados na Tabela 4.2.



(a) Sujeito 1. **(b)** Sujeito 2. **(c)** Sujeito 3.

Figura 4.9: Percurso efetuado por três sujeitos tendo em conta a referência em linha reta.

Tabela 4.2: Desvios absolutos médios e respetivos desvios padrões para a posição da cabeça ao longo de cada percurso linear, e para cada um dos sujeitos.

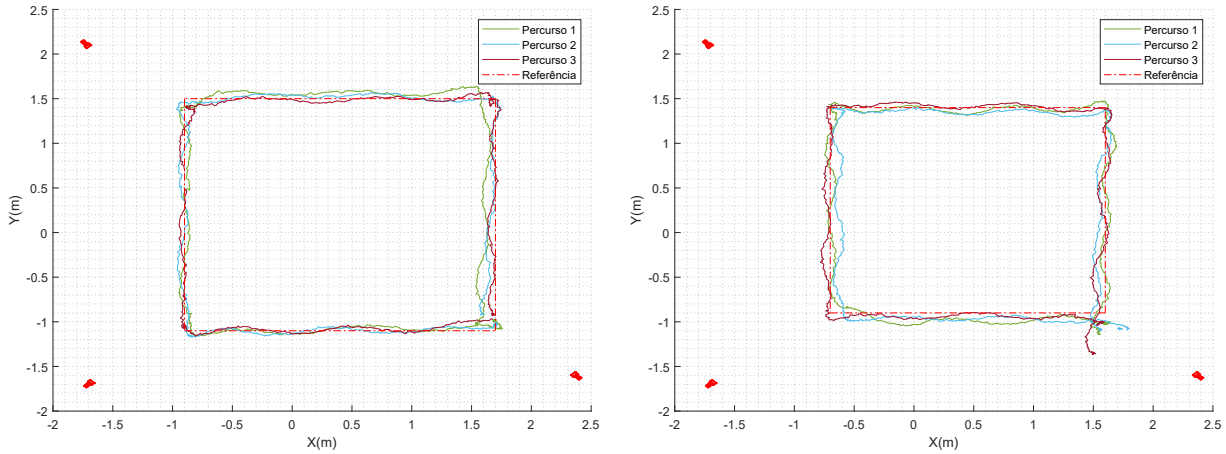
	Sujeito 1					Sujeito 2				
Percurso	1	2	3	4	**T**	1	2	3	4	**T**
DAM (m)	0.0294	0.0424	0.0273	0.0235	**0.0302**	0.0189	0.0328	0.0218	0.0208	**0.0236**
DP (m)	0.0217	0.0210	0.0173	0.0219	**0.0213**	0.0138	0.0197	0.0129	0.0140	**0.0162**

	Sujeito 3					
Percurso	1	2	3	4	**T**	**Geral**
DAM (m)	0.0262	0.0288	0.0244	0.0223	**0.0256**	**0.0265**
DP (m)	0.0169	0.0150	0.0164	0.0162	**0.0213**	**0.0184**

Tendo em conta a referência imposta e todos os percursos realizados foi obtido um desvio absoluto médio de 0.0265 m. Este valor é relativamente baixo sabendo que não se tem em conta o movimento natural do tronco e, consequentemente, da cabeça aquando da deslocação da pessoa (cujo comportamento é bastante visível na Figura 4.9c através do comportamento ondulatório da curva correspondente ao deslocamento da cabeça, com uma amplitude máxima no eixo X de 0.088 m).

Posteriormente, para as referências em formato de quadrado apresentadas na Figura 4.10, os resultados mantiveram-se satisfatórios. Apesar de este apresentar um desvio absoluto médio de 0.042 m, tem de ser tido em conta que o percurso foi efetuado seguindo um

alinhamento assinalado recorrendo apenas a quatro marcadores colocados no chão sinalizando os cantos do quadrado, o que pode levar naturalmente a algumas discrepâncias entre a referência considerada e a posição real da cabeça no mundo.



(a) Quadrado de lado 2.5 m. **(b)** Quadrado de lado 2.3 m.

Figura 4.10: Percurso efetuado por um sujeito tendo em conta a referência em formato de um quadrado.

Tabela 4.3: Desvios absolutos médios e respetivos desvios padrões para a posição da cabeça ao longo de cada percurso em forma de quadrado.

	Quadrado $2.5\ m \times 2.5\ m$				Quadrado $2.3\ m \times 2.3\ m$				
Percurso	1	2	3	**T**	1	2	3	**T**	**Geral**
DAM (m)	0.0504	0.0301	0.0297	**0.0368**	0.0402	0.0624	0.0416	**0.0480**	**0.0420**
DP (m)	0.0378	0.0244	0.0242	**0.0310**	0.0322	0.0367	0.0348	**0.0360**	**0.0341**

Tendo em consideração a existência dos conjuntos de dados deliberadamente exigentes para o funcionamento do módulo de seguimento da cabeça, pode concluir-se que este módulo apresenta um funcionamento bastante satisfatório, quer a nível das estimativas do posicionamento calculadas quer a nível do tempo de processamento.

5 Conclusão

Com os recentes desenvolvimentos no domínio da realidade imersiva, torna-se claro o seu potencial e vasto campo de aplicação. O aumento da capacidade computacional nos últimos tempos, aliado à evolução das capacidades/características dos seus mais variados componentes como ecrãs (quer pelo aumento da resolução, como pela taxa de atualização, entre outros...), sistemas embebidos e sensores, foram fatores cruciais à inovação no campo da realidade virtual, sendo possível verificar que esta área terá um grande impacto no futuro [67]. Uma das limitações dos sistemas comerciais encontra-se na forma como o movimento dos utilizadores no mundo real é efetuado e monitorizado. Tipicamente esse movimento é determinado através de um ou vários controladores manuseados pelo utilizador, tornando-se numa experiência menos imersiva que pode inclusivamente resultar numa experiência nauseante para algumas pessoas. Desta forma, o objetivo da presente tese foi o desenvolvimento de um sistema capaz de segmentar e seguir objetos, bem como efetuar o seguimento da cabeça de todos os utilizadores presentes no ambiente físico que irá ser mapeado no ambiente virtual. Da abordagem proposta resultam três:

- Representação tridimensional da fusão da informação relevante capturada pelos quatro sensores RGB-D, recorrendo a uma abordagem baseada em grelhas tridimensionais;
- Segmentação da informação previamente obtida no domínio 3D para a deteção de todos os objetos na cena, através da aplicação do algoritmo DBSCAN. Posteriormente, e como resposta ao desempenho insatisfatório desta abordagem (devido ao fato da pesquisa consecutiva dos NN para cada ponto na grelha ser inviável computacionalmente para um sistema que funcione em tempo-real e que tenha um baixo custo), novas alternativas foram investigadas. Desta forma, um método alternativo de segmentação, RBNN, foi implementado. Este torna-se, como pretendido, mais eficiente através das associações dos NN a pontos já associados a *clusters*, sem que estes tenham que voltar a fazer uma nova pesquisa.

- Aplicação de filtros de Kalman para o seguimento de todos os objetos presentes no cenário, aplicado na centroide da BB associada a cada objeto. Posteriormente, face à inviabilidade da aplicação de um filtro de Kalman para o seguimento da cabeça do utilizador (devido à baixa correlação existente entre as translações quer do centro de massa dos voxels associados à pessoa quer do centroide da respetiva BB e o movimento real da cabeça), foi proposto um filtro de partículas dada a sua maior flexibilidade para sistemas altamente não-lineares.

Para testar o trabalho desenvolvido, foram realizados uma série de ensaios experimentais através da captura prévia de conjuntos de dados deliberadamente exigentes. A análise dos resultados obtidos utilizando o método proposto demonstra a sua viabilidade e eficiência.

5.1 Trabalho Futuro

O método proposto, no seu estado atual, é capaz de alcançar resultados satisfatórios cumprindo todos os objetivos delineados. Contudo, trata-se de uma fase inicial de prototipagem do processo em MATLAB®. Face à necessidade de este se inserir num sistema de tempo real, a sua implementação numa linguagem mais eficiente do ponto de vista do tempo de execução, que apresente funcionalidades como p.ex. processamento paralelo, é fulcral.

Para além disso, o método apresentado não está apto a retornar uma segmentação correta aquando da interação do utilizador com um objeto dinâmico. Este tipo de situações oferecem ainda um desafio significativo neste tipo de sistemas (especialmente considerando o requisito do funcionamento em tempo real). Contudo, há duas abordagens que se revelam interessantes para desempenhar este requisito: integração da imagem RGB fornecida pelo sensor Kinect (dado atualmente apenas ser feito uso do mapa de profundidade) para a deteção de *features* de interesse afim de facilitar a deteção de objetos; implementação de uma rede neuronal, treinada com informação dos objetos que vão estar presentes no cenário, para melhor segmentar os objetos presentes aquando da utilização do sistema.

Relativamente ao módulo de seguimento de cabeça, poderá ser experimentada uma abordagem que integre a estimativa do KF da pessoa no processo de predição do PF dessa mesma pessoa (mais precisamente, na distribuição gaussiana associada à incerteza de medição). Esta aproximação pode ser vantajosa para a precisão da estimativa, i.e., uma aceleração estimada constante ou nula no vetor de estados retornado pelo KF poderá, teoricamente representar uma correlação mais forte entre a translação do centro de massa dos respetivos *voxels* e a translação real da cabeça no mundo.

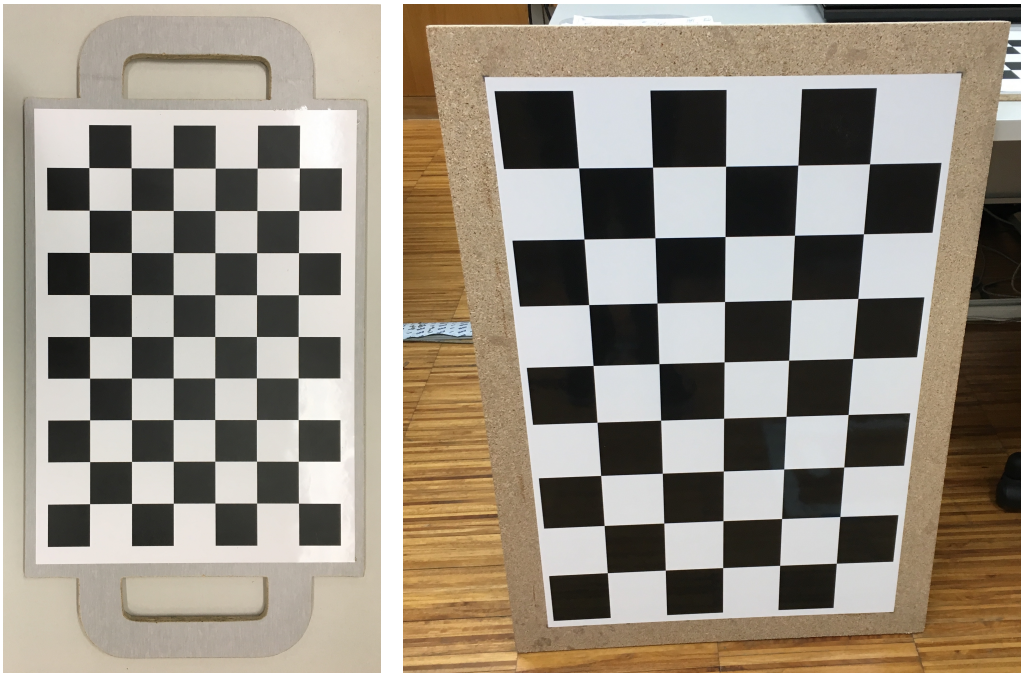
Apêndices

A Comparação das Tecnologias dos Sensores de Visão 3D

Tabela A.1: Comparação das tecnologias de captura de informação 3D (adaptado de [68]).

	Sistemas ToF	Sistemas Visão estéreo	Sistemas SL
Princípio de funcionamento	Pulso IR & medição do tempo de viagem do sinal	Par de sensores RGB com reconstrução por via de, p. ex. Structure from motion	Iluminação de padrão único (Visível ou IR) & detecção de distorção
Geração da nuvem de pontos	Diretamente a partir do chipset	Elevado poder de processamento necessário	Baixo/Médio poder de processamento necessário
Velocidade de aquisição	Rápida (Limitada pela velocidade do sensor)	Média (Limitada pela complexidade do software)	Média (Limitada pela Velocidade da câmara)
Desempenho com baixa iluminação	Bom	Fraco	Bom
Desempenho em ambientes externos	Médio (Depende da potência da iluminação)	Bom	Fraco/Médio (Depende da potência da iluminação)
Alcance	Curto a Longo (Depende da potência laser & modelação)	Médio (Depende do espaçamento entre câmaras)	Muito Curto a Médio (Depende da potência Da iluminação)
Resolução de imagem	QVGA / VGA	Baixa a Alta (Depende das câmaras utilizadas)	Baixa a Média (Depende do padrão projetado)
Precisão da leitura de profundidade	$mm \sim cm$ (Depende da resolução do sensor)	$mm \sim cm$ (Problemas com superfícies lisas)	$\mu m \sim cm$ (Padrões variáveis & diferentes fontes de luz podem melhorar a precisão)
Custo material	Médio	Baixo	Médio/Elevado

B Padrões Axadrezados Construídos para o Procedimento de Calibração



(a) Padrão 7×10 de lado 4 cm, impresso em tamanho A3.

(b) Padrão 6×9 de lado 8 cm, impresso em 50 $cm \times$ 75 cm.

Figura B.1: Padrões de calibração construídos.

C Análise do Erro de Reprojeção

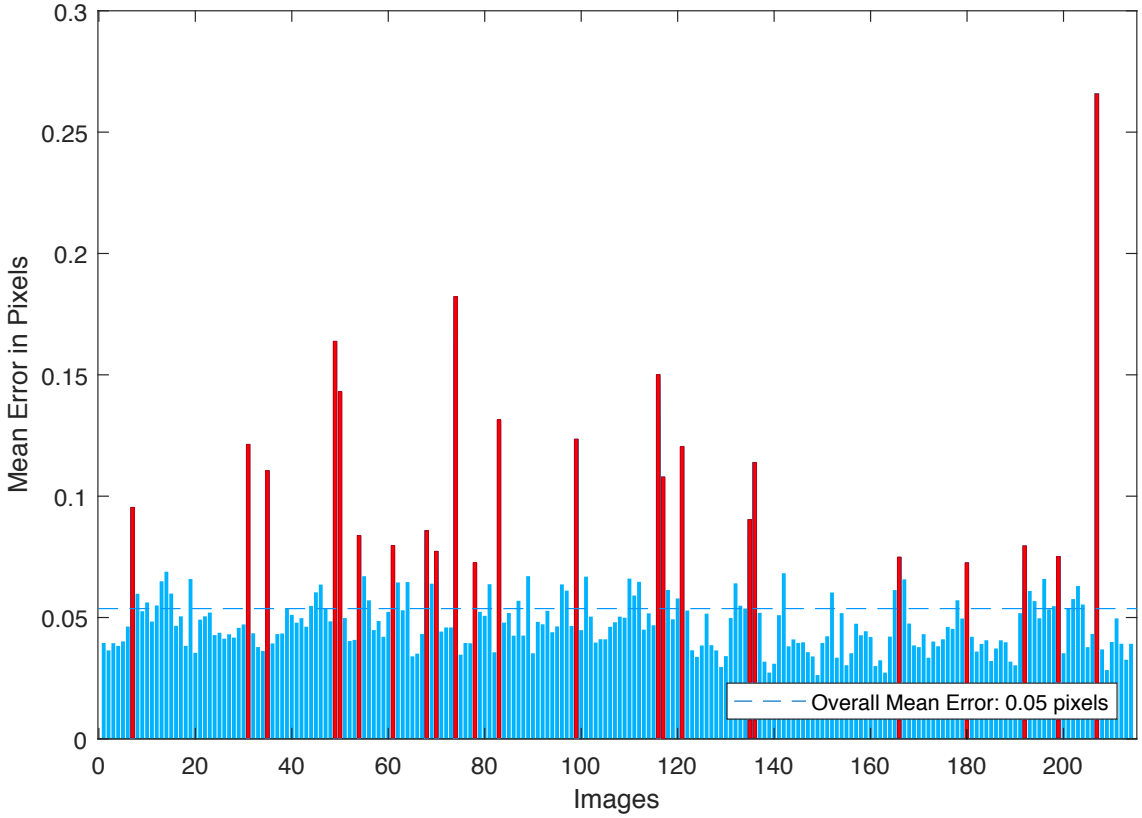


Figura C.1: Erro médio de reprojeção por imagem (as barras a vermelho, corresponde às imagens seleccionadas a serem removidas).

D Pseudocódigo para o Cálculo da Média e do Desvio Padrão do Plano de Fundo

Algorithm 1: Análise da média e do desvio padrão do plano de fundo de uma câmara para a remoção dos pontos pertencentes ao chão

```
AnalyzeFloor (depthImages, intrinsics);
Input  : empty room depthImages,
         intrinsic parameters of the camera;
Output: pointCloud < x, y, z >;
ground < means, stdDeviation > ← zeros(height(depthImages), ...
      width(depthImages));                                  ▷ Pre-Allocation
for k ← 1, ..., num. of depthImages do
    depthImage(:, :, k) = ...                  ▷ For each image of the empty scenario
        undistortImage(depthImages(:, :, k), intrinsics);
end
for i ← 1, ..., width(depthImages) do
    for j ← 1, ..., height(depthImages) do
        depths ← depthImages(j, i, :);
        for k ← 1, ..., size(depths) do
            if depths(k) ≤ 500 then
                delete(depths(k));
            end
        end
        ground.mean(j, i) ← mean(depths);
        ground.stdDeviation(j, i) ← std(depths);
    end
end
return ground
```

E Pseudocódigo para a Construção da Nuvem de Pontos

Algorithm 2: Criação da nuvem de pontos através do mapa de profundidade de uma câmara.

CreatePointCloud ($depthImage, intrinsics, extrinsics, ground$);
Input : $depthImage$,
$intrinsic$ and $extrinsic$ parameters of the camera,
$ground$, mean and standard deviation matrices (background detection);
Output: $pointCloud < x, y, z >$;
$image \leftarrow undistortImage(depthImage, intrinsicsParameters)$;
$image \leftarrow applyMedianFilter(image)$;
$cx \leftarrow$ optical center, x-axis; ▷ Intrinsics Camera Parameters
$cy \leftarrow$ optical center, y-axis;
$fx \leftarrow$ focal length, x-axis;
$fy \leftarrow$ focal length, y-axis;
$R \leftarrow$ rotational matrix; ▷ Extrinsics Camera Parameters
$t \leftarrow$ translation matrix;
$pointCloud < x, y, z > \leftarrow \varnothing$;
for $y \leftarrow 1, \ldots, imageHeight$ **do**
  for $x \leftarrow 1, \ldots, imageWidth$ **do**
    if $image(y, x) \geq 500 \wedge \ldots$ ▷ *Map only valid pixels*
    $(isNotDefined(ground.deviation(y, x)) \vee \ldots$
    $image(y, x) < ground.means(y, x) - 3 * ground.stdDeviation(y, x))$ **then**
      $pointX \leftarrow (x - c_x) \cdot \frac{image(y,x)}{f_x}$; ▷ Transformation of $ICS \rightarrow CCS$
      $pointY \leftarrow (y - c_y) \cdot \frac{image(y,x)}{f_y}$;
      $[X, Y, Z] \leftarrow ([pointX, pointY, image(y, x)] - t) \cdot R^T$ ▷ $CCS \rightarrow WCS$
      $pointCloud \leftarrow pointCloud \bigcup < X, Y, Z >$
    end
  end
end
return $pointCloud$

F Procedimento para o Processamento dos Filtros de Kalman (Perdidos)

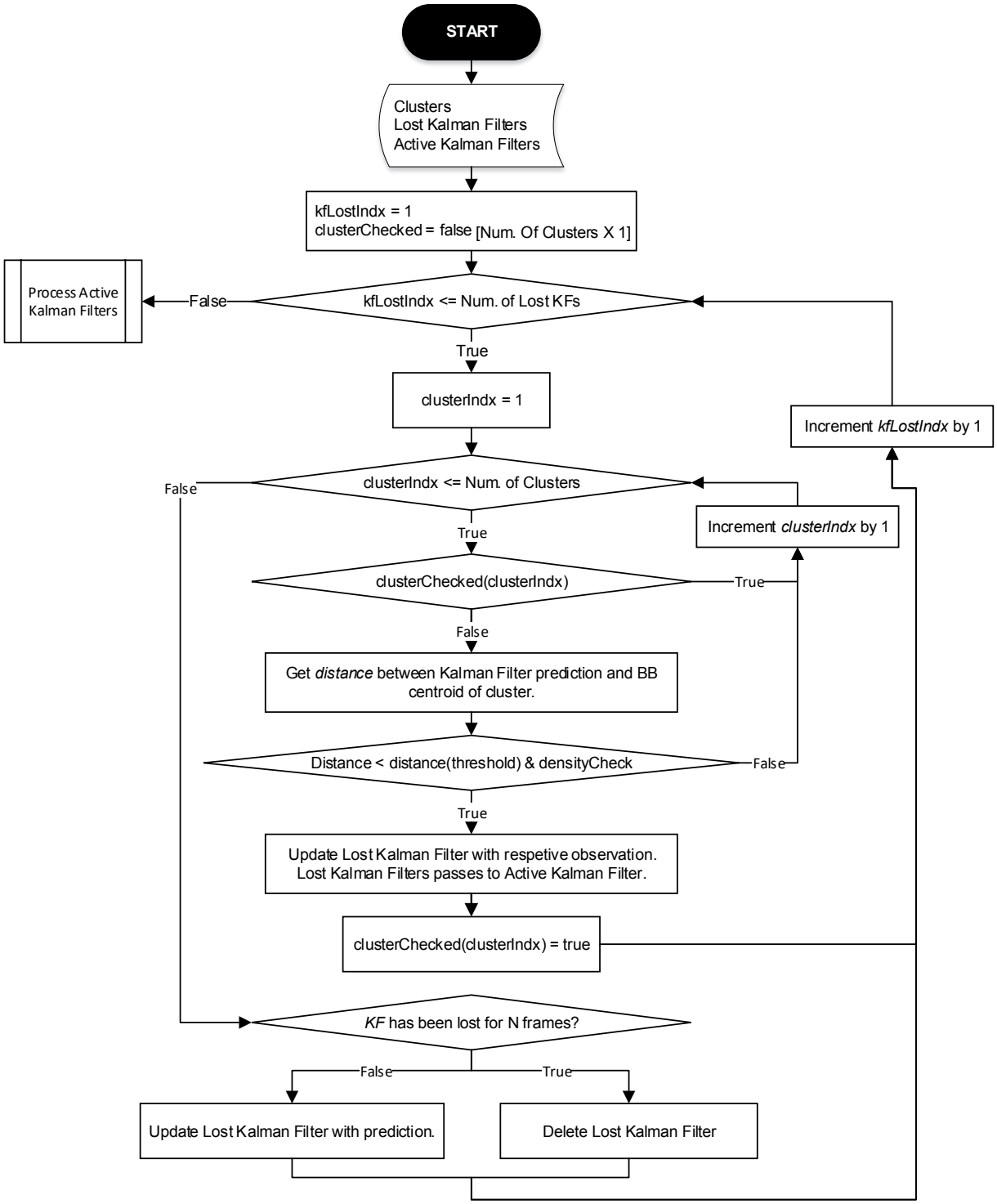


Figura F.1: Procedimento para a atualização dos filtros de Kalman que se encontram sem observação válida.

G Procedimento para o Processamento dos Filtros de Kalman (Ativos)

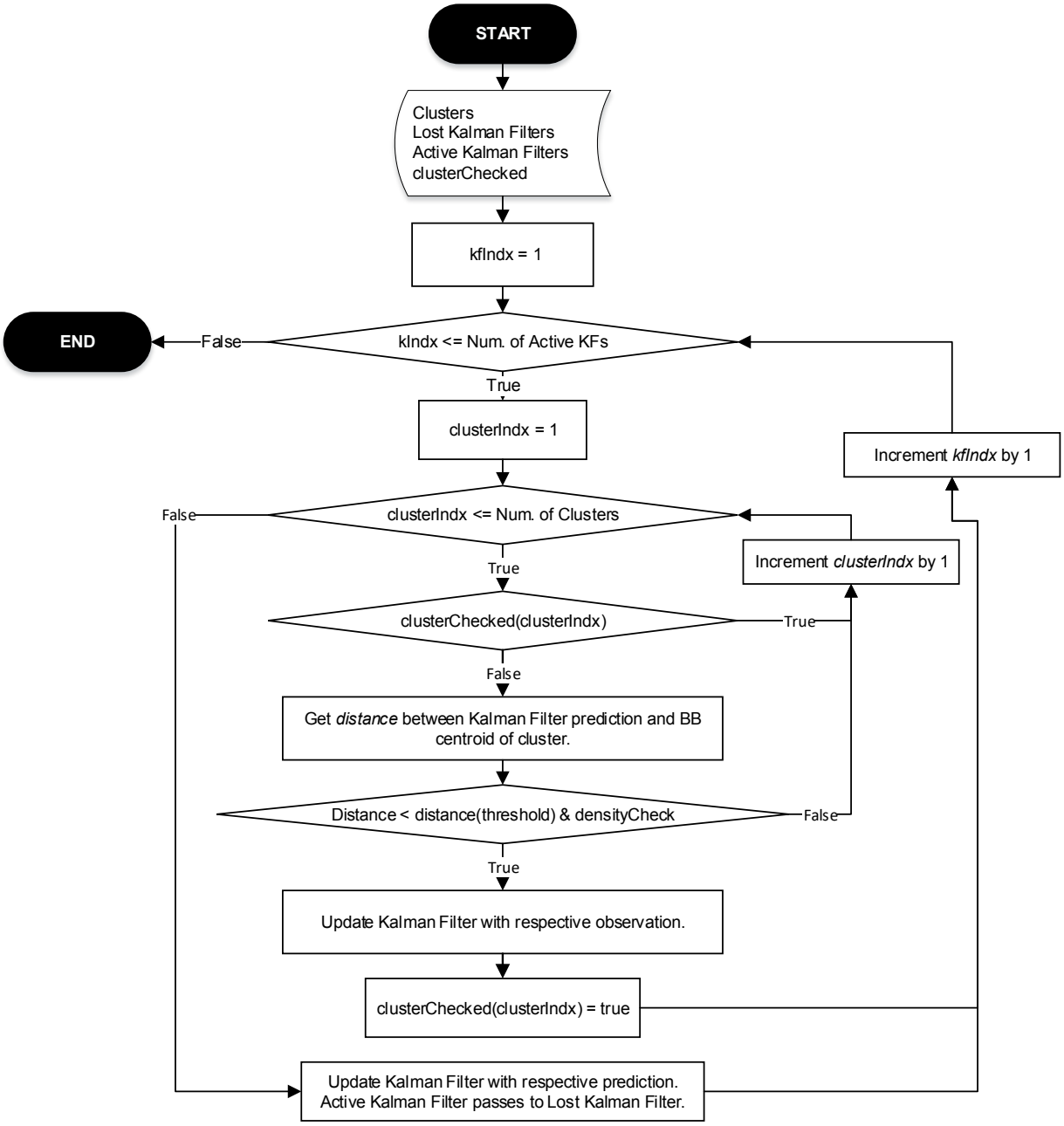


Figura G.1: Procedimento para a atualização dos filtros de Kalman ativos.

Referências

[1] Doug A. Bowman and Ryan P. McMahan. “Virtual reality: How much immersion is enough?” In: *Computer* (2007), pp. 37–39.

[2] Corey J. Bohil, Bradly Alicea, and Frank A. Biocca. *Virtual reality in neuroscience research and therapy*. 2011.

[3] Ivan Sutherland. *The Ultimate Display. Proceedings of IFIP Congress 2*. 1965.

[4] *An Introduction to Enterprise Virtual Reality*. [Conferido em: 12-07-2018]. 2017. URL: `https://www.pwc.com.au/consulting/assets/technology/virtual-reality-may17.pdf`.

[5] Rizwan Macknojia, Alberto Chavez-Aragon, Pierre Payeur, and Robert Laganiere. “Calibration of a network of Kinect sensors for robotic inspection over a large workspace”. In: *2013 IEEE Workshop on Robot Vision, WORV 2013*. 2013.

[6] Jesus Suarez and Robin R. Murphy. “Using the Kinect for search and rescue robotics”. In: *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2012*. 2012.

[7] M. Alnowami, B. Alnwaimi, F. Tahavori, M. Copland, and K. Wells. “A quantitative assessment of using the Kinect for Xbox360 for respiratory surface motion tracking”. In: *Proc. SPIE* (2012).

[8] Dimitris Kastaniotis, George Economou, Spiros Fotopoulos, Gerasimos Kartsakalis, and Panagiotis Papathanasopoulos. “Using kinect for assesing the state of Multiple Sclerosis patients”. In: *Proceedings of the 2014 4th International Conference on Wireless Mobile Communication and Healthcare - "Transforming Healthcare Through Innovations in Mobile and Wireless Technologies", MOBIHEALTH 2014*. 2015.

[9] Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. “Enhanced computer vision with Microsoft Kinect sensor: A review”. In: *IEEE Transactions on Cybernetics* (2013).

[10] Donghwa Lee, Hyongjin Kim, and Hyun Myung. “GPU-based real-time RGB-D 3D SLAM”. In: *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. 2012.

[11] E. Lachat, H. Macher, M. A. Mittet, T. Landes, and P. Grussenmeyer. “First experiences with kinect V2 sensor for close range 3D modelling”. In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*. 2015, p. 5.

[12] Péter Fankhauser, Michael Bloesch, Diego Rodriguez, Ralf Kaestner, Marco Hutter, and Roland Siegwart. “Kinect v2 for mobile robot navigation: Evaluation and modeling”. In: *Proceedings of the 17th International Conference on Advanced Robotics, ICAR 2015*. 2015.

[13] Daniel Freedman, Yoni Smolin, Eyal Krupka, Ido Leichter, and Mirko Schmidt. “SRA: Fast removal of general multipath for ToF sensors”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2014.

[14] Sebastian Thrun and A Bücken. “Integrating Grid-Based and Topological Maps for Mobile Robot Navigation”. In: *Proceedings Of The National Conference On Artificial Intelligence* (1996), pp. 1–2.

[15] H. P. Moravec. “Sensor Fusion in Certainty Grids for Mobile Robots”. In: *AI Magazine* (1988).

[16] Alberto Elfes. “Sonar-Based Real-World Mapping and Navigation”. In: *IEEE Journal on Robotics and Automation* (1987).

[17] Martial Hebert, C. Caillas, Eric Krotkov, In So Kweon, and Takeo Kanade. “Terrain Mapping for a Roving Planetary Explorer”. In: *IEEE International Conference on Robotics and Automation (ICRA ’89), 997-1002* (1989).

[18] Cristiano Premebida, Joao Sousa, Luis Garrote, and Urbano Nunes. “Polar-Grid Representation and Kriging-Based 2.5D Interpolation for Urban Environment Modelling”. In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. 2015, p. 5.

[19] Pant, Abhishek. *Probabilistic Occupancy Grid Mapping Using Monocular Vision*. [Conferido em: 11-08-2018]. 2015. URL: `https://www.youtube.com/watch?v=RhPlzIyTT58`.

[20] Lee, Daniel. *Producing 3D point clouds with a stereo camera in OpenCV*. [Conferido em: 12-08-2018]. 2014. URL: https://erget.wordpress.com/2014/04/27/producing-3d-point-clouds-with-a-stereo-camera-in-opencv/.

[21] PMD Technologies. *World Class 3D Depth Sensing Development Kits. For Everyone.* [Conferido em: 12-08-2018]. URL: https://pmdtec.com/picofamily/.

[22] Wikipedia. *Voxel*. [Conferido em: 12-08-2018]. URL: https://en.wikipedia.org/wiki/Voxel.

[23] Arie Kaufman, Roni Yagel, and Daniel Cohen. "Volume Graphics". In: *Computer* (1993).

[24] Donald Meagher. "Geometric modeling using octree encoding". In: *Computer Graphics and Image Processing* (1982).

[25] Antonin Guttman. "R-trees". In: *Proceedings of the 1984 ACM SIGMOD international conference on Management of data - SIGMOD '84*. 1984.

[26] K M Wurm, a Hornung, M Bennewitz, C Stachniss, and W Burgard. "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems". In: *Proc of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation* (2010).

[27] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. "OctoMap: An efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous Robots* (2013).

[28] Fan Cai, NA Le-Khac, and M-tahar Kechadi. "Clustering Approaches for Financial Data Analysis: a Survey". In: *Proceedings of the 8th International Conference on Data Mining, Las Vegas* (2012).

[29] A.K. Jain, R. P W Duin, and Jianchang Mao. "Statistical pattern recognition: a review". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2000).

[30] Yihang Yin, Fengzheng Liu, Xiang Zhou, and Quanzhong Li. "An Efficient Data Compression Model Based on Spatial Clustering and Principal Component Analysis in Wireless Sensor Networks." In: *Sensors (Basel, Switzerland)* (2015).

[31] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis (Wiley Series in Probability and Statistics)*. 1990.

[32] Lior Rokach and Oded Maimon. “Clustering methods”. In: *Data mining and knowledge discovery handbook*. 2005, pp. 321–352.

[33] J. A. Hartigan and M. A. Wong. “A K-Means Clustering Algorithm”. In: *Applied Statistics* (1979).

[34] David Arthur and Sergei Vassilvitskii. “k-means++: the advantages of careful seeding”. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. 2007.

[35] Sang Su Lee, Dongwoo Won, and Dennis Mcleod. “Discovering Relationships among Tags and Geotags”. In: *Power* (2007).

[36] Alvida Mustika Rukmi and Ikhwan Muhammad Iqbal. “Using k-means++ algorithm for researchers clustering”. In: *AIP Conference Proceedings*. 2017.

[37] Martin Ester, Hans P Kriegel, Jorg Sander, and Xiaowei Xu. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining* (1996).

[38] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. “DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN”. In: *ACM Transactions on Database Systems* (2017), p. 4.

[39] Brian S. Everitt, Sabine Landau, Morven Leese, and Daniel Stahl. *Cluster Analysis*. 2011. arXiv: `arXiv:1011.1669v3`.

[40] *Bayesian Filtering for Location Estimation*. [Conferido em: 09-08-2018]. URL: `http://www.irisa.fr/aspi/legland/ref/fox03a.pdf`.

[41] Rudolf Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Transaction of the ASME - Journal of Basic Engineering* (1960), pp. 35–45.

[42] Alireza Asvadi, Pedro Girão, Paulo Peixoto, and Urbano Nunes. “3D Object Tracking using RGB and LIDAR Data”. In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)* (2016).

[43] Andre Ibisch, Sebastian Houben, Marc Schlipsing, Robert Kesten, Paul Reimche, Florian Schuller, and Harald Altinger. “Towards highly automated driving in a parking garage: General object localization and tracking using an environment-embedded camera system”. In: *IEEE Intelligent Vehicles Symposium, Proceedings*. 2014.

[44] a Doucet, N De Freitas, and N Gordon. “Sequential Monte Carlo Methods in Practice”. In: *Springer New York* (2001).

[45] Michael D. Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. “Robust tracking-by-detection using a detector confidence particle filter”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2009.

[46] Jie Yu. “A particle filter driven dynamic Gaussian mixture model approach for complex process monitoring and fault diagnosis”. In: *Journal of Process Control* (2012).

[47] D. Brown, G. Georgoulas, H. Bae, G. Vachtsevanos, R. Chen, Y. H. Ho, G. Tannenbaum, and J. B. Schroeder. “Particle filter based anomaly detection for aircraft actuator systems”. In: *IEEE Aerospace Conference Proceedings*. 2009.

[48] Lingzhu Xiang et al. *libfreenect2: Release 0.2 (Version v0.2)*. [Conferido em: 10-07-2018]. 2016. URL: https://doi.org/10.5281/zenodo.50641.

[49] *Kinect for Windows SDK 2.0*. [Conferido em: 31-07-2018]. URL: https://www.microsoft.com/en-us/download/details.aspx?id=44561.

[50] *USB Active Booster Extension Cable*. [Conferido em: 29-07-2018]. URL: https://www.datapro.net/products/usb-active-booster-extension-cable.html.

[51] Oliver Wasenmüller and Didier Stricker. “Comparison of kinect v1 and v2 depth images in terms of accuracy and precision”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2017, pp. 4–6, 10.

[52] Zhengyou Zhang. “A flexible new technique for camera calibration”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334.

[53] Bouguet, Jean-Yves. *Camera Calibration Toolbox for Matlab*. [Conferido em: 06-08-2018]. 2015. URL: http://www.vision.caltech.edu/bouguetj/calib_doc/.

[54] Diana Pagliari and Livio Pinto. “Calibration of Kinect for Xbox One and comparison between the two generations of microsoft sensors”. In: *Sensors (Switzerland)* (2015), p. 11.

[55] Jiyoung Jung, Joon-Young Lee, and In So Knweon. *Noise aware depth denoising for a time-of-flight camera*. [Conferido em: 15-07-2018]. 2014. URL: https://joonyoung-cv.github.io/assets/paper/14_fcv_noise_aware.pdf.

[56] Carlos Moreno and Ming Li. "A Comparative Study of Filtering Methods for Point Clouds in Real-Time Video Streaming". In: *World Congress on Engineering and Computer Science 2016 Vol I WCECS 2016* (2016).

[57] Klaas Klasing, Dirk Wollherr, and Martin Buss. "A clustering method for efficient segmentation of 3D laser data." In: *Icra* (2008).

[58] *Gift Wrapping Algorithm*. [Conferido em: 20-07-2018]. URL: https://en.wikipedia.org/wiki/Gift_wrapping_algorithm.

[59] *Graham Scan Algorithm*. [Conferido em: 20-07-2018]. URL: https://en.wikipedia.org/wiki/Graham_scan.

[60] *Static adult human physical characteristics of the head - Average Head Sizes*. [Conferido em: 24-07-2018]. URL: https://upload.wikimedia.org/wikipedia/commons/0/06/AvgHeadSizes.png.

[61] Sebastian Thrun. "Probabilistic robotics". In: *Communications of the ACM* (2002), pp. 85–87.

[62] *Independent random sample with stratified and/or variable probability pros and cons*. [Conferido em: 23-07-2018]. URL: https://groups.nceas.ucsb.edu/monitoring-kb/2-design/2.1_Status_Trend_Design/2.1.1-spatial/2.1.6-survey_design/2.6.1.2.5-irs_strat_variable_proscons.

[63] *Performance Tradeoff - When is MATLAB better/slower than C/C++*. [Conferido em: 31-08-2018]. 2015. URL: https://stackoverflow.com/questions/20513071/performance-tradeoff-when-is-matlab-better-slower-than-c-c.

[64] Thomas Butkiewicz. "Low-cost coastal mapping using Kinect v2 time-of-flight cameras". In: *2014 Oceans - St. John's, OCEANS 2014*. 2015, p. 2.

[65] Hamed Sarbolandi, Damien Lefloch, and Andreas Kolb. "Kinect range sensing: Structured-light versus Time-of-Flight Kinect". In: *Computer Vision and Image Understanding* (2015).

[66] *MATLAB Mobile - Connect to MATLAB from your iPhone, iPad or Android device*. [Conferido em: 31-08-2018]. URL: https://stackoverflow.com/questions/20513071/performance-tradeoff-when-is-matlab-better-slower-than-c-c.

[67] *Virtual Reality (VR) - Statistics & Facts*. [Conferido em: 07-09-2018]. URL: https://www.statista.com/topics/2532/virtual-reality-vr/.

[68] Brading, Michael et al. *Using 3D sensors to bring depth discernment to embedded vision apps*. [Conferido em: 10-08-2018]. 2013. URL: https://www.embedded.com/print/4411991.