TÉCNICO LISBOA

Seguimento 3D em Tempo Real de Objetos Simples com uma Câmara RGB

Lino Cordeiro Pereira

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e Computadores

Orientador: Prof. Alexandre José Malheiro Bernardino

Júri

Presidente: Prof. João Fernando Cardoso Silva Sequeira
Orientador: Prof. Alexandre José Malheiro Bernardino
Vogal: Prof. Pedro Daniel dos Santos Miraldo

Outubro 2020

Declaração

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Agradecimentos

Quero exprimir os meus agradecimentos a todos aqueles que, de alguma forma, permitiram que este projeto se concretizasse. Este projeto apenas se tornou possível derivado do importante apoio e incentivos que recebi ao longo deste período.

Ao meu orientador Professor Alexandre Bernardino quero agradecer por toda a ajuda, orientação e tempo despendido no desenvolvimento deste projeto.

Aos meus colegas de mestrado que me acompanharam ao longo deste percurso, com quem partilhei sucessos e aventuras, o meu agradecimento.

A todos os meus amigos, e em especial à minha namorada, agradeço a motivação que me deram para finalizar mais uma etapa e por estarem sempre disponíveis quando precisei de distração.

Por último, porque sem eles não seria possível, dirijo um agradecimento especial aos meus pais e irmãos pelo apoio incondicional, incentivo e paciência demonstrado no decurso deste projeto.

Resumo

Atualmente, algoritmos que obtêm informação 3D de objetos, apresentam uma elevada importância em aplicações de diversas áreas, como por exemplo no desporto, robótica, medicina, entre outros.

Neste trabalho, pretende-se melhorar um algoritmo de seguimento monocular baseado num método de regiões utilizando uma câmara RGB. O algoritmo que se pretende melhorar, deriva de um filtro de partículas (PF) onde cada partícula representa uma hipótese do estado do objeto em 3D, o que permite o uso de modelos de movimento 3D realistas. Porém, a literatura menciona que o filtro de partículas usa uma distribuição de importância para propagar as partículas bastante limitada, o que facilmente leva a que o filtro degenere e perca o rasto do objeto. Dada a limitação do PF, nesta dissertação é proposto um *unscented particle filter* (UPF) que obtém uma aproximação à distribuição de importância ótima, ao incorporar uma observação corrente do estado.

Apenas a distribuição de importância é diferente, o modelo de observação permanece idêntico, onde um método baseado nas distribuições de cor das regiões interior e exterior da silhueta do objeto é utilizado para calcular a verosimilhança de cada partícula.

De maneira a comparar o algoritmo proposto com o anterior, ambos são implementados neste trabalho e várias experiências reais e simuladas com um objeto simples são realizadas. Pelos resultados obtidos, os filtros demonstram ter êxito, sendo o UPF mais robusto perante degeneração, diferentes números de partículas e diferentes ajustes nas regiões interior e exterior da silhueta do objeto.

Palavras-chave: Seguimento 3D de Objetos, Seguimento Monocular, Método baseado em Regiões, Filtro de Partículas, Unscented Particle Filter

Abstract

Nowadays, algorithms that obtain 3D information of objects, have a high importance in applications in different areas, such as in sports, robotics, medicine, among others.

This work, intends to improve a monocular region-based tracking algorithm using an RGB camera. The algorithm to be improved, derives from a particle filter where each particle represents a hypothesis of the state of the object in 3D, which allows the use of realistic 3D motion models. However, the literature mentions that the particle filter (PF) uses a very limited importance distribution to propagate the particles, which easily leads to the filter degenerating and losing track of the object. Given the limitation of the PF, an unscented particle filter (UPF) is proposed, this one obtains an approximation to the optimal importance distribution, by adding a current observation of the state.

Since, only the importance distribution is different, the observation model remains identical, where a method based on the color distributions of the interior and exterior regions of the object's silhouette, is used to calculate the likelihood of each particle.

In order to compare the proposed algorithm with the previous one, both are implemented in this work and several real and simulated experiments with a simple object are performed. From the results obtained, is shown that the filters are successful, with the UPF being more robust against degeneration, different numbers of particles and different adjustments in the interior and exterior regions of the object's silhouette.

Keywords: 3D object tracking, monocular tracking, region-based methods, particle filter, unscented particle filter

Índice

Agradecimentos v
Resumo vi
Abstract vii
Lista de Tabelas xiii
Lista de Figuras xv
Lista de Abreviações xvii

1 Introdução **1**
1.1 Objetivos e Abordagem ao Problema 2
1.2 Contribuições 2
1.3 Estrutura da Dissertação 3

2 Fundamentação Teórica **5**
2.1 Sistemas e Filtros 6
2.1.1 Estado de um Sistema 6
2.1.2 Filtro de *Bayes* 6
2.2 Filtro de *Kalman* 9
2.2.1 Passo de predição 9
2.2.2 Passo de observação 10
2.3 Abordagem do Filtro de *Kalman* para Problemas Não-Lineares 10
2.3.1 Filtro de *Kalman* Estendido 11
2.3.2 *Unscented Kalman Filter* 13
2.4 Filtro de Partículas 16
2.4.1 Simulação de *Monte Carlo* 16
2.4.2 Amostragem de Importância Sequencial 17
2.4.3 Reamostragem das Partículas 18
2.4.4 Passo de Predição 21
2.4.5 Passo de Observação 21
2.4.6 Passo de Seleção 22
2.5 Melhoria da Distribuição de Importância 23
2.6 Vantagens e Desvantagens dos Filtros Mencionados 25

3 Estado da Arte **26**
3.1 Câmaras e Abordagens para Localização e Seguimento 3D . . . 26
3.2 Métodos de Localização e Seguimento 3D . . . 27
3.2.1 Métodos Baseado em Contornos . . . 28
3.2.2 Métodos Baseado em Características . . . 29
3.2.3 Métodos Baseado em Regiões . . . 30
3.2.4 Métodos Diretos . . . 31
3.2.5 Comparação entre Métodos . . . 31

4 Metodologias **33**
4.1 Algoritmo de Seguimento 3D de Objetos . . . 34
4.1.1 Modelo de predição . . . 35
4.1.2 Modelo de observação . . . 36
4.2 Arquiteturas dos algoritmos . . . 37
4.3 Métodos implementados nos algoritmos . . . 40
4.3.1 Modelos de projeção . . . 40
4.3.2 Obtenção dos pontos das regiões interior e exterior . . . 43
4.3.3 Modelo de movimento incorporando colisões . . . 46

5 Implementação **50**
5.1 Simulador . . . 50
5.1.1 Projeção de uma superfície planar quadrada na imagem . . . 53
5.1.2 Trajetórias Simuladas . . . 54
5.2 Deteção e estimação da pose do obstáculo . . . 56
5.3 Calibração Real de Uma Câmara . . . 58
5.4 Implementação em C++, utilizando a biblioteca *OpenCV* . . . 59

6 Resultados e Discussão **61**
6.1 Obtenção do estado exato em experiências reais . . . 61
6.2 Critérios utilizados para avaliar o erro . . . 62
6.2.1 Parâmetros a testar . . . 63
6.3 Experiências . . . 64
6.3.1 Trajetória A . . . 65
6.3.2 Trajetória B . . . 67
6.3.3 Trajetória C . . . 68
6.4 Trajetórias reais . . . 71
6.5 Testes de eficiência . . . 72

7 Conclusão **75**

Bibliografia **77**

A Algoritmos dos Filtros **81**

A.1 Algoritmo do Filtro de *Kalman* 81

A.2 Algoritmo do Filtro de *Kalman* Estendido 81

A.3 Algoritmo do *Unscented Kalman Filter* 82

A.4 Algoritmo do Filtro de Partículas 82

A.5 Algoritmo do Filtro de Partículas de *Kalman* Estendido 83

A.6 Algoritmo do *Unscented Particle Filter* 83

B Obtenção das matrizes S e V **85**

Lista de Tabelas

3.1 Comparação dos métodos de Localização e Seguimento 3D 32

6.1 Resultados da raiz quadrática do erro quadrático médio em mm, para diferentes números de partículas para a trajetória A. 67

6.2 Resultados da raiz quadrática do erro quadrático médio em mm, para diferentes números de partículas para a trajetória B. 68

6.3 Tempo de execução de cada método 73

Lista de Figuras

1.1 Algoritmos que utilizam informação 3D 1

2.1 Exemplo de um problema de filtragem 5
2.2 Diagrama do filtro de *Bayes* 8
2.3 Ilustração do filtro de *Kalman* 11
2.4 Linearização de uma função arbitrária 13
2.5 *Scaled Unscented Transformation* 15
2.6 Degeneração e empobrecimento do algoritmo de amostragem de importância sequencial 19
2.7 Processos de Reamostragem 20
2.8 Ilustração do algoritmo do Filtro de Partículas 22
2.9 Influência da distribuição proposta 24

3.1 Exemplo de um fundo confuso 28
3.2 Algoritmo RAPID 29
3.3 Algoritmo de deteção de pontos chave SIFT 30
3.4 Algoritmo PWP3D 31

4.1 Imagem criada por uma câmara omnidirecional 34
4.2 Arquitetura do Filtro de Partículas. 37
4.3 Arquitetura do sistema baseado no UPF 39
4.4 Modelo de perspetiva exata 41
4.5 Imagem que apresenta uma grande distorção radial 43
4.6 Projeção de uma quádrica numa imagem em perspetiva 44
4.7 Projeção de uma hipótese de estado na imagem 46
4.8 Impacto central entre dois corpos 47
4.9 Ilustração do modelo de movimento incorporando colisões 49

5.1 Método de projeção da bola na imagem 51
5.2 Resultado da projeção da bola numa imagem 52
5.3 Exemplo de um marcador 54
5.4 Imagem do simulador 55
5.5 Trajetórias simuladas 57

5.6 Padrão de calibração axadrezado 58

6.1 Exemplo de gráficos de precisão 63
6.2 Tamanho das silhuetas 63
6.3 Gráficos de precisão da estimativa da posição 65
6.4 Gráficos de precisão da estimativa da velocidade 66
6.5 Gráfico das 100 estimativas de posição do PF 67
6.6 Gráficos de precisão da estimativa da posição 68
6.7 Gráficos de precisão da estimativa da velocidade 68
6.8 Gráficos de precisão da estimativa da posição e da velocidade sem o modelo de impacto 69
6.9 Gráficos de precisão da estimativa da posição e da velocidade com o modelo de impacto 70
6.10 Gráfico das estimativas da velocidade do PF 70
6.11 Testes para a trajetória pendular real 71
6.12 Testes para a trajetória em queda livre real 72
6.13 Testes de eficiência 73

Lista de Abreviações

2D 2 Dimensões. 40

3D 3 Dimensões. 1

EKF Filtro de *Kalman* Estendido. 11

EKPF Filtro de Partículas de *Kalman* Estendido. 24

GPU Unidade de Processamento Gráfico. 76

KF Filtro de *Kalman*. 9

PF Filtro de Partículas. 2

RAPID *Real-Time Attitude and Position Determination*. 28

RIS Reamostragem de Importância Sequencial. 21

SIFT *Scale Invariant Feature Transform*. 29

SUT *Scaled Unscented transformation*. 13

UKF *Unscented Kalman Filter*. 11

UPF *Unscented Particle Filter*. 2

VANT Veículos Aéreos Não Tripulados. 1

Capítulo 1

Introdução

Atualmente e cada vez mais, algoritmos que obtêm informação 3 Dimensões (3D) de objetos, apresentam uma elevada importância em aplicações em diversas áreas (desporto, área militar, robótica, medicina etc). No desporto, existem casos particulares em que o olho humano não consegue concluir se a bola passou ou não a linha de golo num jogo de futebol, ou se num jogo de ténis a bola ultrapassou a linha de fundo por exemplo. Para isso, tecnologias como o *Hawk-eye* [1] foram desenvolvidas. Esta tecnologia, para analisar se a bola ultrapassou a linha de golo, tem instalada sete câmaras de alta velocidade espalhadas pelo estádio apontadas para uma baliza. O sistema apresenta erros na estimação da bola apenas na ordem dos milímetros [1]. Também a análise de vídeos de desportos, tem sido cada vez mais uma área crescente de pesquisa, para recolher estatísticas de jogo, analisar táticas, analisar a movimentação dos jogadores e as suas decisões, etc [2].

Tem havido uma crescente investigação na área dos Veículos Aéreos Não Tripulados (VANT) para operações de reconhecimento e vigilância. Por exemplo, como os VANT são altamente portáveis, estes podem reunir informação de tráfego rodoviário em regiões problemáticas e de difícil acesso. *Yongzheng Xu et al.* [3] apresentam uma solução que deteta e segue os veículos nas estradas através de imagens aéreas com recurso aos VANT.

(a) *Hawk-eye* em funcionamento [1].

(b) Seguimento de veículos [3].

Figura 1.1: Algoritmos que utilizam informação 3D. Na imagem à esquerda (a), observa-se a localização 3D de uma bola utilizando a tecnologia *Hawk-eye*. À direita (b) é realizado o seguimento de veículos através de imagens aéreas.

Para realizar algoritmos que localizam e seguem objetos em 3D, são necessários dispositivos ca-

pazes de adquirir informação 3D ou dispositivos que obtenham dados que possam ser inferidos para obter informação 3D. Neste trabalho apenas se utiliza uma câmara que obtém dados cromáticos e a partir destes dados, obtém-se informação 3D de um objeto no espaço.

1.1 Objetivos e Abordagem ao Problema

O objetivo principal desta dissertação é o seguimento e estimação do estado de um objeto simples (neste caso uma bola) em 3D, utilizando apenas uma câmara RGB. De forma a realizar esta tarefa, serão desenvolvidos dois algoritmos. O primeiro utiliza o filtro de partículas como núcleo do algoritmo, o que corresponde ao trabalho realizado por *Taiana, M. et al.* [4], o segundo usa o *unscented particle filter* desenvolvido por *Van der Merwe et al.* [5], com o objetivo de melhorar a precisão da estimação do estado da bola, face ao anterior. Assim este trabalho reproduz uma extensão do trabalho de *Taiana, M. et al.* [4].

Perante o objetivo principal da dissertação, assume-se que o objeto é conhecido e assim tem-se acesso à sua forma, dimensão e cor homogénea. Os algoritmos são criados e otimizados apenas para o modelo do objeto de interesse. A arquitetura do primeiro algoritmo baseada no Filtro de Partículas (PF) e o segundo que corresponde a uma extensão do primeiro, o *Unscented Particle Filter* (UPF), representam cada partícula como uma hipótese do que poderá ser o estado 3D exato do objeto. De forma a testar a qualidade do seguimento destes dois filtros, foram criadas trajetórias num simulador e experiências reais para comparar o erro de estimação do estado do objeto. Assim, vão ser avaliados vários parâmetros para se perceber quais as vantagens e desvantagens destes filtros.

Como objetivo secundário, a intenção é criar algoritmos eficientes. Logo, para atingir este propósito serão usados recursos de uma biblioteca dedicada a processamento de imagem e visão computacional. No fim, a rapidez de estimação do objeto será comparada para ambos os filtros.

1.2 Contribuições

As contribuições principais desta dissertação são:

- Uma revisão teórica de abordagens e métodos de localização e seguimento 3D;
- Um simulador aplicando funcionalidades da biblioteca *OpenCV*, para criar imagens de uma bola a percorrer uma trajetória arbitrária. O código pode ser consultado em `https://github.com/linonil/Simulator`;
- Dois algoritmos desenvolvidos em C++, um tendo como base o filtro de partículas e o outro o *unscented particle filter*, para estimar a posição e velocidade em 3D de uma bola ao longo do tempo. Os códigos podem ser consultados em `https://github.com/linonil/PF` e `https://github.com/linonil/UPF`;

1.3 Estrutura da Dissertação

A presente dissertação encontra-se organizada em oito capítulos:

- Neste capítulo, encontra-se a introdução ao trabalho, no qual se enquadra a motivação, se enuncia os objetivos e a abordagem ao problema e se evidencia as contribuições deste trabalho.
- No segundo capítulo é realizada uma revisão teórica dos aspetos básicos que constituem esta dissertação. Assim, são descritos inicialmente conceitos sobre sistemas e filtros, e de uma forma mais pormenorizada, são explicados o filtro de *kalman*, o filtro de partículas e o *unscented particle filter*.
- No capítulo três, uma pesquisa do estado da arte no contexto de localização e seguimento 3D de objetos é efetuada. São evidenciados os principais sensores, abordagens e métodos utilizados em localização e seguimento 3D e referenciam-se alguns trabalhos desenvolvidos nesta área.
- O quarto capítulo expõe a metodologia usada na dissertação. É apresentado e explicado o trabalho de *Taiana, M. et al* [4] no qual este trabalho se baseia, especificando o modelo de predição e o modelo de observação. Posteriormente, são evidenciadas as arquitecturas dos dois algoritmos desenvolvidos e por fim, são descritos métodos fundamentais que estes algoritmos executam.
- No quinto capítulo, são explicados de uma forma sucinta os conceitos que foram implementados para avaliar os filtros no decorrer desta dissertação. Primeiro, é apresentado o simulador que foi desenvolvido para gerar imagens de uma bola a percorrer uma trajetória arbitrária no espaço num ambiente com obstáculos, com a finalidade de mais tarde testar a eficácia dos algoritmos. Seguidamente, é explicado como esses obstáculos são detetados e estimados, é enunciado como a câmara real foi calibrada e por fim de uma forma resumida, são abordados alguns conceitos que se implementaram para realizar os filtros.
- No sexto capítulo, são apresentados resultados de testes efetuados num simulador e experiências reais com uma bola, com o intuito de comparar e examinar a eficácia dos algoritmos no seguimento deste objeto. Depois de se avaliar a eficácia, serão mostrados os resultados da eficiência dos filtros, no qual se discriminam os processos menos e mais eficientes e se constata o tempo médio para se estimar o estado do objeto.
- No sétimo capítulo, são discutidos e concluídos os resultados obtidos e apresentadas algumas perspetivas para trabalhos futuros, de forma a continuar com esta investigação e contribuir para a resolução de alguns problemas que surgiram durante a realização desta dissertação.
- O último capítulo evidencia as referências bibliográficas usadas no decorrer da dissertação.

Capítulo 2

Fundamentação Teórica

Filtragem é o problema de estimar o estado de um sistema à medida que se obtém dados provenientes de sensores. Este é um problema de grande importância em várias áreas da ciência, engenharia e finanças. Para abordar este problema, é necessário modelar matematicamente a evolução do sistema e a incerteza nas medições observadas pelos sensores [5].

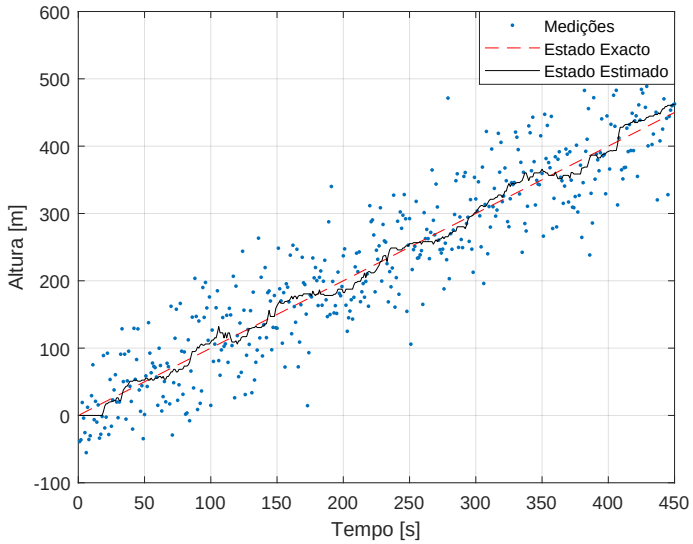


Figura 2.1: Exemplo de um problema de filtragem. Neste caso, um sensor mede com bastante incerteza, a altura de um objeto no ar ao longo do tempo. Constata-se que é possível obter uma estimativa aceitável da altura do objeto, a partir de medições ruidosas provenientes do sensor.

Modelos matemáticos são utilizados para representar sistemas. Contudo, nenhum modelo é exacto. Estes apenas representam uma abstração do mundo real e assim, apenas modelam parcialmente os processos físicos do meio envolvente, pelo que estes apresentam uma incerteza associada [6]. Por outro lado como já foi mencionado, também os sensores têm sempre ruído acoplado que se traduz em incerteza. A ideia chave de utilizar modelos probabilísticos é de representar incerteza explicitamente usando teoria de probabilidades. A informação é representada de uma forma matemática, utilizando distribuições de probabilidades sobre um espaço de hipóteses possíveis. Desta maneira, a estimativa do estado do sistema é representada por uma função de densidade de probabilidade [6]. Antes de

apresentar os filtros que se utilizaram para realizar este trabalho, serão explicados alguns conceitos gerais sobre sistemas e filtros.

2.1 Sistemas e Filtros

Um sistema é uma combinação de componentes que agem em conjunto para atingir um certo objetivo. Este é chamado de dinâmico se o estado deste variar ao longo do tempo. Um sistema dinâmico em termos matemáticos é representado através de um modelo, sendo este modelo definido por um conjunto de equações que representam a dinâmica do sistema [7].

2.1.1 Estado de um Sistema

O estado de um sistema é um conjunto de informação quantitativa, no qual corresponde ao menor número de dados que é necessário conhecer acerca do seu comportamento passado, com o intuito de predizer o seu comportamento no futuro [8]. Na forma de um modelo matemático, esta informação quantitativa representa um conjunto de variáveis (denominadas de variáveis de estado), de aspetos do próprio sistema e do seu meio envolvente, que podem criar impacto no futuro [6]. É definido baseado naquilo que é importante para atingir um determinado objetivo. O estado pode-se representar da seguinte maneira:

$$\mathbf{x}_t = \begin{bmatrix} x_1 & x_2 & ... & x_n \end{bmatrix}^T \tag{2.1}$$

onde $\mathbf{x}_t$ representa o estado no tempo t e as n componentes do vetor, correspondem às respetivas variáveis de estado. Ao espaço n dimensional que corresponde a todos os estados possíveis que um sistema pode tomar, chama-se espaço de estados. É de notar, que estas variáveis não necessitam de ser fisicamente mensuráveis ou quantidades observáveis. Por exemplo, num contexto de seguimento de um objeto, pode-se ter como variáveis de estado a posição e a velocidade do objeto mas apenas obter medidas de posição no meio ambiente. Sendo a velocidade uma variável oculta ou não observável, esta pode ser inferida pelas medições observáveis dos sensores que o sistema utiliza [6].

O estado deve ser entendido como sendo o valor exato do sistema. Este valor é normalmente desconhecido. Dando um exemplo, num problema em que seja necessário conhecer o peso de um objeto (neste caso o peso é o estado do sistema), pode-se utilizar uma balança para obter medições do peso desse objeto. Diz-se que estas medições são observáveis visto que é possível observar diretamente as medidas. Pelo contrário, não é possível observar o peso, mas apenas o estimar através das medições obtidas pela balança [9]. Os filtros são utilizados neste tipo de problemas com o propósito de obter estimativas fidedignas do estado ao longo do tempo.

2.1.2 Filtro de *Bayes*

Filtragem ótima remete à classe de métodos que podem ser utilizados para estimar o estado de uma sistema dinâmico, que é indiretamente observado através de medições ruidosas de um sensor. O termo

ótimo neste contexto, refere-se a otimização estatística. Filtragem de *Bayes* é o conceito de formular um método de filtragem ótima, utilizando a regra de *Bayes*. Desta maneira, pretende-se estimar todos os valores que o estado toma desde o estado inicial $\mathbf{x}_0$ até ao estado final $\mathbf{x}_T$. Na perspetiva de *Bayes*, corresponde a calcular a distribuição conjunta *a posteriori* de todos os estados, condicionado por todas as observações. Em princípio isto pode ser realizado aplicando a regra de *Bayes*:

$$p(\mathbf{x}_{0:T}|\mathbf{z}_{1:T}) = \frac{p(\mathbf{z}_{1:T}|\mathbf{x}_{0:T})p(\mathbf{x}_{0:T})}{p(\mathbf{z}_{1:T})} \tag{2.2}$$

- $p(\mathbf{x}_{0:T}) = p(\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_T)$ é a probabilidade conjunta que corresponde à distribuição *a priori* definida pelo modelo dinâmico do sistema.

- $p(\mathbf{z}_{1:T}|\mathbf{x}_{0:T}) = p(\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_T|\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_T)$ é conhecido por modelo de observação das medições. É expresso como a probabilidade conjunta de todas as medições conhecendo a distribuição de todos os estados.

- $p(\mathbf{z}_{1:T}) = \int p(\mathbf{z}_{1:T}|\mathbf{x}_{0:T})p(\mathbf{x}_{0:T})d\mathbf{x}_{0:T}$ representa a constante de normalização [10].

Infelizmente, esta formulação da distribuição *a posteriori* total tem a séria desvantagem que de cada vez que é obtido uma nova medição, é necessário recalcular a distribuição e à medida que o tempo evolui, a dimensão da distribuição aumenta, o que leva eventualmente a que os cálculos se tornem impraticáveis. Visto que estimar a distribuição *a posteriori* total é inconveniente, foi desenvolvido o filtro de *Bayes* com o intuito de estimar o estado de um sistema ao longo do tempo de uma forma recursiva. Este filtro impõe a restrição de que os modelos dinâmicos necessitam de ser cadeias estocásticas de *Markov*. Se o sistema for uma cadeia de *Markov*, apenas é necessário estimar a distribuição *a posteriori* marginal do estado atual $\mathbf{x}_t$, isto é, apenas se calcula $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ cada vez que o filtro é iterado [10].

Propriedades de *Markov*

A evolução do estado e das observações dos sensores, são governados por leis probabilísticas pelo que o estado é normalmente gerado estocasticamente. A distribuição probabilística que caracteriza a evolução do estado pode ser expressa como:

$$p(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t-1}). \tag{2.3}$$

A fórmula acima indica que para calcular o estado atual $\mathbf{x}_t$ é necessário conhecer todos os estados anteriores $\mathbf{x}_{0:t-1} = \mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{t-1}$ e medições anteriores $\mathbf{z}_{1:t-1} = \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_{t-1}$. Felizmente, se o processo estocástico for uma cadeia de *Markov*, a propriedade de *Markov* dos estados demonstra que o estado atual $\mathbf{x}_t$ conhecendo o estado anterior $\mathbf{x}_{t-1}$, é independente das restantes variáveis anteriores. Deste modo a fórmula resume-se a:

$$p(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{2.4}$$

Também é possível modelar o processo no qual as observações estão a ser geradas e como todas as observações são condicionalmente independentes, a fórmula resulta em:

$$p(\mathbf{z}_t|\mathbf{x}_{0:t}, \mathbf{z}_{1:t-1}) = p(\mathbf{z}_t|\mathbf{x}_t). \tag{2.5}$$

A probabilidade $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ é conhecida por probabilidade de transição de estado e específica como o estado evolui ao longo do tempo. A probabilidade $p(\mathbf{z}_t|\mathbf{x}_t)$ é chamada de probabilidade de observação e especifica a lei probabilística no qual as observações **z** são geradas do estados **x**. São estas duas probabilidades que em conjunto, caracterizam a dinâmica de um sistema dinâmico estocástico [6]. A arquitetura do filtro de *Bayes* pode ser expressa pela seguinte cadeia oculta de *Markov*:

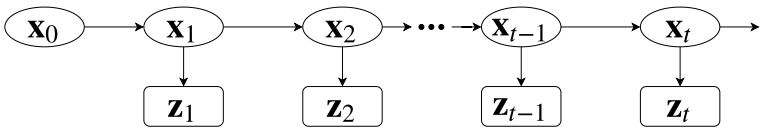


Figura 2.2: Processo estocástico que segue uma cadeia de *Markov*. Neste caso, a sequência dos estados ocultos $\mathbf{x}_0, ..., \mathbf{x}_t$ representados pelas elipses, são indiretamente observados pelas medições $\mathbf{z}_0, ..., \mathbf{z}_t$ dos retângulos.

Para iniciar o filtro de *Bayes* é necessário fornecer uma estimativa do estado inicial $\mathbf{x}_0 = p(\mathbf{x}_0)$, de seguida, a estimativa dos estados seguintes já é calculada a partir das observações.

Algoritmo do filtro de *Bayes*

Algoritmo 1: $p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \text{BAYES}(p(\mathbf{x}_t|\mathbf{z}_{1:t-1}), \mathbf{z}_t)$

Para $t = 0$**:**

1 $p(\mathbf{x}_0)$

Para $t \geq 1$**:**

2 $p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) d\mathbf{x}_{t-1}$

3 $p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \eta\, p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$

O algoritmo do filtro de *Bayes* executa em cada iteração duas etapas essenciais, o modelo de predição (linha 2) e o modelo de observação (linha 3). O filtro necessita que seja inicializado com uma distribuição inicial do estado $p(\mathbf{x}_0)$ (linha 1). Depois, a cada iteração basta fornecer a medição atual $\mathbf{z}_t$, que o algoritmo calcula a distribuição *a posteriori* do estado corrente $\mathbf{x}_t$. Abaixo são explicadas as etapas que constituem o filtro.

- **Modelo de Predição** - É utilizada a probabilidade de transição de estado também denominada por modelo do processo $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, com o intuito de predizer o valor atual do estado antes de incorporar a medição corrente $\mathbf{z}_t$. Com este passo, obtém-se a distribuição de predição $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$.
- **Modelo de Observação** - Este último passo incorpora a observação atual $\mathbf{z}_t$ na estimação, atuando como uma correção ao passo de predição. É aplicada a probabilidade de observação $p(\mathbf{z}_t|\mathbf{x}_t)$, para executar a regra de *Bayes* e assim obtém-se a distribuição *a posteriori* atual do

estado $p(\mathbf{x}_t|\mathbf{z}_{1:t})$. A constante $\eta = \int p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})d\mathbf{x}_t$ é a constante de normalização da regra de *Bayes*.

Apesar do filtro de *Bayes* reduzir drasticamente o número de cálculos necessários para estimar o estado, as equações acima não podem ser implementadas nesta forma. Em particular, é necessário realizar o integral e a multiplicação em forma fechada ou é necessário restringirmos o espaços de estados, para tornar o integral finito. Existem diversas técnicas e algoritmos que são derivados do filtro de *Bayes*. Cada técnica depende de diferentes premissas a respeito das probabilidades de transição de estado, de observação de estado e da distribuição inicial. Essas premissas criam diferentes tipos de distribuições *a posteriori* e os algoritmos que as calculam ou as aproximam, apresentam diferentes características computacionais [6]. Nas secções seguintes serão abordados alguns filtros derivados deste e no final comparam-se as propriedades de cada filtro mencionado.

2.2 Filtro de *Kalman*

O Filtro de *Kalman* (KF) é um estimador recursivo que deriva do filtro de *Bayes*. Este filtro foi desenhado para lidar apenas com problemas de filtragem linear gaussiana, pelo que representa as probabilidades *a posteriori* através de funções *gaussianas*, assim como a estimativa inicial do estado necessita de ser gaussiana $p(\mathbf{x}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{P}_0)$. A restrição de apenas lidar com distribuições gaussianas, trás a vantagem de implementar o filtro de *Bayes* numa forma fechada em espaços contínuos, sem utilizar nenhuma aproximação, tornando o algoritmo exequível [6]. Assim, a estimativa atual do estado é representada pelo dois parâmetros que caracterizam a distribuição gaussiana, a média $\boldsymbol{\mu}_t$ e covariância $\mathbf{P}_t$ [10]:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t, \mathbf{P}_t) \tag{2.6}$$

2.2.1 Passo de predição

No filtro de *Kalman*, a probabilidade de transição de estado $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ é expressa da seguinte forma:

$$\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{q}_t, \quad \mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t). \tag{2.7}$$

A equação acima é utilizada como modelo do sistema. Neste filtro, o modelo do processo é uma função linear com a adição de ruído gaussiano. A matriz $\mathbf{F}_t$, denominada por matriz de transição modela a evolução do sistema. Associado ao modelo de predição, existe a variável aleatória $\mathbf{q}_t$, que modela a incerteza da transição do estado. Esta variável segue uma distribuição normal com média nula e covariância $\mathbf{Q}_t$ conhecida por ruído do processo. As duas equações abaixo propagam a estimativa do estado antes de integrar a medição $\mathbf{z}_t$ e assim, obtém-se a distribuição de predição $p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \mathcal{N}(\mathbf{x}_t|\bar{\boldsymbol{\mu}}_t, \bar{\mathbf{P}}_t)$. A primeira equação propaga a média e a segunda a covariância (as barras acima das

variáveis indicam que a distribuição é de predição) [9][10].

$$\bar{\boldsymbol{\mu}}_t = \mathbf{F}_t \boldsymbol{\mu}_{t-1} \tag{2.8}$$

$$\bar{\mathbf{P}}_t = \mathbf{F}_t \mathbf{P}_{t-1} \mathbf{F}_t^T + \mathbf{Q}_t \tag{2.9}$$

2.2.2 Passo de observação

A probabilidade de observação $p(\mathbf{z}_t|\mathbf{x}_t)$, é dada por:

$$\mathbf{z}_t = \mathbf{H}_t \mathbf{x}_t + \mathbf{r}_t, \quad \mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t). \tag{2.10}$$

Esta corresponde a uma função linear, com a adição de um ruído gaussiano [6]. A matriz $\mathbf{H}_t$ designada por matriz de observação, transforma o valor do estado para o espaço de observação. Todas as medidas adquiridas por sensores são afetadas por ruído. Para modelar a incerteza da observação $\mathbf{z}_t$ gerada pelos sensores, existe a variável aleatória $\mathbf{r}_t$, que segue uma distribuição normal com média nula e covariância $\mathbf{R}_t$, designada por ruído de observação. Este último passo do filtro representado pelas três equações:

$$\mathbf{K}_t = \bar{\mathbf{P}}_t \mathbf{H}_t^T (\mathbf{H}_t \bar{\mathbf{P}}_t \mathbf{H}_t^T + \mathbf{R}_t)^{-1} \tag{2.11}$$

$$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t (\mathbf{z}_t - \mathbf{H}_t \bar{\boldsymbol{\mu}}_t) \tag{2.12}$$

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \bar{\mathbf{P}}_t \tag{2.13}$$

incorpora a observação corrente na estimação. A matriz $\mathbf{K}_t$ apresentada em (2.11) denominada por ganho de *Kalman*, é um fator que calcula um rácio entre a covariância do modelo de predição e a covariância do modelo de observação. O objetivo deste ganho, é de ponderar a importância que a observação $\mathbf{z}_t$ trás para a estimativa do estado. Quanto menor for a magnitude da covariância, mais confiança o filtro terá na estimativa desse modelo. Se a covariância da observação $\mathbf{R}_t$ for muito grande comparando com a covariância da estimativa de predição $\bar{\mathbf{P}}_t$, o ganho será baixo, ignorando em grande quantidade o valor da observação e dando mais relevância ao modelo do processo. Pelo contrário, se a covariância do modelo de observação for baixo, o ganho dará mais relevância às medições, que irão corrigir o modelo do processo. O ganho é de seguida multiplicado pelo residual entre a observação real $\mathbf{z}_t$ e a estimativa da observação $\mathbf{H}_t \bar{\boldsymbol{\mu}}_t$ e é adicionado à predição do estado, o que resulta em $p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t, \mathbf{P}_t)$. Por fim, é atualizada a covariância da estimativa [9, 10]. Na figura 2.3 observam-se os dois passos do filtro. O algoritmo deste filtro pode ser consultado em A.1.

2.3 Abordagem do Filtro de *Kalman* para Problemas Não-Lineares

Uma das grandes desvantagens do filtro de *Kalman* é só poder manipular problemas lineares, o que representa uma grande limitação na sua utilidade perante as não linearidades existentes no mundo [9]. As premissas que a probabilidade de transição de estado e probabilidade de observação, sejam ambos

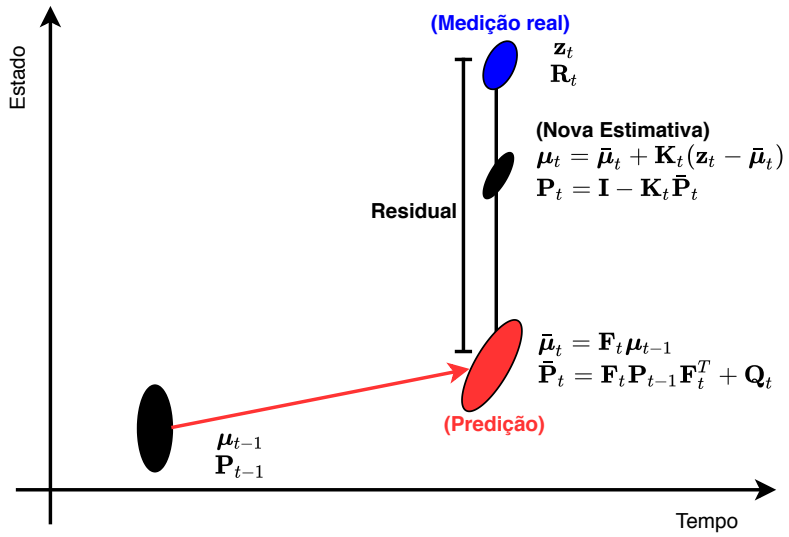


Figura 2.3: Ilustração dos passos de predição e observação do filtro de *Kalman*. Neste caso, a matriz **H** corresponde à matriz identidade e as elipses representam a média e covariância da estimativa. O ganho de *Kalman* escolhe um valor ao longo do residual entre a predição e a medição real $\mathbf{z}_t$ [9].

lineares com a adição de ruído gaussiano são raramente cumpridas em problemas de estimação reais. Por exemplo, um robô que percorre uma trajetória circular não pode ser descrito usando um modelo de predição linear. Este argumento e o facto do filtro de *Kalman* ser unimodal demonstra que o filtro apenas pode ser aplicado a problemas de estimação triviais [6]. Com o intuito de estender a utilidade do KF para problemas não lineares, foram concebidos filtros como o Filtro de *Kalman* Estendido (EKF) e o *Unscented Kalman Filter* (UKF). Ambos superam a premissa de não-linearidade, assumindo que as probabilidades de transição de estado e de observação são governadas por funções não-lineares [6]. Estes filtros apesar de terem como base o filtro de *Kalman* representam uma aproximação ao filtro de *Bayes*, pois aproximam os modelos não lineares das distribuições através da distribuição gaussiana. Ou seja, os modelos de predição e observação podem ser não lineares mas as estimativas são aproximadas e representadas da mesma maneira que no filtro de *Kalman*, $p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t, \mathbf{P}_t)$ [10].

2.3.1 Filtro de *Kalman* Estendido

O filtro de *Kalman* estendido foi uma das primeiras técnicas usadas para problemas não-lineares e continua sendo a técnica mais comum [9]. As probabilidades de transição de estado (2.7) e de observação (2.10) num modelo não-linear, assumindo que os ruídos do processo e de observação são não aditivos, podem ser escritas da seguinte forma [10]:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{q}_t), \quad \mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t) \tag{2.14}$$

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{r}_t), \quad \mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t). \tag{2.15}$$

A abordagem do EKF para lidar com problemas não-lineares é bastante intuitiva. A ideia-chave é o conceito de linearização. Supondo que o modelo do processo f é não-linear, a projeção de uma gaussiana através desta função, resulta tipicamente numa função que não é gaussiana. Linearização consiste em aproximar uma função não-linear por uma função linear tangente a f em torno de um ponto. Através desta aproximação linear, a projeção da gaussiana resulta numa gaussiana. O mesmo argumento aplica-se ao modelo de observação h. Desta maneira com o sistema linearizado, a mecânica de estimação do estado é equivalente ao filtro de *Kalman* [6, 9]. Existem diversas técnicas para linearizar funções não-lineares. O EKF usa o método de expansão em série de *Taylor* de primeira ordem em torno de um ponto. Ou seja, uma aproximação linear local é construída a partir do valor e do declive da função não-linear, onde o declive é calculado a partir das derivadas parciais de primeira ordem nesse ponto (na figura 2.4 pode-se observar como funciona o método). As probabilidades de transição de estado e observação são aproximadas da seguinte maneira [11]:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{q}_t) \approx f(\boldsymbol{\mu}_{t-1}, \mathbf{0}) + \mathbf{F}_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) + \mathbf{G}_t\mathbf{q}_t \tag{2.16}$$

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{r}_t) \approx h(\bar{\boldsymbol{\mu}}_t, \mathbf{0}) + \mathbf{H}_t(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t) + \mathbf{U}_t\mathbf{r}_t \tag{2.17}$$

onde $\mathbf{F}_t(i,j) = \dfrac{\partial f(i)}{\partial \mathbf{x}_{t-1}(j)}(\mathbf{x}_{t-1}, \mathbf{q}_t)\Big|_{\mathbf{x}_{t-1}=\boldsymbol{\mu}_{t-1}, \mathbf{q}_t=0}$ corresponde à matriz jacobiana das derivadas parciais de f em ordem a $\mathbf{x}_{t-1}$ e $\mathbf{G}_t(i,j) = \dfrac{\partial f(i)}{\partial \mathbf{q}_t(j)}(\mathbf{x}_{t-1}, \mathbf{q}_t)\Big|_{\mathbf{x}_{t-1}=\boldsymbol{\mu}_{t-1}, \mathbf{q}_t=0}$ representa a matriz jacobiana das derivadas parciais de f em ordem a $\mathbf{q}_t$, em que ambas são avaliadas em $\mathbf{x}_{t-1} = \boldsymbol{\mu}_{t-1}$ e $\mathbf{q}_t = 0$. E $\mathbf{H}_t(i,j) = \dfrac{\partial h(i)}{\partial \mathbf{x}_t(j)}(\mathbf{x}_t, \mathbf{r}_t)\Big|_{\mathbf{x}_t=\bar{\boldsymbol{\mu}}_t, \mathbf{r}_t=0}$ corresponde à matriz jacobiana das derivadas parciais de h em ordem a $\mathbf{x}_t$ e $\mathbf{U}_t(i,j) = \dfrac{\partial h(i)}{\partial \mathbf{r}_t(j)}(\mathbf{x}_t, \mathbf{r}_t)\Big|_{\mathbf{x}_t=\bar{\boldsymbol{\mu}}_t, \mathbf{r}_t=0}$ representa a matriz jacobiana das derivadas parciais de h em ordem a $\mathbf{r}_t$, em que ambas são avaliadas em $\mathbf{x}_t = \bar{\boldsymbol{\mu}}_t$ e $\mathbf{r}_t = 0$ [11]. A simplicidade do método de linearização permite que a mecânica de estimação do filtro seja equivalente à do filtro de *Kalman* e por isso, o algoritmo do EKF (representado em A.2) é bastante parecido com o KF (A.1) [9].

Equações do Filtro de *Kalman* Estendido

Com a linearização realizada às duas equações acima, para este modelo não-linear, as equações do passo de predição são dadas por:

$$\bar{\boldsymbol{\mu}}_t = f(\boldsymbol{\mu}_{t-1}, \mathbf{0}) \tag{2.18}$$

$$\bar{\mathbf{P}}_t = \mathbf{F}_t\mathbf{P}_{t-1}\mathbf{F}_t^T + \mathbf{G}_t\mathbf{Q}_t\mathbf{G}_t^T \tag{2.19}$$

e as do passo de observação resulta em:

$$\mathbf{K}_t = \bar{\mathbf{P}}_t\mathbf{H}_t^T(\mathbf{H}_t\bar{\mathbf{P}}_t\mathbf{H}_t^T + \mathbf{U}_t\mathbf{R}_t\mathbf{U}_t^T)^{-1} \tag{2.20}$$

$$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t\left(\mathbf{z}_t - \mathbf{h}(\bar{\boldsymbol{\mu}}_t, \mathbf{0})\right) \tag{2.21}$$

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\bar{\mathbf{P}}_t. \tag{2.22}$$

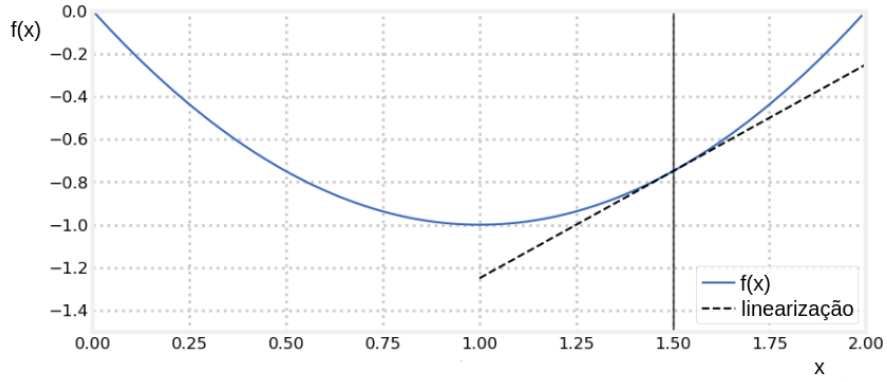


Figura 2.4: Nesta figura, uma função não-linear é linearizada num certo ponto obtendo-se a função linear representada a tracejado [9].

2.3.2 *Unscented Kalman Filter*

A abordagem do UKF às não-linearidades é bastante diferente da abordagem do EKF. Enquanto que no EKF são aplicadas aproximações lineares, no UKF as aproximações são baseadas na *scaled unscented transform*. Neste filtro a distribuição dos estados é também representada por uma variável aleatória gaussiana como no filtro de *Kalman*, mas é especificada utilizando um mínimo conjunto de amostras escolhidas de forma determinística. Estas amostras capturam completamente a média e covariância da variável aleatória gaussiana e quando propagadas pelo sistema não linear, capturam a média e covariância da distribuição de forma precisa, até à segunda ordem de qualquer não-linearidade. Apenas são introduzidos erros para a terceira ou maiores ordens. De maneira a clarificar o funcionamento do filtro, a *scaled unscented transformation*, será explicada [5, 12].

Scaled Unscented Transformation

A *Scaled Unscented transformation* (SUT) é um método para calcular estatísticas de uma variável aleatória que sofre uma transformação não linear. Considerando que se quer propagar uma variável aleatória **x**, através de uma função não-linear arbitrária f, pretende-se calcular as estatísticas depois de aplicar a função (ver figura 2.5). Deste modo, fazendo,

$$\mathbf{y} = f(\mathbf{x}) \tag{2.23}$$

pretende-se calcular as estatísticas de **y**. Assumindo que **x** apresenta como média $\boldsymbol{\mu}_x$ e covariância $\mathbf{P}_x$. Para calcular as estatísticas de **y** utilizando o método, um conjunto de amostras ponderadas, denominadas por pontos sigma, de tamanho $2n_x + 1$, em que n_x representa a dimensão da variável aleatória **x**, é deterministicamente escolhido para extrair a média e covariância de **x**. A cada ponto sigma i, estão associados dois pesos, $w_i^{(m)}$ para a média e $w_i^{(c)}$ para a covariância. Este conjunto de

pontos sigma $\mathbf{S} = \{\boldsymbol{\chi}, \mathbf{w}^{(m)}, \mathbf{w}^{(c)}\}$, deve satisfazer o seguinte esquema de seleção:

$$\lambda = \alpha^2(n_x + k) - n_x \tag{2.24}$$

$$\boldsymbol{\chi}_1 = \boldsymbol{\mu}_x \tag{2.25}$$

$$\boldsymbol{\chi}_i = \boldsymbol{\chi}_1 + \left(\sqrt{(n_x + \lambda)\mathbf{P}_x}\right)_i \qquad i = 2,\ \ldots, n_x + 1 \tag{2.26}$$

$$\boldsymbol{\chi}_i = \boldsymbol{\chi}_1 - \left(\sqrt{(n_x + \lambda)\mathbf{P}_x}\right)_i \qquad i = n_x + 2,\ \ldots, 2n_x + 1 \tag{2.27}$$

$$w_1^{(m)} = \frac{\lambda}{n_x + \lambda} \tag{2.28}$$

$$w_1^{(c)} = \frac{\lambda}{n_x + \lambda} + 1 - \alpha^2 + \beta \tag{2.29}$$

$$w_i^{(m)} = w_i^{(c)} = \frac{1}{2(n_x + \lambda)} \qquad i = 2,\ \ldots, 2n_x + 1 \tag{2.30}$$

o termo $\left(\sqrt{(n_x + \lambda)\mathbf{P}_x}\right)_i$ corresponde à coluna i da raiz quadrada de $(n_x + \lambda)\mathbf{P}_x$. Os valores k, α e β, são constantes que controlam a disposição e o peso dos pontos sigma. O valor k deve ser escolhido sendo $k \geq 0$ para garantir que a matriz de covariância é positiva semi-definida o que simplifica o cálculo da raiz quadrada da matriz. Como este é um parâmetro que não é crítico, costuma ser colocado a zero. Os outros parâmetros devem ser escolhidos no intervalo, $0 \leq \alpha \leq 1$ e $\beta \geq 0$. O parâmetro α controla o espaçamento dos pontos *sigma* e deve idealmente, ser um número pequeno, de forma a amostrar zonas de grande não-linearidade. O parâmetro β é um termo não negativo e é utilizado para controlar o erro de curtose da distribuição não linear. Se **x** for uma distribuição gaussiana, o parâmetro ideal é $\beta = 2$ [5]. Cada ponto sigma é agora propagado pela função não-linear:

$$\boldsymbol{\mathcal{Y}}_i = f(\boldsymbol{\chi}_i) \qquad i = 1,\ \ldots, 2n_x + 1 \tag{2.31}$$

e finalmente, as estimativas da média e da covariância de **y** são calculadas como:

$$\boldsymbol{\mu}_y = \sum_{i=1}^{2n_x+1} w_i^{(m)} \boldsymbol{\mathcal{Y}}_i \tag{2.32}$$

$$\mathbf{P}_y = \sum_{i=1}^{2n_x+1} w_i^{(c)} (\boldsymbol{\mathcal{Y}}_i - \boldsymbol{\mu}_y)(\boldsymbol{\mathcal{Y}}_i - \boldsymbol{\mu}_y)^T. \tag{2.33}$$

Equações do *Unscented Kalman Filter*

Para implementar o UKF o estado é redefinido para um estado aumentado, concatenando o estado original com as variáveis de ruído. Desta maneira, é possível gerar os pontos sigma para o passo de predição e observação numa só operação [10]:

$$\boldsymbol{\chi}_{t-1}^a = \left[\boldsymbol{\mu}_{t-1}^a \quad \boldsymbol{\mu}_{t-1}^a \pm \left(\sqrt{(n_a + \lambda_a)\mathbf{P}_{t-1}^a}\right)_i\right] \tag{2.34}$$

$$\tag{2.35}$$

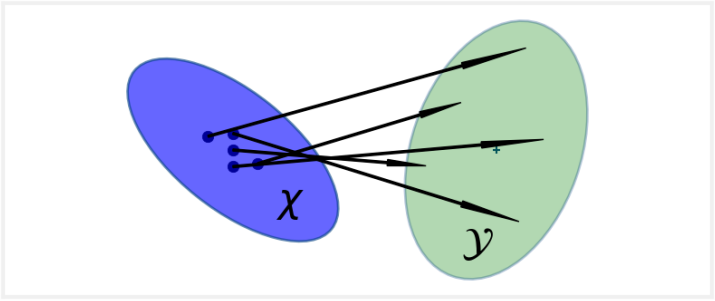


Figura 2.5: Nesta figura, a elipse a azul corresponde à distribuição antes da transformação não linear e a elipse a verde corresponde à distribuição depois da transformação [9].

onde $\boldsymbol{\mu}^a_{t-1} = \begin{bmatrix} \boldsymbol{\mu}_{t-1} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$, $\mathbf{P}^a_{t-1} = \begin{bmatrix} \mathbf{P}_t & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_t \end{bmatrix}^T$, $\lambda_a = \alpha^2(n_a + k) - n_a$ e $n_a = n_x + n_q + n_r$, no qual n_x, n_q e n_r, correspondem respetivamente à dimensão do estado, do ruído do processo e do ruído de observação. Com os pontos sigma gerados, as equações de predição são as seguintes:

$$\boldsymbol{\mathcal{Y}}_t = f\left(\boldsymbol{\mathcal{X}}^x_{t-1}, \boldsymbol{\mathcal{X}}^q_{t-1}\right) \tag{2.36}$$

$$\bar{\boldsymbol{\mu}}_t = \sum_{i=1}^{2n+1} w_i^{(m)} \boldsymbol{\mathcal{Y}}_{i,t} \tag{2.37}$$

$$\bar{\mathbf{P}}_t = \sum_{i=1}^{2n+1} w_i^{(c)} (\boldsymbol{\mathcal{Y}}_{i,t} - \bar{\boldsymbol{\mu}}_t)(\boldsymbol{\mathcal{Y}}_{i,t} - \bar{\boldsymbol{\mu}}_t)^T. \tag{2.38}$$

$\boldsymbol{\mathcal{X}}^x_{t-1}$ denota as n_x primeiras linhas, $\boldsymbol{\mathcal{X}}^q_{t-1}$ as n_q linhas seguintes e $\boldsymbol{\mathcal{X}}^r_{t-1}$ as últimas linhas de $\boldsymbol{\mathcal{X}}^a_{t-1}$, e os fatores $w_i^{(m)}$ e $w_i^{(c)}$ são definidos em função de n_a e λ_a. Os pontos sigma são propagados pela função não-linear f e depois é calculado a média e a covariância da estimativa de predição. O KF integra na estimativa a medição $\mathbf{z}_t$ no espaço de observação. Deste modo, convertem-se os pontos sigma para o espaço de observação, através da função h. De seguida, são calculadas a média e covariância no espaço de observação para obter o ganho de *Kalman* e atualiza-se a estimativa do estado *a posteriori* com a influência da observação corrente $\mathbf{z}_t$. Obtêm-se assim, as equações do passo de observação (o algoritmo deste filtro pode ser consultado em A.3) [9]:

$$\boldsymbol{\mathcal{Z}}_t = h(\boldsymbol{\mathcal{Y}}_t, \boldsymbol{\mathcal{X}}^r_{t-1}) \tag{2.39}$$

$$\boldsymbol{\mu}_z = \sum_{i=1}^{2n+1} w_i^{(m)} \boldsymbol{\mathcal{Z}}_{i,t} \tag{2.40}$$

$$\mathbf{P}_z = \sum_{i=1}^{2n+1} w_i^{(c)} (\boldsymbol{\mathcal{Z}}_{i,t} - \boldsymbol{\mu}_z)(\boldsymbol{\mathcal{Z}}_{i,t} - \boldsymbol{\mu}_z)^T \tag{2.41}$$

$$\mathbf{K}_t = \left(\sum_{i=1}^{2n+1} w_i^{(c)} (\boldsymbol{\mathcal{Y}}_{i,t} - \bar{\boldsymbol{\mu}}_t)(\boldsymbol{\mathcal{Z}}_{i,t} - \boldsymbol{\mu}_z)^T \right) \mathbf{P}_z^{-1} \tag{2.42}$$

$$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - \boldsymbol{\mu}_z) \tag{2.43}$$

$$\mathbf{P}_t = \bar{\mathbf{P}}_t - \mathbf{K}_t \mathbf{P}_z \mathbf{K}_t^T. \tag{2.44}$$

2.4 Filtro de Partículas

Uma popular alternativa às técnicas de filtragem gaussianas são os filtros não-paramétricos. Filtros deste tipo, não necessitam que o formato da densidade de probabilidade *a posteriori* seja um funcional fixo, como é o caso das gaussianas, que apresentam sempre uma função exponencial para definir a densidade da distribuição normal. O seu objetivo principal é estimar a distribuição *a posteriori* das variáveis de estado dadas as variáveis de observação, utilizando um número finito de amostras. Esta representação é aproximada e não-paramétrica, logo, pode ser utilizada para representar uma grande quantidade de distribuições de probabilidades. No PF, as amostras são denominadas por partículas e cada partícula possui um peso denominado por fator de importância. Estas são descritas como:

$$\boldsymbol{\chi}_t = \{\mathbf{x}_t, \mathbf{w}_t\} = \left\{\{x_t^{(1)}, w_t^{(1)}\}, \{x_t^{(2)}, w_t^{(2)}\}..., \{x_t^{(N)}, w_t^{(N)}\}\right\} \tag{2.45}$$

sendo N o número de partículas geradas, cada partícula $x_t^{(i)}$, representa uma hipótese sobre o que pode ser o estado actual do sistema, no tempo t [6]. Para melhor compreender o funcionamento do filtro de partículas, são explicados os conceitos em que o filtro se baseia tais como o método de simulação de *Monte Carlo*, amostragem de importância sequencial e o processo de seleção de partículas. Finalmente são apresentados os passos que constituem o filtro, o passo de predição, o passo de observação e o passo de seleção.

2.4.1 Simulação de *Monte Carlo*

Em simulação de *Monte Carlo*, se for possível gerar N amostras aleatórias independentes e identicamente distribuídas provenientes da distribuição $p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$, uma estimativa empírica da distribuição pode ser aproximada por:

$$p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) \approx \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}_{0:t} - \mathbf{x}_{0:t}^{(i)}) \tag{2.46}$$

onde $\delta(\cdot)$ denota a função delta de *dirac* e $\mathbf{x}_{0:t}^{(i)} \sim p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}), i = 1, ..., N$, são as amostras aleatórias. Consequentemente, qualquer valor expectável da forma

$$E[f(\mathbf{x}_{0:t})|\mathbf{z}_{1:t}] = \int f(\mathbf{x}_{0:t})p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})d\mathbf{x}_{0:t} \tag{2.47}$$

pode ser aproximado pela estimativa seguinte:

$$\hat{E}[f(\mathbf{x}_{0:t})|\mathbf{z}_{1:t}] \approx \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_{0:t}^{(i)}). \tag{2.48}$$

A estimativa é não-enviesada e de acordo com a lei dos grandes números, à medida que o número de amostras aumenta, a variância do erro de aproximação diminui [13][14][5]. Desta maneira é possível estimar a distribuição *a posteriori* total $p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$, utilizando o método de simulação de *Monte Carlo*. Porém, frequentemente esta distribuição é desconhecida ou bastante difícil de amostrar e assim, o

método de amostragem de importância é aplicado.

2.4.2 Amostragem de Importância Sequencial

Amostragem de importância sequencial deriva da técnica de amostragem de importância. Esta é uma técnica utilizada para estimar as propriedades de uma distribuição alvo, utilizando apenas amostras, mas extraídas de uma distribuição diferente e conhecida. A ideia é simples, as amostras são extraídas de uma distribuição de probabilidade conhecida denominada por distribuição de importância, mas a cada amostra está associado um peso e cada peso é avaliado de acordo com a distribuição de interesse. Com isto é possível estimar empiricamente, propriedades como a média ponderada e variância ponderada das amostras [9].

Amostragem de importância baseia-se na seguinte decomposição do valor espectável:

$$\begin{aligned} E[f(\mathbf{x}_{0:t})|\mathbf{z}_{1:t}] &= \int f(\mathbf{x}_{0:t})p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})d\mathbf{x}_{0:t} \\ &= \int \Big[f(\mathbf{x}_{0:t})\frac{p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})}\Big]q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})d\mathbf{x}_{0:t} \end{aligned} \tag{2.49}$$

sendo a distribuição de importância definida como $q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$. Se a condição seguinte for válida para todos os valores possíveis:

$$p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) > 0 \implies q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) > 0 \tag{2.50}$$

pode-se criar uma aproximação de *Monte Carlo* amostrando da distribuição proposta:

$$E[f(\mathbf{x}_{0:t})|\mathbf{z}_{1:t}] \approx \frac{1}{N}\sum_{i=1}^{N}\frac{p(\mathbf{x}_{0:t}^{(i)}|\mathbf{z}_{1:t})}{q(\mathbf{x}_{0:t}^{(i)}|\mathbf{z}_{1:t})}f(\mathbf{x}_{0:t}^{(i)}) = \frac{1}{N}\sum_{i=1}^{N}w^{(i)}f(\mathbf{x}_{0:t}^{(i)}) \tag{2.51}$$

onde os valores $w^{(i)}$ são conhecidos como os fatores de importância:

$$w^{(i)} = \frac{p(\mathbf{x}_{0:t}^{(i)}|\mathbf{z}_{1:t})}{q(\mathbf{x}_{0:t}^{(i)}|\mathbf{z}_{1:t})}. \tag{2.52}$$

Se o fator de normalização da distribuição de interesse for desconhecida $p(\mathbf{x}_{0:t}^{(i)}|\mathbf{z}_{1:t})$, é necessário normalizar os fatores, que pode ser realizado da seguinte maneira [15]:

$$\tilde{w}^{(i)} = \frac{w^{(i)}}{\sum_{k=1}^{N}w^{(k)}}. \tag{2.53}$$

A desvantagem desta amostragem de importância directa é que é necessário avaliar $p(\mathbf{x}_{0:t}^{(i)}|\mathbf{z}_{1:t})$ ou um valor proporcional, o que nem sempre é possível. Amostragem de importância sequencial manuseia um conjunto de partículas para representar a distribuição *a posteriori* do estado $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ de forma recursiva, de maneira a que em cada iteração, a aproximação ao valor expectável de uma função

arbitraria f possa ser calculada como a média ponderada das partículas [10]:

$$E[f(\mathbf{x}_t)|\mathbf{z}_{1:t}] \approx \sum_{i=1}^{N} w_t^{(i)} f(\mathbf{x}_t^{(i)}). \tag{2.54}$$

Para derivar o algoritmo, considera-se a distribuição *a posteriori* total. Felizmente a regra de *Bayes* proporciona o uso de técnicas simples na estimação e assim com recurso à regra e às propriedades de *Markov* não é necessário lidar diretamente com $p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$ nem com a amostragem de importância direta:

$$\begin{aligned} p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) &= \frac{p(\mathbf{z}_t|\mathbf{x}_{0:t}, \mathbf{z}_{1:t-1}) p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} \\ &= \eta\, p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t-1}) p(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1}) \\ &= \eta\, p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1}) \end{aligned} \tag{2.55}$$

se a distribuição de importância for formulada recursivamente:

$$q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) = q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}) q(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1}) \tag{2.56}$$

então a expressão para os fatores de importância pode ser escrita da seguinte forma:

$$w_t^{(i)} \propto \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})} \frac{p(\mathbf{x}_{0:t-1}^{(i)}|\mathbf{z}_{1:t-1})}{q(\mathbf{x}_{0:t-1}^{(i)}|\mathbf{z}_{1:t-1})}. \tag{2.57}$$

Assumindo que já se extraiu as amostras $\mathbf{x}_{0:t-1}^{(i)}$ da distribuição de importância anterior $q(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1})$ e já se calculou os pesos $w_{t-1}^{(i)}$, pode-se gerar as novas hipóteses de estado para o tempo corrente t executando $\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})$. Os fatores de importância da iteração anterior são proporcionais ao último termo da equação e assim tem-se [10]:

$$w_t^{(i)} \propto \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})} w_{t-1}^{(i)}. \tag{2.58}$$

A equação acima oferece um mecanismo de sequencialmente atualizar os fatores de importância dada uma distribuição proposta apropriada $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t})$. Tudo o que é necessário realizar, é gerar um conjunto de amostras e iterativamente calcular os seus fatores de importância [5]. De notar que é conveniente e praticável seleccionar uma distribuição de importância que seja um processo de *Markov* para assim obter $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}) = q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_{1:t})$. Com este formato não é necessário guardar todo o historial dos estados $\mathbf{x}_{0:t}^{(i)}$, apenas os estados correntes $\mathbf{x}_t^{(i)}$[10].

2.4.3 Reamostragem das Partículas

O algoritmo de amostragem de importância sequencial tem uma grande limitação. A variância da importância dos pesos, aumenta estocasticamente ao longo do tempo. Por outras palavras, pretende-se que a variância se situe perto de zero com o intuito de obter estimativas razoáveis. À medida que a variância aumenta, esta cria um impacto negativo na exatidão das estimativas. O aumento da

variância causa a degeneração das partículas, o que na prática indica que tipicamente, depois de algumas iterações do algoritmo de amostragem sequencial, observa-se que um dos fatores de importância normalizado tende para um, enquanto que todos os restantes tendem para zero. Desta maneira, um grande número de amostras tornam-se numericamente insignificantes e frequentemente começam-se a dispersar erradamente no espaço [5]. De maneira a evitar a degeneração do algoritmo, é executado um processo de seleção que replica as partículas com um fator de importância elevado e descarta as partículas com um baixo fator. Executar este passo, tem a importante função de reorientar as partículas para zonas do espaço de estados, que correspondem a elevada probabilidade no modelo de observação [6]. Porém, o processo de seleção de partículas pode introduzir efeitos não desejados, como empobrecimento das amostras. Eliminar totalmente as partículas com baixo peso leva à redução da diversidade das amostras [16]. Por exemplo, se poucas amostras conterem a maior parte do peso, muitas destas que serão replicadas serão semelhantes, isto reduzirá a dispersão das partículas no espaço de estados ficando estas altamente concentradas em poucas zonas [17] como se pode ver na figura 2.6.

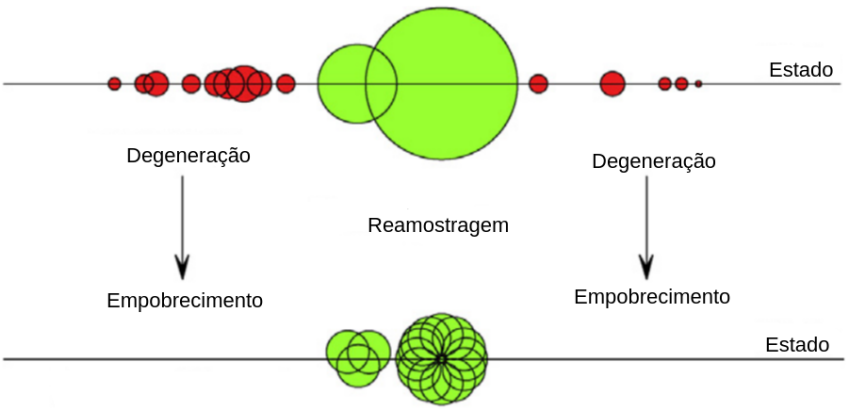


Figura 2.6: Empobrecimento das partículas após o algoritmo degenerar [17].

Existem variados métodos de seleção, sendo os mais conhecidos o método de reamostragem multinomial, reamostragem estratificada e reamostragem sistemática [18]. Pesquisa por algoritmos eficientes ou que impeçam ambos a degeneração e o empobrecimento das amostras, tem sido um grande tópico de investigação nos filtros de partículas [17, 18]. Para melhor compreensão dos algoritmos de reamostragem serão explicados e comparados apenas estes três métodos tradicionais, o multinomial, estratificado e sistemático.

Reamostragem multinomial

A ideia base deste método é de gerar N números aleatórios $u_t^{(n)} \sim \mathcal{U}(0,1), \quad n = 1, ..., N$ e usá-los para selecionar as partículas a serem reamostradas. A partícula $x_t^{(m)}$ é escolhida se satisfazer a seguinte equação:

$$Q_t^{(m-1)} < u_t^{(n)} \leq Q_t^{(m)} \tag{2.59}$$

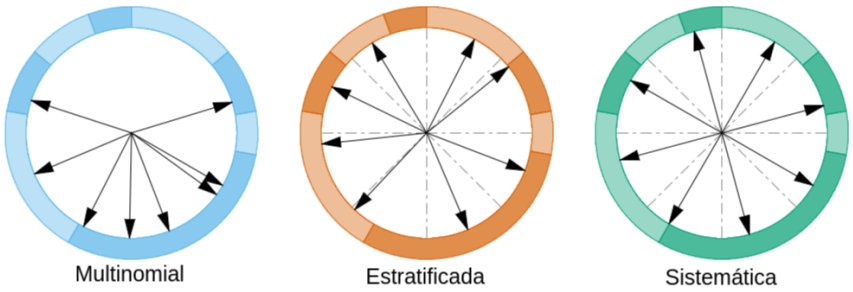


Figura 2.7: Visualização de três processos de reamostragem. Nas coroas circulares, estão evidenciados os pesos normalizados do filtro, uns maiores e outros menores. As 8 setas indicam as partículas que foram selecionadas para serem reamostradas e as linhas a tracejado nos dois métodos mais à direita correspondem aos estratos, isto é, 8 conjuntos que dividem uniformemente o círculo dos pesos [18].

onde $Q_t^{(0)} = 0$ e $Q_t^{(m)} = \sum_{k=1}^{m} \tilde{w}_t^{(k)}$. Assim, a probabilidade de amostrar a partícula $x_t^{(m)}$ é a mesma probabilidade de $u_t^{(n)}$ se situar entre o intervalo limitado pela soma acumulada dos pesos normalizados (2.59) [16]. O método como se pode observar na figura 2.7, tende a escolher as partículas com maior probabilidade mas não descarta as de baixa. A grande desvantagem deste filtro é a eficiência computacional, pois é preciso gerar N números aleatórios e também procurar pelo valor de $u_t^{(n)}$ que satisfaz a equação (2.59), onde se for aplicado uma pesquisa binária a complexidade computacional do algoritmo é de $\mathcal{O}(Nlog(N))$ [19].

Reamostragem estratificada

O processo de reamostragem estratificada divide todo o conjunto de partículas em sub-conjuntos, ou seja, estratifica o intervalo de $[0, 1]$ em N sub-intervalos $\left[0, \frac{1}{N}\right] \cup \left[\frac{1}{N}, \frac{2}{N}\right] \cup ... \cup \left[1 - \frac{1}{N}, 1\right]$. Desta maneira são gerados N números aleatórios que são amostrados independentemente em cada um dos sub intervalos $u_t^{(n)} \sim \mathcal{U}\left(\frac{n-1}{N}, \frac{n}{N}\right), \quad n = 1, 2, ..., N$. Com este algoritmo é possível obter abordagens com complexidade $\mathcal{O}(N)$. Também como na reamostragem multinomial, as partículas são escolhidas de acordo com o intervalo (2.59) [19]. Na figura 2.7 observa-se como as partículas são escolhidas, cada número aleatório seleciona um valor dentro do seu estrato.

Reamostragem sistemática

Este algoritmo explora a ideia do algoritmo de reamostragem estratificada mas de uma maneira diferente. Apenas um número aleatório é gerado da distribuição uniforme e os restantes são obtidos de forma determinística, $u_t^{(1)} \sim \mathcal{U}\left(0, \frac{1}{N}\right), \quad u_t^{(n)} = u_t^{(1)} + \frac{n-1}{N}, \quad n = 2, 3, ..., N$. Esta abordagem é visível na figura 2.7, o primeiro número aleatório escolhe um valor dentro do seu estrato e de seguida todos os outros valores ficam deslocados da mesma maneira que o primeiro. As partículas são reamostradas da mesma maneira que nos métodos anteriores, com recurso à equação (2.59) e este

método também atinge uma complexidade linear $\mathcal{O}(N)$, mas apresenta a vantagem de apenas gerar um número aleatório, sendo assim, o algoritmo mais eficiente dos três enunciados.

Como foi referido, os três métodos baseiam-se em abordagens que selecionam as partículas com a mesma intuição e assim apresentam resultados similares [16]. É possível constatar na figura 2.7, que os três métodos têm em consideração o conceito de degeneração e empobrecimento das amostras.

Algoritmo de reamostragem de importância sequencial

Com o método de reamostragem de partículas, surge o filtro de Reamostragem de Importância Sequencial (RIS). Este adiciona ao método de amostragem de importância sequencial o processo de seleção de partículas [5]. Abaixo encontra-se o algoritmo do RIS.

Algoritmo 2: $\left(\mathbf{x}_t^{(1)}, \ ... \ , \mathbf{x}_t^{(N)}\right) = \text{RIS}\left(\mathbf{x}_{t-1}^{(1)}, \ ... \ , \mathbf{x}_{t-1}^{(N)}\right)$

Para $t = 0$**:**

$\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0), \quad i = 1, ..., N$

$w_0^{(i)} = \frac{1}{N}, \quad i = 1, ..., N$

Para $t \geq 1$**:**

$\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i-1)}, \mathbf{z}_{1:t}), \quad i = 1, ..., N$

$w_t^{(i)} \propto \frac{p(\mathbf{z}_t | \mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)} | \mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)} | \mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})} w_{t-1}^{(i)}, \quad i = 1, ..., N$

$\tilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{k=1}^{N} w_t^{(k)}}, \quad i = 1, ..., N$

REAMOSTRAGEM$(\mathbf{x}_t^{(i)}, \tilde{w}_t)$

$w_t^{(i)} = \frac{1}{N}, \quad i = 1, ..., N$

2.4.4 Passo de Predição

Tendo apresentado o RIS, será agora explicado como funciona o PF. O filtro de partículas corresponde a uma especialização do filtro de reamostragem de importância sequencial, onde a distribuição de importância proposta é à probabilidade de transição de estado $p(\mathbf{x}_t | \mathbf{x}_{t-1})$. Como foi referido na amostragem de importância sequencial, são gerados N hipóteses de estado provenientes da distribuição proposta [6]. Assim as hipóteses de estado são geradas da seguinte maneira:

$$\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i)}). \tag{2.60}$$

2.4.5 Passo de Observação

A vantagem de usar como distribuição de importância a probabilidade de transição de estado é que esta não só é normalmente fácil de implementar como simplifica o cálculo dos fatores de importância

[5, 10]:

$$w_t^{(i)} \propto \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})} w_{t-1}^{(i)} = \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1})^{(i)}} w_{t-1}^{(i)} = p(\mathbf{z}_t|\mathbf{x}_t^{(i)}) w_{t-1}^{(i)}. \tag{2.61}$$

Deste modo, o passo de observação que aplica a cada partícula o modelo de observação, calcula diretamente os fatores de importância. De seguida basta normalizar os fatores:

$$\tilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{k=1}^{N} w_t^{(k)}}. \tag{2.62}$$

Este conjunto de partículas ponderado pelo fator de importância, é usado para aproximar a estimativa do estado *a posteriori* $p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) \approx \sum_{i=1}^{N} \tilde{w}_t^{(i)} \delta(\mathbf{x}_{0:t} - \mathbf{x}_{0:t}^{(i)})$ [6].

2.4.6 Passo de Seleção

Finalmente as partículas são selecionadas para evitar a degeneração do filtro. O passo de seleção recorrendo aos fatores de importância, implementa a relevante função de forçar a distribuição das partículas a assemelharem-se à distribuição *a posteriori* [6]. O algoritmo deste filtro pode ser consultado em A.4. É apresentada a figura 2.8 que exemplifica o comportamento do filtro de partículas ao longo do tempo. Os círculos representam as partículas e o tamanho dos círculos o fator de importância. Abaixo são explicados os passos que estão enumerados na figura:

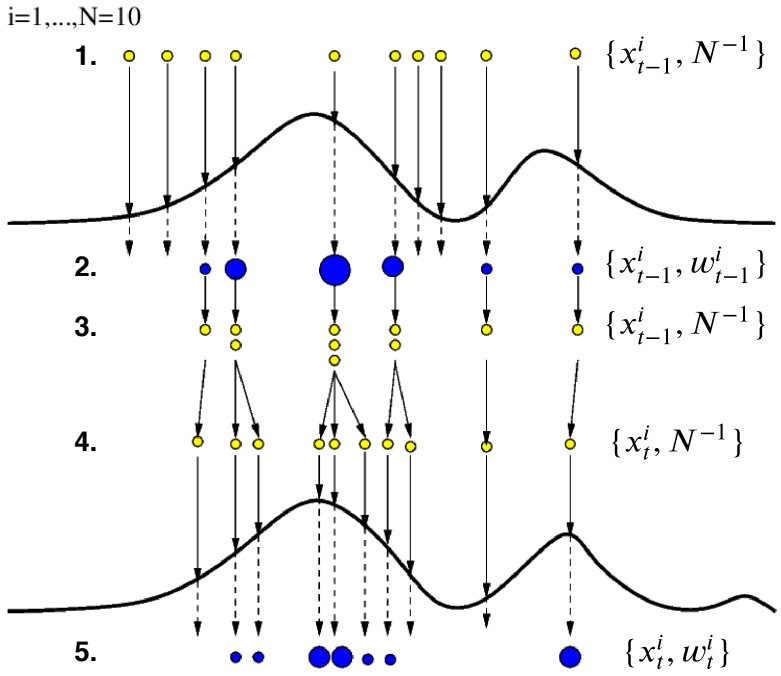


Figura 2.8: Representação do funcionamento do filtro de partículas [5].

1. Na figura 2.8, observa-se o filtro em $t-1$ com a distribuição das partículas representadas a amarelo, todas apresentam o mesmo fator de importância, pois ainda não foi aplicado o modelo de observação.

2. É aplicado o modelo de observação onde são calculados os fatores de importância e de seguida, os fatores são normalizados. Na imagem, existem setas que apontam para nada, porque o fator de importância é muito reduzido. Já com os pesos calculados é possível obter uma estimativa do estado *a posteriori*.

3. É efetuado o passo de seleção que replica as partículas de maior peso e descarta as de menor. É reiniciado o peso das partículas para N^{-1}. Como se pode constatar, a disposição das partículas alterou-se. Agora as partículas estão distribuídas de acordo com a distribuição *a posteriori*.

4. É novamente efectuado o passo de predição, que propaga as partículas no espaço de estados. Como se observa, este processo introduz diversidade na estimativa do estado.

5. É calculado novamente a importância dos pesos, que fornece uma estimativa *a posteriori* do estado para o tempo atual.

2.5 Melhoria da Distribuição de Importância

O desempenho do filtro de partículas depende não só da quantidade de partículas mas também da qualidade da distribuição de importância. Quanto mais a distribuição de importância proposta se assemelhar à distribuição *a posteriori*, melhor será o desempenho do filtro. Num processo de *Markov*, a distribuição de importância ótima em termos da minimização da variância dos pesos é dada por [5]:

$$q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t). \tag{2.63}$$

Esta é chamada de distribuição ótima visto que alcança o mínimo de variância dos pesos sobre as diferentes hipóteses de estado $\mathbf{x}_t^{(i)}$. Como se demonstra nas equações abaixo, os pesos $w_t^{(i)}$ são independentes de $\mathbf{x}_t^{(i)}$ e assim a variância é zero [20]. Fazendo:

$$q(\mathbf{x}_t|\mathbf{x}_{0:t}, \mathbf{z}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{z}_t|\mathbf{x}_{t-1})} \tag{2.64}$$

obtêm-se:

$$\begin{aligned} w_t^{(i)} &\propto \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})} w_{t-1}^{(i)} \\ &= \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})p(\mathbf{z}_t|\mathbf{x}_{t-1}^{(i)})}{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})} w_{t-1}^{(i)} \\ &= p(\mathbf{z}_t|\mathbf{x}_{t-1}^{(i)})w_{t-1}^{(i)}. \end{aligned} \tag{2.65}$$

Todavia, a distribuição proposta mais utilizada e popular em filtros de partículas é a probabilidade de transição de estado $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. Apesar de resultar numa maior variabilidade do que a distribuição proposta ótima (que introduz a mais recente observação), é normalmente mais fácil de implementar [5, 10]. Não só o modelo do processo por norma é trivial, como também o cálculo dos fatores de importância é dado pelo modelo de observação. Porém a grande desvantagem deste modelo é a

ineficácia desta distribuição quanto à variância dos pesos, pelo que poderá ser necessário usar um elevado número de partículas para obter estimativas adequadas [5].

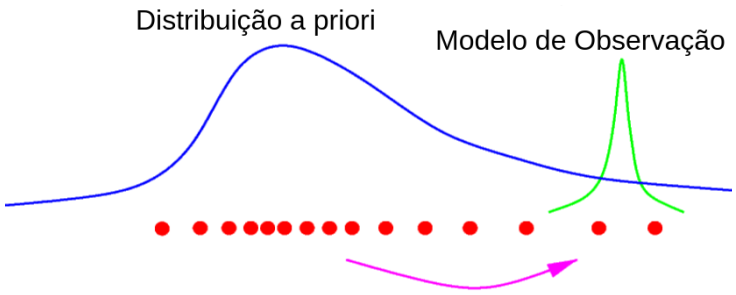


Figura 2.9: Modelo de observação apresenta um erro de medição pequeno e encontra-se numa zona com poucas partículas. É de grande importância que a distribuição proposta propague as partículas para zonas de alta probabilidade [5].

Uma distribuição de importância ótima, permite mover as partículas da distribuição *a priori* para regiões de grande probabilidade do modelo de observação. Isto é bastante importante para casos onde o modelo de observação se situa em zonas de baixa probabilidade *a priori*, que terá consequentemente menos partículas, ou se for muito estreito (se o modelo tiver um erro de medição pequeno), como se observa na figura 2.9. Se a última informação existente para propor novas hipóteses de estado falhar, então apenas algumas partículas serão replicadas [5]. Ao contrário do modelo do processo que é simples de amostrar, no geral, amostrar da distribuição ótima é não trivial, devido à dependência na observação atual $\mathbf{z}_t$. Com o propósito de aproximar a distribuição ótima na filtragem, o Filtro de Partículas de *Kalman* Estendido (EKPF) e o UPF, foram desenvolvidos [5]. Estes aproximam a distribuição ótima, incorporando a observação corrente em conjunto com a estimativa gaussiana do estado. Assim o mecanismo consiste num filtro de partículas que tem como distribuição de importância, o filtro de *Kalman* estendido no caso do EKPF ou o UKF no caso do UPF.

$$q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t}^{(i)}, \mathbf{z}_{1:t}) \approx \mathcal{N}(\boldsymbol{\mu}_t^{(i)}, \mathbf{P}_t^{(i)}) \quad i = 1, ..., N. \tag{2.66}$$

Os algoritmos EKPF e UPF são bastantes semelhantes (estes podem ser comparados em A.5 e A.6), diferem apenas no método de propagação de partículas que num caso é o EKF e no outro o UPF [5]. Ambos apresentam algumas vantagens perante o PF, sendo a mais notória a incorporação da observação atual $\mathbf{z}_t$, o que aproxima de melhor forma as partículas para zonas de alta probabilidade no modelo de observação. Que por consequência, trás a vantagem de se poder utilizar um menor número de partículas e obter estimativas satisfatórias. A grande desvantagem comparando com a distribuição proposta $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ é a elevada complexidade computacional. Não só o processo de gerar as amostras é mais complexo, visto que é necessário iterar o UKF ou o EKF para cada partícula, como também o processo de avaliação dos fatores de importância não é apenas o modelo de observação, mas sim a equação geral recursiva dos fatores de importância dada pela equação (2.58).

2.6 Vantagens e Desvantagens dos Filtros Mencionados

Quando na escolha de um filtro para estimar o estado de um sistema, é necessário ter em conta três propriedades, eficiência computacional, exatidão da solução e facilidade de implementação. Nesta secção são comparados todos os filtros enunciados nesta dissertação, nomeadamente o EKF e o UKF.

Uma característica que diferencia e limita completamente o filtro de *Kalman* é de ser apenas viável para problemas lineares. Por outro lado, devido a esta condicionante, as distribuições são exatamente gaussianas e assim, o KF implementa de forma ótima o filtro de *Bayes* [10]. O EKF e o UKF representam uma extensão do KF e por isso estes três estão limitados a distribuições unimodais. Os restantes três filtros o PF, EKPF e UPF baseiam-se em métodos de reamostragem de importância sequencial, no qual, técnicas de *Monte Carlo* são usadas para obter soluções aproximadas das equações do filtro de *Bayes*. O que permite obter representações de distribuições multimodais [6, 10]. Devido à grande versatilidade destes filtros, uma ampla quantidade de distribuições *a posteriori* podem ser aproximadas. Contudo, para obter melhores estimativas, maior é o número de partículas necessárias, o que diminuí a eficiência computacional, enquanto que nos filtros gaussianos a eficiência depende da dimensão do estado [6]. Dos filtros baseados em partículas o mais eficiente é o PF, pois no EKPF e no UPF é necessário executar respetivamente o EKF e o UKF para cada partícula. Por outro lado, para um mesmo número de partículas ambos o EKPF e o UPF obtém estimativas mais exatas, visto que ambos tentam aproximar a distribuição de importância ótima. Quanto ao EKF e UKF, ambos aproximam a estimativa do estado através de uma distribuição gaussiana. O que os difere, é a abordagem às não-linearidades. O EKF baseia-se no método de linearização local e o UKF na *scaled unscented transform*. Uma grande vantagem do segundo face ao primeiro, é que não se baseia numa aproximação linear em torno de um único ponto, em vez disso, vários pontos são usados para aproximar a não-linearidade [5, 10]. Em vez de se aproximar o modelo do processo e o modelo de observação, a distribuição da estimativa do estado é que é aproximada. Como o EKF apenas usa termos de primeira ordem na expansão da série de *Taylor*, frequentemente introduz erros na estimação, sendo esses erros especialmente evidentes quando os efeitos de ordem superior são significantes. Já a SUT é exata até aos termos de segunda ordem de qualquer função não linear, com erros apenas a serem introduzidos na terceira ou maiores ordens [5]. Outra vantagem é que as funções de predição e de observação não necessitam de ser diferenciáveis nem é preciso calcular as matrizes jacobianas. Pelo contrário, as funções serem diferenciáveis é uma restrição no EKF [6, 10]. As desvantagens do UKF é que apresenta uma complexidade computacional e de implementação superior. Todavia, apesar do UKF apresentar uma eficiência mais baixa, a diferença face ao EKF não é muito acentuada [5]. Pelas razões enunciadas, o UKF é mais robusto e exato (o desempenho deste já foi demonstrado em vários trabalhos [5]) que o EKF, que por consequência torna o UPF num filtro mais exato que o EKPF [5].

Capítulo 3

Estado da Arte

Neste capítulo são enunciadas as principais abordagens e métodos usados em localização e seguimento 3D de acordo com a bibliografia pesquisada. São referidos vários trabalhos baseados em diferentes tipos de câmaras, são mencionadas vantagens e desvantagens de abordar problemas de localização e seguimento 3D com uma só câmara ou várias e por fim, são evidenciados diversos métodos aplicados para localizar objetos, apresentando também vantagens e desvantagens entre os métodos enunciados.

3.1 Câmaras e Abordagens para Localização e Seguimento 3D

Algoritmos de localização e seguimento de objectos necessitam de dispositivos capazes de adquirir informação 3D, ou dispositivos que obtenham dados que possam ser inferidos, para obter informação 3D. Para adquirir dados 3D do mundo real, existem principalmente dois tipos de câmaras: As câmaras RGB (do inglês *Red-Green-Blue*), que como o próprio nome indica geram imagens de cor baseadas nas três cores primárias, vermelho verde e azul e as câmaras RGB-D que geram imagens de cor e correspondentes imagens de profundidade (do inglês *Depth*).

Como as câmaras RGB apenas capturam dados visuais, são necessários algoritmos que avaliem esses dados e a partir destes obtenham informação 3D. A grande vantagem das câmaras RGB-D é que estas capturam dados visuais e geométricos do espaço real. Para cada pixel é obtido um valor de cor de uma zona no espaço e do valor correspondente da distância entre o plano de imagem da câmara e essa zona. Por exemplo, o processo de aquisição da câmara RGB-D *Kinect* como é explicado por Leandro Cruz et al. [21] consiste numa câmara RGB para obter uma imagem de cor e numa técnica de luz estruturada que utiliza um emissor infra-vermelho e outra câmara, para realizar a medição de profundidade. Assim as câmaras RGB-D fornecem dados valiosos de profundidade que simplificam bastante a reconstrução de um cenário 3D o que leva a algoritmos de deteção mais simples. Porém, estas câmaras também apresentam várias desvantagens, falham frequentemente em cenários ao ar livre devido à interferência da luz solar, falham com objetos bastante refletivos ou bastante absorventes, são volumosas, apresentam um maior consumo de energia, apresentam uma menor resolução, têm

um alcance limitado de profundidade e não são tão acessíveis como as câmaras RGB [22]. Existem variados trabalhos na área de deteção, localização e estimação 3D de objetos, uns com câmaras RGB outros com câmaras RGB-D. Por exemplo o trabalho desenvolvido por *Dushyant Mehta et. al* [23] com recurso a apenas uma câmara RGB, utiliza redes neuronais convolucionais para estimar a pose 3D de um humano, já o trabalho de *Andreas Baak et. al* [24] também para estimar a pose 3D de um ser humano utiliza uma câmara RGB-D.

A deteção de objetos através de processamento de imagem e visão computacional utilizando apenas câmaras RGB, segue principalmente duas tendências: Deteção monocular (baseada numa câmara) e deteção estereoscópica (baseada em duas ou mais câmaras).

A abordagem monocular usa técnicas como ajuste de modelos de objetos, segmentação de cores, deteção de características como textura e simetrias, etc. Após esta fase de deteção, a medição de propriedades 3D na imagem é feita através de uma combinação do conhecimento prévio que se tem do objeto, características na imagem e conhecimento sobre os parâmetros de calibração da câmara. Já a abordagem estereoscópica apresenta a vantagem de estimar diretamente as coordenadas 3D de uma característica na imagem, seja este, qualquer coisa desde um ponto até uma estrutura complexa [25]. Estereoscopia consiste em encontrar correspondências de características em imagens de diferentes câmaras previamente calibradas, para estimar as coordenadas dessas características em 3D. No entanto, requer uma calibração muito precisa da montagem rígida das câmaras.

Vantagens de utilizar um sistema com apenas uma câmara em vez de um sistema que integra múltiplas é que permite uma configuração simples, não é necessário calibrar todo o sistema de câmaras, nem instalar todo o equipamento complementar. É mais rápido, não só é necessário adquirir e processar todos os dados provenientes das câmaras como também os métodos de visão estéreo existentes, são principalmente lentos na parte de correspondências e assim apresentam uma grande limitação no desenvolvimento de aplicações em tempo real [25]. É bastante portável, este sistema pode ser facilmente integrado num robô móvel, é acessível, entre outras. Claramente também existem desvantagens, apenas um limitado campo de visão está acessível, oclusão do objeto de interesse acontece frequentemente, pelo contrário ter várias câmaras aumenta a redundância do sistema [4].

3.2 Métodos de Localização e Seguimento 3D

Um grande número de métodos de localização e seguimento 3D de objectos têm sido propostos nas últimas décadas. Estes métodos podem ser classificados em quatro grandes categorias: métodos baseados em contornos, baseados em características, baseados em regiões e métodos directos [26]. Nos próximos tópicos, serão apresentadas as vantagens e desvantagens destes métodos, no que consistem, e alguns algoritmos baseados nestes. Por fim, é apresentado uma conclusão de todos os métodos, com uma tabela de comparação.

3.2.1 Métodos Baseado em Contornos

Segundo a literatura, as primeiras abordagens que apareceram para localização e seguimento de objectos 3D eram, maioritariamente, baseadas em extração de contornos, pois são métodos de fácil implementação e computacionalmente eficientes. São também naturalmente estáveis perante variações de iluminação, mesmo para superfícies polidas, uma vez que não é necessário considerar os pixeis internos que pertencem ao objecto [27]. Para poderem funcionar de forma correta, estes tipos de métodos dependem de margens perceptíveis, contudo frequentemente debatem-se com numerosos mínimos locais, especialmente quando são aplicados em meios ambientes que apresentam fundos confusos e complexos [26], como o da figura 3.1.

Figura 3.1: Na imagem apresentada é possível detetar bastantes contornos no meio envolvente ao objeto alvo, neste caso a chávena. [28]

Um algoritmo baseado neste tipo, é o algoritmo *Real-Time Attitude and Position Determination* (RAPID) [29]. Por causa da sua baixa complexidade computacional, este foi um dos primeiros algoritmos a ser utilizado com sucesso para seguimento 3D em tempo real [27]. A ideia chave do algoritmo é que partindo do modelo 3D do objecto, é possível selecionar-se um conjunto de pontos 3D, denominados por pontos de controlo, extraídos de contornos de grande contraste no objecto alvo. Estes pontos devem ser ativados ou desativados conforme o movimento do objecto, uma vez que existem contornos que podem ficar ocultos com a rotação do objecto [29]. Posto isto, o movimento 3D entre duas imagens consecutivas pode ser recuperado através da deslocação 2D dos pontos de controlo. Estes são comparados nos contornos do objecto da nova imagem e a nova postura é estimada a partir destas correspondências. No entanto, utilizar apenas heurísticas dos contornos torna o algoritmo fraco. Este algoritmo não previne que contornos pouco distintos e contornos de meios envolventes confusos, sejam expostos. Por outro lado, também a própria textura do objecto pode expor contornos, e por conseguinte afetar o algoritmo de seguimento do objecto [27].

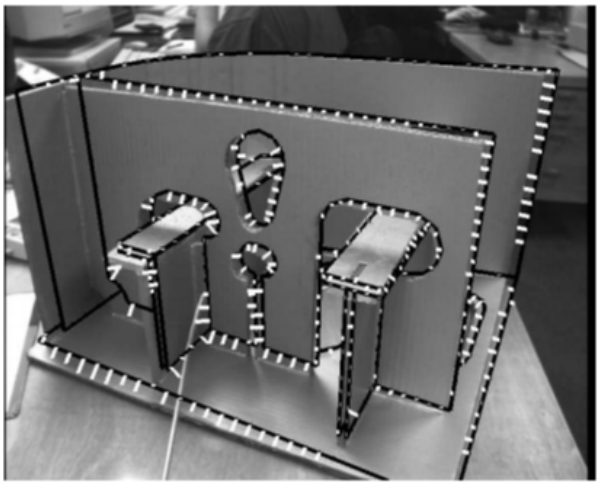


Figura 3.2: Algoritmo RAPID. As linhas a preto representam os contornos da imagem anterior. Nesta nova imagem, os pontos de controlo são amostrados dos contornos e os segmentos a branco representam o deslocamento das antigas posições dos pontos de controlo para as posições correntes.[30]

3.2.2 Métodos Baseado em Características

Métodos baseados em características, são métodos que utilizam características locais que pertencem ao objecto de interesse [27]. Estes métodos têm o mesmo princípio de correspondência que os métodos baseados em contornos, mas em vez da correspondência ser com os contornos do objecto, é com as características locais. A grande vantagem destes métodos é que estas características são individuais, isto é, cada característica local apresenta atributos que a faz diferenciar de todas as outras. Assim, é mais fácil de tornar o sistema robusto contra oclusão parcial ou erros de correspondência. Invariância à iluminação é também algo favorável de obter com estes métodos, comparando com os métodos acima referidos. Fundos complexos não confundem o algoritmo devido à descriminação dos pontos locais, pelo que é possível explorar mais informação na imagem [27]. A grande desvantagem é o facto dos objectos utilizados terem de apresentar textura suficiente para extrair uma considerável quantidade de pontos de interesse, de forma a estabelecer-se correspondências. Logo, estes métodos não são capazes de lidar com objectos pobres em texturas [26].

Aliados a estes métodos, estão os algoritmos que utilizam *Scale Invariant Feature Transform* (SIFT) [31]. Este processo consiste em extrair características invariantes e distintas a partir de imagens. Estas características que são extraídas, são invariantes à escala e à rotação da imagem, e são robustas contra distorções, mudanças no ponto de vista, ruído e variações de iluminação. Para detetar estes pontos característicos, também denominados de pontos chave, o sistema percorre quatro etapas [31]:

- A primeira etapa é identificar potenciais pontos invariantes à escala e à orientação através de uma função específica;
- De seguida, a cada candidato um modelo detalhado é ajustado para determinar a localização e a escala do ponto. Os pontos chave são selecionados para a fase seguinte, conforme a sua medida

de estabilidade;

- Posteriormente, uma ou mais orientações são atribuídas ao ponto chave, utilizando as direções do gradientes locais na imagem à volta do ponto;
- Para finalizar, a magnitude dos gradientes são medidos à escala previamente selecionada e são criados histogramas de orientação em torno do ponto local. Assim, é criado um descritor para cada ponto-chave que descriminada cada característica local.

Aplicando este processo a cada imagem, é possível maximizar a correlação das correspondências de cada ponto, e deste modo localizar e seguir o objecto como representado na figura 3.3.

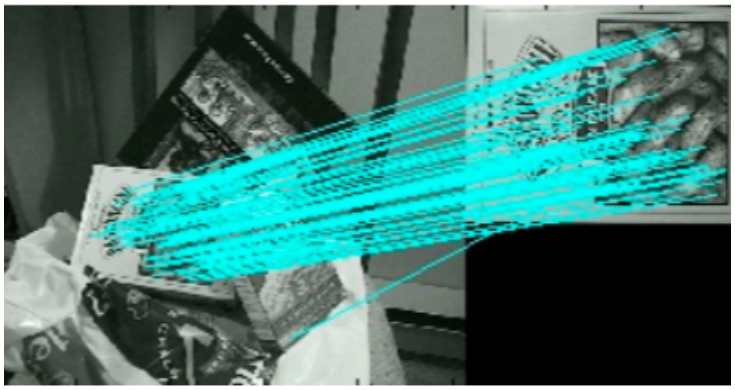


Figura 3.3: Algoritmo de deteção de pontos chave SIFT. Como se observa na figura, as linhas a azul correspondem à posição dos pontos chave extraídos do livro para as diferentes imagens [32].

3.2.3 Métodos Baseado em Regiões

Estes métodos focam-se nas propriedades estatísticas de diferentes regiões nas imagens. Estes iterativamente evoluem projectando o contorno 2D do modelo do objecto 3D, otimizando os parâmetros da postura do mesmo, de forma a que haja maximização das discrepâncias estatísticas entre a região do objecto e a região de fundo. Estes métodos são vantajosos no seguimento de objectos homogéneos com baixa riqueza em texturas, com meios envolventes simples a médio complexos. Para meios ambientes confusos, com bastantes obstruções e objectos heterogéneos, o desempenho do algoritmo degrada-se devido à variação estatística do objecto e do plano de fundo, pois as discrepâncias das regiões não são tão expressivas. Também estes métodos sofrem de ambiguidade na postura para objectos simétricos, pois eles determinam a posição 3D do objecto através da informação da silhueta, e apenas com esta informação, a orientação do objecto não é possível de se auferir [26].

Um dos algoritmos que utiliza este tipo de métodos, é o algoritmo PWP3D. Este foi o primeiro sistema a ser implementado que alcançou um desempenho vantajoso em tempo real, ao utilizar este tipo de método. Este algoritmo consiste em maximizar a discriminação estatística entre os modelos interiores e exteriores ao objecto, otimizando os parâmetros de postura do objecto, utilizando o método

do gradiente descendente [33]. A partir da figura 3.4 é possível perceber como o algoritmo funciona. Primeiro, com o intuito de separar a região interior da exterior, é projetado na imagem o contorno C, que representa a postura inicial estimada 3D do objecto. De seguida, é calculada uma função de energia, que tem a finalidade de determinar a qualidade de correlação dos pixeis da imagem com os modelos interiores e exteriores. Esta função é constantemente minimizada adaptando a projeção do contorno C na imagem. No fim desta minimização estar concluída, espera-se obter a verdadeira postura do objecto [33].

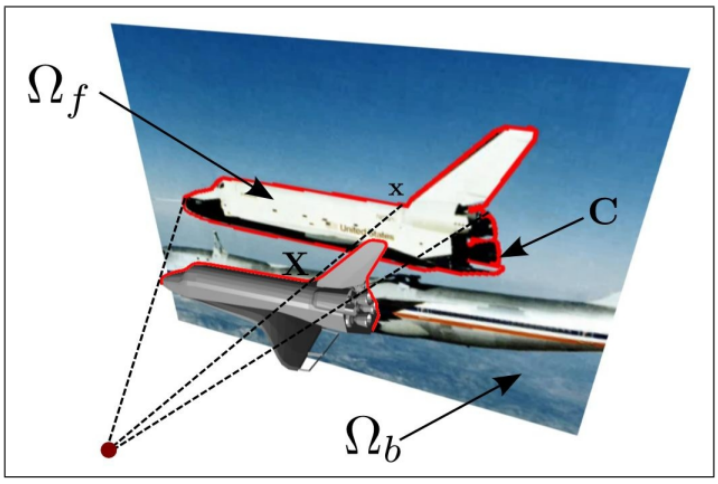


Figura 3.4: Algoritmo PWP3D. Nesta imagem observa-se o contorno C e as regiões interior e exterior [33].

3.2.4 Métodos Diretos

O objetivo principal dos métodos diretos é minimizar o erro fotométrico através do alinhamento direto da imagem corrente, com um conjunto de imagens de referência e o seu respetivo modelo 3D, de forma a calcular-se o movimento 3D efectuado. Apenas se utiliza a informação bruta dos pixeis do objecto, ignorando completamente as propriedades estatísticas do plano do objecto e do plano de fundo. Esta premissa de ignorar as propriedades estatísticas, leva à grande desvantagem deste método que é a constância fotométrica. Esta constância é frequentemente afectada por variações de iluminação, o que leva a erros na estimação do objecto alvo [26].

3.2.5 Comparação entre Métodos

Recapitulando, para estes quatro grupos de métodos, temos a seguinte tabela de comparação:

	Contornos	**Caraterísticas**	**Regiões**	**Diretos**
Pixeis usados	Pixeis nos contornos	Pixeis interiores	Pixeis à volta dos contornos	Pixeis interiores
Informação usada	Contornos do objeto	Caraterísticas locais	Informação estatística das regiões	Valores brutos dos pixeis
Regiões consideradas	Contornos do objeto	Interior	Interior e exterior	Interior
Vantagens	Objetos homogéneos e fundos simples	Objetos com textura	Objetos homogéneos e fundos simples	Objetos com textura
Desvantagens	Objetos heterógeneos ou fundos complexos	Objetos homogéneos	Objetos heterógeneos ou fundos complexos	Variações na iluminação

Tabela 3.1: Comparação dos métodos apresentados [26].

Capítulo 4

Metodologias

Baseado no algoritmo desenvolvido por *Taiana et al.* [4] e com o intuito de melhorar o seu desempenho, propôs-se a implementação do UPF explicado na secção 2.5 como núcleo do filtro. Uma das grandes vantagens deste filtro, é o facto de inserir no modelo de predição uma observação corrente completamente independente do modelo de observação, com o propósito de propagar com mais eficácia as partículas para zonas de alta verosimilhança. Inserir esta observação corrente, permite não só predizer melhor o próximo estado do objecto, como para situações de grande imprevisibilidade, permite um ajuste mais eficaz do filtro ao movimento do objeto. Já no trabalho de *Taiana et al.* [4] é aplicado o filtro de partículas convencional que apenas usa o modelo de transição de estado para propagar as partículas e por isso, pode ter problemas quando o movimento real do objeto se desvia do modelo.

No capítulo anterior mencionaram-se vários métodos que se aplicam em problemas de localização e seguimento de objetos. Ora, um destes tipos de métodos é o de regiões, que é usado no trabalho de *Taiana et al*. [4]. Como o próprio nome indica, o método considera duas regiões, uma interior ao objeto e outra exterior. A teoria é que maximizando a discrepância entre as regiões e maximizando a semelhança entre a região interior com o conteúdo real do objeto, indicará que seja mais provável que o objeto se encontre nesse local. Dito isto, é para problemas de seguimento de objetos homogéneos que este método melhor se adequa, pois à medida que as regiões apresentam mais textura as discrepâncias tornam-se menos expressivas. Na secção 1.1, referiu-se que o objeto que se pretende seguir é uma esfera de cor homogénea e que se sabe qual é a sua cor e dimensão. Sendo o objeto uma esfera, as regiões consideradas estão ao redor da sua silhueta. Na secção seguinte será revelado de uma forma geral como funciona o algoritmo desenvolvido por *Taiana et al.* [4], algoritmo este, que serviu como base para o desenvolvimento deste trabalho. Seguidamente são apresentadas as arquiteturas dos algoritmos desenvolvidos neste trabalho, a arquitetura do filtro de partículas que reproduz o trabalho de *Taiana et al.* [4] e a do *unscented particle filter*, algoritmo proposto para melhorar a eficácia do primeiro. Para finalizar, são explicados métodos que os algoritmos executam na estimação do estado do objeto.

4.1 Algoritmo de Seguimento 3D de Objetos

O algoritmo desenvolvido por *Taiana et.al* [4], estima o estado de um objecto em 3D ao longo do tempo baseado em métodos de regiões. De uma maneira resumida, este sistema foi desenvolvido para funcionar com um grande número de sensores óticos, isto é, o algoritmo é aplicável para diversos modelos de projeção de imagem. A arquitetura do sistema é baseada num filtro de partículas, onde cada partícula representa uma hipótese do estado 3D do objeto, em vez do seu estado na imagem. Isto permite superar a inversão da não-linearidade causada pelo modelo de projeção das câmaras (como por exemplo, o uso de câmaras omnidirecionais como se observa na figura 4.1 que apresenta grande distorções na imagem) [4].

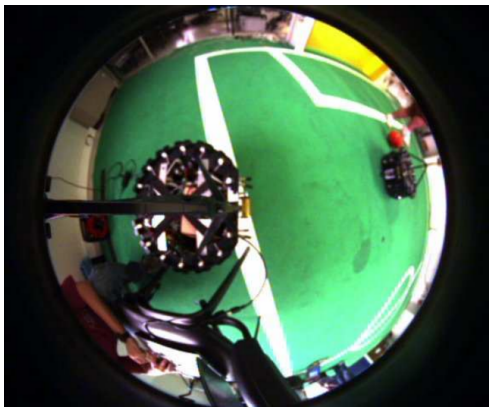


Figura 4.1: Imagem proveniente de uma câmara omnidirecional. Apesar da câmara ter um grande campo de visão, esta apresenta uma profunda distorção radial o que explica a aparência esférica das imagens [34].

Frequentemente, para imagens com estas distorções, métodos de otimização não-linear são utilizados para identificar o objeto nas imagens, pelo que estas abordagens ou apresentam limites na convergência ou necessitam de modelos de predição muito precisos. Com este algoritmo tenta-se superar estes problemas utilizando um filtro de partículas, uma vez que todas as não-linearidades estão concentradas no modelo de observação. No modelo de observação, cada partícula projeta dois conjuntos de pontos na imagem segundo um modelo 3D do objeto. Projetar apenas os pontos isoladamente e não os contornos completos, facilita bastante na projeção de modelos não-lineares. Um conjunto forma a região interior e o outro a região exterior. Como apenas operações com pixeis são aplicadas, usar rotinas de processamento de imagem como extração de características locais ou contornos, que são sensíveis ao movimento e a distorções óticas, não são necessárias. Destes pontos são criados histogramas de cor, que são correlacionados com o histograma de cor do objeto e com esta informação, avalia-se cada hipótese de estado[4]. Na seção 2.5, referiu-se que a distribuição de importância mais popular e utilizada em filtros baseado em partículas é o modelo do processo $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ e desta forma, o calculo dos fatores de importância de cada partícula é dado inteiramente pelo modelo de observação $p(\mathbf{z}_t|\mathbf{x}_t)$. Ora precisamente, é esta distribuição que *Taiana et al.* [4] implementou no seu algoritmo. O algoritmo apresenta quatro etapas:

- Predição: É efetuada uma aproximação a $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$ propagando cada partícula $\mathbf{x}_t^{(i)}, i = 1, ..., N$

através do modelo de movimento do objecto.

- Observação: É aplicado o modelo de observação a cada partícula através de dados da imagem.

- Atualização: Cada fator de importância é calculado através do modelo de observação, $w_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{z}_t|\mathbf{x}_t^{(i)})$.

- Seleção: É realizado um processo de reamostragem de partículas em que a probabilidade de seleção de uma partícula é diretamente proporcional ao seu fator de importância [4].

4.1.1 Modelo de predição

Como o algoritmo foi desenhado com o intuito de se aplicar a qualquer objeto de interesse e a qualquer trajetória, dada a variabilidade dos objetos e da sua movimentação, um modelo de predição simples é adotado para lidar com as incertezas da evolução do estado. Mais especificamente, assume-se que as componentes do estado evoluem segundo modelos de movimento linear com ruído independentes e gaussianos [35]. Ou seja:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t|\mathbf{F}\mathbf{x}_{t-1}, \Lambda) \tag{4.1}$$

onde $\mathcal{N}(.|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denota a distribuição gaussiana com média $\boldsymbol{\mu}$ e covâriancia $\boldsymbol{\Sigma}$, e $\boldsymbol{\Lambda}$ corresponde à matriz diagonal que contem as variâncias do modelo de movimento das diversas componentes de estado do objeto [35][4]. Para o caso do seguimento de uma bola, o estado do sistema consiste nas coordenadas cartesianas 3D de posição $\mathbf{p}_t$ e velocidade $\dot{\mathbf{p}}_t$, ou seja:

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t & \dot{\mathbf{p}}_t \end{bmatrix}^T = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T \tag{4.2}$$

O modelo de movimento escolhido para o seguimento $\mathbf{x}_t = f(\mathbf{x}_{t-1}) + \mathbf{w}_t$, onde $\mathbf{w}_t$ representa o ruído do processo para lidar com a incerteza do modelo, é o modelo de velocidade constante expresso da seguinte maneira:

$$\mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{w}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{a}_t, \quad \mathbf{F} = \begin{bmatrix} \mathbf{I}_3 & \Delta_t \mathbf{I}_3 \\ \mathbf{0} & \mathbf{I}_3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \frac{\Delta_t^2}{2}\mathbf{I}_3 \\ \Delta_t \mathbf{I}_3 \end{bmatrix}, \quad \mathbf{a}_t = \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}) \tag{4.3}$$

onde $\mathbf{I}_3$ representa a matriz identidade 3×3, Δ_t o tempo de amostragem e o ruído do processo $\mathbf{B}\mathbf{a}_t$ corresponde a um ruído de aceleração onde $\mathbf{a}_t$ adiciona às variáveis de estado, valores aleatórios gaussianos de aceleração onde $\boldsymbol{\Lambda}$ representa a matriz diagonal que contem as variâncias dos valores aleatórios. As variâncias destes valores são frequentemente ajustadas experimentalmente e como as coordenadas do modelo de movimento são coordenadas reais, estas podem ser ajustadas em unidades métricas o que é bastante conveniente e intuitivo [4].

4.1.2 Modelo de observação

Para o modelo de observação como já foi referido, é proposta uma metodologia que calcula a verosimilhança de cada partícula, utilizando o modelo geométrico e o modelo de cor do objecto. Para calcular a verosimilhança de uma hipótese de estado, são projetados na imagem um conjunto de pontos interior e um conjunto de pontos exterior ao objeto. A projeção destes pontos corresponde a pixeis na imagem e se bastantes pixeis na região interior e poucos na região exterior apresentarem cores semelhantes às cores do objeto alvo, então as partículas obtêm valores elevados. Para o caso de uma bola, as regiões interior e exterior correspondem respetivamente às fronteiras interior e exterior da silhueta da bola. De seguida, com a informação cromática dos conjuntos de pontos interior e exterior, são formados os histogramas de cor interior $\mathbf{H}_{IN}$ e exterior $\mathbf{H}_{EX}$. Também o histograma do modelo da cor do objeto $\mathbf{H}_{MO}$ necessita de ser criado, para se comparar as similaridades ou discrepâncias entre os dados da imagem e a cor do objeto [4]. Estes são criados no modelo de cor Tom-Saturação-Intensidade do inglês, *Hue-Saturation-Intensity* (HSI), que representa as componentes vermelho, azul e verde em componentes cromáticas dadas pelo tom e saturação, e acromáticas dada pela intensidade [36]. A vantagem deste modelo é que o tom e a saturação são invariantes perante variações de intensidade de iluminação, o que torna o HSI bastante robusto face a esta adversidade [37]. Os histogramas são normalizados e o número de barras por componente tom, saturação e intensidade são respetivamente dados por B_H, B_S e B_I. Deste modo, o histograma de dimensão $B_H \times B_S \times B_I$ pode ser calculado da seguinte maneira:

$$\mathbf{H}(i,j,k) = \beta \sum_{n=1}^{N} \delta_{i,b(\mathbf{p}_n^H)} \delta_{j,b(\mathbf{p}_n^S)} \delta_{k,b(\mathbf{p}_n^I)}, \quad i \in \{1,...,B_H\},\ j \in \{1,...,B_S\},\ k \in \{1,...,B_I\} \tag{4.4}$$

onde N corresponde ao número de pontos utilizados para compor o histograma, δ representa a função delta de *Kronecker*, os termos $b(\mathbf{p}_n^H)$, $b(\mathbf{p}_n^S)$ e $b(\mathbf{p}_n^I)$ indicam o índice da barra do histograma associado aos componentes H, S e I da cor do pixel $\mathbf{p}_n$ e β é a constante que normaliza o histograma [35]. Para quantificar a similaridade entre histogramas, uma métrica baseada no coeficiente de *Bhattacharyya* é definida. A métrica $\mathcal{D}$ estabelecida é a seguinte:

$$\mathcal{D} = \frac{1 - S_{IM} + KS_{IE}}{1 + K} \tag{4.5}$$

onde S_{IM} corresponde à similaridade entre o histograma interior e o histograma do modelo e S_{IE} corresponde à similaridade entre o histograma interior e exterior. Ambos são calculados através do coeficiente de *Bhattacharyya*:

$$S_{IM} = \sum_{i=1}^{B_H} \sum_{j=1}^{B_S} \sum_{k=1}^{B_I} \sqrt{\mathbf{H}_{IN}(i,j,k)\mathbf{H}_{MO}(i,j,k)} \qquad S_{IE} = \sum_{i=1}^{B_H} \sum_{j=1}^{B_S} \sum_{k=1}^{B_I} \sqrt{\mathbf{H}_{IN}(i,j,k)\mathbf{H}_{EX}(i,j,k)} \tag{4.6}$$

O parâmetro K permite balancear a influência das contribuições das similaridades e foi ajustado para $K = 1.5$. A métrica deve ser próxima de zero, quando o histograma interior H_{IN} é semelhante ao histograma do modelo H_{MO} e ao mesmo tempo, diferente do histograma exterior H_{EX}. Por fim, o

modelo de observação é modelado segundo uma distribuição laplaciana sobre a distância $\mathcal{D}$:

$$p(\mathbf{z}_t|\mathbf{x}_t^{(i)}) \propto e^{-\frac{\mathcal{D}}{\epsilon}} \tag{4.7}$$

onde o valor de ϵ foi ajustado para $\epsilon = 1/30$ [4].

4.2 Arquiteturas dos algoritmos

Tendo-se explicado na seção anterior o algoritmo de seguimento que suporta este trabalho, agora são apresentadas as arquiteturas dos algoritmos desenvolvidos. Primeiro é enunciado o sistema baseado no filtro de partículas e de seguida o sistema baseado no UPF.

Primeiro Algoritmo

Na figura 4.2 é apresentada a arquitetura baseada no filtro de partículas. O sistema é composto por seis blocos:

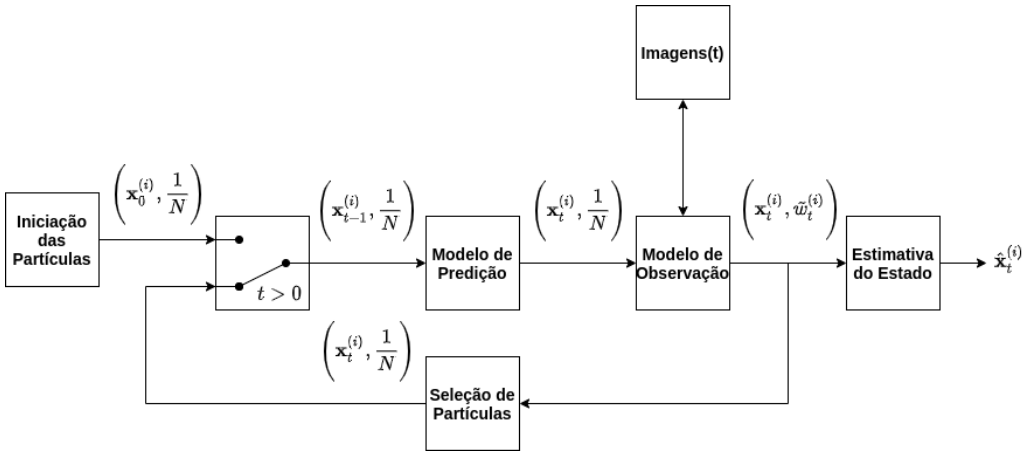


Figura 4.2: Arquitetura do sistema baseado no filtro de partículas.

- Iniciação das Partículas: Este bloco consiste em iniciar todas as N partículas do algoritmo. Como este trabalho apenas se foca no seguimento e não na localização, as partículas são iniciadas ao redor do estado inicial exato que é conhecido, segundo uma distribuição gaussiana. Se assim não fosse, desconhecendo o estado inicial, o número de partículas a ser utilizado para se obter estimativas adequadas teria de ser um número elevado para cobrir uma grande parte do espaço 3D visível pela câmara. Todos os fatores de importância são iniciados com o valor de $\frac{1}{N}$, dado que ainda não existem dados das observações. As partículas são iniciadas da seguinte maneira:

$$\begin{cases} \mathbf{x}_0^{(i)} = \mathcal{N}(\mathbf{x}_0, \mathbf{P}_0) \\ w_0^{(i)} = \frac{1}{N} \end{cases} \qquad i = 1, ..., N \tag{4.8}$$

onde $\mathbf{x}_0$ representa o estado inicial exato da bola e $\mathbf{P}_0$ a matriz de covariância.

- Modelo de Predição: Este bloco propaga as partículas através de um modelo de movimento:

$$\mathbf{x}_t^{(i)} = f\left(\mathbf{x}_{t-1}^{(i)}\right) + \mathbf{w}_t \qquad i = 1, ..., N \tag{4.9}$$

  O modelo de movimento que foi aplicado é o modelo de velocidade constante explicado na secção anterior, todavia, para ambientes onde o objeto pode colidir com obstáculos, é acrescentado ao modelo anterior um modelo de impacto. Este será explicado na seguinte secção 4.3.3.

- Modelo de Observação: Este modelo tem como propósito calcular a verosimilhança de cada partícula. Para o modelo de predição referido atrás, os factores de importância podem ser calculados directamente dos valores de verosimilhança. Como foi mencionado na secção anterior, dois conjuntos de pontos interior e exterior são projetados na imagem para cada partícula e com a informação cromática desses pontos, são criados histogramas de cor no modelo de cor HSI. Na secção seguinte 4.3.2 será explicado como os pontos das regiões interior e exterior são obtidos. O número de barras que constituem os histogramas são estáticos e foram escolhidos de acordo com [34], pelo que $B_H = 12$, $B_S = 12$ e $B_I = 4$. O menor número de barras para a componente de intensidade, foi optado com o intuito de alcançar uma maior robustez perante variações de iluminação [34]. Os fatores de importância são dados pela seguinte equação:

$$w_t^{(i)} = e^{-\frac{|\mathcal{D}|}{\epsilon}} \qquad i = 1, ..., N \tag{4.10}$$

  e no fim, os fatores são normalizados:

$$\tilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{k=1}^{N} w_t^{(k)}} \qquad i = 1, ..., N \tag{4.11}$$

- Imagens: Este bloco representa a interação do modelo de observação com as imagens adquiridas por uma câmara.

- Estimativa do Estado: Para se obter a estimativa do estado aplica-se a aproximação de *Monte Carlo* ao valor expectável [4]:

$$\hat{\mathbf{x}}_t = \sum_{i=1}^{N} \mathbf{x}_t^{(i)} \tilde{w}_t^{(i)} \tag{4.12}$$

- Seleção de Partículas: É aplicado o método de reamostragem sistemática enunciado na seção 2.4.3. Neste método, a probabilidade de replicar uma partícula é diretamente proporcional ao seu fator de importância, o que permite evitar a degeneração e empobrecimento das partículas. Durante este passo de reamostragem os fatores de importância são recolocados a $\frac{1}{N}$.

Segundo Algoritmo

A arquitetura do algoritmo baseado no *Unscented Particle Filter* está representado na figura 4.3, onde se evidenciam oito blocos. Como o UPF é uma extensão do filtro anterior, existem blocos que de-

sempenham as mesmas operações ou operações bastante semelhantes. Neste sistema, cada partícula contem a hipótese de estado $\mathbf{x}_t^{(i)}$, matriz de covariância $\mathbf{P}_t^{(i)}$ e fator de importância $w_t^{(i)}$.

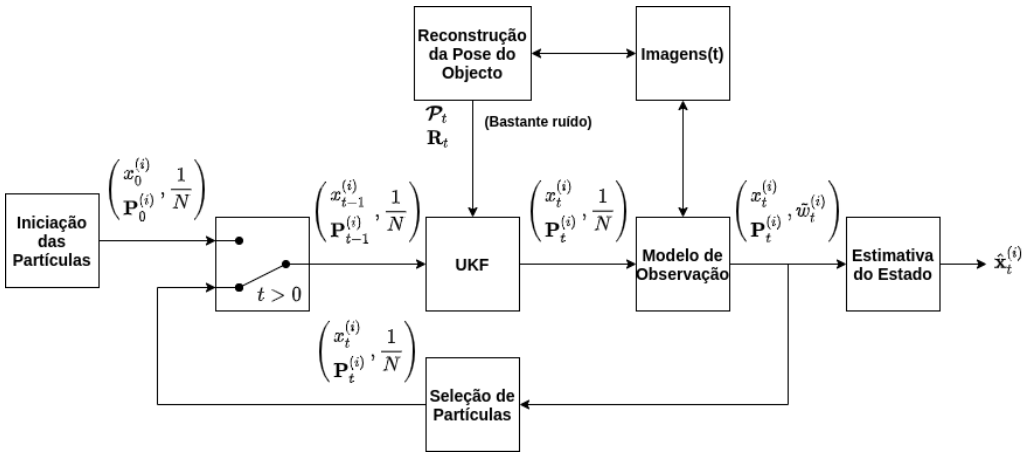


Figura 4.3: Arquitetura do sistema baseado no *unscented particle filter*.

- Iniciação das Partículas: Este bloco é semelhante ao da arquitetura anterior, apenas é acrescentada a iniciação trivial da matriz de covariância para cada partícula.

- Reconstrução da pose do objeto: Este bloco tem o objetivo de estimar a pose atual do objecto no espaço, utilizando um método baseado em contornos. Sendo o objeto uma bola homogénea, o objetivo é estimar apenas a posição da bola. De uma forma generalista, o método efetua um processo de segmentação de cor na imagem atual, obtendo várias hipóteses para a pose do objeto. Seguidamente, são aplicados operadores morfológicos e outras técnicas que selecionam a melhor hipótese para a pose atual da bola. Por fim, com os contornos remanescentes, a pose da bola no espaço é reconstruída e com isso é estimada a sua posição. Este método não foi desenvolvido neste trabalho, mas será utilizado para fornecer observações da posição atual do objeto ao UPF, apesar de apresentar estimativas ruidosas principalmente na dimensão de profundidade em relação à câmara.

- UKF: Uma vez que, a distribuição de importância é o UKF, a propagação de cada partícula é dada pela seguinte equação:

$$\mathbf{x}_t^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}_t^{(i)}, \mathbf{P}_t^{(i)}) \qquad i = 1, ..., N \tag{4.13}$$

  onde $\boldsymbol{\mu}_t^{(i)}$ e $\mathbf{P}_t^{(i)}$ correspondem respetivamente à média e covariância de cada hipótese de estado após aplicar o UKF. Deste modo, para cada partícula são gerados pontos sigma e a estes é aplicado o modelo de movimento do objeto. De seguida, aplicando o modelo de observação do UKF, aos pontos sigma e com a estimativa da posição da bola $\mathcal{P}_t$ e $\mathbf{R}_t$ (como se observa na figura 4.3), obtém-se a média e covariância apresentada na equação acima, do qual se amostra para propagar a partícula.

- Modelo de Observação: Este bloco é similar ao bloco da arquitetura anterior, porém como agora a distribuição de importância é o UKF, os fatores de importância $w_t^{(i)}$ são calculados a partir da

equação geral recursiva dos fatores de importância, dado pela equação (2.58). No fim os fatores são normalizados.

- Imagens: Equivalente ao bloco da arquitetura anterior.
- Estimativa do Estado: Equivalente ao bloco da arquitetura anterior.
- Seleção de Partículas: As partículas são selecionadas da mesma forma do que na arquitetura anterior, mas agora também é replicada a matriz de covariância $\mathbf{P}_t^{(i)}$.

4.3 Métodos implementados nos algoritmos

Como mencionado na secção anterior, o cerne do primeiro algoritmo consiste num filtro de partículas e o segundo num *unscented particle filter*. Nas arquiteturas apresentadas na secção anterior, existem blocos que aplicam métodos genéricos do PF ou do UPF, como por exemplo, a iniciação das partículas, a seleção das partículas a estimativa do estado, etc. Porém, existem blocos que usam métodos específicos como o bloco do modelo de observação, onde são projetados pontos das regiões interior e exterior na imagem, e o bloco do modelo de predição no primeiro algoritmo e o bloco UKF do segundo algoritmo, que acrescentam ao modelo de velocidade constante, um modelo de impacto. Nesta secção, são evidenciados esses métodos específicos que os algoritmos aplicam. O primeiro método refere-se a modelos de projeção. É importante abordar este tópico, porque como existem diferentes tipos de câmaras, existem também variados modelos de projeção. É substancial usar um modelo de projeção adequado à câmara utilizada, para projetar os pontos das regiões interior e exterior de forma apropriada. Depois de se explicar o modelo de projeção utilizado nesta dissertação, será explicado como são obtidos os pontos da região interior e exterior para cada hipótese de estado e finalmente, é explicado o modelo de movimento do objeto incorporando um modelo de impacto.

4.3.1 Modelos de projeção

Calibração de uma câmara é um etapa fundamental em aplicações de visão computacional, mais especificamente em problemas de extração de informação 3D a partir de imagens digitais [38]. Este procedimento consiste em determinar um conjunto de parâmetros que descrevem o mapeamento entre o referencial 3D de coordenadas do espaço tridimensional e as coordenadas 2 Dimensões (2D) das imagens [39]. Estes parâmetros provêm de modelos de projeção desenhados com o intuito de modelar fielmente o comportamento real das câmaras [40]. Normalmente os parâmetros são divididos em extrínsecos e intrínsecos, os parâmetros extrínsecos relacionam o referencial do espaço com o referencial da câmara e os parâmetros intrínsecos transformam as coordenadas dos pontos projetados no plano de imagem em pixeis [39]. Existem diversos modelos de câmara sendo os mais utilizados, os modelos que capturam perspetiva. Um dos modelos mais conhecido e mais utilizado em visão computacional é o modelo de perspetiva exata. Neste modelo, cada ponto no espaço é projetado em linha reta até ao centro ótico intersetando o plano de imagem (ver figura 4.4). Contudo este apenas costuma ser

válido para casos onde as distorções que as lentes das câmaras provocam nas imagens são insignificantes, pelo que normalmente, o modelo de perspetiva exata costuma ser extendido com o propósito de se corrigir essas distorções, nomeadamente distorções radiais e tangenciais [39].

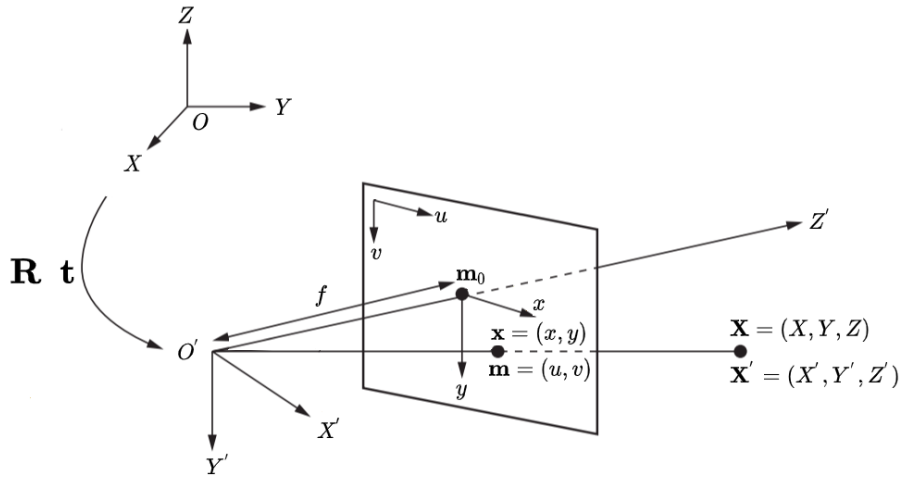


Figura 4.4: Projeção de um ponto do espaço no plano de imagem através do modelo de perspetiva exata [41].

Modelo de projeção perspetiva exata

Este modelo projeta um ponto 3D $\mathbf{X} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$ no plano de imagem, o que resulta no correspondente ponto 2D $\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T$. Porém, normalmente o referencial da câmara não é o mesmo que o referencial do espaço tridimensional, pelo que antes de realizar a simples operação de projeção, é necessário transformar as coordenadas do ponto **X** expressas no sistema de coordenadas do espaço O para o sistema de coordenadas da câmara $O^{'}$ (como se pode observar na figura 4.4), através da matriz de rotação **R** e o vetor de translação **t**, conhecidos por parâmetros extrínsecos:

$$\mathbf{X}^{'} = \mathbf{RX} + \mathbf{t}. \tag{4.14}$$

Deste modo, com o ponto 3D no referencial da câmara, a sua correspondente projeção no plano de imagem aplicando o modelo de perspetiva exata, é facilmente calculado por semelhança de triângulos [42]:

$$\mathbf{x} = \frac{f}{Z^{'}} \begin{bmatrix} X^{'} \\ Y^{'} \end{bmatrix}. \tag{4.15}$$

O próximo passo é converter o ponto para coordenadas de imagem. Normalmente, a origem do sistema de coordenadas da imagem é diferente do da câmara, costuma encontrar-se na esquina superior esquerda da própria imagem, como se observa na figura 4.4. Por vezes, também um possível enviesamento entre os eixos da imagem pode existir. Dito isto, é com os parâmetros intrínsecos que os pontos no plano são transformados em pixeis. Estes são a distância focal f, os fatores de escala d_u e d_v que transformam as unidades métricas em pixeis, o ponto principal $\mathbf{m}_0 = (u_0, v_0)$ e o parâmetro de

enviesamento s. Convertendo o ponto **x** para coordenadas de imagem, obtém-se o ponto **m**:

$$\mathbf{m} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} d_u x + sy + u_0 \\ d_v y + v_0 \end{bmatrix} \tag{4.16}$$

Frequentemente, este modelo costuma ser apresentado integrando os parâmetros extrínsecos e intrínsecos numa só matriz designada por matriz de projeção **P**. Esta representação matricial apenas é possível usando coordenadas homogéneas. Em coordenadas homogéneas um ponto de dimensão n é representado por um vetor de dimensão $n+1$. Por exemplo, o ponto 2D $\mathbf{x} = (x, y)$, em coordenadas homogéneas é representado por $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) = \tilde{w}(x, y, 1)$. O ponto $\tilde{\mathbf{x}}$ pode ser convertido para coordenadas cartesianas dividindo a última coordenada homogénea $\tilde{w}$ pelas restantes, $\mathbf{x} = \left(\frac{\tilde{x}}{\tilde{w}}, \frac{\tilde{y}}{\tilde{w}}\right)$ [40]. Desta forma, a projeção de um ponto no espaço para o plano de imagem em coordenadas homogéneas é dada por:

$$\tilde{\mathbf{m}} \sim \mathbf{P}\tilde{\mathbf{X}} = \mathbf{K}\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}\tilde{\mathbf{X}} \tag{4.17}$$

onde a matriz **K** é uma matriz triangular superior que contém os parâmetros intrínsecos e é dada por [40]:

$$\mathbf{K} = \begin{bmatrix} fd_u & s & u_0 \\ 0 & fd_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.18}$$

e $\sim$ indica que o ponto 2D $\tilde{\mathbf{m}}$ é diretamente proporcional a $\mathbf{P}\tilde{\mathbf{X}}$. Para suprimir a constante de proporcionalidade basta transformar o ponto $\tilde{\mathbf{m}}$ de coordenadas homogéneas para cartesianas.

Introduzindo correções para distorções radiais e tangenciais

O modelo anterior não tem em consideração possíveis distorções que as lentes das câmaras provocam nas imagens. Em aplicações que requerem grande precisão, é fundamental corrigir estas distorções para se obter resultados adequados [39]. Neste trabalho apenas serão referidas distorções radiais e distorções tangenciais. A distorção radial é a deformação mais frequentemente presente nas câmaras, esta pode ser aproximada pela seguinte expressão:

$$\begin{bmatrix} \delta x^{(r)} \\ \delta y^{(r)} \end{bmatrix} = \begin{bmatrix} x(k_1 r^2 + k_2 r^4 + ... + k_n r^{2n}) \\ y(k_1 r^2 + k_2 r^4 + ... + k_n r^{2n}) \end{bmatrix} \tag{4.19}$$

onde $\delta x^{(r)}$ e $\delta y^{(r)}$ representam a componente de distorção radial nos eixos x e y, $k_1, k_2, ..., k_n$ são coeficientes que parametrizam a distorção radial e $r = \sqrt{x^2 + y^2}$. Tipicamente, os primeiros dois coeficientes são suficientes para compensar a distorção [39]. Na figura 4.5 pode-se observar o efeito que a distorção radial cria na imagem.

Um outro tipo de distorção comum é a distorção tangencial, esta surge quando os centros óticos das lentes não se encontram bem alinhados [43]. A expressão para a distorção tangencial é normalmente

Figura 4.5: Nesta figura, um tabuleiro axadrezado é apresentado em frente a uma câmara. É bastante notório o efeito esférico que a distorção radial causa nos quadrados do tabuleiro [43].

escrita da seguinte forma:

$$\begin{bmatrix} \delta x^{(t)} \\ \delta y^{(t)} \end{bmatrix} = \begin{bmatrix} 2p_1xy + p_2(r^2 + 2x^2) \\ p_1(r^2 + 2y^2) + 2p_2xy \end{bmatrix} \tag{4.20}$$

onde $\delta x^{(t)}$ e $\delta y^{(t)}$ representam a componente de distorção tangencial nos eixos x e y e p_1 e p_2 são coeficientes que parametrizam a distorção tangencial. Tendo em conta estas distorções, um modelo de câmara mais apropriado pode ser derivado. Inicialmente um ponto 3D **X** é transformado para o referencial da câmara a partir dos parâmetros extrínsecos e depois é efetuada a projeção do ponto no plano de imagem obtendo-se **x**, através do modelo de projeção em perspetiva exata. De seguida são aplicadas as correções ao ponto projetado, da seguinte forma [39]:

$$\mathbf{x}^{'} = \begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = \begin{bmatrix} x + \delta x^{(r)} + \delta x^{(t)} \\ y + \delta y^{(r)} + \delta y^{(t)} \end{bmatrix} \tag{4.21}$$

e finalmente, os parâmetros intrínsecos são aplicados ao ponto corrigido:

$$\mathbf{m} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} d_u x^{'} + s y^{'} + u_0 \\ d_v y^{'} + v_0 \end{bmatrix} \tag{4.22}$$

4.3.2 Obtenção dos pontos das regiões interior e exterior

Um processo que projete uma hipótese de estado da bola na imagem é essencial. É com esta projeção que os histogramas de cor são calculados com o intuito de avaliar a importância da partícula na estimação do estado. Para este propósito, pretende-se compor uma técnica de projeção rigorosa mas também eficiente, porque esta operação será realizada para cada partícula. Uma bola é uma superfície esférica e uma superfície esférica é um caso particular de uma quádrica. No trabalho desenvolvido por *Cross e Zisserman* [44] constata-se que a projeção da silhueta de uma quádrica numa imagem em perspetiva, resulta numa cónica e para o caso particular de uma superfície esférica, a cónica resulta numa elipse (como se observa na figura 4.6).

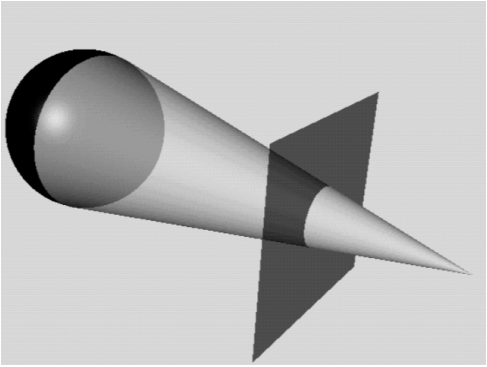


Figura 4.6: Projeção de uma superfície esférica num plano de imagem, que resulta numa elipse [44].

Uma quádrica é uma superfície representada por uma matriz 4×4 simétrica **Q**, de tal modo que qualquer ponto 3D em coordenadas homogéneas $\tilde{\mathbf{X}} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$ na superfície, satisfaz a seguinte restrição quadrática:

$$\tilde{\mathbf{X}}^T \mathbf{Q} \tilde{\mathbf{X}} = 0. \tag{4.23}$$

Para o caso de uma superfície esférica centrada em $\mathbf{x}_0 = \begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}^T$, com raio R tem-se [45]:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{I}_3 & -\mathbf{x}_0 \\ -\mathbf{x}_0^T & ||\mathbf{x}_0||^2 - R^2 \end{bmatrix}. \tag{4.24}$$

A silhueta de uma quádrica numa imagem em perspetiva é uma cónica. Uma cónica é representada por uma matriz 3×3 simétrica **C** no qual, qualquer ponto $\tilde{\mathbf{x}} = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$ que lhe pertence, satisfaz a seguinte restrição [44]:

$$\tilde{\mathbf{x}}^T \mathbf{C} \tilde{\mathbf{x}} = 0. \tag{4.25}$$

Ora, existe uma expressão que relaciona uma cónica projetada na imagem, com a silhueta de uma quádrica e é dada por [46]:

$$k\mathbf{C}^{-1} = \mathbf{P}\mathbf{Q}^{-1}\mathbf{P}^T \tag{4.26}$$

onde a matriz $\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}$ corresponde à matriz de projeção e k é um fator de escala desconhecido. Admitindo que o referencial da câmara coincide com o referencial do espaço (isto é, os parâmetros extrínsecos correspondem a $\mathbf{R} = \mathbf{I}_3$, $\mathbf{t} = \mathbf{0}$) e que a quádrica é a superfície esférica representada em (4.24), com a inversa de **Q** representada abaixo [45]:

$$\mathbf{Q}^{-1} = \begin{bmatrix} \mathbf{I}_3 - \mathbf{x}_0\mathbf{x}_0^T/R^2 & -\mathbf{x}_0/R^2 \\ -\mathbf{x}_0^T/R^2 & -1/R^2 \end{bmatrix} \tag{4.27}$$

obtém-se:

$$\mathbf{C} = k \cdot \mathbf{K}^{-T} \left(\mathbf{I}_3 - \frac{\mathbf{x}_0\mathbf{x}_0^T}{R^2} \right)^{-1} \mathbf{K}^{-1}. \tag{4.28}$$

Dado que $\tilde{\mathbf{x}}^T\mathbf{C}\tilde{\mathbf{x}} = 0$ e $\tilde{\mathbf{x}}^T k\mathbf{C}\tilde{\mathbf{x}} = 0$ representam a mesma cónica, o fator de escala k pode ser ignorado. Assim, para um ponto na imagem $\tilde{\mathbf{m}} = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$ pertencer à projeção da superfície esférica,

necessita de satisfazer a seguinte equação:

$$\begin{aligned} \tilde{\mathbf{m}}^T \mathbf{K}^{-T} \left(\mathbf{I}_3 - \frac{\mathbf{x}_0 \mathbf{x}_0^T}{R^2} \right)^{-1} \mathbf{K}^{-1} \tilde{\mathbf{m}}^T = 0 \\ \tilde{\mathbf{m}}^T \mathbf{K}^{-T} \underbrace{\left(\mathbf{I}_3 - \frac{\mathbf{x}_0 \mathbf{x}_0^T}{||\mathbf{x}_0||^2 - R^2} \right)}_{=\mathbf{H}} \mathbf{K}^{-1} \tilde{\mathbf{m}}^T = 0. \end{aligned} \tag{4.29}$$

Sabendo que a cónica resultante é uma elipse, é possível encontrar uma solução simples e bastante eficiente que transforma uma circunferência centrada na origem, na elipse arbitrária representada pela matriz **C**. Esta solução é exequível através da decomposição em valores próprios da matriz **H** e fazendo com que a matriz que contém os vetores próprios **V**, seja ortonormal. Se **V** for ortonormal, então $\mathbf{V}^{-1} = \mathbf{V}^T$ e assim $\mathbf{H} = \mathbf{V}\mathbf{S}\mathbf{V}^T$, onde a matriz **S** é a matriz diagonal que contém os valores próprios. Executando a decomposição em valores próprios da forma mencionada, obtém-se **S** e **V** (a obtenção destas matrizes podem ser consultadas no apêndice B):

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \dfrac{1}{1 - \left|\left| \dfrac{\mathbf{x}_0}{R} \right|\right|^2} \end{bmatrix}, \qquad \mathbf{V} = \begin{bmatrix} \dfrac{z_0}{k_1} & \dfrac{x_0 y_0 + y_0^2 + z_0^2}{k_2} & \dfrac{x_0}{k_3} \\ \dfrac{z_0}{k_1} & -\dfrac{x_0 y_0 + x_0^2 + z_0^2}{k_2} & \dfrac{y_0}{k_3} \\ -\dfrac{x_0 + y_0}{k_1} & \dfrac{z_0(y_0 - x_0)}{k_2} & \dfrac{z_0}{k_3} \end{bmatrix} \tag{4.30}$$

onde:

$$k_1 = \sqrt{2z_0^2 + (x_0 + y_0)^2} \tag{4.31}$$

$$k_2 = \sqrt{(x_0 y_0 + y_0^2 + z_0^2)^2 + (x_0 y_0 + x_0^2 + z_0^2)^2 + z_0^2 (y_0 - x_0)^2} \tag{4.32}$$

$$k_3 = \sqrt{x_0^2 + y_0^2 + z_0^2}. \tag{4.33}$$

Deste modo, alterando a equação acima (4.29) para:

$$\tilde{\mathbf{m}}^T \mathbf{K}^{-T} \mathbf{V} \mathbf{S} \mathbf{V}^T \mathbf{K}^{-1} \tilde{\mathbf{m}} = 0 \tag{4.34}$$

e efetuando a transformação $\tilde{\mathbf{m}}_p = \begin{bmatrix} u_p & v_p & 1 \end{bmatrix}^T = \mathbf{V}^T \mathbf{K}^{-1} \tilde{\mathbf{m}}$, como os maiores valores próprios apresentados em **S** são ambos idênticos e unitários é possível mapear a elipse numa circunferência:

$$\begin{aligned} \tilde{\mathbf{m}}_p^T \mathbf{S} \tilde{\mathbf{m}}_p = 0 \\ u_p^2 + v_p^2 = \frac{1}{\left|\left| \dfrac{\mathbf{x}_0}{R} \right|\right|^2 - 1}. \end{aligned} \tag{4.35}$$

Assim, obter um conjunto de pontos que projetam a elipse resultante é realizado amostrando pontos desta circunferência e invertendo a transformação que se fez. Como foi dito anteriormente, é gerado um conjunto de pontos interior e um conjunto de pontos exterior à superfície esférica. Para isso, em vez de se utilizar o raio R exato da bola, é usado um raio inferior para gerar os pontos interiores, por exemplo

$0.95R$ e um raio superior para os pontos exteriores, por exemplo $1.05R$ (na figura 4.7 encontra-se um exemplo da projeção destes pontos). O método pode ser efetuado da seguinte forma:

São amostrados N pontos da circunferência, com raio interior $r_{IN} = \sqrt{1/\left(\left|\left|\frac{\mathbf{x}_0}{(1-p)R}\right|\right|^2 - 1\right)}$ e raio exterior $r_{EX} = \sqrt{1/\left(\left|\left|\frac{\mathbf{x}_0}{(1+p)R}\right|\right|^2 - 1\right)}$ com $0 < p < 1$, entre $0 \leq \theta_i < 2\pi,\ i = 1, ..., N$:

$$\tilde{\mathbf{m}}_p^{IN} = \begin{bmatrix} r_{IN}\cos(\theta_1) & ... & r_{IN}\cos(\theta_N) \\ r_{IN}\sin(\theta_1) & ... & r_{IN}\sin(\theta_N) \\ 1 & ... & 1 \end{bmatrix}, \qquad \tilde{\mathbf{m}}_p^{EX} = \begin{bmatrix} r_{EX}\cos(\theta_1) & ... & r_{EX}\cos(\theta_N) \\ r_{EX}\sin(\theta_1) & ... & r_{EX}\sin(\theta_N) \\ 1 & ... & 1 \end{bmatrix} \tag{4.36}$$

de seguida, multiplicando os pontos pela inversão da transformação $\left(\mathbf{V}^T\mathbf{K}^{-1}\right)^{-1} = \mathbf{KV}$, obtém-se os pontos na elipse, $\tilde{\mathbf{m}}^{IN} = \mathbf{KV}\tilde{\mathbf{m}}_p^{IN}$ e $\tilde{\mathbf{m}}^{EX} = \mathbf{KV}\tilde{\mathbf{m}}_p^{EX}$. Por fim, converte-se de coordenadas homogéneas para cartesianas.

Figura 4.7: Projeção de uma hipótese de estado que coincide com o estado exato da bola. Observa-se a branco a silhueta exata da bola e a preto os pontos da região interior e exterior [4].

4.3.3 Modelo de movimento incorporando colisões

Neste trabalho pretende-se expandir o modelo de movimento com o objetivo de abordar colisões com obstáculos. Num impacto, os corpos que colidem, exercem grandes forças um no outro, pelo que as velocidades dos corpos após o impacto podem ser bastante distintas das velocidades antes do embate. Nestas circunstâncias, o modelo de velocidade constante enunciado em 4.1.1, poderá não propagar as partículas devidamente. Na sua generalidade, um modelo de impacto pode ser dividido em vários [47]:

- Perfeitamente elástico, não ocorrem perdas de energia.
- Parcialmente elástico, há perdas de energia, mas não há deformação permanente dos corpos.

- Parcialmente plástico, ocorre perda de energia e deformação permanente.
- Perfeitamente plástico, toda a energia é dissipada no impacto [47].

O coeficiente de restituição e, é um parâmetro utilizado para determinar a perda de energia que ocorre durante um impacto. O valor deste está contido entre $0 \leq e \leq 1$. Se $e = 0$, a colisão é perfeitamente plástica (total perda de energia), se $e = 1$ a colisão é perfeitamente elástica (não ocorrem perdas de energia). A vantagem deste coeficiente é que engloba várias contribuições dos fenómenos que ocorrem no impacto sem ter que os conhecer e modelar, tais como as propriedades materiais dos corpos, as suas geometrias, velocidades dos corpos, duração da colisão, etc [48]. Na figura 4.8 observa-se um

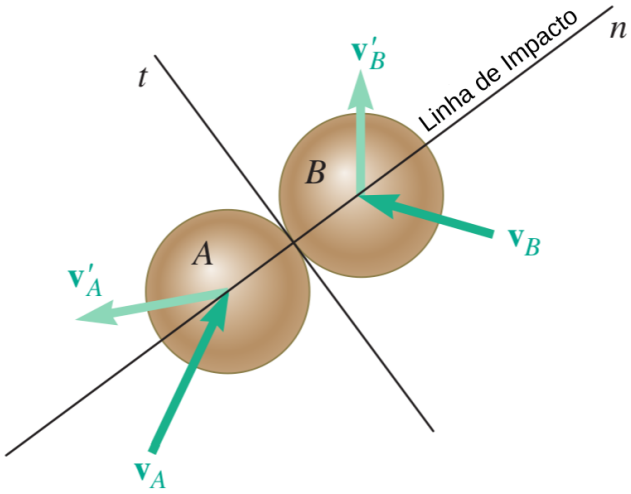


Figura 4.8: Impacto central entre os corpos A e B. $\mathbf{v}'_A$ e $\mathbf{v}'_B$ representam as velocidades após o impacto [47].

impacto central onde a normal comum às superfícies em contacto no embate é chamada de linha de impacto. Este é chamado de central se os centros de massa dos corpos que colidem, se localizam nesta linha. O propósito do coeficiente de restituição é obter a velocidade relativa dos corpos depois do impacto. Num impacto central, existem casos especiais em que se pode assumir que os corpos são perfeitamente lisos e que apresentam superfícies com coeficientes de atrito desprezáveis. Nestas condições, apenas as componentes normais das velocidades são afetadas e assim para os corpos A e B, tem-se:

$$e = -\frac{(\mathbf{v}'_A)_n - (\mathbf{v}'_B)_n}{(\mathbf{v}_A)_n - (\mathbf{v}_B)_n} \tag{4.37}$$

$$(\mathbf{v}'_A)_t = (\mathbf{v}_A)_t, \quad (\mathbf{v}'_B)_t = (\mathbf{v}_B)_t \tag{4.38}$$

onde n e t indicam respetivamente as componentes da velocidade normal e tangencial à superfície de contacto na colisão, $\mathbf{v}_A$ e $\mathbf{v}_B$ correspondem às velocidades dos corpos imediatamente antes do impacto e $\mathbf{v}'_A$ e $\mathbf{v}'_B$, representam as velocidades imediatamente após o impacto [47]. Tendo-se explicado o coeficiente de restituição, pretende-se modelar o movimento de uma bola tendo em conta que esta possa colidir com obstáculos imóveis. Assumindo que as colisões são parcialmente elásticas, centrais e que as superfícies de impacto da bola e do obstáculo apresentam coeficientes de atrito desprezáveis,

como o obstáculo permanece estático, a velocidade da bola após o impacto $\mathbf{v}'_{Bola}$ é dada por:

$$(\mathbf{v}'_{Bola})_n = -e(\mathbf{v}_{Bola})_n, \qquad (\mathbf{v}'_{Bola})_t = (\mathbf{v}_{Bola})_t \tag{4.39}$$

combinando as duas equações, obtém-se:

$$\mathbf{v}'_{Bola} = \mathbf{v}_{Bola} - (1+e)\big({\mathbf{v}_{Bola}}^T\mathbf{n}\big)\mathbf{n} \tag{4.40}$$

onde **n** corresponde ao vetor normalizado na direção da linha de impacto.

Tendo-se mencionado o modelo de colisão, irá se explicar como este é incorporado no modelo de velocidade constante referido em 4.1.1. Abaixo apresenta-se o algoritmo deste modelo que complementará a explicação apresentada neste parágrafo. Dito isto, primeiro as partículas são propagadas normalmente através do modelo de velocidade constante (linha 1 do algoritmo 3), mas caso uma ultrapasse o obstáculo (como se constata na figura 4.9), isto indica que houve uma colisão (linha 2) e que deve ser aplicado o modelo de impacto. Para isso, é calculado o tempo t_c (linha 3) no qual se dá a colisão e em vez de se propagar a partícula usando o tempo de amostragem Δ_t, é usado t_c (linha 4). De seguida, é aplicada a equação acima (4.40) para obter a velocidade imediatamente após o impacto (linha 5) e por fim, é iterado novamente o modelo de velocidade constante mas usando o tempo $\Delta_t - t_c$ (linha 6).

Algoritmo 3: MODELO DE VELOCIDADE CONSTANTE INCORPORANDO COLISÕES

1 $\mathbf{x}_t^{(i)} = \begin{bmatrix} \mathbf{I}_3 & \Delta_t\mathbf{I}_3 \\ \mathbf{0} & \mathbf{I}_3 \end{bmatrix} \mathbf{x}_{t-1}^{(i)} + \begin{bmatrix} \frac{\Delta_t^2}{2}\mathbf{I}_3 \\ \Delta_t\mathbf{I}_3 \end{bmatrix} \mathbf{a}_t^{(i)}, \qquad i = 1, ..., N$

2 SE COLISÃO $\left(\mathbf{x}_t^{(i)}\right), \qquad i = 1, ..., N$

{

3 $\quad t_c = \text{TEMPO_DE_COLISÃO}\left(\mathbf{x}_{t-1}^{(i)}\right)$

4 $\quad \mathbf{x}_t^{'(i)} = \begin{bmatrix} \mathbf{I}_3 & t_c\mathbf{I}_3 \\ \mathbf{0} & \mathbf{I}_3 \end{bmatrix} \mathbf{x}_{t-1}^{(i)} + \begin{bmatrix} \frac{t_c^2}{2}\mathbf{I}_3 \\ t_c\mathbf{I}_3 \end{bmatrix} \mathbf{a}_t^{(i)}$

5 $\quad \mathbf{x}_t^{''(i)} = \begin{bmatrix} \mathbf{p}_t^{'(i)} \\ \dot{\mathbf{p}}_t^{'(i)} - (1+e)\left(\dot{\mathbf{p}}_t^{'(i)^T}\mathbf{n}\right)\mathbf{n} \end{bmatrix}$

6 $\quad \mathbf{x}_t^{(i)} = \begin{bmatrix} \mathbf{I}_3 & (\Delta_t - t_c)\mathbf{I}_3 \\ \mathbf{0} & \mathbf{I}_3 \end{bmatrix} \mathbf{x}_t^{''(i)} + \begin{bmatrix} \frac{(\Delta_t - t_c)^2}{2}\mathbf{I}_3 \\ (\Delta_t - t_c)\mathbf{I}_3 \end{bmatrix} \mathbf{a}_t^{(i)}$

{

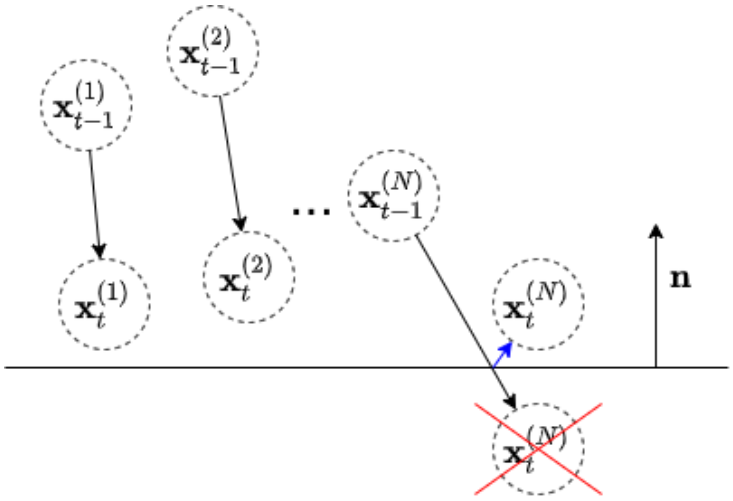


Figura 4.9: Impacto entre a partícula N e um obstáculo. Nesta situação, o modelo de impacto é aplicado.

Capítulo 5

Implementação

Neste capítulo são evidenciados todos os conceitos que foram implementados com o objetivo de criar e avaliar os algoritmos. Primeiro será apresentado um simulador que foi criado para simular uma câmara a adquirir imagens de uma bola percorrendo uma trajetória. A vantagem de se criar um simulador é que desta maneira o estado exato da bola é conhecido e assim, é possível avaliar de forma exata a eficácia dos algoritmos. De seguida, como foi enunciado no capítulo anterior, pretende-se inserir no modelo de movimento um modelo de impacto, por isso, também se integra no simulador a possibilidade de inserir obstáculos nas trajetórias simuladas e depois é explicado como os algoritmos detetam e estimam o obstáculo. Posteriormente é mencionado como a câmara real foi calibrada para executar as experiências reais e finalmente são mencionados de forma resumida, alguns conceitos que se implementaram no código para realizar os filtros.

5.1 Simulador

Para criar o simulador usaram-se as funcionalidades da biblioteca *openCV*. A biblioteca especializa-se na área de visão computacional e por isso apresenta bastantes funções de manipulação de imagens. Para simular uma experiência real de uma bola seguindo uma trajetória, é necessário modelar a câmara e a bola. Para simular a câmara usou-se o modelo de perspetiva exata mencionado em 4.3.1. Com este modelo, assumindo que o referencial de coordenadas do espaço coincide com o referencial da câmara, apenas é necessário definir os parâmetros intrínsecos para modelar a câmara. Para a bola, é necessário definir o seu modelo geométrico e o seu modelo de cor. O modelo geométrico considerado corresponde à equação cartesiana da superfície esférica no espaço:

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = R^2 \tag{5.1}$$

onde $\mathbf{x}_0 = (x_0, y_0, z_0)$ e R representam respetivamente o centro e o raio da superfície esférica e qualquer ponto $\mathbf{x} = (x, y, z)$ que satisfazer a equação, pertence à superfície esférica. O modelo de cor da bola é representado por um histograma de cor e apesar de se referir que neste trabalho apenas se lida

com objetos de cor homogénea, não será definida apenas uma cor para colorir a bola. Num ambiente real, uma bola de cor homogénea não apresenta somente uma cor, pois existem fatores como variações de iluminação e ruído que afeta o sensor ótico da câmara. Dito isto, tiraram-se fotos a uma bola de cor homogénea e usaram-se apenas os pixeis dessas imagens que pertencem à bola, para formar o histograma de cor da bola. Com a geometria e cor definidas, um método que pode ser aplicado para projetar a bola na imagem, consiste em usar os pontos 3D que satisfazem a equação da superfície esférica e projetar estes na imagem, colorindo os pixeis correspondentes usando cores aleatórias geradas de acordo com a distribuição dada pelo histograma de cor da bola. Porém, existem infinitos pontos que satisfazem a equação e devido a isso, decidiu-se aplicar um método diferente. Este, consiste em gerar uma recta que passe no centro ótico da câmara e que atravesse o centro de um pixel, caso a recta intersecte também a equação da superfície esférica, indica que a bola está a ser projetada nesse pixel e por isso, é colorido com uma cor proveniente do modelo de cor da bola (uma ilustração do funcionamento do método pode ser observado na figura 5.1). Este método é realizado para todos os pixeis da imagem.

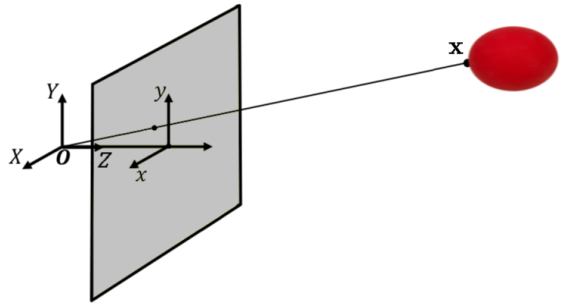


Figura 5.1: Reta que atravessa um pixel no plano de imagem de uma câmara e intersecta um ponto **x** da bola.

A equação de uma recta que passe no centro ótico e que atravesse um pixel **m** no plano de imagem, pode ser escrita em função desse pixel em coordenadas homogéneas:

$$\lambda \mathbf{K}^{-1}\tilde{\mathbf{m}} = \lambda \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{5.2}$$

onde a matriz $\mathbf{K}^{-1}$ corresponde à inversa da matriz dos parâmetros intrínsecos, que nesta equação transforma o pixel para coordenadas métricas. Substituindo a equação (5.2) em (5.1) e resolvendo em ordem a λ é possível encontrar as raízes da equação quadrática. Se as raízes forem reais, a reta interseta a superfície esférica. A seguinte inequação se for satisfeita, demonstra que as raízes da equação são reais:

$$\left(\tilde{\mathbf{m}}^T \mathbf{K}^{-T} \mathbf{x}_0\right)^2 - \tilde{\mathbf{m}}^T \mathbf{K}^{-T} \mathbf{K}^{-1} \tilde{\mathbf{m}} \left(\mathbf{x}_0^T \mathbf{x}_0 - R^2\right) \geq 0. \tag{5.3}$$

Desta maneira, é efetuado um varrimento a todos os pixeis e os que satisfazem a inequação são coloridos. Na figura abaixo 5.2(a), pode-se observar o resultado deste método, no qual é bastante

visível o efeito de pixelização na silhueta da bola. Pois, embora os pixeis sejam pequenos, estes representam à mesma uma área na imagem, pelo que existem situações em que a projeção da bola interseta parcialmente o pixel, e se a inequação for satisfeita, todo o pixel é colorido, pelo contrário não é colorido e o pixel mantém a cor que já tinha. Uma técnica que se pode usar para mitigar o efeito de pixelização na silhueta da bola, é interpolação de cor. Entre duas cores, a técnica pode ser usada aplicando o método de interpolação linear entre as componentes R, G e B de cada cor. Desta forma a cor resultante **c** é dada por:

$$\mathbf{c} = a\mathbf{c}_M + (1-a)\mathbf{c}_F = a \begin{bmatrix} c_M(R) \\ c_M(G) \\ c_M(B) \end{bmatrix} + (1-a) \begin{bmatrix} c_F(R) \\ c_F(G) \\ c_F(B) \end{bmatrix} \tag{5.4}$$

onde a representa a percentagem de interseção da projeção da bola com o pixel, $\mathbf{c}_M$ corresponde a uma cor do modelo de cor da bola e $\mathbf{c}_F$ corresponde à cor corrente do pixel antes da projeção da bola. Se $a = 1$, a projeção da bola cobre todo o pixel e assim $\mathbf{c} = \mathbf{c}_M$, se $a = 0$ nenhuma porção da bola é projectada sobre o pixel $\mathbf{c} = \mathbf{c}_F$, ou seja, o pixel permanece com a cor que tinha (na figura 5.2(b) observa-se o resultado da técnica). Para calcular a percentagem de interseção é usado o método de integração de *Monte Carlo*. Desta maneira, para cada pixel **m** são gerados N pontos aleatórios, distribuídos de forma uniforme entre os limites das coordenadas que definem o pixel:

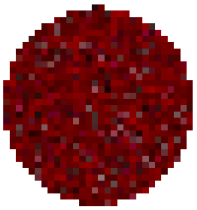
$$\tilde{\mathbf{m}}_i = \begin{bmatrix} \mathcal{U}(u, u+1) \\ \mathcal{U}(v, v+1) \\ 1 \end{bmatrix}, \qquad i = 1, ..., N \tag{5.5}$$

a percentagem de interseção é dada pelo número de pontos que satisfazem a inequação (5.3), sobre o número total de pontos. Ou seja:

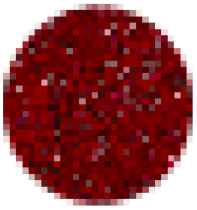
$$a = \frac{\sum_{i=0}^{N} H(\tilde{\mathbf{m}}_i)}{N} \tag{5.6}$$

onde:

$$H(\tilde{\mathbf{m}}_i) = \begin{cases} 1, & \left(\tilde{\mathbf{m}}^T \mathbf{K}^{-T} \mathbf{x}_0\right)^2 - \tilde{\mathbf{m}}^T \mathbf{K}^{-T} \mathbf{K}^{-1} \tilde{\mathbf{m}} \left(\mathbf{x}_0^T \mathbf{x}_0 - R^2\right) \geq 0 \\ 0, & \text{contrário} \end{cases} . \tag{5.7}$$



(a) Sem interpolação de cor



(b) Com interpolação de cor

Figura 5.2: Resultado da projeção da bola numa imagem. Na figura 5.2(a) é apresentada a projeção da bola sem interpolação de cor e na figura 5.2(b) com interpolação de cor.

5.1.1 Projeção de uma superfície planar quadrada na imagem

Com a finalidade de testar as vantagens do modelo de colisões introduzido anteriormente em 4.3.3, simularam-se obstáculos com o qual a bola pode colidir. Neste caso os obstáculos serão superfícies planares quadradas imóveis de cor homogénea. Da mesma forma que se projetou a bola, o mesmo se realiza para projetar a superfície planar quadrada. Começa-se por definir a equação cartesiana que define um plano no espaço:

$$ax + by + cz = ax_0 + by_0 + cz_0 \tag{5.8}$$

onde $\mathbf{x}_0 = (x_0, y_0, z_0)$ corresponde a um ponto no plano, $\mathbf{n} = (a, b, c)$ à normal ao plano e qualquer ponto $\mathbf{x} = (x, y, z)$ que satisfazer a equação acima, pertence ao plano. Substituindo a equação (5.2) em (5.8) e resolvendo em ordem a λ obtém-se:

$$\lambda = \frac{\mathbf{n}^T \mathbf{x}_0}{\mathbf{n}^T \mathbf{K}^{-1} \tilde{\mathbf{m}}} \tag{5.9}$$

desta forma facilmente se obtém o ponto **x** no plano, no qual a reta interseta:

$$\mathbf{x} = \lambda \mathbf{K}^{-1} \tilde{\mathbf{m}} = \frac{\mathbf{n}^T \mathbf{x}_0}{\mathbf{n}^T \mathbf{K}^{-1} \tilde{\mathbf{m}}} \mathbf{K}^{-1} \tilde{\mathbf{m}}. \tag{5.10}$$

Definindo um quadrado contido no plano, aplicando a equação (5.10) para cada pixel, é possível testar se este intersecta o quadrado, caso intersecte, o pixel é colorido. Conhecendo já a normal **n**, para definir um quadrado basta conhecer o ponto central e um vértice. Tomando como centro do quadrado o ponto $\mathbf{x}_0$ e definindo outro ponto como vértice $\mathbf{v}_1$, os restantes vértices $\mathbf{v}_2$, $\mathbf{v}_3$ e $\mathbf{v}_4$ são facilmente determinados aplicando a formula de *Rodrigues*:

$$\mathbf{v}_i = \mathbf{x}_0 + (\mathbf{v}_1 - \mathbf{x}_0)\cos(\theta) + (\hat{\mathbf{n}} \times (\mathbf{v}_1 - \mathbf{x}_0))\sin(\theta) + \hat{\mathbf{n}}(\hat{\mathbf{n}} \cdot (\mathbf{v}_1 - \mathbf{x}_0))(1 - \cos(\theta)), \quad \theta = \frac{\pi}{2}(i-1), \quad i = 2, 3, 4 \tag{5.11}$$

onde $\hat{\mathbf{n}}$ representa o vetor normal à superfície quadrada normalizado. Para saber se o ponto **x** pertence ao quadrado, será usada a seguinte equação que expressa a posição do ponto **x** em relação ao primeiro vértice do quadrado $\mathbf{v}_1$ a partir da combinação linear:

$$\mathbf{x} = \mathbf{v}_1 + \alpha(\mathbf{v}_2 - \mathbf{v}_1) + \beta(\mathbf{v}_4 - \mathbf{v}_1) \tag{5.12}$$

onde $\mathbf{v}_2$ e $\mathbf{v}_4$ são os vértices adjacentes a $\mathbf{v}_1$. Transformando a equação para matricial e invertendo a matriz, α e β são dados por:

$$\begin{bmatrix} 1 \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 - \mathbf{v}_1 & \mathbf{v}_4 - \mathbf{v}_1 \end{bmatrix}^{-1} \mathbf{x}. \tag{5.13}$$

Se os escalares $\alpha, \beta \in [0, 1]$, facilmente se deduz que o ponto pertence ao quadrado e assim, os pixeis que satisfazem esta restrição são coloridos. O mesmo método de interpolação de cor que se aplicou para projetar a bola é replicado para o quadrado.

Adição de marcadores

Quando as imagens simuladas forem usadas para testar os algoritmos, estes terão que ter conhecimento da pose do obstáculo. Para este fim são inseridos quatro marcadores especiais, um por cada vértice do quadrado, todos colocados sobre as diagonais do quadrado a distâncias idênticas dos vértices. O uso de marcadores é uma abordagem bastante atrativa pois são simples de detetar e atingem resultados bastante precisos de forma eficiente [49] na estimação da sua pose. Ou seja, inserindo os marcadores no obstáculo e obtendo a pose destes, será possível estimar a pose do obstáculo. Os marcadores utilizados, pertencem a um módulo implementado no *OpenCV* baseado na biblioteca *ArUCo* desenvolvida por *S. Garrido-Jurado et al* [49]. O módulo contém funções para gerar marcadores, detetar e estimar a sua pose. Um marcador é um quadrado com tamanho específico l e com textura única como se observa na figura 5.3, onde cada ponto 2D no marcador apresenta uma cor específica.

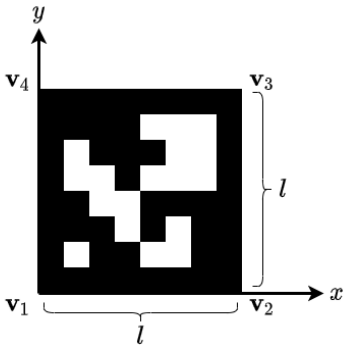


Figura 5.3: Exemplo de um marcador desenvolvido por *S. Garrido-Jurado et al* [49], onde **v1**, **v2**, **v3** e **v4** representam os vértices e l o tamanho do marcador.

Dado que o modelo geométrico de um marcador é um quadrado, as equações que se usaram para projetar o obstáculo podem ser reutilizadas. Porém como a textura não é uniforme, é necessário mapear o ponto 3D que intersecta o marcador para o ponto 2D correspondente e aceder à sua cor para colorir o pixel. Com o sistema de coordenadas representado na figura 5.3, onde um ponto 2D $\mathbf{p} = (x, y)$ no marcador é referenciado em relação ao primeiro vértice de forma semelhante à enunciada na equação (5.12), facilmente se comprova que o ponto 2D correspondente é dado por $\mathbf{p} = l\,(\alpha, \beta)$. Por fim, o mesmo processo de interpolação de cor já usado anteriormente é também aplicado para colorir os pixeis dos marcadores. Na figura 5.4 encontra-se um exemplo de uma imagem simulada, no qual se projeta uma bola e um quadrado com os marcadores.

5.1.2 Trajetórias Simuladas

Tendo-se mencionado os conceitos mais importantes do simulador, criaram-se três trajetórias arbitrárias diferentes, a trajetória A, B e C ilustradas na figura 5.5. Estas trajetórias serão usadas para mais tarde testar e quantificar a precisão dos algoritmos desenvolvidos neste trabalho:

- **Trajetória A:** Esta trajetória foi criada para simular uma bola num movimento circular uniforme,

Figura 5.4: Exemplo de uma imagem, onde se projeta uma bola e um quadrado com marcadores.

mantendo constante a coordenada de profundidade relativamente à câmara:

$$\mathbf{t}_A(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} = \begin{bmatrix} x_c + A_x \sin(2\pi f t) & y_c + A_y \cos(2\pi f t) & z_c \end{bmatrix}^T \tag{5.14}$$

com $f = 1\,\text{Hz}$, $x_c = y_c = 0\,\text{mm}$, $z_c = 1000\,\text{mm}$ e $A_x = A_y = 300\,\text{mm}$. O estado exato da bola é representado pelas coordenadas de posição e velocidade ao longo do tempo, isto é, $\mathbf{x}(t) = \begin{bmatrix} x(t) & y(t) & z(t) & \dot{x}(t) & \dot{y}(t) & \dot{z}(t) \end{bmatrix}^T$. Amostrando $\mathbf{x}(t)$ para gerar N imagens da bola seguindo a trajetória $\mathbf{t}_A$, designando como tempo de amostragem $t = t_s$, obtêm-se os estados exatos da bola $\mathbf{x}_j$ para os instantes de tempo $0, t_s, 2t_s, ..., (N-1)t_s$:

$$\mathbf{x}_j = \mathbf{x}(j\ t_s), \qquad j = 0, ..., N-1 \tag{5.15}$$

onde $N = 30$ e $t_s = 1/N\,\text{s}$.

- **Trajetória B:** Esta é também uma trajetória circular, mas desta vez a coordenada de profundidade é alterada ao longo do tempo:

$$\mathbf{t}_B(t) = \begin{bmatrix} x_c + A_x \sin(2\pi f t) & y_c + A_y \sin(2 \cdot 2\pi f t) & z_c + A_z \cos(2\pi f t) \end{bmatrix}^T \tag{5.16}$$

com $f = 1\,\text{Hz}$, $x_c = y_c = 0\,\text{mm}$, $z_c = 800\,\text{mm}$, $A_x = 300\,\text{mm}$, $A_y = 100\,\text{mm}$ e $A_z = 400\,\text{mm}$. A trajetória é amostrada da mesma maneira que a anterior, com $N = 100$ e $t_s = 1/N\,\text{s}$.

- **Trajectória C:** Esta trajetória simula uma bola em queda livre a colidir com um obstáculo num local com vento lateral constante. Para construir esta trajetória são usadas três equações. A equação da queda livre:

$$\mathbf{x}(t) = \mathbf{p}_i + \mathbf{v}_i(t - t_i) + \frac{\mathbf{a}}{2}(t - t_i)^2 \tag{5.17}$$

onde $\mathbf{p}_i$ e $\mathbf{v}_i$ correspondem à posição e velocidade da bola em $t = t_i$ e **a** representa a aceleração da bola. A equação do modelo de impacto (4.40) explicada na secção 4.3.3 e a equação do tempo de colisão entre a bola e o quadrado (5.19). Admitindo que a bola colide sempre com o obstáculo

então a trajetória é dada por:

$$\mathbf{t}_C(t) = \begin{cases} \mathbf{x}(t) = \mathbf{p}_i + \mathbf{v}_i(t-t_i) + \dfrac{\mathbf{a}}{2}(t-t_i)^2, & t_i < t < t_{i+1} \\ \mathbf{p}_{i+1} = \mathbf{x}(t_{i+1}), \quad \mathbf{v}_{i+1} = \dot{\mathbf{x}}(t_{i+1}) - (1+e)(\dot{\mathbf{x}}(t_{i+1})^T\mathbf{n})\mathbf{n}, & t = t_{i+1} \end{cases}, \quad i = 0, 1, 2, ... \tag{5.18}$$

onde t_{i+1} é dado por:

$$t_{i+1} = \max \left\{ \frac{-\mathbf{n}^T(\mathbf{v}_i - t_i\mathbf{a}) \pm \sqrt{\left(\mathbf{n}^T(\mathbf{v}_i - t_i\mathbf{a})\right)^2 - 2\mathbf{n}^T\left(\frac{t_i^2}{2}\mathbf{a} - t_i\mathbf{v}_i + \mathbf{p}_i - R\mathbf{n} - \mathbf{k}\right)\mathbf{n}^T\mathbf{a}}}{\mathbf{n}^T\mathbf{a}} \right\} \tag{5.19}$$

no qual, **R** é o raio da bola, **k** corresponde a qualquer ponto no quadrado, e é o coeficiente de restituição e **n** representa o vetor na direção da linha de impacto, que devido às formas regulares da bola e do quadrado, **n** coincide sempre com o vetor normal ao obstáculo. Para gerar a trajetória definiu-se $t_0 = 0$ s, $\mathbf{p}_0 = (300, -180, 1200)$ mm, $\mathbf{v}_0 = (-1000, 0, -1000)$ mm/s, $\mathbf{a} = (1000, 9810, 1000)$ mm/s^2, $R = 30$ e $e = 0,7$. A trajetória é amostrada da mesma maneira que as anteriores, com $N = 75$ e $t_s = 1/50$ s.

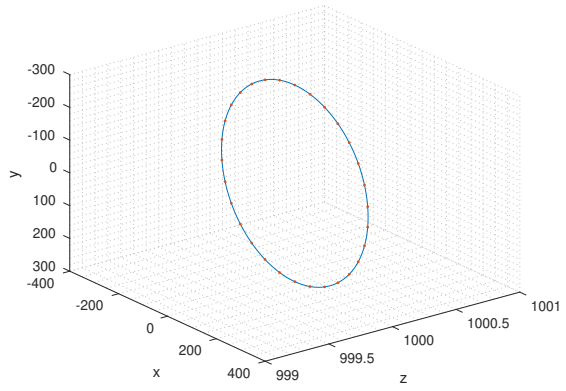
Por fim com as trajetórias amostradas, estas são introduzidas no simulador e depois este produz imagens coerentes para cada trajetória.

5.2 Deteção e estimação da pose do obstáculo

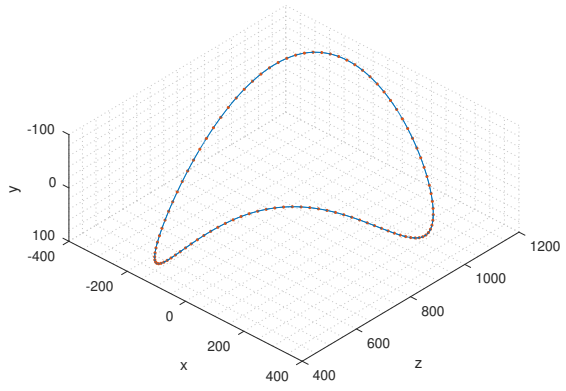
Para localizar os marcadores é aplicado uma rotina de deteção que analisa a imagem com o objetivo de encontrar estruturas com formatos quadrangulares. Depois de se obter vários quadrados candidatos, a textura de cada um é analisada para validar se são marcadores. Se o marcador for válido é aplicado um algoritmo que estima a sua pose no espaço [49]. Estas rotinas para estimar a pose de um marcador estão implementadas no *openCV* onde a pose deste pode ser expressa através de um vetor de translação **t**, que indica a posição central do marcador no espaço e uma matriz de rotação **R** que determina a sua orientação. De notar que são conhecidas previamente as posições nas quais os marcadores foram incorporados no obstáculo e o tamanho s destes. Como os marcadores fora colocados sobre as diagonais e a distâncias idênticas dos vértices do quadrado, como se verifica na figura 5.4, efetuando a média dos vetores de translação dos quatro marcadores, facilmente se obtém a posição central do quadrado $\bar{\mathbf{t}}$:

$$\bar{\mathbf{t}} = \frac{1}{4}\sum_{i=1}^{4}\mathbf{t}_i. \tag{5.20}$$

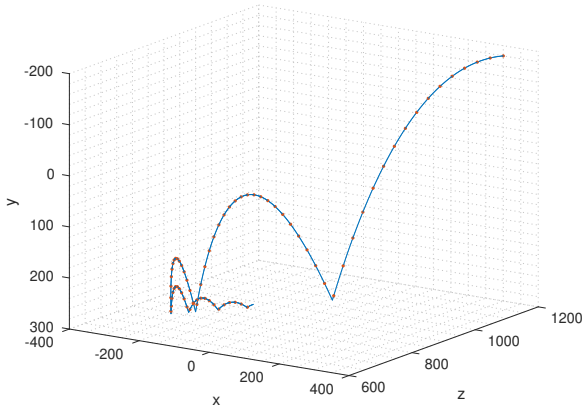
Já o processo de obter a orientação é mais complexo. Com os marcadores obtêm-se quatro matrizes de rotação, ora fazer a média das matrizes para estimar a matriz de rotação do quadrado, faz todo o sentido. Contudo o grupo das matrizes de rotação não é um espaço euclidiano, pelo que a noção de média não é um conceito óbvio [50]. O trabalho de *F. Landis Markley* [51] apresenta uma solução



(a) Trajetória A.



(b) Trajetória B.



(c) Trajetória C.

Figura 5.5: Figuras das três trajetórias simuladas. Os pontos a vermelho representam as amostras extraídas da trajetória.

usando quaterniões. Nesse trabalho para obter a orientação do quadrado, transformam-se as matrizes de rotação de cada marcador i para quaterniões $\mathbf{q}_i$ e calcula-se o quaternião médio $\bar{\mathbf{q}}$, que é dado pelo vetor próprio do maior valor próprio da matriz $\mathbf{M}$, onde [51]:

$$\mathbf{M} = \sum_{i=1}^{n} \mathbf{q}_i \mathbf{q}_i^T. \tag{5.21}$$

Invertendo o quaternião resultante $\bar{\mathbf{q}}$, obtém-se a matriz de rotação resultante $\bar{\mathbf{R}}$. Ao se definir os vértices do quadrado como $\mathbf{v}_1^{'} = (s, s, 0)$, $\mathbf{v}_2^{'} = (-s, s, 0)$, $\mathbf{v}_3^{'} = (-s, -s, 0)$ e $\mathbf{v}_4^{'} = (s, -s, 0)$, multiplicando-os pela matriz de rotação $\bar{\mathbf{R}}$ e adicionando $\bar{\mathbf{t}}$ obtêm-se os vértices do quadrado no espaço:

$$\mathbf{v}_i = \bar{\mathbf{R}}\mathbf{v}_i^{'} + \bar{\mathbf{t}} \qquad i = 1, ..., 4 \tag{5.22}$$

e o vetor normal ao quadrado **n** é dado por:

$$\mathbf{n} = \bar{\mathbf{R}} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} . \tag{5.23}$$

5.3 Calibração Real de Uma Câmara

Para realizar experiências reais com o intuito de demonstrar a utilidade prática deste trabalho, é usado uma câmara real e portanto é necessário efetuar a calibração da câmara. O modelo de projeção que será usado para caracterizar a câmara foi explicado na secção anterior 4.3.1. Nessa secção mencionaram-se que as distorções mais significativas que aparecem frequentemente nas câmaras são as distorções radiais e tangenciais. Estas distorções não devem ser negligenciadas, para não prejudicarem a precisão das soluções apresentadas nesta dissertação. Dito isto, a biblioteca *OpenCV* contém um algoritmo baseado no trabalho de *Jean-Yves Bouguet* [38] que estima os parâmetros intrínsecos e os coeficientes de distorção radial e tangencial de uma câmara. O processo de calibração divide-se em duas fases. A primeira fase consiste em detetar pontos-chave que pertencem a um padrão de calibração conhecido, neste caso utilizou-se um padrão axadrezado como se visualiza na figura 5.6. Este procedimento deve ser realizado várias vezes com o padrão em diferentes locais, com o intuito de tornar a calibração mais precisa e robusta. De seguida, o algoritmo é inicializado e é realizado um método de otimização não-linear que minimiza o erro total de reprojeção ajustando os parâmetros inerentes da câmara [38]. No fim o algoritmo devolve os parâmetros intrínsecos, os três primeiros coeficientes da distorção radial e os dois coeficientes de distorção tangencial.

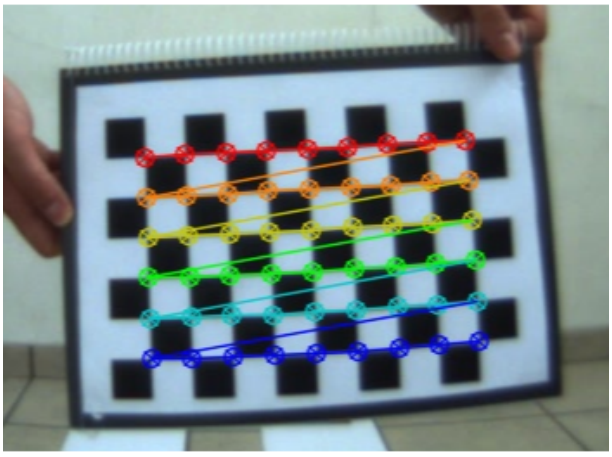


Figura 5.6: Exemplo da deteção dos pontos chave do padrão de calibração que serão usados para calibrar a câmara.

5.4 Implementação em C++, utilizando a biblioteca *OpenCV*

Com o intuito de tornar o algoritmo bastante eficiente e portável, decidiu-se desenvolver todo o código recorrendo à biblioteca *OpenCV*. Esta biblioteca está bastante otimizada para métodos de processamento de imagem, visão computacional e operações com matrizes. O código do PF pode ser consultado em `https://github.com/linonil/PF` e o do UPF em `https://github.com/linonil/UPF`.

De maneira a poder reutilizar o código do PF, visto que bastantes das funções implementadas no UPF provêm do PF, criaram-se duas classes principais PF e UPF, onde a classe PF é a classe base e a classe UPF é a classe derivada. Assim, para o código do UPF funcionar, este necessita de alguns ficheiros do código do PF.

Para cada código, antes de se inicializar os filtros, é implementada uma rotina de estimação da pose de um obstáculo como se refere em 5.2. Esta rotina é iniciada procurando marcadores, e no caso de se encontrar algum, estes são usados para estimar a pose do obstáculo. Para os testes que se efetuaram, os marcadores estão localizados no chão, mas também podiam estar numa parede, ou noutro obstáculo arbitrário.

Como se referiu, a biblioteca encontra-se bastante otimizada para operações com matrizes. Uma grande vantagem do modelo de velocidade constante do PF que usa a seguinte equação para propagar cada partícula:

$$\mathbf{x}_t^{(i)} = \mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{w}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{a}_t, \quad \mathbf{F} = \begin{bmatrix} \mathbf{I}_3 & \Delta_t \mathbf{I}_3 \\ \mathbf{0} & \mathbf{I}_3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \frac{\Delta_t^2}{2}\mathbf{I}_3 \\ \Delta_t \mathbf{I}_3 \end{bmatrix}, \quad \mathbf{a}_t = \mathcal{N}(\mathbf{0}, \mathbf{\Lambda}) \tag{5.24}$$

é que facilmente se integram todas as partículas numa só matriz. Aumentando o vetor da hipótese do estado $\mathbf{x}_t$ para uma matriz em que as colunas são as partículas, isto é:

$$\mathbf{X}_t = \begin{bmatrix} \mathbf{x}_t^1 & \mathbf{x}_t^2 & \mathbf{x}_t^3 & ... & \mathbf{x}_t^N \end{bmatrix} \tag{5.25}$$

e realizando o mesmo para o ruído de aceleração:

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{a}_t^1 & \mathbf{a}_t^2 & \mathbf{a}_t^3 & ... & \mathbf{a}_t^N \end{bmatrix} \tag{5.26}$$

então, é possível com:

$$\mathbf{X}_t = \mathbf{F}\mathbf{X}_{t-1} + \mathbf{B}\mathbf{A}_t \tag{5.27}$$

calcular todas as partículas com apenas duas operações matriciais, o que é mais eficiente do que calcular cada partícula iterativamente. Já para o UPF o mesmo não é possível pois é necessário criar e propagar os pontos sigma para cada partícula.

Capítulo 6

Resultados e Discussão

Com o intuito de analisar o desempenho do filtro de partículas e do *unscented particle filter* no seguimento de objetos, são realizados diversos testes recorrendo às trajetórias simuladas A, B e C ilustradas na figura 5.5 e realizadas duas experiências reais: na primeira, uma bola vermelha descreve uma trajetória pendular e na segunda, uma bola verde descreve uma trajetória em queda livre onde colide várias vezes com o chão. Para criar estas duas trajetórias, uma câmara real foi calibrada e de seguida foi usada para adquirir imagens do movimento real da bola. Para as trajetórias em queda livre, são realizados testes aplicando o modelo de impacto. Os objetivos destes testes consistem em determinar a eficácia de cada filtro avaliando e quantificando o erro de estimação e determinar a eficiência dos algoritmos especificando o tempo necessário para estimar o estado do objeto. Deste modo, neste capítulo é explicado, primeiramente, como é obtida a posição exata da bola nas experiências reais, que critérios se utilizaram para avaliar o erro e que parâmetros de ajuste foram testados (pois a inicialização destes afeta a qualidade de seguimento do objeto). Por fim, os resultados dos testes com diferentes parâmetros de inicialização são apresentados, comparados e discutidos.

6.1 Obtenção do estado exato em experiências reais

Para se obter o erro de estimação dos filtros, é necessário conhecer o estado exato do objeto. A grande vantagem de usar experiências simuladas é que o estado exato do objeto é conhecido. Já numa experiência real, tal não é possível e por isso necessita de ser medido. Todavia, o simulador que foi criado apresenta também uma grande desvantagem, porque muitos dos efeitos reais a que as câmaras e os objetos estão expostos, não são incorporados. Por exemplo, variações de iluminação, ruído no sensor de aquisição de imagem, desfoque do objeto na imagem, que é provocado quando este se move rapidamente no momento de aquisição de uma imagem, entre outros. Ou seja, são as experiências reais que demonstram verdadeiramente a utilidade e o desempenho real dos algoritmos. Desta forma, para determinar o estado exato de um objeto móvel em experiências reais, é necessário recorrer a um sensor de alta precisão bastante dispendioso para medir a distância da bola, e de seguida, analisar extensamente a trajetória do objeto para determinar também a sua velocidade e outros termos

que façam parte do estado do objeto. Não tendo acesso a um sensor mais preciso, para se obter a posição exata da bola é usado um processo de tentativa erro que consiste em manualmente, para cada imagem, definir um ponto em 3D $\mathbf{p} = (x, y, z)$ (que representa a posição da bola no espaço), projetar a silhueta da bola nesse ponto para a imagem e iterativamente ajustar os valores de x, y e z, de forma a que a silhueta coincida com a posição verdadeira da bola. Este processo é pouco rigoroso e apresenta problemas na reprodutibilidade, pois não só é difícil que a silhueta coincida perfeitamente com a bola, como também todo o processo depende da exatidão da calibração da câmara e do observador. Este processo foi realizado apenas para a posição da bola, pelo que apenas serão apresentados resultados da estimativa do erro de posição da bola para os testes reais.

6.2 Critérios utilizados para avaliar o erro

Analisar o desempenho de um algoritmo de seguimento através de métricas quantitativas é uma tarefa complexa, porque existem inúmeros fatores que afetam o seguimento, o que pode levar a que uma métrica não avalie de forma justa um filtro [52]. Por exemplo, uma métrica bastante utilizada para quantificar o erro de um estimador numa dada experiência é o erro quadrático médio, que calcula o desvio médio ao quadrado da estimativa do filtro em relação ao valor exato [53], isto é:

$$\text{MSE} = \frac{1}{N} \sum_{i=0}^{N} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 \tag{6.1}$$

onde N representa o número de estimativas que fazem parte de uma experiência, e $\mathbf{x}_i$ e $\hat{\mathbf{x}}_i$ correspondem, respetivamente, ao estado exato e à estimativa do estado i. Porém este valor pode não capturar devidamente o desempenho de um filtro. Por exemplo, se um filtro na maior parte do tempo estima com grande precisão o estado de um objeto, mas porventura perde-lhe o rasto nas últimas imagens da trajetória, então o erro quadrático médio poderá ser maior que num filtro que segue o objeto em toda a trajetória, mas de uma forma não tão precisa [54]. Este cenário é bastante provável de acontecer para este tipo de filtros, sendo importante evidenciá-los, para isso foram criados gráficos de precisão. Estes expressam a percentagem de estimativas que se encontram abaixo de um dado limiar de erro, à medida que esse limiar de erro aumenta [52]. Para esclarecer o funcionamento destes gráficos, é exemplificado na figura 6.1, um estimador muito preciso que perde o rasto no fim, dado pela linha azul, e um estimador pouco preciso que não perde o rasto ao objeto, expresso a vermelho.

Pela figura 6.1 observa-se que o estimador mais preciso atinge 80% de estimativas abaixo do limiar de erro e_1, enquanto que o outro para o mesmo limiar apenas atinge 25%. Contudo, à medida que se consideram limiares de erro maiores, o estimador menos preciso atinge os 100%, enquanto que o azul acaba por estagnar. Para além de gráficos de precisão, são também exibidos alguns resultados de análise local para determinadas situações e a raiz quadrada da métrica definida em cima, $\sqrt{\text{MSE}}$, é aplicada para caraterizar a evolução do erro à medida que o número de partículas aumenta.

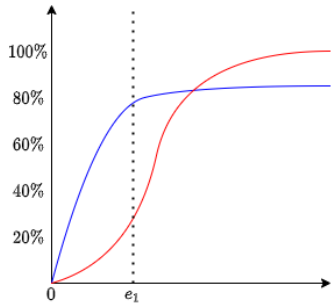


Figura 6.1: Gráficos de precisão para dois filtros diferentes.

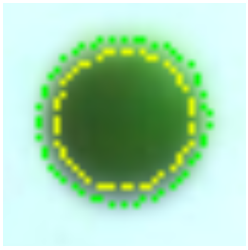
6.2.1 Parâmetros a testar

Para ambos os filtros, existem parâmetros de ajuste que afetam a eficácia e eficiência das estimativas do objeto. Os três principais e únicos parâmetros testados são:

- **Número de partículas:** Como foi referido na secção 2.6, este parâmetro é um *trade-off* entre exatidão da estimativa e eficiência computacional. Para filtros como o PF e o UPF, quanto maior for o número de partículas N, melhor é a aproximação que estes filtros fazem à distribuição real do estado [6], portanto obtêm-se resultados mais exatos;

- **Tamanho das silhuetas:** Este parâmetro consiste em ajustar os raios da silhueta interior R_{IN} e exterior R_{EX}, expressos de forma normalizada em relação ao raio exato da bola. Por exemplo, colocar $R_{IN} = 0,8$ corresponde a projetar uma silhueta de uma bola com um raio de 80% do raio exato da bola. São estas silhuetas que criam o modelo de observação das partículas e, como se pode observar na figura 6.2, quanto mais próximas forem (por exemplo $R_{IN} = 0.9$ e $R_{EX} = 1.1$) mais rigoroso é o modelo de observação. Todavia, tornar o erro de medição mais pequeno, tem o custo de aumentar a probabilidade de degeneração dos filtros, como foi mencionado na secção 2.5. Deste modo, são realizados testes variando o tamanho das silhuetas;

(a) $R_{IN} = 0,7$ e $R_{EX} = 1,3$.



(b) $R_{IN} = 0,9$ e $R_{EX} = 1,1$.

Figura 6.2: Tamanho das silhuetas interior e exterior.

- **Ruído do modelo do processo:** O ruído do processo controla a magnitude de dispersão das partículas no espaço de estados. Este parâmetro é bastante importante, porque por exemplo, se

o modelo do processo não for adequado e um objeto descrever uma trajetória rápida e imprevisível e a dispersão das partículas for baixa ou pelo contrário, se o movimento do objeto for lento mas houver uma grande dispersão das partículas, os filtros podem divergir facilmente do estado exato. Por norma, ajustar este parâmetro é uma tarefa bastante difícil, na prática costumam-se efetuar simulações e a partir dos resultados obtidos, tenta-se encontrar um valor apropriado para este parâmetro [9]. Contudo este depende totalmente da trajetória que o objeto segue e assim, para uns casos o ruído do processo pode estar ajustado adequadamente, no entanto para outros tal pode não ser verdade.

6.3 Experiências

Dada as infinitas possibilidades de teste, antes de se apresentarem os resultados, são referidas certas escolhas que se tomaram. Para criar os gráficos de precisão, a percentagem de erro relativo δ foi definida como limiar de erro. Desta maneira é usada a seguinte equação para calcular a percentagem de estimativas que possuem um erro inferior ao limiar de erro δ dado pela variável F:

$$F = \frac{100}{N} \sum_{i=1}^{N} H\left(\delta - \frac{||\mathbf{x}_i - \hat{\mathbf{x}}_i||}{||\mathbf{x}_i||}\right) \quad [\%] \tag{6.2}$$

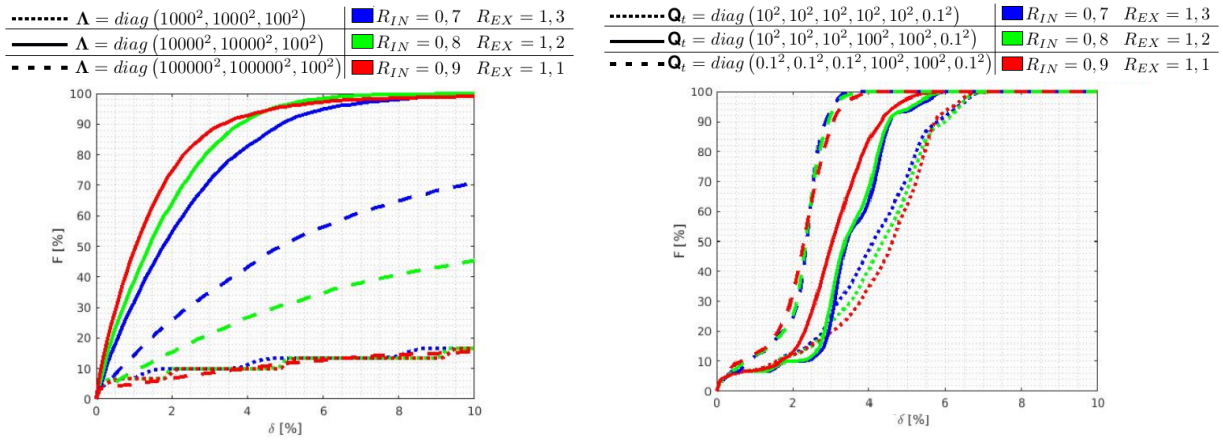
onde N representa o número de estimativas que fazem parte de uma experiência, $\mathbf{x}_i$ e $\hat{\mathbf{x}}_i$ correspondem respetivamente, ao estado exato e à estimativa do estado para a imagem i e a função H, a função de *Heaviside*. Como os filtros lidam com variáveis aleatórias, realizar um teste com os mesmos parâmetros origina resultados diferentes, embora estes possam ser semelhantes. Porém, podem também haver situações de instabilidade onde o filtro pode degenerar, e por este motivo, cada teste é repetido 100 vezes.

Este projeto apenas se foca no seguimento do objeto e não na sua localização, assim, em todos os testes, as partículas são iniciadas de forma aleatória com valores bastante semelhantes ao do estado inicial exato da bola. Quanto ao número de partículas, o valor padrão que se usou foi $N = 1024$, no entanto foram também usados valores de $N = 128$, $N = 256$ e $N = 4096$ para avaliar a exatidão das estimativas e a eficiência em função do número de partículas. Para avaliar a influência do tamanho das silhuetas, efetuaram-se testes para três diferentes distâncias entre silhuetas, umas bastante próximas com raio interior e exterior $R_{IN} = 0,9$ e $R_{EX} = 1,1$ e as outras duas como $R_{IN} = 0,8$ e $R_{EX} = 1,2$, e $R_{IN} = 0,7$ e $R_{EX} = 1,3$. Sobre o parâmetro do ruído do processo, este difere bastante perante o filtro utilizado. Para o PF, tal como mencionado em 4.1.1, este parâmetro corresponde a adicionar ao modelo do processo um termo aleatório de aceleração $\mathbf{a}_t = \mathcal{N}(\mathbf{0}, \mathbf{\Lambda})$, onde $\mathbf{\Lambda} = diag\left(\sigma_x^2, \sigma_y^2, \sigma_z^2\right)$. Já para o UPF, que usa o filtro UKF como distribuição de importância para propagar cada partícula, o ruído do modelo do processo do UKF corresponde à matriz de covariância $\mathbf{Q}_t = diag\left(\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_{\dot{x}}^2, \sigma_{\dot{y}}^2, \sigma_{\dot{z}}^2\right)$. Enquanto que no PF, o ruído do processo está diretamente relacionado com a aceleração da bola, no UPF o ruído está diretamente relacionado com a incerteza do modelo do processo perante o movimento real do objeto, logo nas trajetórias os valores de magnitude de ruído para ambos os filtros são bastante

diferentes. Para cada trajetória e para cada filtro, são realizados testes com três diferentes ruídos do processo. No caso do PF, são utilizadas configurações de ruído do processo, para cada trajetória, onde uma é ajustada experimentalmente e as outras duas correspondem a valores de ordem de grandeza superior e inferior a esta. No UPF as mesmas três configurações de ruído são usadas para todas as trajetórias, apenas é ajustada a componente σ_z^2 dependendo da trajetória utilizada. Como se referiu anteriormente, o UKF adiciona um modelo de observação (o método de reconstrução da posição da bola mencionado em 4.2) que tem também um ruído associado, representado pela matriz de covariância $\mathbf{R}_t$. Para facilitar a execução dos testes, em todos os testes simulados o ruído de observação foi colocado como $\mathbf{R}_t = diag\left(10^2, 10^2, 10^2\right)$ e para os testes reais aumentou-se para $\mathbf{R}_t = diag\left(50^2, 50^2, 50^2\right)$, porque o método de reconstrução da posição da bola é bastante afetado por ruído e por variações de iluminação, devido ao processo de segmentação na cor que é efetuado. Para finalizar, nas trajetórias em queda livre, para ambos os filtros, o modelo de impacto referido em 4.3.3 é incorporado no modelo do processo juntamente com o termo da aceleração gravítica, com o intuito de analisar o benefício deste modelo. Para os algoritmos detetarem o chão nas experiências reais, colocaram-se os marcadores especiais referidos em 5.1.1 e de seguida aplicou-se o método de estimação da pose do obstáculo mencionado em 5.2.

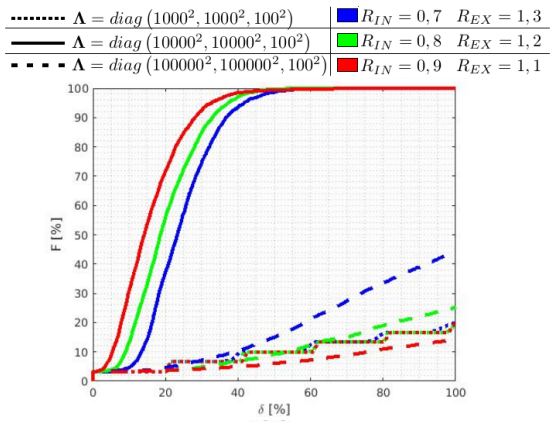
6.3.1 Trajetória A

Ao analisar os quatro gráficos de precisão expostos abaixo (Figuras 6.3 e 6.4), é bastante perceptível que nos gráficos de posição existe um menor erro que nos gráficos de velocidade, para ambos os filtros. Isto porque o modelo de observação apenas considera a posição das partículas, ignorando de certo modo a velocidade. Por exemplo, uma partícula que possui um valor de posição exato mas um valor de velocidade inadequado, será favorecida pelo modelo de observação, no qual este lhe atribuirá um alto valor de fator de importância $w_t^{(i)}$, o que afeta e desvia a estimativa da velocidade da bola.

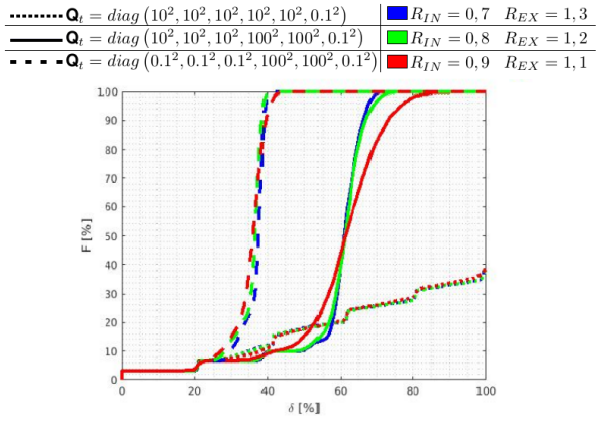


(a) PF. (b) UPF.

Figura 6.3: Gráficos de precisão da estimativa da posição.



(a) PF.



(b) UPF.

Figura 6.4: Gráficos de precisão da estimativa da velocidade.

Outro aspeto bastante evidente nos gráficos de precisão da estimativa da bola (Figura 6.3), é a maior sensibilidade do PF para diferentes magnitudes de desvio padrão. Para uma ordem de magnitude abaixo ou acima do desvio padrão daquele que é usado para os resultados das linhas sólidas, o filtro perde o rasto ao objeto e degenera, o que resulta em estimativas de posição e velocidade bastante incorretas. Para as linhas a pontilhado, o problema reside na pouca dispersão das partículas causado pelos baixos valores de variância do modelo do processo diante do movimento rápido do objeto. Com pouca dispersão, o filtro não consegue acompanhar o movimento do objeto e acaba por lhe perder o rasto. Por outro lado, para as linhas a tracejado, os elevados valores de variância, dispersam as partículas para zonas distantes da posição exata da bola, que acaba por causar a degeneração do filtro. Já no UPF, para diferentes valores de variância, os resultados obtidos na estimativa da posição são todos bastante adequados e bastante próximos entre si, contudo o mesmo não acontece para as estimativas da velocidade (figura 6.4(b)). Uma possível explicação para o problema na estimação da velocidade, é o ruído do processo não ser o mais apropriado para um movimento circular como este e por isso, obtêm-se melhores resultados utilizando variâncias maiores para a velocidade, como se observa nas linhas sólidas e a tracejado da figura 6.4(b).

No PF as linhas sólidas que representam um valor de desvio padrão adequado, obtêm resultados promissores tanto na posição como na velocidade, como se verifica nas figuras 6.3(a) e 6.4(a). A diferença reside no tamanho das silhuetas. A linha vermelha que corresponde às silhuetas mais próximas apresenta uma pequena melhoria face às restantes ao se analisar a percentagem de estimativas, por exemplo para um erro relativo de 2% na posição ou 20% na velocidade, e à medida que se consideram erros relativos maiores, as três linhas aproximam-se de 100%. Contudo, um pormenor que não é possível observar no gráfico de posição é o facto das linhas verde e azul atingirem os 100% primeiro que a linha vermelha. Isto acontece porque das 100 vezes que se realizou a simulação com os parâmetros da linha sólida vermelha, houve situações onde perto do fim o filtro degenerou, como se observa na figura 6.5. Uma vez que, tal como se referiu em 6.2.1, ao se usar tamanhos de silhuetas próximos, há um maior risco de o filtro perder o rasto ao objeto e degenerar e assim, pelo contrário,

obtêm-se melhores resultados para variâncias mais elevadas usando silhuetas mais afastadas. No UPF, esta propriedade não é tão visível, porque a introdução de uma observação corrente do estado do objeto na distribuição de importância, impede de melhor forma que o filtro degenere.

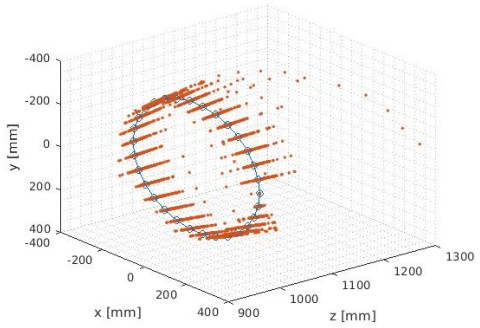


Figura 6.5: Gráfico das 100 estimativas de posição do PF. Observam-se que algumas estimativas de posição distanciam-se bastante dos valores exatos da trajetória A, representados pela linha azul.

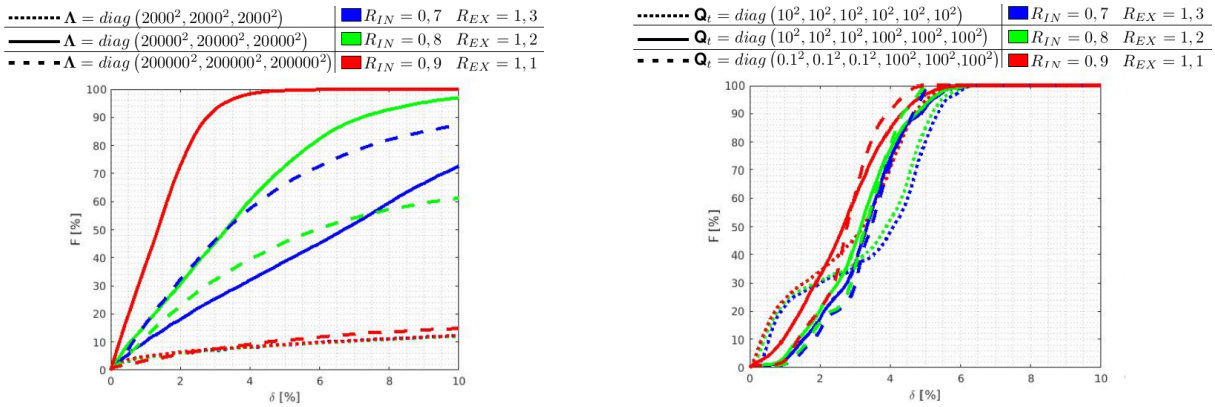
Na tabela 6.1 estão expostos os resultados de experiências com parâmetros idênticos variando apenas o número de partículas. Tal como referido em 6.2.1, pode-se verificar que estes resultados estão em concordância com a literatura, uma vez que à medida que o número de partículas aumenta, menor é o erro das estimativas.

	N	128	256	1024	4096
Posição	PF: ———	$6,32 \times 10^8$	$392,35$	$24,13$	$15,36$
	UPF: - - - - - -	$27,05$	$26,04$	$23,98$	$22,99$
Velocidade	PF: ———	$6,32 \times 10^8$	$1319,17$	$415,82$	$372,89$
	UPF: - - - - - -	$673,49$	$666,71$	$660,97$	$660,86$

Tabela 6.1: Resultados da raiz quadrática do erro quadrático médio em mm, para diferentes números de partículas para a trajetória A.

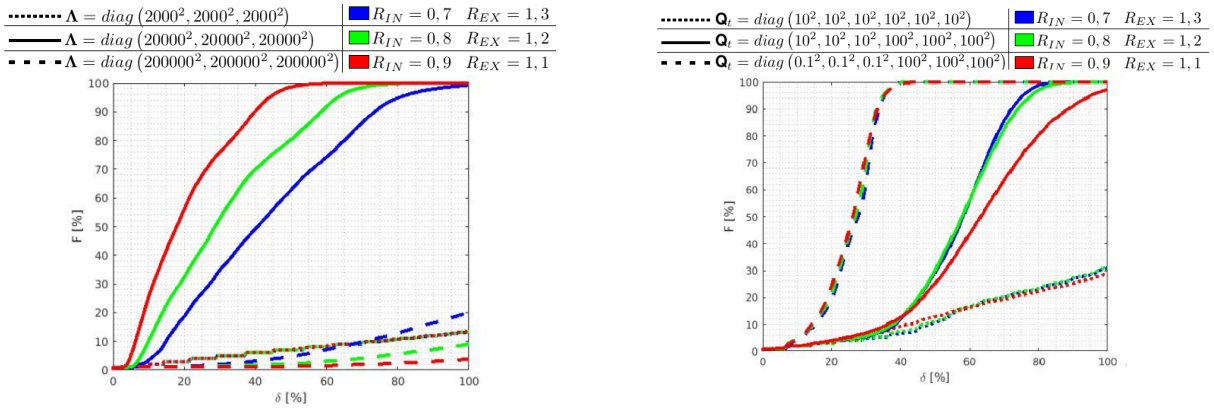
6.3.2 Trajetória B

Ao avaliar a trajetória B, verifica-se que os pontos enunciados na trajetória A são válidos também para esta trajetória. Ou seja, comprova-se que as estimativas de posição apresentam valores de erro bastante menores do que as estimativas de velocidade. Para o PF, o conceito do tamanho das silhuetas referido na secção anterior também se aplica na trajetória B, e desta vez, a linha vermelha sólida, a linha que representa o uso de uma variância adequada e tamanhos de silhueta próximos, atinge os 100% primeiro que as restantes, o que indica que o filtro não degenerou em nenhuma das 100 simulações. Quanto ao UPF o que acontece na trajetória A repete-se para esta, isto é, obtêm-se estimativas de posição satisfatórias, o tamanho das silhuetas não é tão sensível como no PF e o ruído do modelo do processo não está bem ajustado para este tipo de trajetórias. Comprova-se também que as estimativas do objeto melhoram à medida que o número de partículas aumenta (observar tabela 6.2).



(a) PF. (b) UPF.

Figura 6.6: Gráficos de precisão da estimativa da posição.



(a) PF. (b) UPF.

Figura 6.7: Gráficos de precisão da estimativa da velocidade.

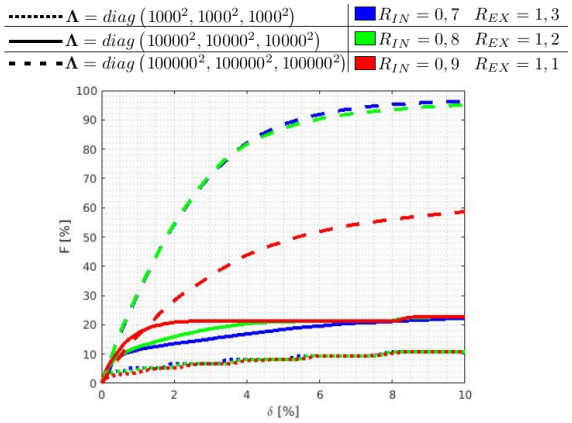
	N	128	256	1024	4096
Posição	PF: ———	$5,81 \times 10^9$	$2,68 \times 10^9$	$38,55$	$30,32$
	UPF: - - - - - -	$30,12$	$29,92$	$29,88$	$29,87$
Velocidade	PF: ———	$5,81 \times 10^9$	$2,68 \times 10^9$	$824,09$	$790,30$
	UPF: - - - - - -	$636,15$	$624,91$	$616,88$	$615,08$

Tabela 6.2: Resultados da raiz quadrática do erro quadrático médio em mm, para diferentes números de partículas para a trajetória B.

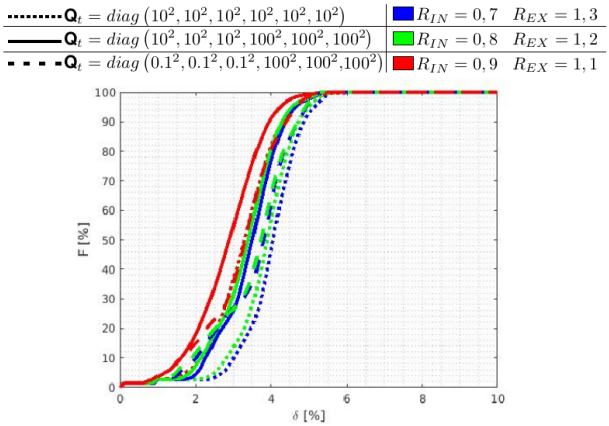
6.3.3 Trajetória C

A trajetória C é uma trajetória no qual a bola perfaz um movimento em queda livre e colide com um obstáculo várias vezes. Para esta trajetória o modelo de impacto mencionado na secção 4.3.3 pode ser aplicado. Com o intuito de entender o benefício que este modelo traz para as estimativas, serão

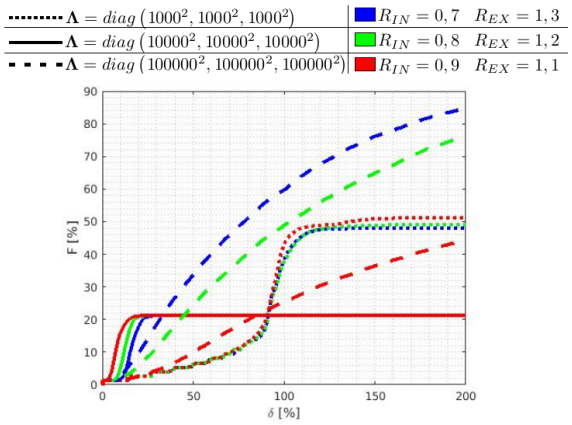
exibidos resultados apenas com o modelo de velocidade constante e também resultados adicionando o modelo de impacto juntamente com a constante de aceleração gravítica. Analisando primeiro os gráficos da estimativa para o PF, se não se usar o modelo de impacto, apenas usando um alta variância é que é possível que o filtro acompanhe o movimento da bola após as colisões com o obstáculo (ver figuras 6.8(a) e 6.8(c)), mesmo assim, das vezes que se simulou algumas das estimativas perderam o rasto da bola ao longo do tempo. Usando o modelo de impacto (figuras 6.9(a) e 6.9(c)), obtém-se um drástico aumento de precisão tanto na posição como na velocidade. Na figura 6.10, é exibido para 100 simulações diferentes as estimativas da velocidade após a primeiro colisão, para o caso sem o modelo de impacto e com o modelo, onde se pode constatar a grande melhoria que se obtém nas estimativas aplicando o modelo. Para o UPF, a melhoria da estimativa da posição é incremental (figuras 6.8(b) e 6.9(b)), mas para a velocidade a diferença já é significativa (figuras 6.8(d) e 6.9(d)).



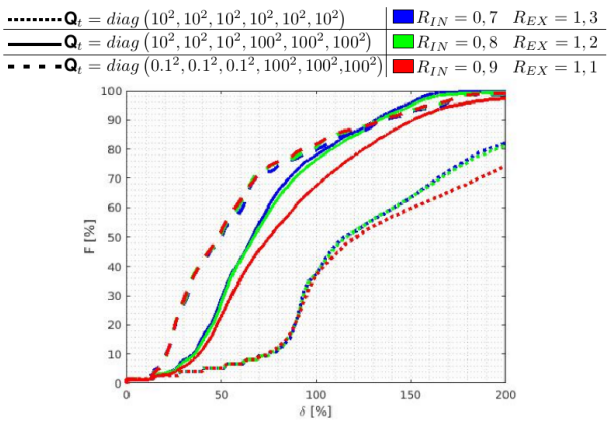
(a) Estimativa da posição do PF sem modelo de impacto.



(b) Estimativa da posição do UPF sem modelo de impacto.

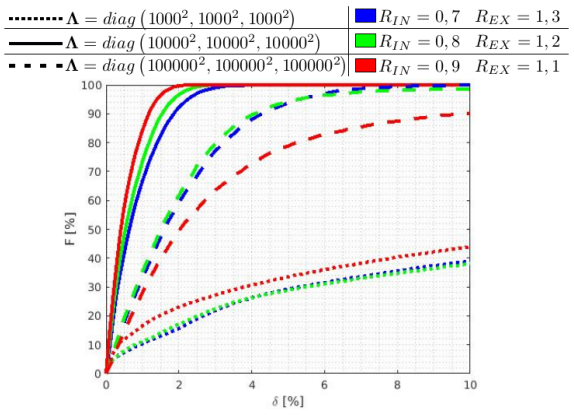


(c) Estimativa da velocidade do PF sem modelo de impacto.

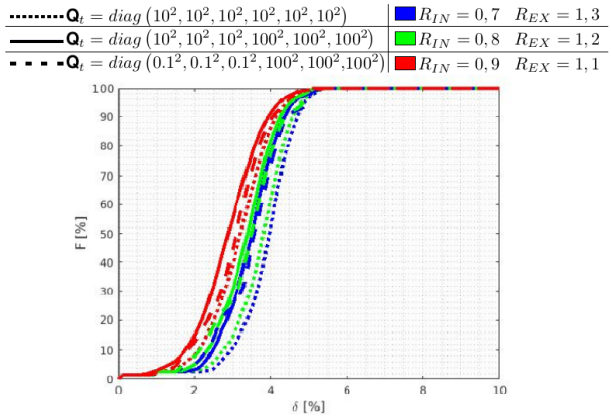


(d) Estimativa da velocidade do UPF sem modelo de impacto.

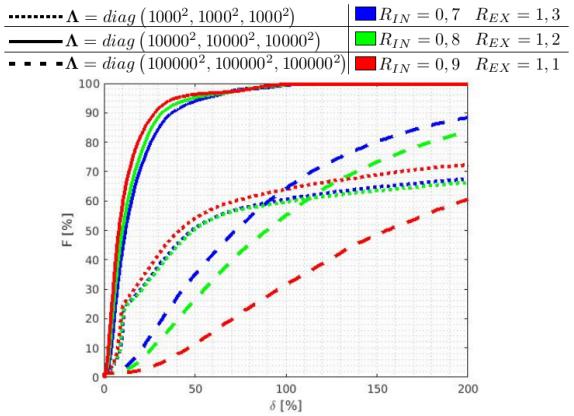
Figura 6.8: Gráficos de precisão da estimativa da posição e da velocidade sem o modelo de impacto.



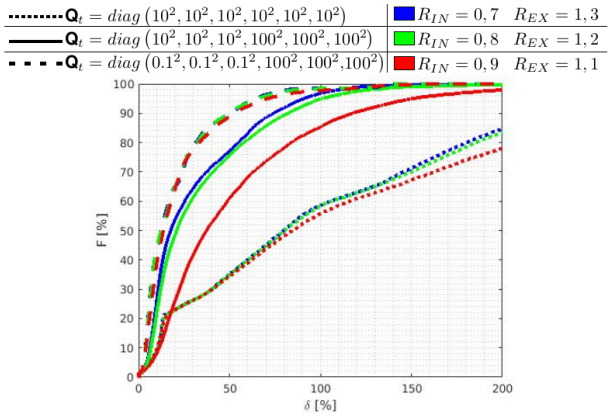
(a) Estimativa da posição do PF com modelo de impacto.



(b) Estimativa da posição do UPF com modelo de impacto.



(c) Estimativa da velocidade do PF com modelo de impacto.



(d) Estimativa da velocidade do UPF com modelo de impacto.

Figura 6.9: Gráficos de precisão da estimativa da posição e da velocidade com o modelo de impacto.

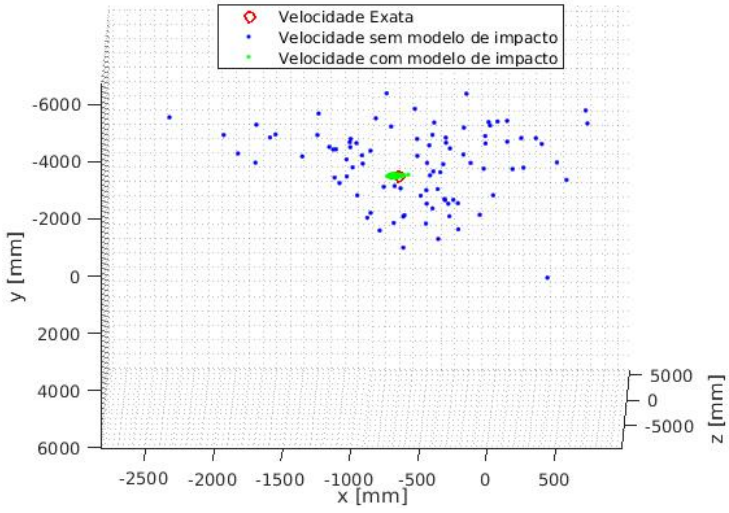
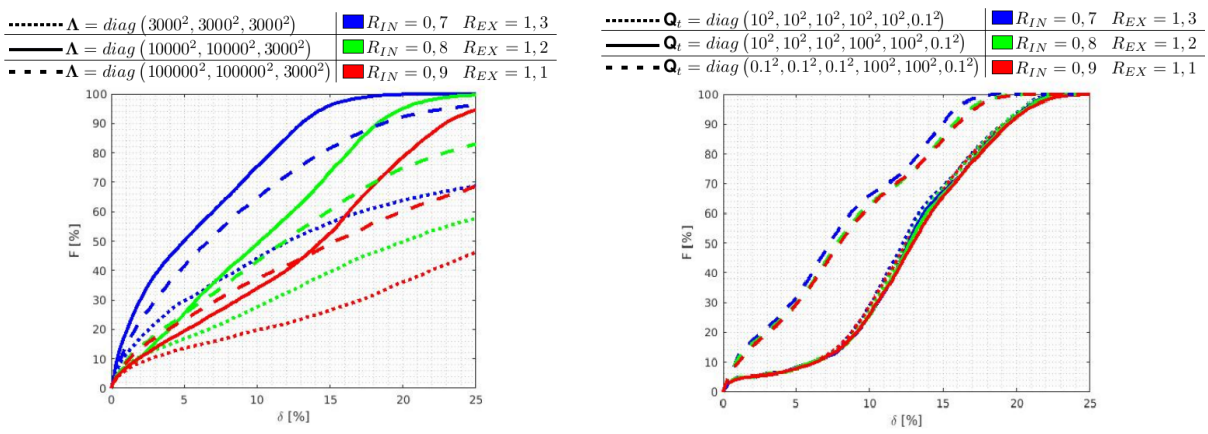


Figura 6.10: Estimativas da velocidade do PF após o primeiro impacto.

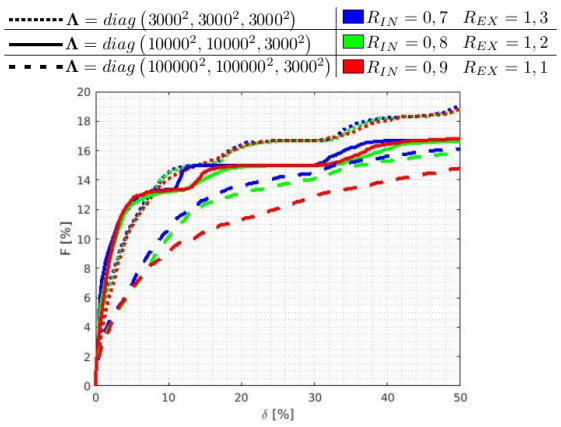
6.4 Trajetórias reais

Como se referiu, para os resultados reais apenas são apresentados resultados da estimativa da posição. Uma diferença bastante visível comparando os gráficos das experiências reais com os gráficos das experiências simuladas é que para as experiências reais o erro é maior, pelo menos o dobro, o que faz sentido, visto que o simulador não tem em conta os fenómenos reais, também mais uma vez, o uso do modelo de impacto aumenta a precisão dos resultados, principalmente no PF. Fazendo uma análise mais pormenorizada, tanto para o PF como para o UPF, repara-se que desta vez os melhores resultados obtém-se usando as silhuetas mais afastadas. Uma possível explicação para isto talvez seja devido ao ruído ou ao desfoque de movimento que a bola apresenta no momento de aquisição de uma imagem. Estes fatores aumentam de forma indesejada o contorno da bola, pelo que o uso de silhuetas mais próximas não é apropriado. Com estes resultados comprova-se a utilidade prática dos algoritmos para problemas reais, apesar do erro ser o dobro, estes valores continuam a ser bastante pequenos como se pode constatar.

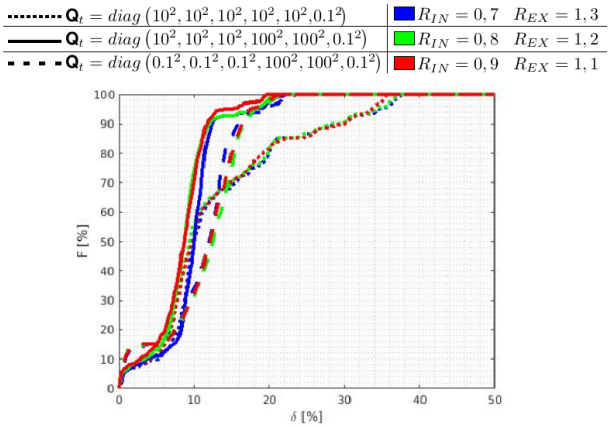


(a) PF. (b) UPF.

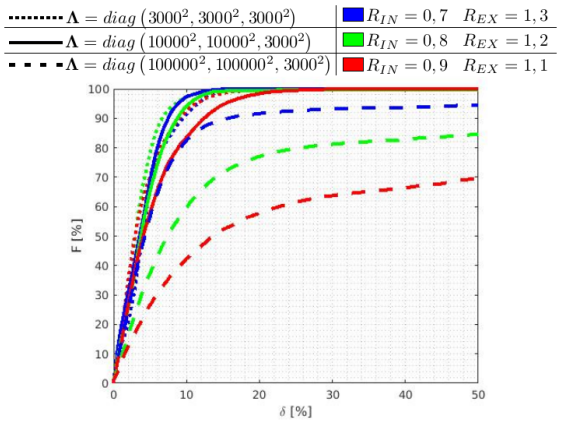
Figura 6.11: Gráficos de precisão da estimativa da posição para o movimento pendular.



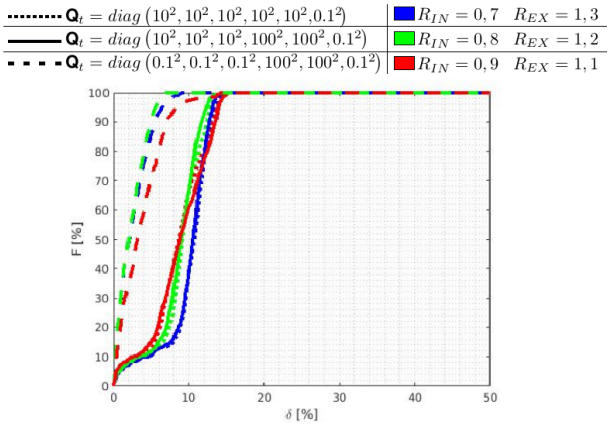
(a) PF sem modelo de impacto.



(b) UPF sem modelo de impacto.



(c) PF com modelo de impacto.



(d) UPF com modelo de impacto.

Figura 6.12: Gráficos de precisão da estimativa da posição para a trajetória em queda livre com e sem o modelo de impacto.

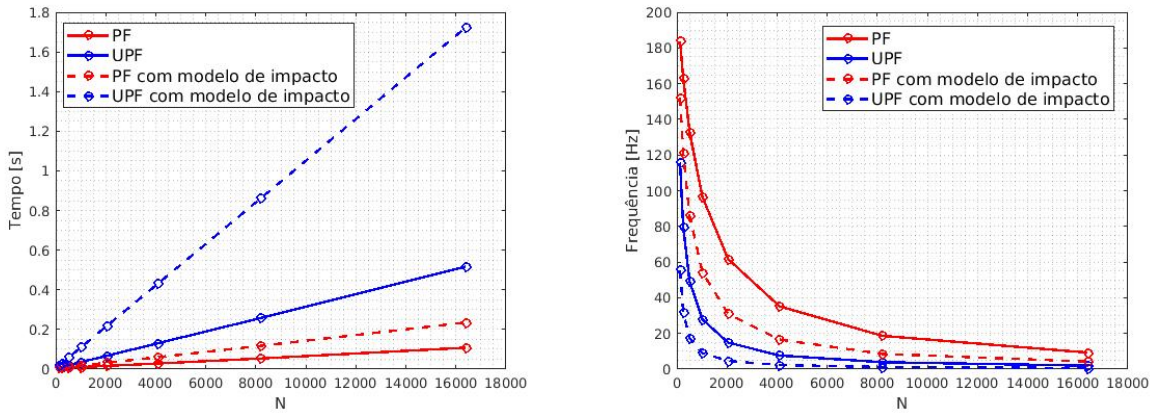
6.5 Testes de eficiência

Uma propriedade importante de um filtro é a sua eficiência computacional. Esta propriedade pode ser crítica em problemas que necessitam que as estimativas sejam calculadas rapidamente. Os filtros baseados em partículas, apresentam uma complexidade computacional que cresce com o número de partículas, logo para avaliar a eficiência dos algoritmos desenvolvidos foram realizados testes variando este parâmetro.

Os testes efetuados foram realizados numa máquina *Intel Core* i7-8700 CPU @ 3.20GHz e estão representados na tabela 6.3 e nos gráficos 6.13(a) e 6.13(b). De notar que para os testes onde se usa o modelo de impacto, todos os resultados apresentados correspondem ao pior caso possível, no qual, o modelo de impacto necessita de ser aplicado a todas as partículas.

N = 1024	PF		UPF	
	Sem Modelo de Impacto	Com Modelo de Impacto	Sem Modelo de Impacto	Com Modelo de Impacto
Predição	$32,90\ \mu s$	$8,13\ ms$	$25,56\ ms$	$100,85\ ms$
Aquisição de imagem	$4,75\ ms$			
Projeção das Silhuetas	$2,02\ ms$			
Cálculo dos Histogramas	$3,53\ ms$			
Seleção de Partículas	$85,81\ \mu s$		$170,86\ \mu s$	

Tabela 6.3: Tempo de execução de cada método, para 1024 partículas.



(a) Tempo *vs* número de partículas.

(b) Frequência *vs* número de partículas.

Figura 6.13: Na figura 6.13(a) é exibido o tempo necessário para calcular uma estimativa do estado e na figura 6.13(b) é apresentada a evolução da frequência à medida que o número de partículas aumenta.

Dos resultados obtidos, conclui-se que o UPF é menos eficiente que o PF e que o modelo de impacto pode adicionar uma grande carga computacional aos algoritmos. Como se referiu, tanto na tabela como nos gráficos, os resultados com o modelo de impacto são apresentados para o pior caso possível. Para esse pior caso, como se pode observar na tabela, para 1024 partículas o aumento do tempo médio no cálculo da predição aumenta cerca de 250 vezes no PF, por outro lado no UPF o aumento é quase 4 vezes superior. Este elevado acréscimo no PF, deve-se à maneira diferente que tem que ser usada para propagar as partículas aplicando o modelo de impacto. Pois sem o modelo, a propagação das partículas é realizada efetuando apenas duas operações matriciais para todas as N partículas, enquanto que com o modelo é necessário iterar cada partícula para testar se essa colidiu com um obstáculo e realizar também mais algumas operações. Os restantes métodos referidos na tabela exibem tempos iguais, porque são idênticas para ambos os filtros. Apenas no UPF, a seleção de partículas demora o dobro do tempo, visto que para além de se selecionar o estado das partículas, também é necessário selecionar as matrizes de covariância.

Das figuras 6.13(a) e 6.13(b) é evidenciado o tempo que os filtros precisam para gerar uma estima-

tiva do estado. Ora, para 1024 partículas, o PF sem modelo de impacto regista uma frequência de 96 Hz, enquanto que ao usar o modelo de impacto, para o caso de pior cenário, a frequência baixa para os 54 Hz. Com recurso ao UPF, a velocidade do filtro sem o modelo ronda os 28 Hz e recorrendo ao modelo de impacto, para o pior caso, a frequência cai para os 9 Hz. Como se pode observar na tabela, a grande diferença na eficiência entre o PF e o UPF encontra-se sobretudo no método de predição dos filtros, enquanto no primeiro as operações são triviais, no segundo é necessário calcular o UKF para cada partícula. Esta diferença causa, no total, a que o UPF seja quase 3,5 vezes mais lento que o PF sem o modelo de impacto e 6 vezes mais lento com o modelo de impacto.

Capítulo 7

Conclusão

Os resultados obtidos nesta dissertação permitem concluir que os filtros que foram implementados funcionam com êxito, porém os parâmetros dos filtros devem ser ajustados devidamente para o tipo de trajetória. Tal como se observou nos resultados, o filtro de partículas é bastante sensível aos parâmetros de inicialização, enquanto que o *unscented particle filter* é menos sensível, uma vez que para o PF, mesmo com um ruído do processo ajustado à trajetória, usar poucas partículas normalmente resultou em degeneração, enquanto que no UPF isso não aconteceu, mesmo com poucas partículas as estimativas apresentavam valores adequados. Verificou-se também nos gráficos que o segundo filtro (UPF) é bastante robusto perante diferentes tamanhos de silhuetas, assim como para diferentes ruídos do modelo do processo. Porém para a estimativa da velocidade, os resultados obtidos não foram tão exatos como se pretendia e como se observou, pois os resultados já dependem mais dos parâmetros de ajuste. Outra informação que se retira dos resultados obtidos é o grande êxito do modelo de impacto, que para ambos os filtros, melhorou consideravelmente a exatidão das estimativas. Quanto aos testes de eficiência, o filtro de partículas é mais rápido que o *unscented particle filter*, logo para uma aplicação que necessite rapidamente de estimativas, o UPF pode não ser adequado.

Esta dissertação apresenta também algumas limitações. Uma das limitações foi o facto da estimativa da velocidade do objeto não ter sido estimada para trajetórias reais, também o método utilizado para obter a posição exata dessas trajetórias não apresenta muito rigor, pelo que os resultados obtidos podem não ser exatos e dada a infinita possibilidade de testes diferentes, poucos testes foram realizados. Outra limitação, foi que propriedades como o seguimento após o filtro perder o rasto da bola, o que acontece bastante para ambientes com obstáculos onde a bola pode facilmente ficar oculta ou parcialmente oculta, não foram estudadas.

Como trabalho futuro, e dadas as limitações dos métodos baseados em regiões mencionadas na tabela 3.1, que indicam que estes apenas funcionam adequadamente para objetos homogéneos, um trabalho a considerar é o trabalho desenvolvido por *Leisheng Zong et. al* [26], que combina um método baseado em regiões com um método direto, criando um modelo híbrido, na tentativa de incorporar as vantagens de ambos os métodos. Ora, o trabalho de *Leisheng Zong et. al* [26] pode ser adaptado aos algoritmos desenvolvidos neste projeto e assim estender o trabalho para lidar com objetos mais

complexos. Do ponto de vista de eficiência, uma propriedade bastante interessante dos algoritmos desenvolvidos nesta dissertação é que todas as partículas são calculadas da mesma maneira e de forma independente. Estas duas características permitem que o código seja implementado de forma paralela, o que em princípio aceleraria bastante o tempo de estimação de ambos os filtros. Os algoritmos desenvolvidos neste trabalho foram implementados em série, portanto se se quiser avançar para seguimento de outros tipos de objetos mais complexos e que necessitem de modelos de propagação das partículas e modelos de observação mais complicados, um código implementado em série já não conseguirá funcionar em tempo real. Por essa razão, como trabalho futuro poderá ser considerado desenvolver o código na Unidade de Processamento Gráfico (GPU), este que é um processador composto por centenas ou milhares de pequenos e simples núcleos especializados, otimizados para efetuar instruções simples em paralelo.

Bibliografia

[1] Hawk-Eye Innovations Ltd. Goal Line Technology. Consultado a 05/01/2020. Disponível em `https://www.hawkeyeinnovations.com/products/ball-tracking/goal-line-technology`.

[2] Upendra Rao M. e Umesh C. Pati. A Novel Algorithm for Detection of Soccer Ball and Player. Em *2015 International Conference on Communications and Signal Processing (ICCSP)*, páginas 344–348. IEEE, 2015.

[3] Y. Xu, G. Yu, X. Wu,Y. Wang e Y. Ma. An Enhanced Viola-Jones Vehicle Detection Method From Unmanned Aerial Vehicles Imagery. *IEEE Transactions on Intelligent Transportation Systems*, 18 (7):1845–1856, Julho 2017.

[4] M. Taiana, J. Santos, J. Gaspar, J. Nascimento, A. Bernardino e P. Lima. Tracking objects with generic calibrated sensors: An algorithm based on color and 3d shape features. *Robotics and Autonomous Systems*, 58(6):784–795, Junho 2010.

[5] R. Van Der Merwe, A. Doucet, N. de Freitas e E. Wan. The unscented particle filter. Em *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, páginas 563–569, Cambridge, MA, USA, 2000. MIT Press.

[6] S. Thrun, W. Burgard e D. Fox. *Probalistic robotics*. The MIT Press, 1ª edição, 2005.

[7] K. Ogata. *Modern Control Engineering*. Prentice Hall, USA, 4ª edição, 2001.

[8] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, Março 1960.

[9] R. Labbe. Kalman and Bayesian Filters in Python, 2014. Disponível em `https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python`.

[10] S. Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 1ª edição, Outubro 2013.

[11] G. Welch and G. Bishop. An introduction to the kalman filter, 1995.

[12] S. J. Julier. The scaled unscented transformation. Em *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, volume 6, páginas 4555–4559. IEEE, Maio 2002.

[13] A. Doucet, N. de Freitas e N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, New York, NY, 1ª edição, 2001.

[14] A. Doucet, e A. Johansen. A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later. *Handbook of Nonlinear Filtering*, 12(3):656–704, Janeiro 2009.

[15] B. Ristic, S. Arulampalam e N. Gordon. *Beyond the Kalman filter: Particle filters for tracking applications*. Artech House, 2004.

[16] T. Li, M. Bolic e P. Djuric. Resampling methods for particle filtering: Classification, implementation, and strategies. *IEEE Signal Processing Magazine*, 32(3):70–86, 2015.

[17] T. Li, S. Sun, T. Sattar e C. Rodríguez. Fight sample degeneracy and impoverishment in particle filters: A review of intelligent approaches. *Expert Systems with Applications*, 41(8):3944–3954, Junho 2014.

[18] L. Murray, A. Lee e P. Jacob. Parallel resampling in the particle filter. *Journal of Computational and Graphical Statistics*, 25(3):789–805, Julho 2016.

[19] J. Carpenter, P. Clifford e P. Fearnhead. Improved particle filter for nonlinear problems. *IEE Proceedings - Radar, Sonar and Navigation*, 146(1):2–7, 1999.

[20] C. Snyder. Particle filters, the ”optimal”proposal and high-dimensional systems. Em *Proceedings of the ECMWF Seminar on Data Assimilation for atmosphere and ocean*, páginas 161–170. ECMWF, 2011.

[21] L Cruz, D. Lucio e L. Velho. Kinect and rgbd images: Challenges and applications. Em *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials*, páginas 36–49. IEEE, Agosto 2012.

[22] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H. Seidel, W. Xu, D. Casas e C. Theobalt. Vnect: Real-Time 3D Human Pose Estimation with a Single RGB Camera. *ACM Transactions on Graphics*, 36(4):1–14, Julho 2017.

[23] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu e C. Theobalt. Monocular 3D Human Pose Estimation in the Wild Using Improved CNN Supervision. Em *2017 international conference on 3D vision (3DV)*, páginas 506–516. IEEE, Outubro 2017.

[24] A. Baak, M. Müller, G, Bharaj, H. Seidel e C. Theobalt. A Data-Driven Approach for Real-Time Full Body Pose Reconstruction from a Depth Camera. Em *2011 International Conference on Computer Vision*, páginas 1092–1099. IEEE, Novembro 2011.

[25] N. Touzene. Real-Time Obstacle Detection Approach using Stereoscopic Images. *International Journal of Information Engineering and Electronic Business*, 6(1):42–48, Fevereiro 2014.

[26] L. Zhong e L. Zhang. A Robust Monocular 3D Object Tracking Method Combining Statistical and Photometric Constraints. *International Journal of Computer Vision*, 127:973–992, Agosto 2019.

[27] V. Lepetit e P. Fua. Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, 2005.

[28] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab e V. Lepetit. Multimodal Templates for Real-Time Detection of Texture-Less Objects in Heavily Cluttered Scenes. *2011 International Conference on Computer Vision*, páginas 858–865, 2011.

[29] C. Harris, C. Stennett. Rapid - A Video Rate Object Tracker. Em *Proceedings of the British Machine Vision Conference*, páginas 73–77. BMVA Press, 1990.

[30] T. Drummond e R. Cipolla. Real-Time Visual Tracking of Complex Structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932–946, Julho 2002.

[31] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60:91–110, Novembro 2004.

[32] David G. Lowe. Object Recognition from Local Scale-Invariant Features. Em *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, páginas 1150–1157. IEEE, Setembro 1999.

[33] V. Prisacariu e I. Reid. PWP3D: Real-Time Segmentation and Tracking of 3D Objects. Em *Proceedings of the British Machine Vision Conference*, volume 47, páginas 1–10. BMVA Press, Setembro 2009.

[34] M. Taiana. 3D model-based tracking with one omnidirectional camera and particle filters. Thesis, Electrical and Computer Engineering, IST and Politecnico di Milano. Outubro 2007. Disponível em `http://users.isr.ist.utl.pt/~mtaiana/publications.html`.

[35] P. Perez, J. Vermaak e A. Blake. Data Fusion for Visual Tracking with Particles. *Proceedings of the IEEE*, 92(3):495–513, Novembro 2004.

[36] A. Weeks e G. Hague. Color Segmentation in the HSI Color Space using the K-means Algorithm. Em *Nonlinear Image Processing VIII*, volume 3026, páginas 143–154. International Society for Optics and Photonics, SPIE, Abril 1997.

[37] U. Jau, C. Teh e G. Ng. A Comparison of RGB and HSI Color Segmentation In Real - Time Video Images: A Preliminary Study On Road Sign Detection. Em *2008 International Symposium on Information Technology*, volume 4, páginas 1–6. IEEE, Agosto 2008.

[38] J. Bouguet. Camera Calibration Toolbox for Matlab. Computational Vision at CALTECH: J. Bouguet. 14 de Outubro 2015. Disponível em `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[39] J Heikkila e O. Silven. A Four-Step Camera Calibration Procedure with Implicit Image Correction. Em *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, páginas 1106–1112. IEEE, Junho 1997.

[40] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[41] P. Huang, Z. Meng, J. Guo e F. Zhang. *Tethered Space Robot: Dynamics, Measurement, and Control*. Academic Press, 2018.

[42] B. Poling. A tutorial on camera models. páginas 1–10, University of Minnesota, 2015.

[43] C. Ricolfe-Viala e A. Sanchez-Salmeron. Lens Distortion Models Evaluation. *Applied optics*, 49 (30):5914–5928, Outubro 2010.

[44] G. Cross e A. Zisserman. Quadric Reconstruction from Dual-Space Geometry. Em *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, páginas 25–31. IEEE, Janeiro 1998.

[45] N. Greggio, J. Gaspar, A. Bernardino e J. Santos-Victor. Monocular vs Binocular 3D Real-Time Ball Tracking from 2D Ellipses. Em *Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO*, páginas 67–73. INSTICC, SciTePress, 2011.

[46] S. Ma e L. Li. Ellipsoid Reconstruction from Three Perspective Views. Em *Proceedings of 13th International Conference on Pattern Recognition*, volume 1, páginas 344–348, Agosto 1996.

[47] E. Johnston, F. Beer e E. Eisenberg. *Vector Mechanics for Engineers: Statics and Dynamic.* McGraw-Hill, New York, 12ª edição, 2009.

[48] M. Ahmad, K. Ismail e F. Mat. Impact models and coefficient of restitution: A review. *ARPN Journal of Engineering and Applied Sciences*, 11(10):6549–6555, Maio 2016.

[49] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas e M. Marín-Jiménez. Automatic Generation and Detection of Highly Reliable Fiducial Markers Under Occlusion. *Pattern Recognition*, 47(6): 2280–2292, Junho 2014.

[50] M. Moakher. Means and Averaging in the Group of Rotations. *SIAM Journal on Matrix Analysis and Applications*, 24(1):1–16, Julho 2006.

[51] F. Markley, Y. Cheng, J. Crassidis e Y. Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193–1197, Julho 2007.

[52] Y. Wu, J. Lim e M. Yang. Online Object Tracking: A Benchmark. Em *2013 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 2411–2418. IEEE, Junho 2013.

[53] S. Kay. *Fundamentals of statistical signal processing:Estimation Theory*. Prentice Hall, USA, 1993.

[54] B. Babenko, M. Yang e S. Belongie. Robust Object Tracking with Online Multiple Instance Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011.

Apêndice A

Algoritmos dos Filtros

No capítulo 2 foram mencionados vários filtros e explicado as suas equações. Neste apêndice com o intuito de complementar a explicação dos filtros, são apresentados os seguintes algoritmos:

A.1 Algoritmo do Filtro de *Kalman*

Algoritmo 4: $(\boldsymbol{\mu}_t, \mathbf{P}_t) = \mathrm{KF}(\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}, \mathbf{z}_t)$

Para $t = 0$**:**

$\boldsymbol{\mu}_0 = E[\mathbf{x}_0] \qquad \mathbf{P}_0 = E\left[(\mathbf{x}_0 - \boldsymbol{\mu}_0)(\mathbf{x}_0 - \boldsymbol{\mu}_0)^T\right]$

Para $t \geq 1$**:**

$\bar{\boldsymbol{\mu}}_t = \mathbf{F}_t \boldsymbol{\mu}_{t-1}$

$\bar{\mathbf{P}}_t = \mathbf{F}_t \mathbf{P}_{t-1} \mathbf{F}_t^T + \mathbf{Q}_t$

$\mathbf{K}_t = \bar{\mathbf{P}}_t \mathbf{H}_t^T (\mathbf{H}_t \bar{\mathbf{P}}_t \mathbf{H}_t^T + \mathbf{R}_t)^{-1}$

$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}_t \bar{\boldsymbol{\mu}}_t)$

$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t)\bar{\mathbf{P}}_t$

A.2 Algoritmo do Filtro de *Kalman* Estendido

Algoritmo 5: $(\boldsymbol{\mu}_t, \mathbf{P}_t) = \mathrm{EKF}(\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}, \mathbf{z}_t)$

Para $t = 0$**:**

$\boldsymbol{\mu}_0 = E[\mathbf{x}_0] \qquad \mathbf{P}_0 = E\left[(\mathbf{x}_0 - \boldsymbol{\mu}_0)(\mathbf{x}_0 - \boldsymbol{\mu}_0)^T\right]$

Para $t \geq 1$**:**

$\bar{\boldsymbol{\mu}}_t = f(\boldsymbol{\mu}_{t-1}, \mathbf{0})$

$\bar{\mathbf{P}}_t = \mathbf{F}_t \mathbf{P}_{t-1} {\mathbf{F}_t}^T + \mathbf{G}_t \mathbf{Q}_t \mathbf{G}_t^T$

$\mathbf{K}_t = \bar{\mathbf{P}}_t \mathbf{H}_t^T (\mathbf{H}_t \bar{\mathbf{P}}_t \mathbf{H}_t^T + \mathbf{U}_t \mathbf{R}_t \mathbf{U}_t^T)^{-1}$

$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t\left(\mathbf{z}_t - \mathbf{h}(\bar{\boldsymbol{\mu}}_t, \mathbf{0})\right)$

$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t)\bar{\mathbf{P}}_t$

A.3 Algoritmo do *Unscented Kalman Filter*

Algoritmo 6: $(\boldsymbol{\mu}_t, \mathbf{P}_t) = \text{UKF}(\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}, \mathbf{z}_t)$

Para $t = 0$**:**

$$\boldsymbol{\mu}_0 = E[\mathbf{x}_0] \qquad \mathbf{P}_0 = E\left[(\mathbf{x}_0 - \boldsymbol{\mu}_0)(\mathbf{x}_0 - \boldsymbol{\mu}_0)^T\right]$$

Para $t \geq 1$**:**

$$\boldsymbol{\chi}^a_{t-1} = \left[\boldsymbol{\mu}^a_{t-1} \quad \boldsymbol{\mu}^a_{t-1} \pm \left(\sqrt{(n_a + \lambda_a)\mathbf{P}^a_{t-1}}\right)_i\right]$$

$$\boldsymbol{\mathcal{Y}}_t = f(\boldsymbol{\chi}^x_{t-1}, \boldsymbol{\chi}^q_{t-1})$$

$$\bar{\boldsymbol{\mu}}_t = \sum_{i=0}^{2n} w_i^{(m)} \boldsymbol{\mathcal{Y}}_{i,t}$$

$$\bar{\mathbf{P}}_t = \sum_{i=0}^{2n} w_i^{(c)} (\boldsymbol{\mathcal{Y}}_{i,t} - \bar{\boldsymbol{\mu}}_t)(\boldsymbol{\mathcal{Y}}_{i,t} - \bar{\boldsymbol{\mu}}_t)^T$$

$$\boldsymbol{\mathcal{Z}}_t = h(\boldsymbol{\mathcal{Y}}_t, \boldsymbol{\chi}^r_{t-1})$$

$$\boldsymbol{\mu}_z = \sum_{i=1}^{2n+1} w_i^{(m)} \boldsymbol{\mathcal{Z}}_{i,t}$$

$$\mathbf{P}_z = \sum_{i=1}^{2n+1} w_i^{(c)} (\boldsymbol{\mathcal{Z}}_{i,t} - \boldsymbol{\mu}_z)(\boldsymbol{\mathcal{Z}}_{i,t} - \boldsymbol{\mu}_z)^T$$

$$\mathbf{K}_t = \left(\sum_{i=1}^{2n+1} w_i^{(c)} (\boldsymbol{\mathcal{Y}}_{i,t} - \bar{\boldsymbol{\mu}}_t)(\boldsymbol{\mathcal{Z}}_{i,t} - \boldsymbol{\mu}_z)^T\right) \mathbf{P}_z^{-1}$$

$$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - \boldsymbol{\mu}_z)$$

$$\mathbf{P}_t = \bar{\mathbf{P}}_t - \mathbf{K}_t \mathbf{P}_z \mathbf{K}_t^T$$

A.4 Algoritmo do Filtro de Partículas

Algoritmo 7: $\left(\mathbf{x}_t^{(1)}, \ \ldots, \mathbf{x}_t^{(N)}\right) = \text{PF}\left(\mathbf{x}_{t-1}^{(1)}, \ \ldots, \mathbf{x}_{t-1}^{(N)}\right)$

Para $t = 0$**:**

  Para $i = 1, ..., N$**:**

1 $\quad \mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$

2 $\quad w_0^{(i)} = \dfrac{1}{N}$

Para $t \geq 1$**:**

  Para $i = 1, ..., N$**:**

3 $\quad \mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i)})$

4 $\quad w_t^{(i)} \propto p(\mathbf{z}_t | \mathbf{x}_t^{(i)}) w_{t-1}^{(i)}$

5 $\quad \tilde{w}_t^{(i)} = \dfrac{w_t^{(i)}}{\sum_{k=1}^{N} w_t^{(k)}}$

6 $\quad$ REAMOSTRAGEM$(\mathbf{x}_t^{(i)}, \tilde{w}_t)$

7 $\quad w_t^{(i)} = \dfrac{1}{N}$

A.5 Algoritmo do Filtro de Partículas de *Kalman* Estendido

Algoritmo 8: $\left(\left\{\mathbf{x}_t^{(1)}, \mathbf{P}_t^{(1)}\right\}, \ \ldots, \left\{\mathbf{x}_t^{(N)}, \mathbf{P}_t^{(N)}\right\}\right) = \text{EKPF}\left(\left\{\mathbf{x}_{t-1}^{(1)}, \mathbf{P}_{t-1}^{(1)}\right\}, \ \ldots, \left\{\mathbf{x}_{t-1}^{(N)}, \mathbf{P}_{t-1}^{(N)}\right\}, \mathbf{z}_t\right)$

Para $t = 0$**:**

  Para $i = 1, ..., N$**:**

1 $\quad \mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0), \qquad w_0^{(i)} = \dfrac{1}{N}$

2 $\quad \mathbf{P}_0^{(i)} = E[(\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T]$

Para $t \geq 1$**:**

  Para $i = 1, ..., N$**:**

3 $\quad (\boldsymbol{\mu}_t^{(i)}, \mathbf{P}_t^{(i)}) = \text{EKPF}(\mathbf{x}_{t-1}^{(i)}, \mathbf{P}_{t-1}^{(i)}, \mathbf{z}_t)$

4 $\quad \mathbf{x}_t^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}_t^{(i)}, \mathbf{P}_t^{(i)})$

5 $\quad w_t^{(i)} \propto \dfrac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})} w_{t-1}^{(i)}$

6 $\quad \tilde{w}_t^{(i)} = \dfrac{w_t^{(i)}}{\sum_{k=1}^{N} w_t^{(k)}}$

7 $\quad \text{REAMOSTRAGEM}(\mathbf{x}_t^{(i)}, \mathbf{P}_t^{(i)}, w_t^{(i)})$

8 $\quad w_t^{(i)} = \dfrac{1}{N}$

A.6 Algoritmo do *Unscented Particle Filter*

Algoritmo 9: $\left(\left\{\mathbf{x}_t^{(1)}, \mathbf{P}_t^{(1)}\right\}, \ \ldots, \left\{\mathbf{x}_t^{(N)}, \mathbf{P}_t^{(N)}\right\}\right) = \text{UKF}\left(\left\{\mathbf{x}_{t-1}^{(1)}, \mathbf{P}_{t-1}^{(1)}\right\}, \ \ldots, \left\{\mathbf{x}_{t-1}^{(N)}, \mathbf{P}_{t-1}^{(N)}\right\}, \mathbf{z}_t\right)$

Para $t = 0$**:**

  Para $i = 1, ..., N$**:**

1 $\quad \mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0), \qquad w_0^{(i)} = \dfrac{1}{N}$

2 $\quad \mathbf{P}_0^{(i)} = E[(\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T]$

Para $t \geq 1$**:**

  Para $i = 1, ..., N$**:**

3 $\quad (\boldsymbol{\mu}_t^{(i)}, \mathbf{P}_t^{(i)}) = \text{UKF}(\mathbf{x}_{t-1}^{(i)}, \mathbf{P}_{t-1}^{(i)}, \mathbf{z}_t)$

4 $\quad \mathbf{x}_t^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}_t^{(i)}, \mathbf{P}_t^{(i)})$

5 $\quad w_t^{(i)} \propto \dfrac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})} w_{t-1}^{(i)}$

6 $\quad \tilde{w}_t^{(i)} = \dfrac{w_t^{(i)}}{\sum_{k=1}^{N} w_t^{(k)}}$

7 $\quad \text{REAMOSTRAGEM}(\mathbf{x}_t^{(i)}, \mathbf{P}_t^{(i)}, w_t^{(i)})$

8 $\quad w_t^{(i)} = \dfrac{1}{N}$

Apêndice B

Obtenção das matrizes S e V

Neste capítulo pretende-se descrever a decomposição em valores próprios da matriz que representa a cónica no plano da imagem, **H**, sendo esta apresentada na equação (4.29), e reescrita aqui por conveniência:

$$\mathbf{H} = \left(\mathbf{I}_3 - \frac{\mathbf{x}_0\mathbf{x}_0^T}{||\mathbf{x}_0||^2 - R^2}\right) = \frac{1}{||\mathbf{x}_0||^2 - R^2}\begin{bmatrix} y_0^2 + z_0^2 - R^2 & -x_0y_0 & -x_0z_0 \\ -x_0y_0 & x_0^2 + z_0^2 - R^2 & -y_0z_0 \\ -x_0z_0 & -y_0z_0 & x_0^2 + y_0^2 - R^2 \end{bmatrix}. \tag{B.1}$$

Como a matriz **H** é simétrica, é também diagonalizável e vetores próprios associados a valores próprios diferentes são ortogonais. E sendo a matriz diagonalizável, sabe-se que é sempre possível definir vetores próprios ortogonais associados a valores próprios iguais. Como os vetores próprios são também definidos a menos de uma constante multiplicativa, estes podem ser definidos de forma a que tenham norma unitária. Por outras palavras, é sempre possível associar uma matriz diagonalizante **V** ortonormal a uma matriz simétrica. A matriz **H** e a sua matriz diagonal **S** relacionam-se por

$$\mathbf{H} = \mathbf{V}\mathbf{S}\mathbf{V}^T. \tag{B.2}$$

Resolvendo a equação caraterística $\mathbf{det}(\mathbf{H} - \lambda\mathbf{I}_3) = 0$ em ordem a λ, os valores próprios $\lambda_1 = 1$, $\lambda_2 = 1$ e $\lambda_3 = 1/(1 - \left|\left|\frac{\mathbf{x}_0}{R}\right|\right|^2)$ são obtidos, pelo que a matriz **S** é dada por

$$\mathbf{S} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \dfrac{1}{1 - \left|\left|\frac{\mathbf{x}_0}{R}\right|\right|^2} \end{bmatrix}. \tag{B.3}$$

Como a multiplicidade algébrica do último valor próprio é 1 (aparece apenas uma vez na matriz diagonal **S**), a sua multiplicidade geométrica é também 1, e o respetivo vetor próprio define-se a menos de um parâmetro livre, que pode ser escolhido de forma a que a sua norma seja unitária. Ora, resolvendo a

equação que define o vetor próprio $\mathbf{v}_3 = \begin{bmatrix} v_{3,1} & v_{3,2} & v_{3,3} \end{bmatrix}^T$, isto é, $(\mathbf{H} - \lambda_3 \mathbf{I}_3)\mathbf{v}_3 = 0$, vem:

$$\begin{cases} \left(y_0^2 + z_0^2\right)\, v_{3,1} - x_0 y_0\, v_{3,2} - x_0 z_0\, v_{3,3} = 0 \\ -x_0 y_0\, v_{3,1} + \left(x_0^2 + z_0^2\right)\, v_{3,2} - y_0 z_0\, v_{3,3} = 0 \\ -x_0 z_0\, v_{3,1} - y_0 z_0\, v_{3,2} + \left(x_0^2 + y_0^2\right)\, v_{3,3} = 0 \end{cases} \Leftrightarrow \begin{cases} v_{3,1} = \frac{x_0}{z_0} v_{3,3} \\ v_{3,2} = \frac{y_0}{z_0} v_{3,3} \\ v_{3,3} \in \mathbb{R} \end{cases} \tag{B.4}$$

sendo $v_{3,3}$ o parâmetro livre. Arbitrando um valor $v_{3,3} = z_0/||\mathbf{x}_0||$, obtém-se um vetor próprio normalizado:

$$\mathbf{v}_3 = \frac{1}{||\mathbf{x}_0||} \begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}^T. \tag{B.5}$$

O valor próprio $\lambda_1 = 1$ apresenta multiplicidade algébrica 2 (aparece duas vezes na matriz diagonal **S**) e portanto também multiplicidade geométrica 2, já que **H** é diagonalizável. O respetivo vetor próprio é assim definido a menos de dois parâmetros livres. Ora, resolvendo a equação que define o vetor próprio $\mathbf{v}_1 = \begin{bmatrix} v_{1,1} & v_{1,2} & v_{1,3} \end{bmatrix}^T$, isto é, $(\mathbf{H} - \lambda_1 \mathbf{I}_3)\mathbf{v}_1 = 0$, vem:

$$\begin{cases} -x_0^2\, v_{1,1} - x_0 y_0\, v_{1,2} - x_0 z_0\, v_{1,3} = 0 \\ -x_0 y_0\, v_{1,1} - y_0^2\, v_{1,2} - y_0 z_0\, v_{1,3} = 0 \\ -x_0 z_0\, v_{1,1} - y_0 z_0\, v_{1,2} - z_0^2\, v_{1,3} = 0 \end{cases} \Leftrightarrow \begin{cases} v_{1,1} \in \mathbb{R} \\ v_{1,2} \in \mathbb{R} \\ v_{1,3} = -\frac{x_0}{z_0} v_{1,1} - \frac{y_0}{z_0} v_{1,2} \end{cases} \tag{B.6}$$

sendo $v_{1,1}$ e $v_{1,2}$ os dois parâmetros livres. Como os valores próprios λ_1 e λ_2, são iguais, equações idênticas às do sistema (B.6) obtêm-se para a definição do vetor próprio $\mathbf{v}_2 = \begin{bmatrix} v_{2,1} & v_{2,2} & v_{2,3} \end{bmatrix}^T$ associado a λ_2. Desta forma, os vetores próprios $\mathbf{v}_1$ e $\mathbf{v}_2$ são dados por

$$\mathbf{v}_1 = \begin{bmatrix} v_{1,1} & v_{1,2} & -\frac{x_0}{z_0} v_{1,1} - \frac{y_0}{z_0} v_{1,2} \end{bmatrix}^T, \tag{B.7}$$

$$\mathbf{v}_2 = \begin{bmatrix} v_{2,1} & v_{2,2} & -\frac{x_0}{z_0} v_{2,1} - \frac{y_0}{z_0} v_{2,2} \end{bmatrix}^T, \tag{B.8}$$

respetivamente. Pretende-se que os vectores próprios $\mathbf{v}_1$, $\mathbf{v}_2$ e $\mathbf{v}_3$ sejam ortonormais entre si, pelo que as seguintes condições devem de ser satisfeitas:

$$\begin{cases} \mathbf{v}_1 \cdot \mathbf{v}_2 = 0 \\ ||\mathbf{v}_1|| = 1 \\ ||\mathbf{v}_2|| = 1 \end{cases} \tag{B.9}$$

tendo em conta que $\mathbf{v}_3$ já apresenta norma unitária, e que $\mathbf{v}_1$ e $\mathbf{v}_3$ são necessariamente ortogonais entre si assim como $\mathbf{v}_2$ e $\mathbf{v}_3$, já que estes estão associados a valores próprios diferentes e a matriz **H** é simétrica. Uma forma simples de satisfazer (B.9) é arbitrando valores para três dos quatro parâmetros livres e resolvendo a primeira condição em ordem ao quarto, e posteriormente normalizando os vectores $\mathbf{v}_1$ e $\mathbf{v}_2$ obtidos dividindo estes pelas suas normas. Ora, assumindo $v_{1,1} = v_{1,2} = v_{2,1} = 1$, obtém-se da

primeira condição

$$v_{2,2} = -\frac{x_0 y_0 + x_0^2 + z_0^2}{x_0 y_0 + y_0^2 + z_0^2}, \tag{B.10}$$

o que resulta em

$$\mathbf{v}_1 = \begin{bmatrix} 1 & 1 & -\frac{x_0}{z_0} - \frac{y_0}{z_0} \end{bmatrix}^T, \tag{B.11}$$

$$\mathbf{v}_2 = \begin{bmatrix} 1 & -\frac{x_0 y_0 + x_0^2 + z_0^2}{x_0 y_0 + y_0^2 + z_0^2} & -\frac{z_0(y_0 - x_0)}{x_0 y_0 + y_0^2 + z_0^2} \end{bmatrix}^T, \tag{B.12}$$

Dividindo estes vectores pelas suas normas, é possível obter a seguinte matriz diagonalizante ortonormal:

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} \frac{z_0}{k_1} & \frac{x_0 y_0 + y_0^2 + z_0^2}{k_2} & \frac{x_0}{k_3} \\ \frac{z_0}{k_1} & -\frac{x_0 y_0 + x_0^2 + z_0^2}{k_2} & \frac{y_0}{k_3} \\ -\frac{x_0 + y_0}{k_1} & \frac{z_0(y_0 - x_0)}{k_2} & \frac{z_0}{k_3} \end{bmatrix}, \tag{B.13}$$

com k_1, k_2 e k_3 dados por:

$$k_1 = \sqrt{2z_0^2 + (x_0 + y_0)^2}, \tag{B.14}$$

$$k_2 = \sqrt{(x_0 y_0 + y_0^2 + z_0^2)^2 + (x_0 y_0 + x_0^2 + z_0^2)^2 + z_0^2 (y_0 - x_0)^2}, \tag{B.15}$$

$$k_3 = \sqrt{x_0^2 + y_0^2 + z_0^2}, \tag{B.16}$$

respetivamente.