TÉCNICO LISBOA

Reconhecimento e Estimação de Trajetórias de Obstáculos em Condução de Veículos à Escala

José Guilherme Penas Silvério

Dissertação para obtenção do Grau de Mestre em

Engenharia Mecânica

Orientadores: Prof. Paulo Jorge Coelho Ramalho Oliveira

Prof. Carlos Baptista Cardeira

Júri

Presidente: Prof. Paulo Rui Alves Fernandes

Orientador: Prof. Paulo Jorge Coelho Ramalho Oliveira

Vogais: Prof. João Rogério Caldas Pinto

Junho de 2017

"Ou é porque o sal não salga, ou porque a terra se não deixa salgar."

Padre António Vieira – Sermão de Santo António aos peixes

Resumo

Num ambiente de laboratório, existem muitas aplicações onde o reconhecimento do espaço em volta de um determinado sistema é necessário para que este se movimente no espaço sem colidir. Este reconhecimento é geralmente feito através de sensores implementados no próprio sistema, como câmaras ou lasers. O sistema estudado nesta dissertação é um veículo *radio controlled (RC)* à escala no qual se pretendia implementar uma única câmara e desenvolver software que permitisse ao sistema (RC e sensor) mover-se autonomamente.

Dado que o reconhecimento de profundidade, através de visão monocular, é matematicamente impossível, desenvolveu-se um método que aproveita o movimento do RC e a captação de várias imagens para fazer deteção de obstáculos e para calcular distâncias aos mesmos. Este método envolve o emparelhamento de pontos de interesse nas imagens captadas e relações de deslocamento destes pontos para a segmentação de obstáculos e mapeamento dos mesmos. Implementou-se depois do mapeamento para a localização de obstáculos no espaço um método de planeamento de rota (algoritmo A*) para definir os caminhos do sistema no mapa.

Os resultados em 30 testes efetuados demonstram que o algoritmo implementado faz com que o sistema colida com algum obstáculo 2.9% das vezes em que o mapa é atualizado. Estes resultados devem-se a pequenas diferenças entre as posições esperadas e as posições medidas do sistema em cada momento, que influenciam diretamente os processos de *clustering* e que por consequência levam a um mapeamento deficiente do ambiente.

Palavras-chave

Sistema autónomo; Visão monocular; Pontos de interesse; Segmentação; Planeamento de rota

Abstract

In a laboratory environment, there are many applications where the recognition of the space around a given system is necessary for it to move in said space without colliding. This recognition usually relies on sensors implemented in the system itself, such as cameras or lasers. The system studied in this dissertation is a scale radio controlled vehicle in which it was intended to implement a single camera and to develop software that would allow the system (RC and sensor) to move autonomously.

Since depth recognition through monocular vision is mathematically impossible, a method has been developed that takes advantage of the RC movement and of the capture of multiple images, to detect obstacles and to calculate distances to them. This method involves the pairing of points of interest in the captured images and the displacement relations of these points for the segmentation of obstacles and their mapping. A path planning method (algorithm A *) was implemented after mapping the location of obstacles in space, to define the system paths on the map.

The results in 30 performed tests demonstrate that the implemented algorithm causes the system to collide with some obstacle 2.9% of the times the map is updated. These results are due to small differences between the expected position and the measured position of the system at each time, which directly influence the clustering processes which consequently leads to a poor mapping of the environment.

Keywords

Autonomous system; Monocular vision; Interest points; Segmentation; Path Planning

Índice

CAPÍTULO 1 **1**

1.1 Introdução **1**

1.2 Trabalho relacionado **2**

1.3 Contribuições **4**

CAPÍTULO 2 – PROCESSAMENTO DE IMAGEM **5**

2.1 Geometria e cálculos **5**
2.1.1 Modelo de Pinhole 5
2.1.2 Deduções trigonométricas 7

2.2 Pontos de interesse e descritores **9**
2.2.1 Características SIFT e SURF - Introdução 10
2.2.2 SURF 13

2.3 Clustering **14**
2.3.1 DBSCAN 15

2.4 Validação **16**
2.4.1 1º Teste 16
2.4.2 2º Teste 24

CAPÍTULO 3 – PLANEAMENTO DE ROTA E ALGORITMO **31**

3.1 Planeamento de rota **31**
3.1.1 Formulação do problema 31
3.1.2 Espaço de configuração (Configuration Space) 32
3.1.3 Algoritmo 33
3.1.4 Grid-Based Search 34
3.1.5 A* algorithm 34
3.1.6 Mapeamento dos obstáculos 35
3.1.7 Definição dos pontos iniciais e finais e aplicação 36

3.2 Algoritmo global **37**
3.2.1 Movimentação no mapa 37
3.2.2 Inicialização do algoritmo 38
3.2.3 Ciclo 39

3.3 Hardware e comunicação **39**
3.3.1 Pi Camera 39
3.3.2 Limitações 40

CAPÍTULO 4 – IDENTIFICAÇÃO DO SISTEMA **43**

4.1 Hardware e Setup **44**

4.2 Ensaios realizados **45**
4.2.1 Dados recebidos pela IMU (GY-80) 45
4.2.2 Linha Reta 47
4.2.3 Em curva 51

CAPÍTULO 5 – TESTES E RESULTADOS **55**

5.1 Definição das dimensões da malha **55**

5.2 Testes ao movimento e parâmetros **58**

5.3 Testes ao sistema **62**
5.3.1 1º teste 62
5.3.2 2ºteste 65

5.4 Resultados dos testes ao sistema **68**
5.4.1 Tempos de processos 68
5.4.2 Avaliação do sistema 68

CAPÍTULO 6 – CONCLUSÕES E TRABALHO FUTURO **71**

6.1 Conclusões **71**

6.2 Trabalho Futuro **73**

Lista de Tabelas

Tabela 1 22
Tabela 2 23
Tabela 3 25
Tabela 4 26
Tabela 5 27
Tabela 6 29
Tabela 7 50
Tabela 8 58
Tabela 9 62
Tabela 10 64
Tabela 11 65
Tabela 12 68
Tabela 13 69
Tabela 14 69

Lista de Figuras

Figura 1 - Sistema 1
Figura 2 – Modelo Pinhole 5
Figura 3 – Plano perpendicular ao eixo X2 6
Figura 4 - Representação do fenómeno de ilusão de ótica que acontece com uma câmara apenas e a resolução desse problema com a obtenção de nova imagem 7
Figura 5 - Medidas necessárias à dedução das equações 8
Figura 6 – Deteção de características [1] 10
Figura 7 - Sensibilidade do método de Harris a variações de escala [15] 11
Figura 8 – Pirâmide DoG 12
Figura 9 - Comparação de pixels com pixels adjacentes na mesma escala e em escalas diferentes 12
Figura 10 - Filtros de Gauss (primeira linha); Simplificação após passagem de filtro de caixa (segunda linha); Filtros Haar (terceira linha) 14
Figura 11 - métodos de clustering [19] 15
Figura 12 - Diagrama explicativo do método DBSCAN [21] 16
Figura 13 - Planta da disposição dos obstáculos e da posição da câmara para o 1º teste 17
Figura 14 - imagem captada da disposição dos obstáculos 17
Figura 15 - Pontos de interesse encontrados na imagem (grayscale) 18
Figura 16 - Emparelhamento de pontos de interesse de duas imagens 18
Figura 17 - Vetores formados pelos pontos emparelhados 19
Figura 18 - Clustering dos vetores de pontos emparelhados 20
Figura 19 - Iterações para cálculo de distâncias 21
Figura 20 - Histogramas referentes às distâncias calculadas para os dois obstáculos 21
Figura 21 - Gráficos de dispersão e média calculada para os dois obstáculos 22
Figura 22 - Emparelhamento de pontos de interesse onde o clustering não identificou os obstáculos 23
Figura 23 - Clustering dos vetores de pontos emparelhados na figura 15 24
Figura 24 - Planta da disposição dos obstáculos e da posição da câmara para o 2º teste 25
Figura 25 - Imagem captada da disposição dos obstáculos na posição frontal 26
Figura 26 - Imagem captada da disposição dos obstáculos na posição rodada 26
Figura 27 - Mapeamento de obstáculos 27
Figura 28 - Clustering no intervalo de distâncias [20,40] em linha reta 28
Figura 29 - Emparelhamento de pontos de interesse no intervalo de distâncias [20,40] em linha reta 28
Figura 30 - Exemplo de espaço de configuração 33
Figura 31 - Posições calculadas de obstáculos (à esquerda); normalização para a malha (à direita) 36
Figura 32 - Aplicação do algoritmo A* ao problema 37
Figura 33 – Movimentos possíveis do sistema quando θ = 0º 38
Figura 34 – Movimentos possíveis quando: θ = -45º (esquerda); θ = 45º (direita) 38
Figura 35 – Pi Camera (esquerda); Ligação da câmara à Rpi 40

Figura 36 - Esquema global do algoritmo 41
Figura 37 -Laboratórios da Área Científica de Controlo, Automação e Informática Industrial (ACCAII) – Pavilhão de Mecânica III - IST 43
Figura 38 - RC 44
Figura 39 – RC com câmara acoplada 44
Figura 40 - Resposta a um degrau (power = 0.3) 45
Figura 41 - Resposta a um degrau (power = 0.75) 46
Figura 42 – Variação da posição em linha reta com o tempo 48
Figura 43 – Regressão linear 48
Figura 44 – Regressão polinomial de 3ªordem 49
Figura 45 – Desvio da regressão polinomial de terceira ordem 50
Figura 46 - Regressões sobrepostas (à esquerda) e regressão final composta (à direita) 50
Figura 47 – Movimento circular com direção e potência constante 51
Figura 48 – Aproximação de elipse ao movimento em curva à esquerda até orientação igual a 90º 52
Figura 49 – Tangente à elipse quando a orientação do RC é igual a -45º 53
Figura 50 - Aproximação de elipse ao movimento em curva à direita até orientação igual a 90º 53
Figura 51 - Tangente à elipse quando a orientação do RC é igual a 45º 54
Figura 52 – testes ao sistema 55
Figura 53 – Ilustração do movimento de curva à direita 56
Figura 54 – Movimento alternativo ao definido no algoritmo 57
Figura 55 – 1ºteste 59
Figura 56 - Erro na coordenada x (à esquerda) e erro na coordenada y (à direita) 59
Figura 57 – 2º teste 60
Figura 58 - Erro na coordenada x (à esquerda) e erro na coordenada y (à direita) 60
Figura 59 - 3º teste 61
Figura 60 - Erro na coordenada x (à esquerda) e erro na coordenada y (à direita) 61
Figura 61 - Disposição do ambiente no 1º teste 63
Figura 62 - Emparelhamento de figuras na inicialização do sistema 63
Figura 63 - Clustering da fase da inicialização 64
Figura 64 – Disposição do ambiente no segundo teste 65
Figura 65 - Emparelhamento de pontos de interesse no 2º teste 66
Figura 66 - Posições calculadas de obstáculos (à esquerda) e caminho estimado pelo algoritmo (à direita) na inicialização 66
Figura 67 - Emparelhamento de pontos de interesse no 2º teste (atualização do mapa) 67
Figura 68 - Posições calculadas de obstáculos (à esquerda) e caminho estimado pelo algoritmo (à direita) na atualização do mapa 67

Lista de abreviações

RC Radio Controlled (vehicle)

IDMEC Instituto de Engenharia Mecânica

IST Instituto Superior Técnico

USB Universal Serial Bus

IMU Inertial Measurement Unit

DBSCAN Density-based Spatial Clustering of Applications with Noise

3D Três Dimensões

SIFT Scale-invariant Feature Transform

SURF Speeded Up Robust Features

DoG Difference of Gaussian

RPi Raspberry Pi

VGA Video Graphics Array

CSI Camera Serial Interface

fps frames por segundo

NiMH Nickel–Metal Hydride Battery

MLU Movimento Linear Uniforme

RMSD Root Mean Square Deviation

Glossário

ε Raio Máximo

$\boldsymbol{C_{free}}$ Espaço Livre

$\boldsymbol{C_{obs}}$ Espaço onde se localizam os obstáculos

$\mathbf{C}(\mathbf{x})$ Custo de ir de x_i até x

$\boldsymbol{G}(\boldsymbol{x})$ Custo associado a ir de x até um estado Xg

$\widehat{\boldsymbol{G}}^{*}(\boldsymbol{x})$ Estimativa o mais próxima possível do custo ótimo, sem que ultrapasse esse mesmo custo

Capítulo 1

Neste capítulo pretende-se, primeiramente, introduzir o tema da dissertação, a motivação que leva à sua realização e a forma como está disposto o trabalho ao longo dos capítulos. De seguida, apresenta-se o trabalho relacionado com os temas abordados neste projeto, desenvolvido nos últimos anos, por outros autores, e por fim, apresentam-se as contribuições desta dissertação para a comunidade cientifica.

1.1 INTRODUÇÃO

Na última década tem-se assistido a uma expansão no aparecimento de veículos de condução autónoma. Parece certo que esta expansão irá alargar o seu impacto na sociedade [1]. A evolução dos meios de transporte tem seguido principalmente o rumo de melhorar as condições de segurança, redução do consumo energético e diminuição dos gases poluentes emitidos pelos transportes. A investigação nestas áreas levou a um aumento de interesse no desenvolvimento de veículos autónomos [2]. À semelhança dos transportes de pessoas e mercadorias, também nos veículos de reconhecimento, como drones ou veículos terrestres à escala têm sido alvo de desenvolvimento cientifico. É neste enquadramento que se encontra esta dissertação.

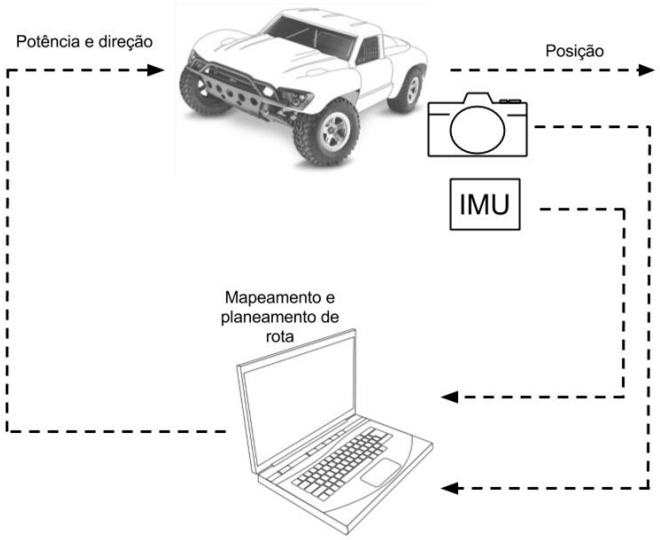

Figura 1 - Sistema

Este trabalho foi realizado com o objetivo de desenvolver software para implementar num sensor a aplicar num modelo RC à escala, para que este se pudesse mover autonomamente no espaço. Para este efeito, o sistema teria que interpretar a informação captada pelo sensor e criar um mapa por onde

se podia movimentar. O sensor a acoplar no RC trata-se de uma câmara. No entanto, com apenas uma câmara, a identificação de objetos e cálculo de distâncias aos mesmos não é um problema trivial, pelo que a primeira parte da dissertação (capítulo 2) recai sobre o método desenvolvido para resolver este problema. O que se propõe é uma forma de contornar as ambiguidades que advém da captação de uma imagem só, com a captação de várias imagens em diferentes posições do sistema.

A movimentação do sistema no espaço é estudada no terceiro capítulo, introduzindo os princípios de planeamento de rota e o algoritmo desenvolvido para o sistema global funcionar, seguido de um capítulo de identificação do sistema onde são estudados os parâmetros de movimento do RC.

Para testar o funcionamento do sistema, foram realizados vários testes onde se averigua a precisão com que o RC efetua um determinado caminho delineado pelo algoritmo de planeamento de rota. Depois realizaram-se alguns testes com obstáculos dispostos no espaço e testou-se validade do sistema global em funcionamento, tendo como objetivo perceber a qualidade com que o sistema identifica os obstáculos no seu caminho e quais os problemas que surgem. Estes resultados são discutidos no penúltimo capítulo e, finalizando com as conclusões e trabalho futuro.

O RC é um modelo à escala 1/18 no qual já foi desenvolvido software anteriormente [3] para o controlo do mesmo, remotamente, através de Matlab. No próprio RC existe uma Raspberry Pi 3 Model B que trata da comunicação entre os comandos Matlab enviados por um computador remoto e o controlador de potência e direção do RC. Existe ainda uma IMU ligada à raspberry Pi presente no RC, equipada com acelerómetro, magnetómetro e giroscópio.

A câmara utilizada neste projeto é uma *Pi camera* [4]*,* que se trata de uma câmara específica para a Raspberry Pi com ligação direta à mesma, para que a transmissão de informação seja mais eficiente do que no caso de uma câmara com ligação USB, por exemplo.

O código desenvolvido neste trabalho foi escrito maioritariamente em *Matlab*. Apenas a comunicação com o RC envolveu código em *python.*

1.2 TRABALHO RELACIONADO

O ser humano e a maior parte dos mamíferos, têm na visão, o seu sentido mais importante para perceção da realidade que os rodeia. É impressionante a facilidade com que o cérebro interpreta o mundo tridimensional através dos nossos olhos. Sem aparente esforço, é possível identificar objetos, contar o número de objetos, ter noção de profundidade ou distinguir as várias cores e transparências [5]. A área de investigação denominada Visão Computacional visa a aproximação por câmaras e meios computacionais às capacidades de visão dos seres vivos.

Dentro da área de visão computacional, um dos principais desafios prende-se com a recuperação da tridimensionalidade do mundo a partir de uma imagem. Entre os vários métodos baseados em imagens para a medição de distâncias a objetos, encontram-se a visão stereo (duas câmaras), uma câmara

apenas, luz estruturada, *time of flight cameras*, entre outros. Muita da bibliografia existente aborda este tema com visão binocular (stereo) [6] [7] [8], dado que é uma forma relativamente precisa de calcular profundidade. No entanto, devido à necessidade de obtenção e processamento de duas imagens simultaneamente, torna-se num método mais carregado computacionalmente. No âmbito de robótica existem já soluções mais simples só com uma camara, pelo que o uso da visão stereo é agora mais limitado [9]. O recurso a camaras *time of flight* permite a obtenção de informação acerca das distâncias a objetos através da medição do tempo que a luz demora a percorrer a distância desde a câmara ao objeto e de volta novamente à câmara. No entanto, estas camaras ainda apresentam custos proibitivos estando o seu uso limitado a aplicações específicas.

O recurso a luz estruturada, também reservado a aplicações específicas, vulgarizou-se bastante, em particular, através do sensor *Microsoft Kinect*.

Contudo, neste projeto pretendia-se implementar um método que utilizasse apenas uma câmara, dado que o espaço disponível no RC para acoplar sensores é reduzido, o que leva a que seja necessário pensar em sensores mais simples. Representa ainda um desafio interessante a tentativa de encontrar um método para contornar o facto de ser matematicamente impossível extrair profundidade de uma só imagem [6].

Os trabalhos existentes relacionados com perceção de profundidade usando uma câmara apenas se focam em algoritmos de aprendizagem (*machine learning*) [6], ou na adição de um laser ou movimento lateral da câmara para obter duas perspetivas laterais e adaptar a visão stereo [7] [8].

O desafio do trabalho consiste em utilizar o movimento do sistema (RC e sensor) como ajuda à identificação e localização de obstáculos.

Depois de definir um método de localização dos obstáculos pela câmara, é necessário formular um método que crie um caminho em volta dos obstáculos encontrados. Este caminho é definido por um algoritmo de planeamento de rota. É importante notar que o caminho criado pelo algoritmo de planeamento de rota terá que ser atualizado à medida que a câmara vai obtendo mais informação sobre o espaço. Os algoritmos de planeamento de rota têm como principio a deslocação de uma configuração inicial do sistema até uma configuração final. Para a definição do algoritmo a usar, é importante definir se o algoritmo será introduzido num sistema offline ou online e qual a dimensionalidade do espaço em que o sistema se pode movimentar. Estes algoritmos são processos de otimização, visto que, numa grande parte das vezes se pretende que os caminhos encontrados sejam ótimos ou sub-ótimos no que diz respeito a tempo, distância ou energia.

O problema de planeamento de rotas no âmbito da robótica é um problema que continua a ser estudado e para o qual muitas soluções já foram propostas [9] e vários tipos diferentes de métodos aplicados, como métodos de espaço C, métodos de campo potencial ou redes neuronais [10]. Para o problema presente neste trabalho, dado se tratar de um problema onde o espaço de configuração em que o sistema se pode mover é bidimensional usou-se a um algoritmo baseado num método de espaço C e que não sobrecarregasse o sistema computacionalmente. De modo a resolver o problema, recorreu-se

maioritariamente a [11] [12], onde Steven M. LaValle apresenta uma extensa investigação e explicação de métodos relacionados com planeamento de rotas.

1.3 CONTRIBUIÇÕES

As principais contribuições deste trabalho são:

- Estudo e implementação de um algoritmo de visão monocular para a obtenção de estruturas do ambiente a partir do movimento do robô;
- Estudo e implementação de planeamento de trajetórias para evitar obstáculos;

Capítulo 2 – Processamento de Imagem

Neste capítulo será introduzida a vertente de processamento de imagem desenvolvida neste projeto, começando por uma parte de explicação da geometria e funcionamento do processo de captação de imagem (modelo de pinhole). Depois disto, surgem algumas deduções trigonométricas que levam a que seja possível o cálculo de distâncias a obstáculos não usando visão stereo no seu sentido tradicional. De seguida introduz-se o que são pontos de interesse e descritores de uma imagem, incluindo a descrição de métodos e o método escolhido para o trabalho. No tópico seguinte aborda-se o problema de classificação de obstáculos através de clustering, onde se explica o princípio de escolha que levou a usar o método DBSCAN e como este funciona. Por fim, os dois subtópicos finais abordam dois testes de validação do método de mapeamento introduzido no capítulo.

2.1 Geometria e cálculos

2.1.1 Modelo de *Pinhole*

Na abordagem ao problema do cálculo das distâncias entre o sensor (câmara) e os obstáculos, é necessário compreender primeiro como funciona a captação de imagem pela câmara.

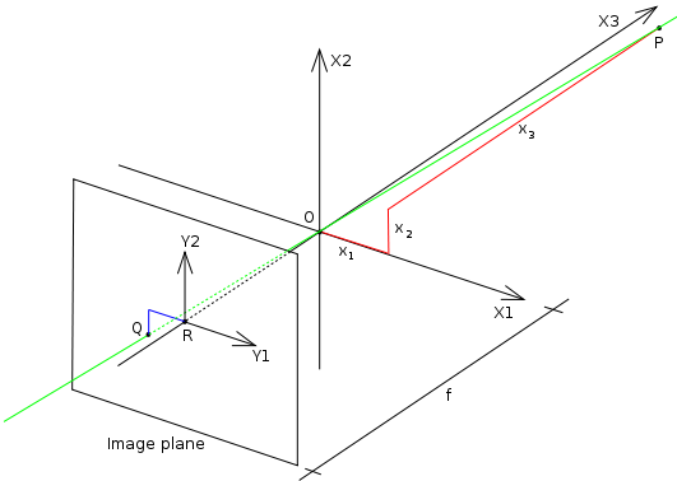

Figura 2 – Modelo *Pinhole*

O modelo de *pinhole* usa semelhança de triângulos para fazer corresponder matematicamente um ponto numa imagem a um ponto físico no mundo real tridimensional. Na figura anterior está representado um esquema representativo do modelo de *pinhole.* Estão representados 3 eixos ortogonais (X1, X2 e X3) onde o ponto de interceção dos mesmos corresponde ao centro ótico da câmara (O). O eixo X3 é chamado eixo principal da câmara, e o plano representado, perpendicular ao plano principal a uma distância *f* (distância focal), é denominado plano focal.

O ponto **P** tem coordenadas (x1, x2, x3) e está projetado no plano focal, onde se identifica pelo ponto **Q** cujas coordenadas nos eixos ordenados locais (Y1, Y2) são (y1, y2).

De maneira a encontrar as relações entre as coordenadas do ponto **P** e do ponto **Q**, representa-se uma projeção da figura 2 no eixo X2.

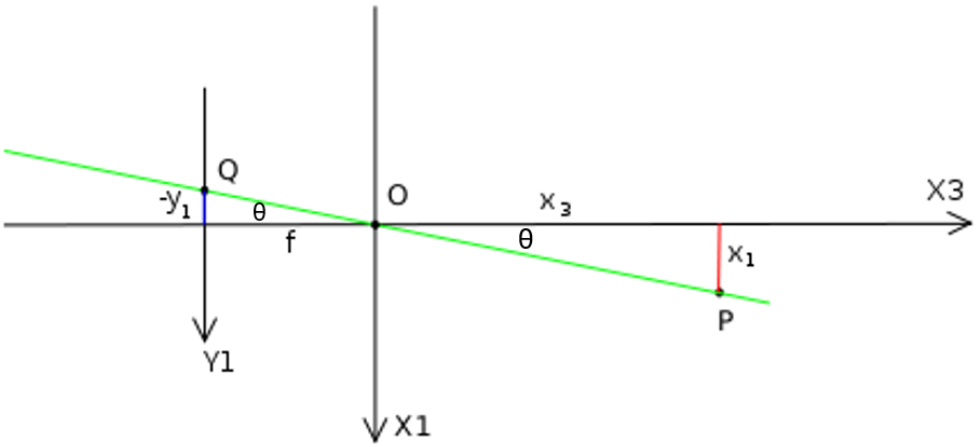

Figura 3 – Plano perpendicular ao eixo X2

A semelhança de triângulos referida anteriormente é explicita na figura, podendo retira-se algumas relações importantes da figura.

$$\tan(\theta_1) = \frac{x_1}{x_3} \quad (1)$$

$$\tan(\theta_1) = \frac{-y_1}{f} \quad (2)$$

De forma análoga, se for feita uma projeção no eixo X1, deduz-se,

$$\tan(\theta_2) = \frac{x_2}{x_3} \quad (3)$$

$$\tan(\theta_2) = \frac{-y_2}{f} \quad (4)$$

Nota: para as deduções seguintes é importante realçar que a distância focal é uma distância constante em todos os processos.

2.1.2 DEDUÇÕES TRIGONOMÉTRICAS

Apenas com uma câmara, não é possível a determinação de distâncias a objetos, a menos que se tenha alguma referência ou se saiba informação prévia sobre os objetos (altura, área, por exemplo). Não tendo informação relativa ao objeto captado ou nenhum outro ponto de referência, algumas opções são viáveis, como foi referido anteriormente, por exemplo, a adição de outra câmara de modo a obter uma captação stereo do objeto e fazer triangulação da informação nas duas imagens captadas a cada instante. Na impossibilidade de acrescentar uma nova câmara, usou-se um método de obtenção de duas imagens seguidas, após movimento linear da câmara.

De maneira a perceber melhor que através de uma imagem apenas, não é possível retirar informação suficiente sobre a posição de obstáculos, representa-se na figura 4 uma situação em que dois elementos, a distâncias distintas podem parecer ter a mesma altura na visão da câmara, quando na verdade isso não é o que acontece.

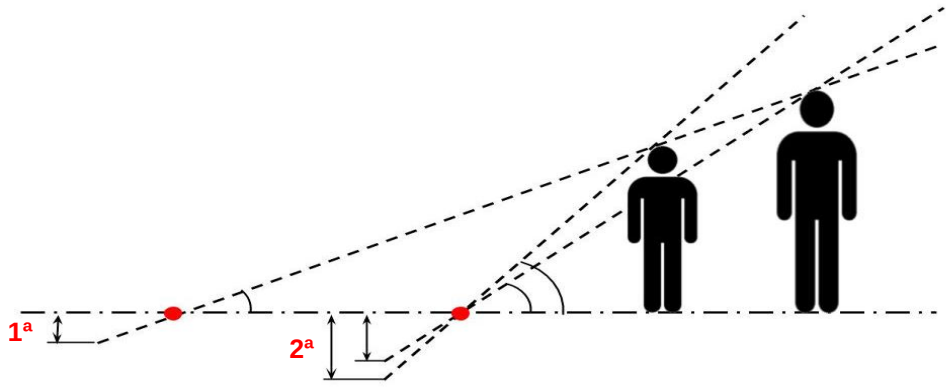

Figura 4 - Representação do fenómeno de ilusão de ótica que acontece com uma câmara apenas e a resolução desse problema com a obtenção de nova imagem

Fazendo uma pequena análise da figura anterior, consegue observar-se o que foi descrito anteriormente. Caso fosse obtida apenas a imagem na primeira posição, o número de pixels correspondente às alturas dos dois elementos seria o mesmo, o que seria obviamente errado assumir em quaisquer cálculos para determinação de distâncias aos elementos.

Com a adição da segunda imagem, permanecendo ambos os elementos na mesma posição, o fenómeno de ilusão de ótica referido anteriormente, já não se verifica, tendo agora duas posições diferentes para as alturas de cada um dos elementos. Assim, conclui-se que com duas imagens obtidas após movimento linear da câmara não existem ambiguidades quanto às posições dos elementos. É baseado nesta premissa que está assente o método de cálculo de distâncias a objetos proposto neste trabalho.

A figura 5, análoga à anterior, pretende introduzir algumas medidas necessárias aos cálculos feitos posteriormente.

Na figura, é então possível ver, novamente, duas posições diferentes para a câmara, simulando posições diferentes após o movimento linear do Sistema, mas neste caso, apenas um objeto, de altura **a,** é identificado. Considera-se os valores da distância $\mathbf{x_1}$ (mm), $\mathbf{p_1}$ e $\mathbf{p_2}$ (pixels), conhecidos. As outras medidas representadas são incógnitas do problema, sendo que o objetivo é calcular a distância $\mathbf{x_2}$ (distância entre a 2ª posição da câmara e o objeto).

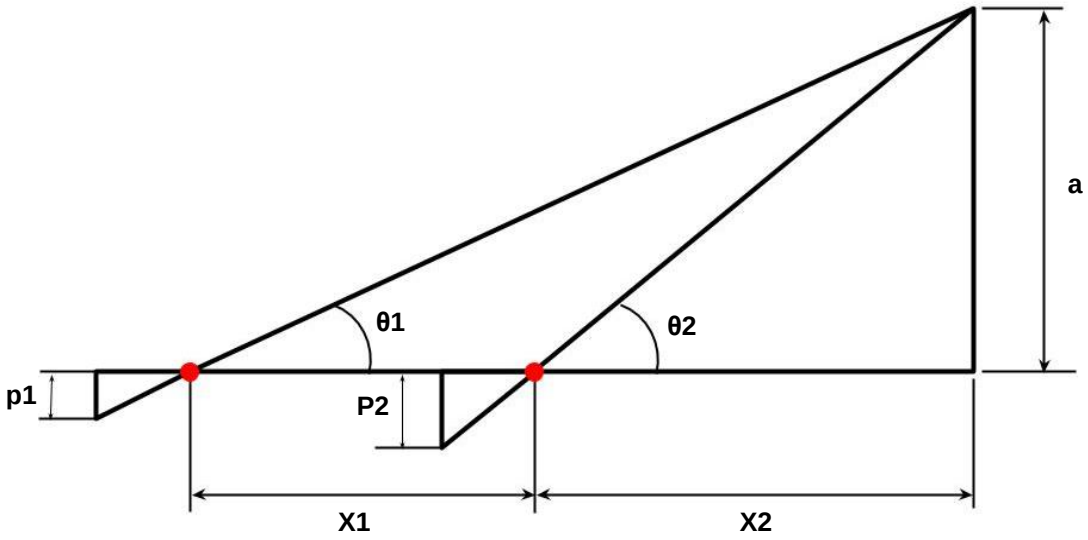

Figura 5 - Medidas necessárias à dedução das equações

Analisando a figura e recorrendo novamente a noções de trigonometria e semelhança de triângulos, tal como foi feito em relação à figura 1, conseguem retirar-se as seguintes relações:

$$\tan(\theta_1) = \frac{a}{x_1 + x_2} \tag{5}$$

$$\tan(\theta_2) = \frac{a}{x_2} \tag{6}$$

Isolando e resolvendo ambas as equações anteriores em ordem a a, chega-se aos seguintes resultados:

$$\mathrm{a} = (x_1 + x_2)\tan(\theta_1) \tag{7}$$

$$\mathrm{a} = x_2 \tan(\theta_2) \tag{8}$$

Igualando as equações (7) e (8) e isolando x_2,

$$(x1 + x2)\tan(\theta_1) = x2\tan(\theta_2) \quad <=>$$

$$x2 = \frac{x1\ tan(\theta_1)}{tan(\theta_2) - tan(\theta_1)} \tag{9}$$

Recorrendo à equação (2), e substituindo os valores de θ1 e p1 e de θ2 e p2, obtêm-se relações válidas para as tangentes dos ângulos:

$$\tan(\theta_1) = \frac{p_1}{f} \tag{10}$$

$$\tan(\theta_2) = \frac{p_2}{f} \tag{11}$$

Substituindo (8) e (9) na equação (7), obtém-se:

$$x_2 = \frac{x_1 \frac{p_1}{f}}{\frac{p_2}{f} - \frac{p_1}{f}} \tag{12}$$

Como a distância focal **f** é uma propriedade intrínseca da câmara, é constante nas diferentes imagens, pelo que a equação (12) se simplifica em:

$$x_2 = \frac{x_1 \, p_1}{p_2 - p_1} \tag{13}$$

A equação (13) apenas depende de variáveis conhecidas, as distâncias medidas em pixels nas duas captações de imagens (p1 e p2) e a distância percorrida pela câmara entre esses dois pontos (x1), pelo que esta é a relação encontrada para calcular distâncias a objetos neste projeto.

2.2 PONTOS DE INTERESSE E DESCRITORES

Encontrar correspondências entre duas imagens referentes a um mesmo cenário é uma área muito estudada pelo ramo de visão computacional e que tem diversas aplicações, tais como: Calibração de câmara, reconstrução 3D ou identificação de objetos [13]. Este processo denomina-se deteção de características *(feature detection)* e emparelhamento de características *(feature matching)*. As características mais facilmente reconhecíveis entre duas imagens serão por exemplo cantos de edifícios, ou formas peculiares identificáveis nas imagens que estão a ser analisadas. Características como orientação ou aparência local, encontram-se em pontos de interesse da imagem e, para além de permitirem o emparelhamento de objetos em imagens diferentes de um cenário semelhante, podem ainda ajudar a identificar as fronteiras dos objetos. O conjunto de características associadas a um determinado ponto de interesse são englobadas no descritor desse ponto de interesse. Os descritores podem conter tipos de características diferentes, consoante o método utilizado.

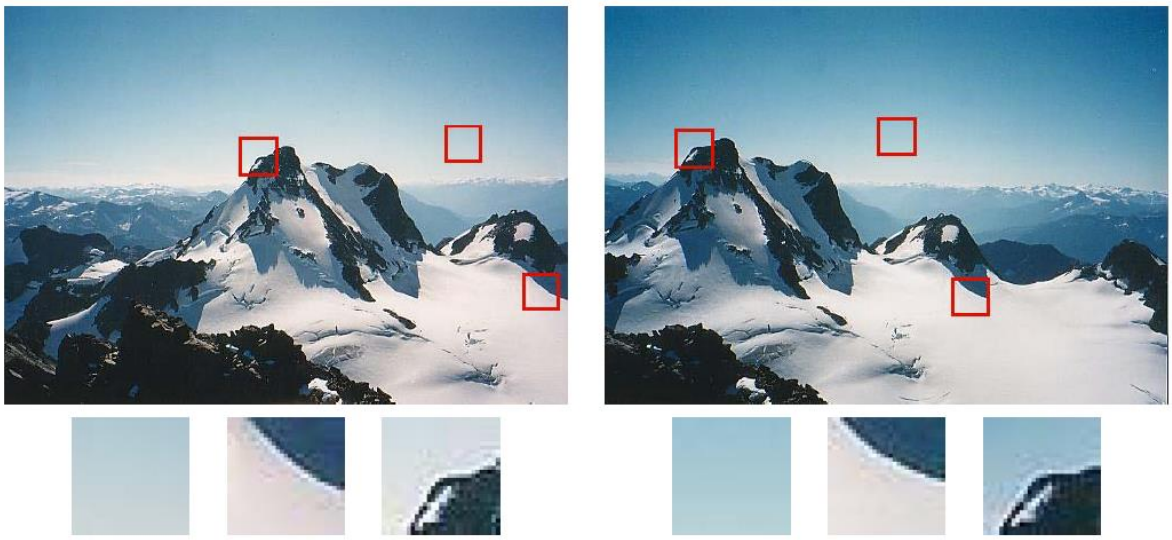

Figura 6 – Deteção de características [1]

Na figura 6 podemos observar duas imagens diferentes de uma mesma paisagem onde estão assinalados três fragmentos em cada imagem. Analisando apenas a olho nu os três fragmentos representados, é fácil perceber que existem zonas mais identificáveis que outras. Os fragmentos, como o primeiro, em que não existe praticamente textura, não são bons pontos de interesse, ao contrário do terceiro fragmento por exemplo, onde as variações de contraste são grandes, o que permite melhores correspondências entre imagens.

2.2.1 Características SIFT e SURF - Introdução

Um dos primeiros métodos introduzidos e ainda um dos mais estudados e utilizados para fazer deteção de pontos de interesse é o método de deteção de cantos – Harris *corner detection* [14], um método que se caracteriza, tal como o próprio nome indica, por identificar pontos de interesse nos cantos representados numa imagem. Este método tem como característica não ser sensível à rotação de imagem, querendo com isto dizer que, mesmo que a imagem esteja rodada, serão identificados os mesmos pontos de interesse que se identificariam numa outra configuração de imagem, no entanto é um método sensível a variações de escala nas imagens, o que é um ponto crucial para este trabalho. Por exemplo, no caso da figura 6, o canto representado numa escala menor, quando aumentado já não apresenta características de canto, segundo o tamanho de janela representado, pelo que o método de Harris já não encontraria semelhanças entre a informação recolhida.

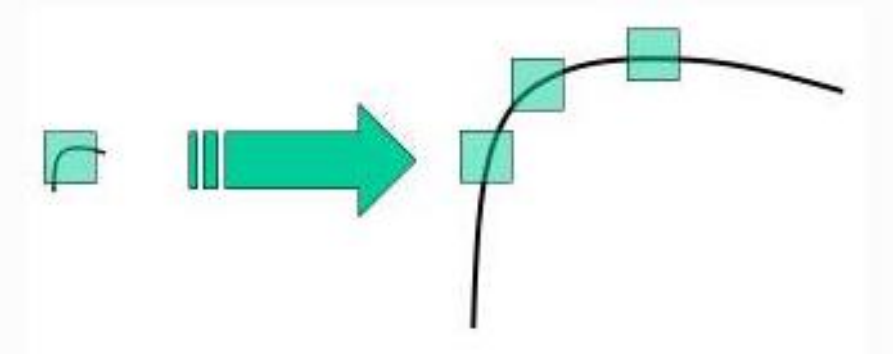

Figura 7 - Sensibilidade do método de Harris a variações de escala [15]

Neste projeto, é necessário usar um método que não seja sensível a variações de escala, pelo que o método de Harris não foi considerado.

Desde a introdução do método de Harris, outros métodos de deteção de pontos de interesse foram introduzidos com outras caracteristicas, muitos deles não sendo sensíveis a variações de escala. O método SIFT – *Scale-invariant Feature Transform*, introduzido em 2004 [16] e o método SURF – *Speeded Up Robust Features* introduzido em 2006 [13] são métodos invariantes a rotação e escala. O método SURF é uma versão menos pesada computacionalmente do que o método SIFT.

Como foi referido, da figura 7, percebe-se que para detetar cantos numa escala maior, não pode usar-se o mesmo tamanho de janela, sendo necessário uma janela maior também. Para isto, é necessário aplicar um filtro de espaço de escala. O método SIFT obtém os seus pontos de interesse a partir dos extremos de um espaço de escala de difference of Gaussian (DoG) dentro de uma pirâmide de *difference of Gaussian* [17] [18].

O processo de construção da pirâmide de *DoG* é composto por várias suavizações de valor σ de uma imagem, fazendo depois a diferença entre os níveis de suavização adjacentes. Os pontos de interesse são depois obtidos a partir dos pontos extremos das *DoG* tendo em conta não só a posição espacial, como também a escala. Basicamente, σ funciona como um parâmetro de escala. Uma *difference of Gaussia*n D (x,y,σ) é dada por:

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma) \tag{14}$$

onde, L(x,y,kσ) corresponde à convolução da imagem original I(x,y) com a suavização de Gauss G(x,y,kσ), portanto,

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y) \tag{15}$$

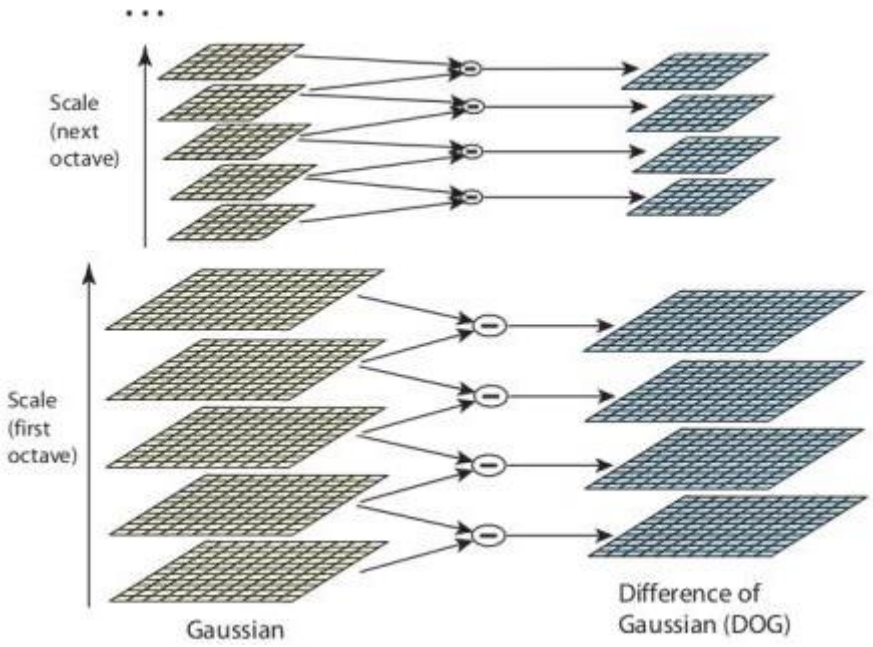

Figura 8 – Pirâmide DoG [16]

Quando estão definidas as imagens da pirâmide, os pontos de interesse são identificados através de máximos ou mínimos locais das imagens ao longo das diferentes escalas. Estes extremos são identificados pela comparação de cada pixel com os seus 8 pixels adjacentes numa mesma escala e com os 9 pixels correspondentes de escalas diferentes. Os pixels que sejam máximos ou mínimos no fim das comparações são dados como candidatos a pontos de interesse.

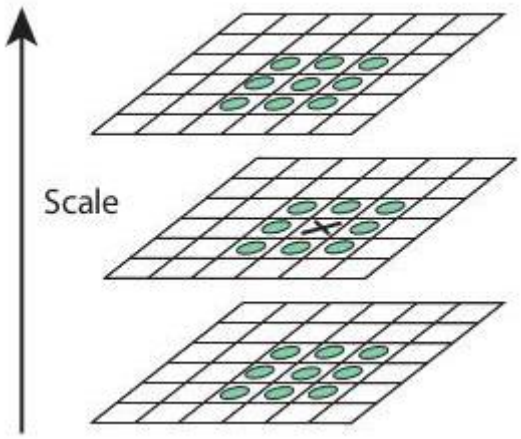

Figura 9 - Comparação de pixels com pixels adjacentes na mesma escala e em escalas diferentes [16]

De modo a obter insensibilidade à rotação, é atribuída uma orientação a cada ponto de interesse considerado. Esta orientação é atribuída da seguinte forma.

Para cada ponto de interesse identificado através do método anterior atribui-se um descritor. O descritor SIFT é composto por informação relativa a direções e magnitudes de gradientes locais nas imagens suavizadas.

Para cada imagem suavizada L(x,y) correspondente a um ponto de interesse na sua escala σ é calculada a magnitude m(x,y) e a orientação θ(x,y) do gradiente:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y)^2 + (L(x,y+1) - L(x,y+1)^2} \quad (16)$$

$$\theta(x,y) = atan2\big(L(x,y+1) - L(x,y-1), L(x+1,y) - L(x-1,y)\big) \quad (17)$$

Os cálculos de magnitudes e direções para o gradiente são feitos para cada pixel da vizinhança do ponto de interesse na imagem L. Cria-se um histograma com 36 *bins* a cobrir os 360º em redor do ponto de interesse em questão.

Cria-se então um descritor para cada ponto de interesse. Numa vizinhança de 16x16 em volta do ponto de interesse é dividida em sub-blocos de tamanho 4x4 e para cada sub-bloco cria-se um histograma de 8 *bins*, totalizando assim 128 *bins* por descritor.

Definidos os mecanismos de obtenção de descritores, pode aplicar-se o método a diferentes imagens de um mesmo cenário e depois fazer o emparelhamento dos descritores encontrados nas várias imagens. O emparelhamento consiste em cada ponto numa imagem encontrar na outra imagem o ponto que minimiza a distância euclidiana entre os descritores representados.

2.2.2 SURF

O método *SURF* foi introduzido depois do método *SIFT* de maneira a diminuir o processamento necessário e, como tal, a acelerar o processo de obtenção de pontos de interesse e descritores.

O método *SIFT*, como foi referido, utiliza *difference of Gaussian* como aproximação a *Laplacian of Gaussian (LoG)* para a deteção de pontos de interesse, enquanto que o método SURF usa filtro de caixa baseado na matriz Hessiana, também como aproximação a *LoG*. A vantagem de usar filtro de caixa em relação aos métodos para deteção de pontos de interesse noutras abordagens deve-se à possibilidade de assim poder usar-se a imagem integral, fazendo com que o tempo despendido na deteção de pontos de interesse não dependa do tamanho do filtro, enquanto que no caso de filtros de Gauss, o tempo necessário para realizar o mesmo processo é da ordem de grandeza do tamanho da imagem multiplicado pelo tamanho do filtro. O método SURF subdivide a vizinhança de cada ponto de interesse em quadrados de 20x20 pixels, subdivididos ainda em conjuntos de 4x4 pixels e em cada um desses conjuntos passa um filtro de Haar. Para cada subdivisão de 4x4 pixels, são calculados valores para a soma das respostas dos filtros e para o módulo das respostas dos filtros, na direção horizontal e vertical, $v = \big(\sum d_x, \sum d_y, \sum|d_x|, \sum|d_y|\big)$. O vetor de características deste método tem então apenas 64 dimensões, ou seja, metade, quando comparado com o método SIFT.

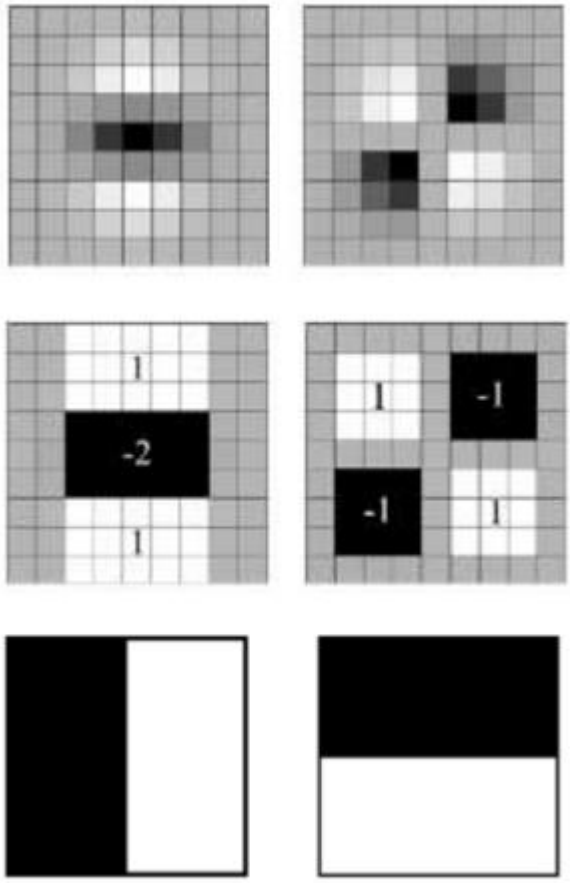

Figura 10 - Filtros de Gauss (primeira linha); Simplificação após passagem de filtro de caixa (segunda linha); Filtros Haar (terceira linha)

2.3 CLUSTERING

No ramo de visão computacional, mais especificamente na vertente de segmentação, já foram estudados e desenvolvidos muitos algoritmos destinados a diferentes utilizações. Para o trabalho em questão, a necessidade de distinguir diferentes objetos no espaço leva a que seja necessário recorrer a métodos de segmentação de imagem ou *clustering*.

Os dados que se pretende agrupar em *clusters* advêm dos pontos de interesse encontrados nas imagens, tendo como objetivo agrupar os pontos pertencentes a cada objeto diferente, em *clusters* diferentes também. Assim, presume-se que os pontos recolhidos de cada objeto estejam agrupados próximos uns dos outros em aglomerado.

Tendo em conta o objetivo do trabalho, a escolha dos métodos a implementar nos algoritmos teria que ser adequada ao pretendido, mas com a condição de não poder ser um método muito exaustivo computacionalmente, dado que para o sistema funcionar, a partir do momento em que a câmara adquire informação, não pode passar muito tempo até que o sistema tenha informação sobre a localização dos obstáculos. Assim, avaliou-se os possíveis métodos de *clustering*.

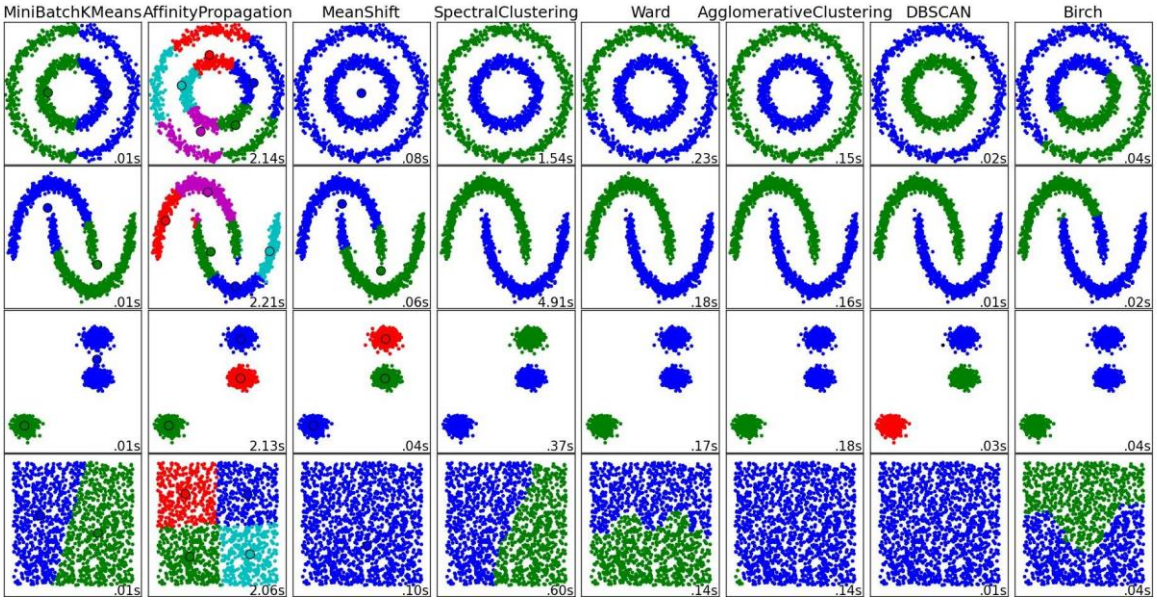

Figura 11 - métodos de *clustering* [19]

Na figura anterior estão representados vários métodos de *clustering* aplicados em iguais conjuntos de dados. De entre os conjuntos de dados apresentados, os esquemas que mais se assemelham aos dados que se pretende agrupar são, principalmente, a terceira linha, e em casos específicos, a segunda. Posto isto, excluímos imediatamente os métodos: *Affinity Propagation* e *Birch.* Tendo em conta que o número de obstáculos a identificar é aleatório, o método de clusterização não pode ter como input um número de clusters. Assim, elimina-se também o método *K-means, Ward* e *Agglomerative Clustering.* Para além do já referido, os métodos *DBSCAN e Mean Shift* são os que apresentam menor tempo de computação nos esquemas pretendidos, 0.01s a 0.03s e 0.04s a 0.06s, respetivamente. De entre estes dois métodos, escolheu-se o método DBSCAN, por apresentar melhores características quando comparando os dois métodos em conjuntos de dados semelhantes aos que se encontram na segunda linha da imagem.

2.3.1 DBSCAN

Density-based spatial clustering of applications with noise [20] é o significado da sigla DBSCAN, que se trata de um algoritmo de *clustering* baseado em zonas de maior e menor densidade de pontos.

Dado um certo grupo de pontos, o algoritmo agrupa os pontos que estão mais próximos (pontos com muitos vizinhos), deixando os pontos que se encontram sós (pontos mais próximos, demasiado longe) ou em zonas de menor densidade de dados como *outliers*.

O método requer 2 parâmetros: ε, raio máximo possível para a vizinhança do ponto *p,* e o número mínimo de pontos necessários para formar um cluster (minPts). Começa por um ponto *p* aleatório e agrega a sua vizinhança. Se contiver pontos suficientes para satisfazer as condições impostas, começa um cluster. Caso contrário, este ponto é classificado como *noise.*

Se um ponto pertencer a um cluster, toda a sua vizinhança ε pertence também a esse cluster. Quando todos os pontos de um determinado cluster estão definidos, o algoritmo segue para um ponto não-visitado e segue o mesmo procedimento.

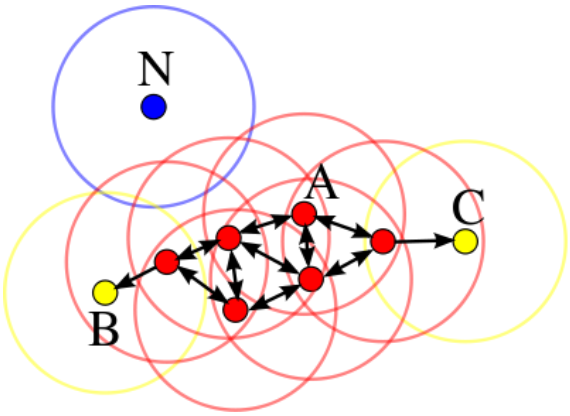

Figura 12 - Diagrama explicativo do método DBSCAN [21]

Na figura acima estão representados alguns pontos de um grupo de dados. Foi definido que o número mínimo de pontos necessário para a formação de um cluster seria minPts = 4 e que o raio máximo da vizinhança de cada ponto seria representado pelas circunferências em volta de cada ponto. O ponto A, juntamente com todos os pontos vermelhos representados, pertencem ao núcleo do cluster, dado que todos eles têm, na sua vizinhança ε, pelo menos 4 pontos (incluindo o próprio ponto). Os pontos B e C, apesar de não terem 4 pontos na sua vizinhança, estão na vizinhança de pontos nucleares, pelo que pertencem também ao cluster formado por pontos vermelhos. O ponto N é um ponto de noise, porque na sua vizinhança não existe nenhum ponto pertencente ao núcleo do cluster.

2.4 VALIDAÇÃO

De maneira a testar o método de identificação de pontos de interesse e o de *clustering*, procedeu-se a dois testes de identificação de obstáculos, cálculo de distâncias aos mesmos e mapeamento.

2.4.1 1º TESTE

Numa primeira fase, pretendeu-se apenas validar o método de cálculo das distâncias a objetos e a identificação de diferentes objetos. Para isto, a ideia era avançar, em linha reta, a distâncias conhecidas uma câmara e tirar várias imagens de um mesmo cenário composto por mais do que um objeto.

Deste modo, criou-se um plano com obstáculos a serem identificados e avaliada a distância até aos mesmos. Para este teste, foram então dispostos dois cadernos (tamanho A5) a distâncias diferentes da posição de captação das imagens. Com isto, pretendia-se ter dois objetos a detetar e a calcular as distâncias.

O caderno da direita encontra-se a 60 cm da parede, enquanto que o da esquerda se encontra a 30 cm da mesma. Foram captadas 5 imagens (150 cm, 140 cm, 130 cm, 110 cm e 100 cm) do ambiente demonstrado, sendo as distâncias referentes às mesmas relativas à parede.

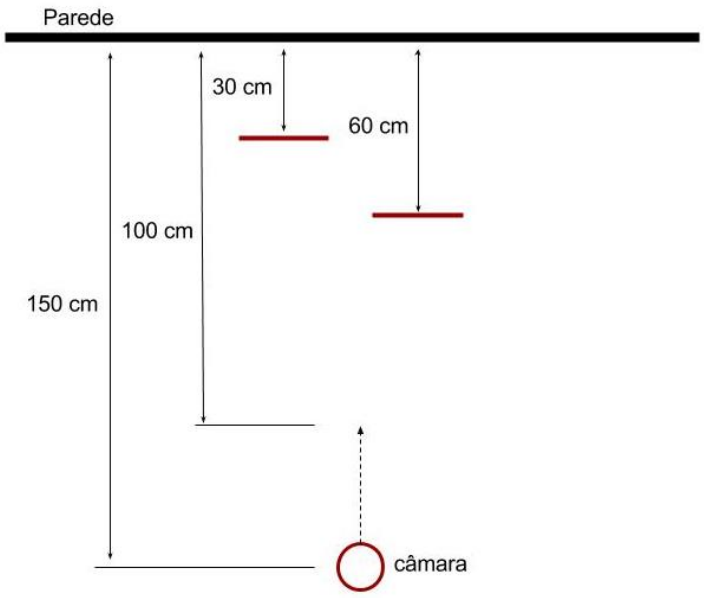

Figura 13 - Planta da disposição dos obstáculos e da posição da câmara para o 1º teste

A figura 13 é uma das imagens captadas para o estudo realizado, neste caso a uma distância da parede de 150 cm.

Figura 14 - imagem captada da disposição dos obstáculos

Como está descrito anteriormente, após a captação e leitura das imagens pela câmara, o algoritmo procura encontrar pontos de interesse através do método SURF nas duas imagens, separadamente, de modo a seguidamente usar descritores desses mesmos pontos de interesse e posteriormente fazer o emparelhamento dos descritores compatíveis encontrados na primeira e na segunda imagem.

Na figura seguinte está representada a imagem anterior, com os pontos de interesse encontrados representados sobre a mesma.

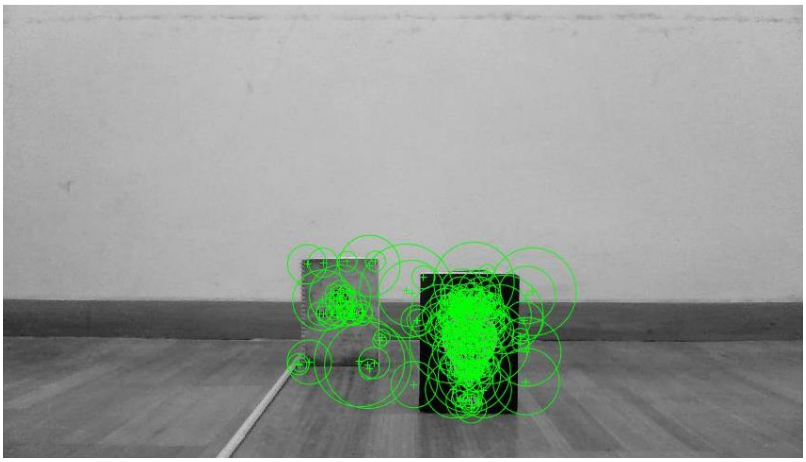

Figura 15 - Pontos de interesse encontrados na imagem (grayscale)

Analisando a figura 15 pode imediatamente perceber-se que a maioria dos pontos de interesse encontrados se encontram nos objetos dispostos, o que vai de encontro ao pretendido. É possível notar ainda que o objeto da direita tem uma incidência maior de pontos de interesse sobre o mesmo.

Seguindo os passos do algoritmo, após a definição dos descritores, procede-se então ao emparelhamento e à remoção dos pontos de *noise*. Este processo reduz em grande parte os pontos encontrados no processo anterior, como se pode facilmente deprender da imagem seguinte.

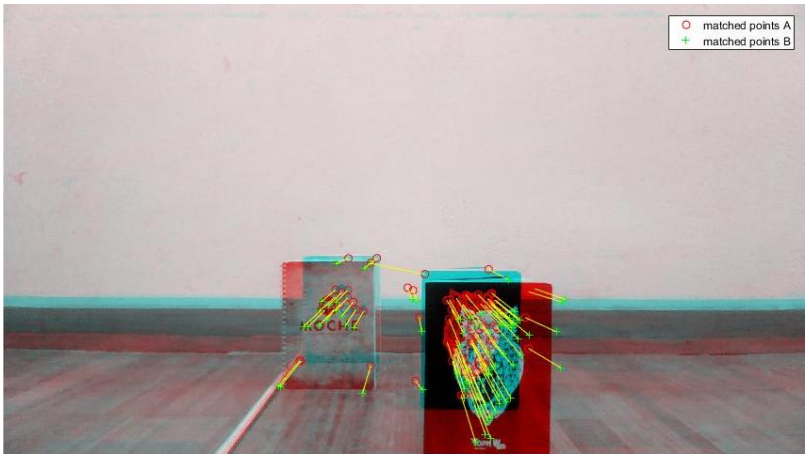

Figura 16 - Emparelhamento de pontos de interesse de duas imagens

Analisando a figura 16, e sabendo que os pontos representados por cruzes (+) verdes são referentes aos pontos de interesse encontrados na segunda imagem captada, neste caso a 130 cm de distância da parede, e os pontos representados por círculos (o) vermelhos são referentes à primeira imagem captada, neste caso a 150 cm de distância da parede, podem tirar-se algumas conclusões.

Tendo em conta que objetos mais distantes da câmara terão necessariamente menor diferença de posição em relação à posição da câmara, quando comparados com objetos mais próximos, seria de esperar que os deslocamentos dos pontos emparelhados no objeto da direita (mais próximo) estejam mais afastados dos seus pares emparelhados, em comparação com os pontos do objeto da esquerda (mais afastado).

Outro aspeto que deve ser notado no emparelhamento dos descritores é a direção das linhas que unem os pontos emparelhados.

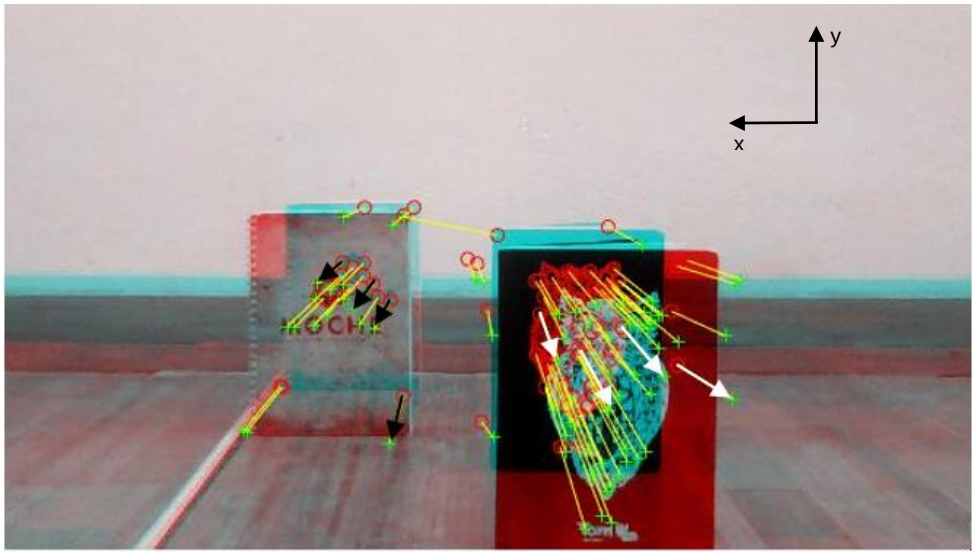

Figura 17 - Vetores formados pelos pontos emparelhados

As evidências referidas no parágrafo anterior estão realçadas na figura 17 na forma de setas. Se se descrever as setas representadas na imagem como vetores, e tendo em conta o referencial representado no canto superior direito da imagem, percebemos que os vetores de cor preta têm direção de coordenada x no sentido positivo e coordenada y no sentido negativo, enquanto que os vetores de cor branca, têm ambas as direções de coordenadas, x e y, no sentido negativo.

As conclusões tiradas desta análise são úteis na formação de clusters que vem de seguida, porque a variação de normas e direções dos vetores vai posicionar o conjunto de vetores referentes a cada objeto em zonas diferentes do referencial cartesiano, o que se espera que seja suficiente para que o método de *clustering* faça a distinção entre os pontos referentes a cada objeto.

Os pontos que se pretende agrupar são os vetores referidos nos parágrafos anteriores.

Os parâmetros usados na função DBSCAN são os seguintes:

- Raio máximo dos pontos da vizinhança de um ponto p, pertencente a um cluster (ε) = 9px
- Mínimo de pontos por cluster = 5

Nota: Estes valores foram obtidos através de várias iterações com todas as combinações das 5 imagens captadas, para otimização dos resultados.

Para o caso de exemplo que está a ser descrito, apresenta-se de seguida o gráfico onde estão representados os pontos dos vetores da figura 17.

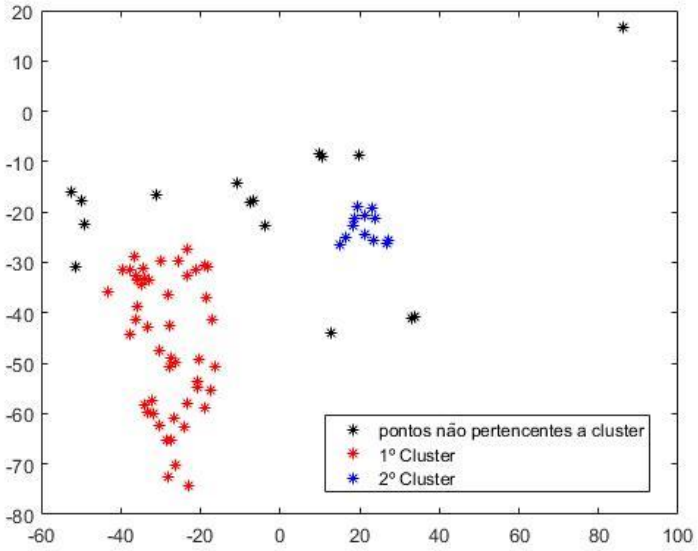

Figura 18 - *Clustering* dos vetores de pontos emparelhados

De acordo com o esperado, os pontos azuis estão colocados em zonas onde a coordenada x é positiva e a coordenada y é negativa, enquanto que os pontos vermelhos estão numa zona em que ambas as coordenadas x e y são negativas, pelo que é fácil fazer a correspondência entre cada conjunto de pontos pertencentes a clusters e o objeto que identificam.

É de notar:

Uma maior quantidade de pontos referentes ao objeto da direita conseguiu passar pelos processos de redução de pontos irrelevantes. Este ponto é concordante com a imagem referente aos pontos de interesse encontrados, onde é possível observar-se que há uma maior concentração de pontos encontrados no objeto da direita.

É possível ver-se um ponto na zona positiva de ambas as coordenadas x e y, que se trata claramente de um ponto de *noise*, que conseguiu chegar até esta fase do processo, no entanto, é eliminado neste processo de *clustering*, o que é benéfico para que não interfira negativamente nos cálculos de distâncias aos objetos.

Os pontos pertencentes a cada cluster são então os utilizados para o processo de cálculo de distâncias aos respetivos objetos.

A partir das localizações dos pontos emparelhados, em pixels, para cada cluster são calculadas distâncias entre pontos da mesma imagem e efetuados os cálculos através de (13), onde x_1 é a distância percorrida entre imagens, p_1 e p_2 são distâncias entre dois pontos da primeira imagem e da segunda, respetivamente, e x_2 é a distância pretendida até ao objeto.

De maneira a obter melhores resultados, calculam-se vários valores de x_2, usando várias distâncias p_1 e p_2.

Na pequena amostra de pontos representada nas figuras seguintes, pode observar-se que são calculadas todas as distâncias entre pontos, possíveis.

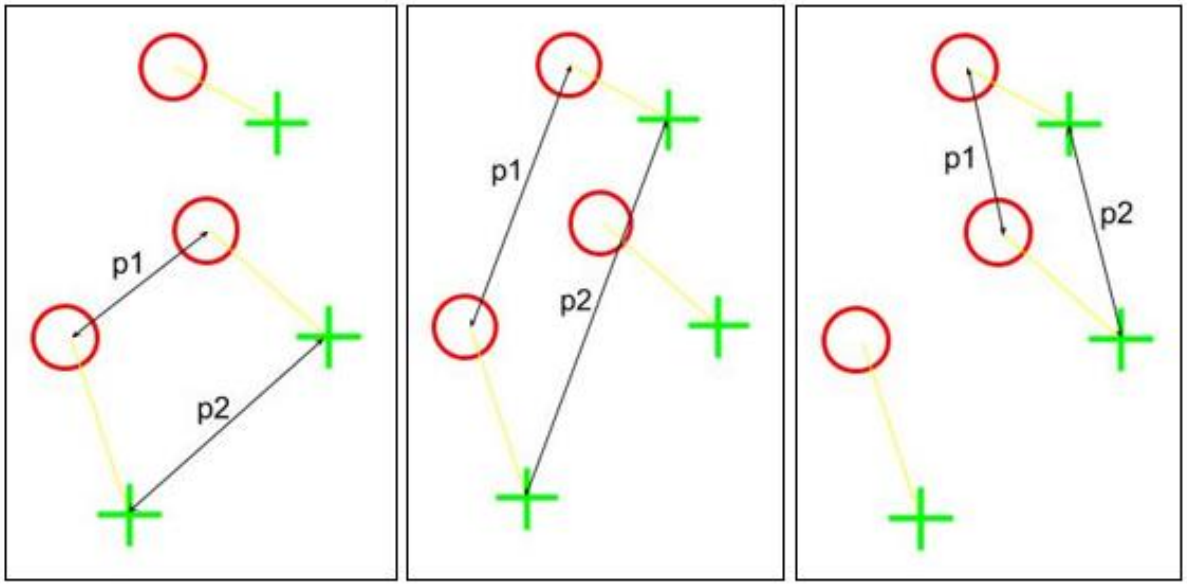

Figura 19 - Iterações para cálculo de distâncias

O processo começa por calcular todos os valores possíveis de p_1 e de p_2, partindo de um ponto e calculando as distâncias desse ponto a todos os outros pontos da mesma imagem. Na figura 19 está representado o método que leva ao cálculo de todos os valores de p_1 e p_2 que serão usados para obter valores de x_2.

Obter vários valores de p_1 e p_2 faz com que se obtenham igualmente vários valores de x_2. Os valores obtidos de x_2 são dispostos num vetor e dispostos num histograma, de modo a eliminar valores de x_2 que estejam desajustados.

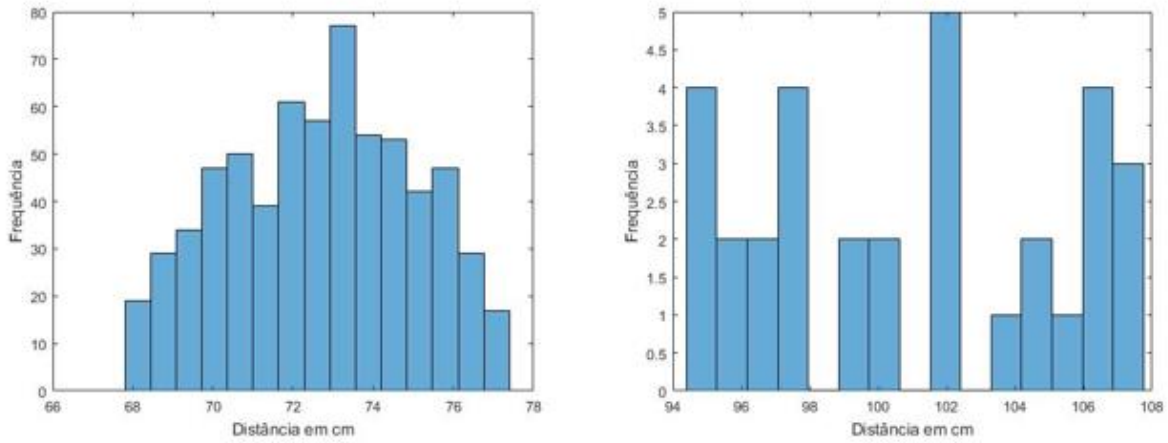

Figura 20 - Histogramas referentes às distâncias calculadas para os dois obstáculos

Os histogramas representados na figura 20 são referentes aos valores de x_2 obtidos nos dois clusters obtidos e já não apresentam os bins considerados excedentes e que iam alterar negativamente os resultados.

Os valores excluídos tratam-se de *bins* sem vizinhança de outros *bins* num espaço de 10 *bins*, ou seja, se, por exemplo, um dos valores de x_2 for 300 cm e não houver nenhum ponto num espaço de 10 *bins* adjacentes, então este ponto é eliminado e já não aparece no histograma.

Após esta exclusão de valores, procede-se ao cálculo da média dos valores de e x_2, sendo esse o valor final de distância calculada para a distância ao objeto identificado pelo cluster.

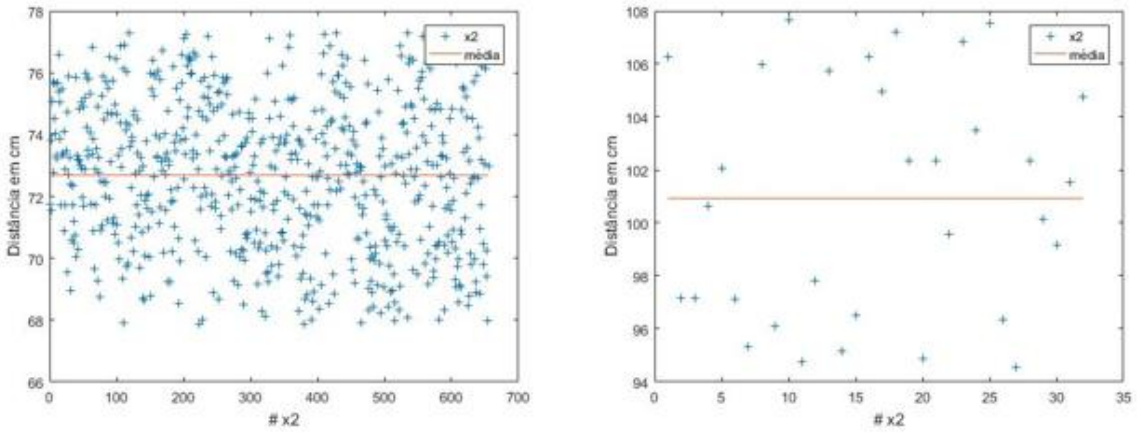

Figura 21 - Gráficos de dispersão e média calculada para os dois obstáculos

Os gráficos representados na figura 21 mostram a dispersão dos pontos x_2 calculados juntamente com a linha média dos valores. Mais uma vez, é de notar a menor quantidade de pontos encontrados no segundo cluster que foram usados para calcular a distância.

Usando as 5 imagens captadas, procedeu-se à aplicação do algoritmo e verificação de resultados. A tabela 1 apresenta os valores pretendidos e os obtidos, juntamente com os erros relativos e absolutos para cada caso.

Tabela 1

Distâncias da câmara	Valor Esperado (esquerda)	Valor Esperado (direita)	Valor Calculado (esquerda)	Valor Calculado (direita)	Erro Absoluto (esquerda)	Erro Absoluto (direita)	Erro Relativo (esquerda)	Erro Relativo (direita)
150cm-100cm	70	40	71,38	38,38	-1,38	1,62	0,0197	0,0405
150cm-110cm	80	50	81,57	48,62	-1,57	1,38	0,0196	0,0276
150cm-130cm	100	70	97,55	72,21	2,45	-2,21	0,0245	0,0316
150cm-140cm	110	80	84,79	84,79	25,21	-4,79	0,2292	0,0599
140cm-100cm	70	40	71,05	38,37	-1,05	1,63	0,0150	0,0408
140cm-110cm	80	50	80,65	49,09	-0,65	0,91	0,0081	0,0182
140cm-130cm	100	70	78,59	78,59	19,35	-8,59	0,1935	0,1227
130cm-100cm	70	40	69,68	37,12	0,32	2,88	0,0046	0,0720
130cm-110cm	80	50	78,28	46,82	1,72	3,18	0,0215	0,0636
110cm-100cm	70	40	72,35	37,04	-2,35	2,96	0,0336	0,0740

Nota: As unidades dos valores de todas as colunas da tabela anterior, exceto as duas últimas, estão em cm.

Analisando a tabela anterior, a primeira coisa que salta à vista são os valores calculados quando as imagens usadas foram as captadas nos pares de distâncias, 150cm-140cm e 140cm-130cm, dado que estes valores apresentam erros muito elevados comparando com a média e desvio padrão. É de notar ainda que os valores calculados para o objeto da esquerda e para o objeto da direita são os mesmos nestes dois casos referidos, o que significa que o método de *clustering* apenas considerou que estivesse um obstáculo no caminho. No cálculo dos restantes valores, está tudo conforme o esperado, visto que o módulo do erro absoluto máximo é 3.18 cm. Tendo em conta que podem ter sido cometidos erros humanos nas medições, este valor de erro é aceitável para o processo que se pretende implementar.

Na tabela 2 estão representados os valores médios dos erros obtidos no teste anterior.

Tabela 2

	Erro Absoluto (esquerda)	**Erro Absoluto (direita)**	**Erro Relativo (esquerda)**	**Erro Relativo (direita)**
Média	4,21	-0,10	0,0569	0,0551
Desvio Padrão	9,24	3,70	-	-

A discrepância maior na média e no desvio padrão dos erros absolutos do objeto da esquerda e do objeto da direita também se deve ao problema descrito no parágrafo anterior. De notar, os valores calculados, quando o *clustering* não identifica os objetos corretamente, estão mais próximos do objeto da direita, o que resulta num erro menor para os cálculos da distância para o objeto da direita e, por conseguinte, os erros maiores encontrados para os valores do objeto da esquerda vão resultar numa média e desvio padrão mais altos para os cálculos do objeto da esquerda.

De forma a analisar os casos em que o *clustering* não foi efetuado com sucesso, representa-se, de seguida na figura 22, com o emparelhamento dos pontos dos descritores encontrados nas imagens captadas a 150 cm e 140 cm e também o gráfico, figura 23, com a dispersão dos vetores criados por esses mesmos pontos.

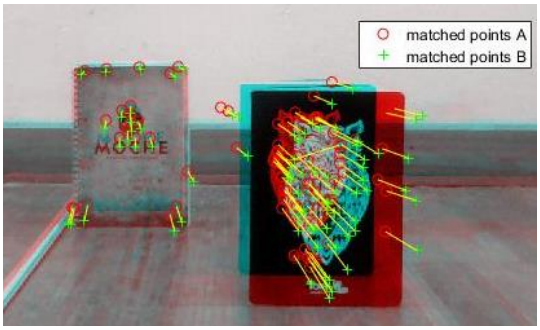

Figura 22 - Emparelhamento de pontos de interesse onde o clustering não identificou os obstáculos

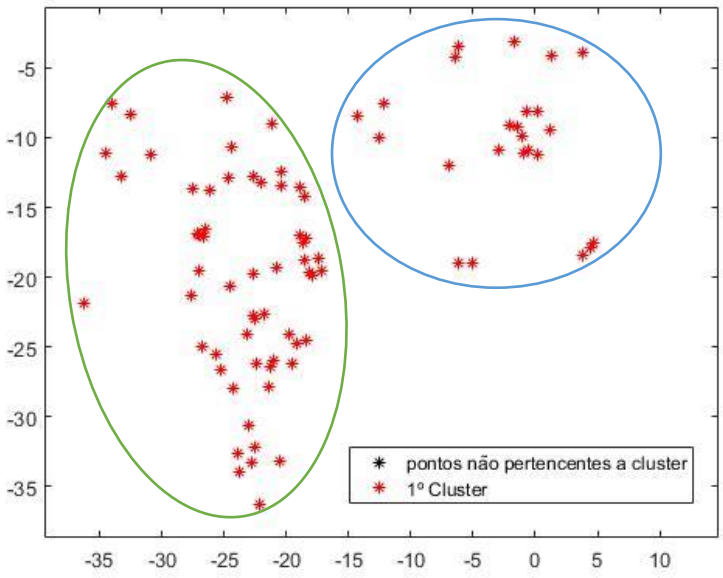

Figura 23 - *Clustering* dos vetores de pontos emparelhados na figura 15

Com os parâmetros utilizados para classificação de clusters, neste caso, a classificação é incorreta. Isto acontece, porque os parâmetros otimizados não são bons para este caso específico. No gráfico, da figura 23, estão desenhadas duas regiões distintas, que deveriam ter sido identificadas como clusters diferentes, mas como a condição de distância máxima entre pontos de um mesmo cluster (ε = 9 px) é sempre satisfeita, o algoritmo considera apenas um cluster. Este problema seria resolvido para este caso específico se se modificasse o valor de ε, no entanto isto criaria outros casos de classificação errada.

2.4.2 2º TESTE

Após testar a validade do processo de cálculo de distâncias no eixo dos *yy* numa trajetória simples em linha reta, era necessário, para introduzir as vertentes de cálculo de distâncias no eixo dos *xx,* um teste ao movimento de translação noutra direção que não em linha reta e orientação da câmara. Com este teste, a ideia era obter um mapeamento de obstáculos no espaço ao invés de apenas distâncias verticais entre a câmara e os obstáculos.

De forma análoga ao cálculo de distâncias para o eixo dos *yy,* o cálculo da posição dos obstáculos no eixo dos *xx* baseia-se nas deduções trigonométricas referidas no segundo capítulo, no entanto, neste caso é necessário o valor da distância focal, dado que as equações (3) e (4) não podem ser igualadas como foi feito para o cálculo das distâncias aos obstáculos. O valor que se pretende obter nessas equações é o valor de *a,* como tal, pode recorrer-se simplesmente às equações (1) e (2) para resolver o problema. Igualando as duas equações, obtemos:

$$\frac{p}{f} = \frac{a}{b} \tag{12}$$

Sendo *p* o número de pixels, *a*, o valor que se pretende obter, *f,* a distância focal e b, a distância entre a câmara e o objeto, é então apenas preciso saber o valor da distância focal *f* para ser possível obter *a.*

Usando a equação (12) com valores conhecidos de *a, b e p*, consegue deduzir-se a distância focal da câmara. Para isso, usou-se um objeto de tamanho conhecido colocado a uma distância também conhecida e foi captada uma imagem. Este processo foi repetido algumas vezes de modo a obter melhores resultados. Assim, foi possível obter-se o valor da distância focal. Os valores encontrados e usados no cálculo da mesma, estão na tabela seguinte.

Tabela 3

coordenadas de pixels				**distância em pixels**	**distância da câmara ao objeto (cm)**	**Distância real medida (cm)**	**distância focal**
x1	y1	x2	y2				
711	331	712	254	77,01	34,1	2	1313
715	335	714	262	73,01	36	2	1314
719	341	716	272	69,07	37,9	2	1309
709	345	706	278	67,07	39,4	2	1321
710	346	704	282	64,28	40,6	2	1305

Usou-se a partir daqui o valor médio das distâncias focais calculadas, f=1312. Com uma mudança de direção, a orientação da câmara deixa de ser a mesma, pelo que, se se pretende fazer um mapeamento de obstáculos, é preciso ter isso em conta e converter as distâncias calculadas após a alteração de direção, para o referencial fixo definido inicialmente. Este problema é resolvido através da multiplicação dos valores obtidos para as distâncias (em x e y) por uma matriz de rotação *R.*

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{13}$$

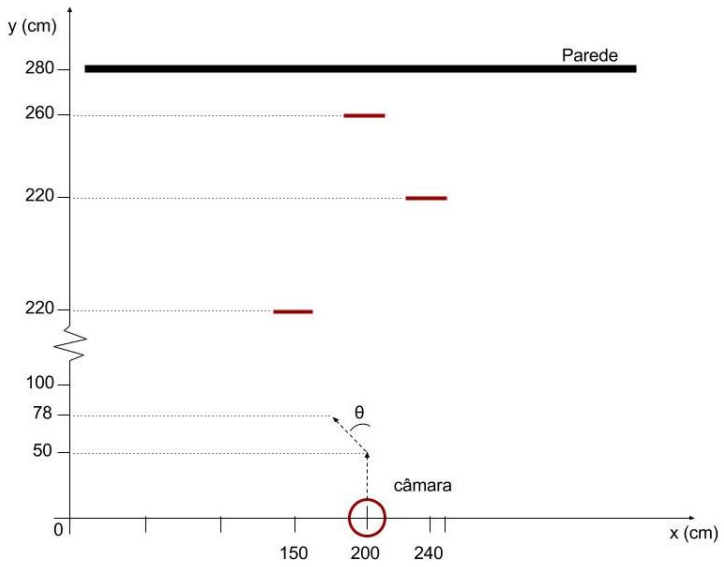

Figura 24 - Planta da disposição dos obstáculos e da posição da câmara para o 2º teste

Na figura 24 está representa a disposição do ambiente no segundo teste. Foram novamente utilizados os dois cadernos de tamanho A5, que já tinham sido usados no teste anterior e juntou-se mais um obstáculo (caixa preta, objeto do centro). Como podemos verificar no esquema, os três objetos foram dispostos a distâncias diferentes, e desta vez, estão definidas também as posições laterais dos objetos. Como foi referido, também a mudança de orientação da câmara seria testada, pelo que no esquema, pode perceber-se a alteração de orientação da câmara, no seu trajeto.

Figura 25 - Imagem captada da disposição dos obstáculos na posição frontal

Figura 26 - Imagem captada da disposição dos obstáculos na posição rodada

As figuras 25 e 26 mostram a disposição referida no esquema da figura 24. A imagem na figura 25 representa a primeira imagem captada, na posição inicial e com a orientação inicial da câmara, enquanto que a imagem da figura 26 foi obtida após translação e mudança de orientação da câmara.

Para este teste foram usadas sete imagens, três imagens obtidas em linha reta (y=20, y=40 e y=50) e quatro imagens após uma rotação θ = 19.8º (deslocamentos nessa direção de 10 cm entre imagens).

Introduzindo as imagens referidas no algoritmo, foram obtidos os seguintes resultados:

Tabela 4

intervalo de posição (x,y) em cm	**objeto esquerda**				**Objeto central**				**objeto direita**			
	esperado		calculado		esperado		calculado		esperado		calculado	
	x	y	x	y	x	y	x	y	x	y	x	y
(200,20) - (200,40)	150	160	146,8	162,3	200	260	203,4	240,9	240	220	240,9	208,9
(200,40) - (200,50)			150,7	153,4			202,2	257,5			241,9	216,6
(197,59) - (193,69)			151,6	162,1			198,1	223,6			-	-
(193,69) - (190,78)			153,1	157,8			196,5	225,9			-	-

Através dos dados da tabela 4, consegue perceber-se que, ao contrário do que aconteceu no primeiro teste de validação, não houve erros no método de *clustering*, dado que não existem valores calculados iguais para nenhum dos casos, em mais do que um dos objetos. Deve notar-se ainda que o objeto central e mais afastado da câmara apresenta erros maiores. Isto pode dever-se a várias razões, tais como:

- Distância elevada até à câmara, o que faz com que uma pequena diferença de pixels possa significar um erro no cálculo elevado.
- Poucos pontos pertencentes ao cluster.
- Pontos de cluster muito próximos.
- Pequena variação da posição dos pixels dada a posição central do objeto.

Repare-se ainda que o objeto da direita não tem valores de distâncias calculadas nas duas últimas linhas da tabela, pois após a rotação, este deixa de aparecer na imagem.

Tabela 5

	Objeto esquerda		**Objeto central**		**Objeto direita**	
	x	y	x	y	x	y
Erro absoluto médio (cm)	2,15	3,3	2,75	23,025	1,4	7,25
Erro relativo médio	0,014	0,021	0,014	0,089	0,006	0,033

Na tabela 5, é possível corroborar o que foi dito anteriormente, vendo que a distância y ao objeto central apresenta um erro absoluto médio muito mais elevado do que os dois outros objetos, perfazendo um erro médio relativo de 8.9%, o que, comparando com os erros para qualquer outra distância calculada, onde o máximo é 3.3% é muito elevado.

Visualmente, os resultados representados da tabela 4, ficam:

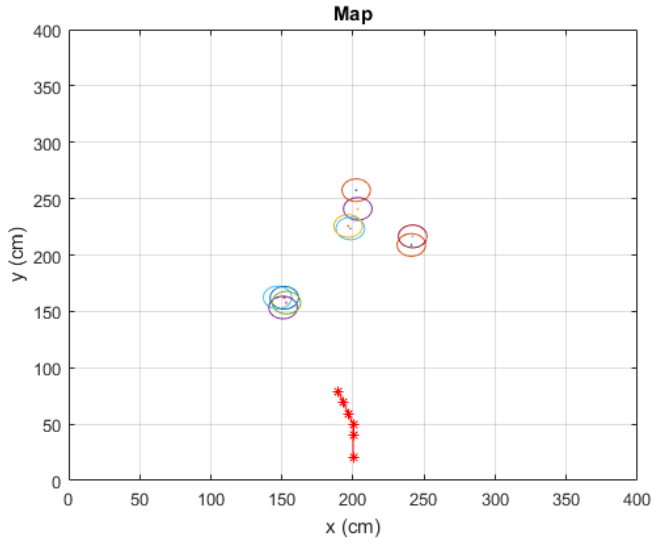

Figura 27 - Mapeamento de obstáculos

De forma a tentar perceber melhor quais os fatores que podem ter influenciado as diferenças de cálculos da posição do objeto central, nas figuras seguintes estão representados, o gráfico do método de *clustering* aplicado numa imagem do emparelhamento de pontos de interesse, num dos intervalos de deslocamento, e a respetiva imagem, com o emparelhamento representado.

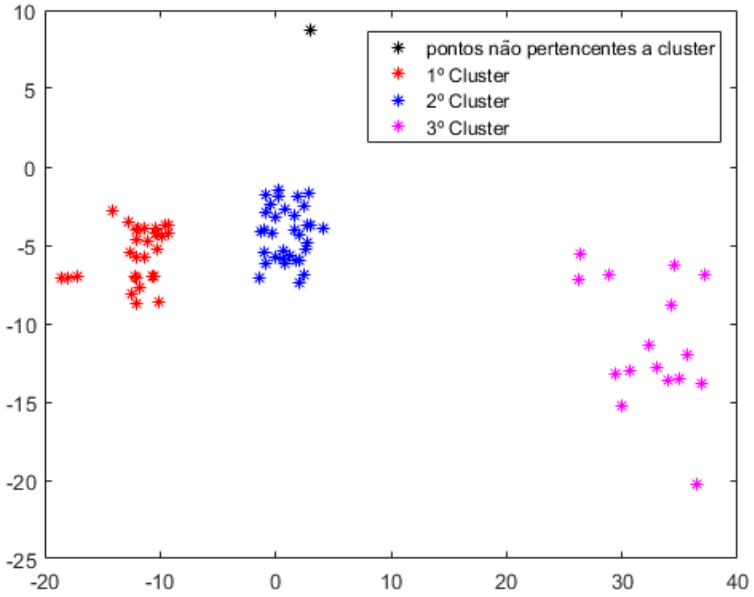

Figura 28 - Clustering no intervalo de distâncias [20,40] em linha reta

Nota: O primeiro cluster corresponde ao objeto da direita, o segundo ao objeto central e o terceiro ao objeto da esquerda.

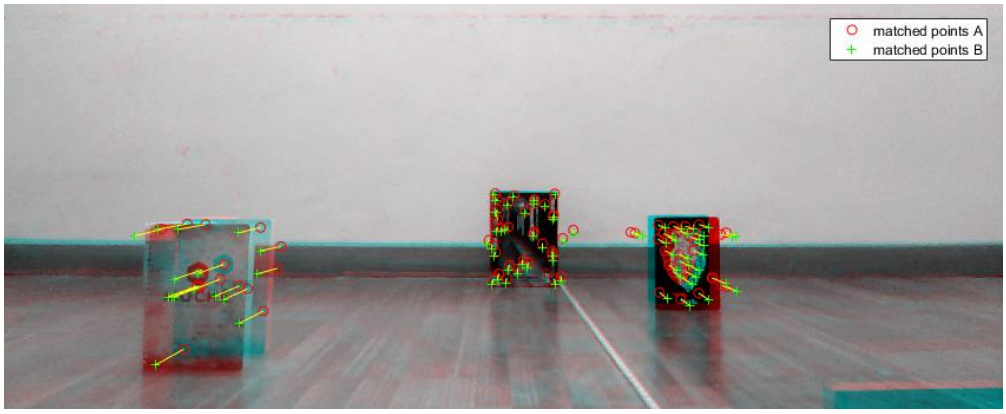

Figura 29 - Emparelhamento de pontos de interesse no intervalo de distâncias [20,40] em linha reta

Olhando de novo para a tabela 4, e calculando o erro relativo apenas para o caso específico do intervalo de distâncias [20,40], obtemos a tabela seguinte.

Tabela 6

	Objeto esquerda	Objeto central	Objeto direita
erro relativo	0,014	0,073	0,050

Dos dados da tabela 6 percebe-se que apesar do erro no cálculo da distância ao objeto central ser o mais elevado, o objeto da direita também teve um erro elevado, pelo que pode tentar encontrar-se semelhanças entre os dois casos de erros mais grosseiros comparando com o caso do objeto da esquerda, que apresentou um erro baixo.

Apoiando a análise do gráfico da figura 28 na visualização da imagem da figura 29 e avaliando os pontos referidos na página anterior, relativamente às possíveis causas do cálculo deficiente da distância ao objeto central, pode eliminar-se a hipótese de haver poucos pontos nos clusters, visto que, segundo o gráfico, os objetos central e da direita até apresentam um maior número de pontos em cada cluster do que o objeto da esquerda que obteve melhores resultados. Pode também eliminar-se a hipótese que refere que a posição central do objeto prejudica o método, uma vez que o objeto da direita também apresenta um erro elevado e não está centralizado. Deste modo, as restantes duas hipóteses parecem complementar-se, ou seja, o facto de os objetos estarem a uma distância elevada da câmara leva a que os pontos dos clusters estejam muito próximos (pouca diferença de imagem para imagem na posição dos pontos de interesse emparelhados), provocando que as distâncias entre pixels sejam pequenas e como tal, que pequenos erros na posição da câmara aquando da captação das imagens, leve a erros de maior magnitude.

Capítulo 3 – Planeamento de rota e algoritmo

Após o mapeamento dos obstáculos, é necessário definir um ponto objetivo e uma trajetória que faça a ligação entre o ponto onde se encontra o RC e esse mesmo ponto objetivo, evitando os obstáculos. Este processo designa-se planeamento de rota ou, em inglês, *path planing.* Existem vários tipos de algoritmos desenvolvidos ao longo dos anos que resolvem este problema. Este capítulo está dividido em duas subdivisões principais. Uma primeira onde se aborda o tema de planeamento de rota e que começa com uma introdução teórica dos seus princípios, e uma explicação do método escolhido para o trabalho, seguida do algoritmo desenvolvido. Nesta fase apresenta-se ainda um teste de validação do método proposto. E uma segunda subdivisão onde se introduz a maneira como o sistema global funciona. Nesta segunda subdivisão, numa primeira fase estão explicados os movimentos que o RC pode efetuar, assim como as posições e orientações que pode tomar. De seguida estão explicadas as duas divisões do algoritmo, uma primeira que se executa apenas uma vez no início do processo, denominada, inicialização do sistema, e uma segunda fase em que um ciclo fechado de processos é executado. Depois, introduz-se a câmara utilizada no sistema, algumas limitações que a comunicação wifi, entre os subsistemas, introduz no sistema, e a influência que estas limitações têm na implementação final, incluindo o modo como se implementou o algoritmo no sistema de modo a contornar estas mesmas limitações. Por fim, apresenta-se um esquema de ações interligadas que descrevem o globalmente o algoritmo.

3.1 PLANEAMENTO DE ROTA

3.1.1 FORMULAÇÃO DO PROBLEMA

A formulação do problema é feita recorrendo a modelos de espaço de estados. O princípio básico é que cada situação específica seja definida por um estado *x,* e que o conjunto de todos os estados possíveis seja denominado *X*. Cada estado *x* pode sofrer alterações quando aplicada uma ação *u,* produzindo assim um novo estado *x'.* Esta transformação pode ser definida por uma função *f,* chamada função de transição de estado. Assim, pode definir-se a equação de transição de estado discreta da seguinte forma:

$$x' = f(x, u) \tag{14}$$

À semelhança do que foi definido anteriormente para o conjunto de estados X, *U(x)* define o espaço de ações para cada estado *x,* que representa o conjunto de todas as ações possíveis. É de notar que para *x, x' ∈ X, U(x) e U(x')* não são necessariamente diferentes. A mesma ação, pode ser aplicada em diferentes estados. Desta forma, é conveniente definir o espaço *U* correspondente a todas as ações possíveis sobre todos os estados.

$$\mathrm{U} = \bigcup_{x \in \mathrm{X}} U(x) \tag{15}$$

Parte da formulação do problema envolve definir um conjunto de estados objetivo $X_g \in X$. O propósito de um algoritmo de planeamento é encontrar uma sequência finita de ações que transforme o estado inicial x_1 num estado final x_g.

Sumarizando, a formulação de um problema de planeamento discreto,

- Definir um espaço de estados *X,* que é um conjunto de estados finitos.
- Para cada estado $x \in X$, definir um espaço finito de ações *U(x).*
- Definir uma função de transição de estado *f* que produza um estado $x' \in X$ para cada estado $x \in X$ e $u \in U(x)$. A equação de transição de estado é dada por x' = f(x,u).
- Definir um estado inicial $x_1 \in X$.
- Definir um conjunto objetivo $X_g \in X$.

3.1.2 ESPAÇO DE CONFIGURAÇÃO (CONFIGURATION SPACE)

Como foi referido anteriormente, o objetivo do planeamento de rota é criar um caminho que faça a ligação entre uma configuração inicial x_1 e uma configuração final X_g, evitando obstáculos.

O RC e os obstáculos são definidos no espaço e o caminho é representado no chamado espaço de configuração. O espaço de configuração engloba todas as configurações possíveis que o RC pode tomar no espaço. No caso apresentado neste trabalho, considera-se que o espaço é 2D (O RC move-se sempre no plano do chão, z=0), no entanto a configuração do RC tem que ser definida por três parâmetros, dado que se trata de um corpo rígido com orientação relevante. Assim, cada configuração é definida por dois valores de posição (x,y) e um valor de orientação (θ).

Dentro do conjunto de configurações C, dois subgrupos complementares relevantes podem ser definidos, C_{free} e C_{obs}.

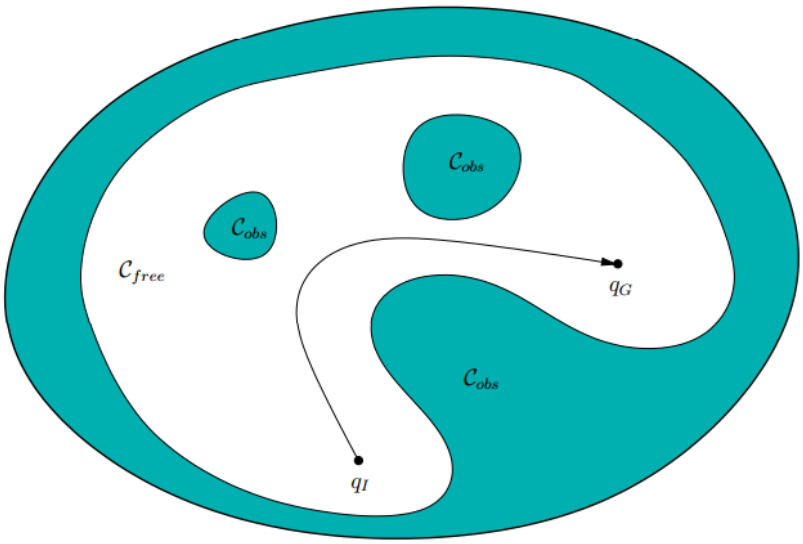

Figura 30 - Exemplo de espaço de configuração

C_{free} é o chamado espaço livre (free space) e é composto pelas configurações que evitam os obstáculos, enquanto que C_{obs} é exatamente o oposto, sendo o conjunto das configurações que coincidem com os obstáculos. Na figura 30 pode ver-se um exemplo dos subconjuntos referidos e de um caminho que faz a ligação entre um ponto inicial e um ponto objetivo evitando os obstáculos.

3.1.3 ALGORITMO

Como foi referido no início deste capítulo, existem vários algoritmos para tentar resolver o problema de planeamento de rota consoante o que se pretende. De entre os muitos tipos de algoritmos presentes, destacam-se: *Grid-Based Search* [11] *e Sampling-Based algorithms* [12].

Sampling-Based algorithms são algoritmos muito usados em robótica, porque funcionam bem em espaços de configuração com várias dimensões. O modo de funcionamento deste tipo de algoritmos evita a formulação explicita de um espaço C_{obs}, em vez disso, experimenta várias configurações em C que são guardadas como marcos. Depois, é criado um caminho entre duas das configurações anteriormente experimentadas, por exemplo A e B, e caso exista uma linha que una as duas configurações, um caminho é criado dentro do espaço C_{free}. Usa-se deteção de colisões para verificar se o caminho pertence ou não ao espaço C_{free}. Caso não seja encontrado um caminho entre A e B pertencente a C_{free}, pode não significar que não existe um caminho, pode apenas significar que não foram amostradas configurações suficientes para encontrar um caminho que chegue da primeira configuração até à configuração objetivo.

Quanto mais elevado for o número de configurações amostradas, maior é a probabilidade de que um caminho entre duas configurações pertença ao espaço C_{free}, no entanto, este o aumento do número de configurações amostradas também aumenta o tempo de computação necessária para a aplicação do algoritmo.

No caso presente neste trabalho, o espaço de configuração é apenas bi-dimensional, portanto não é necessário recorrer a um *Sampling-Based algorithm,* que é mais eficaz para configurações de maior dimensão, pelo que se recorreu a um algoritmo do tipo *Grid-Based Search*, que está explicado em maior detalhe de seguida, e que é menos pesado computacionalmente e ainda assim permite atingir bons resultados para o caso a ser estudado.

3.1.4 GRID-BASED SEARCH

O princípio por detrás deste tipo de algoritmos consiste em definir uma malha de posições por cima do espaço em que se pretende navegar, e cada posição da malha (i,j) corresponde a uma configuração possível. De entre cada posição da malha, o RC pode mover-se para qualquer ponto adjacente à posição em que se encontra.

A definição do tamanho da malha resulta de um *trade-off* entre tempo de computação e precisão de movimentos que se pretende, ou seja, quanto maior for a espaço em cada posição da malha, mais rápido será a computação do caminho, no entanto, uma malha com divisões muito grandes pode não admitir caminhos entre obstáculos por estes estarem em posições adjacentes da malha, quando na realidade, existe espaço suficiente para passar entre eles. Este problema pode resultar em caminhos que se afastam do caminho ótimo ou na não existência de solução.

O processo para definir o tamanho da malha é iterativo, tentando resolver o problema consumindo o menor tempo de computação possível.

Sendo o processo de obtenção de caminhos, um processo discreto, dado que o RC se movimenta de uma posição da malha para outra, cada caminho é composto por uma sequência de posições. Este aspeto faz com que algoritmos como o *A** [18] [22] (derivado do conhecido *Dijkstra's algorithm* [23]) sejam adequados.

3.1.5 A* ALGORITHM

O algoritmo *A** segue um modelo de procura de caminhos denominado *forward search* (busca para a frente), que segue a seguinte ordem de procedimentos:

```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```

O pseudo-código anterior trata-se apenas de um *template* para os algoritmos que se regem pelo método de *forward search*.

A cada momento da busca de caminhos, existem três tipos de estados:

Não visitados – Estados que, tal como o nome indica, ainda não foram testados pelo algoritmo. Todos os pontos são pontos não visitados inicialmente, à exceção do ponto inicial x_i.

Fechados – Estados já visitados e que para os quais, já todos os possíveis estados consequentes foram testados. Um estado consequente x é um estado x' para o qual existe uma ação u ∈ U(x) tal que x' = f(x,u).

Abertos – Estados já visitados, mas que ainda podem ter estados consequentes não visitados. Inicialmente o único estado aberto é o estado inicial x_i.

Este algoritmo é uma extensão do *Dijkstra's algortithm* que tenta reduzir o número de estados explorados através da introdução de uma estimação heurística do custo para atingir o estado objetivo. A sua formulação é feita através da criação de árvores de caminhos possíveis com um custo associado, partindo de um ponto específico e continuando cada um desses caminhos até que se chegue ao ponto objetivo desejado.

Considere-se $C(x)$ o custo de ir de x_i até x e ainda que $G(x)$ denota o custo associado a ir de x até um estado x_g. $C^*(x)$ devia ser possível de calcular através de programação dinâmica, no entanto, não é possível saber antecipadamente o valor de $G^*(x)$. O que acontece é que em muitas aplicações, é possível estimar, por baixo, de forma razoável, este custo. Um exemplo de uma estimativa deste tipo seria considerar a distância em linha reta entre um ponto inicial e um ponto final, visto que esta estimativa de custo ignora possíveis obstáculos. Quando são introduzidos obstáculos, este custo só pode aumentar. O objetivo será ter uma estimativa o mais próxima possível do custo ótimo, sem que ultrapasse esse mesmo custo. Seja $\hat{G}^*(x)$ esta estimativa. Neste algoritmo, usa-se a soma $C^*(x) + \hat{G}^*(x)$, implicando que a ordem de procura de caminhos seja feita a partir do custo estimado $\hat{G}^*(x)$.

Nota: se $\hat{G}^*(x)$ for efetivamente uma estimativa abaixo do valor real do custo ótimo para todo o $x \in X$, o algoritmo garante que o caminho ótimo é encontrado.

À medida que se aproxima o valor de $\hat{G}^*(x)$, menor tempo vai ser despendido até se encontrar a solução. Nem sempre é fácil encontrar uma heurística que permita ter um balanço entre a procura de caminhos e eficiência.

3.1.6 MAPEAMENTO DOS OBSTÁCULOS

A informação obtida pela câmara e transformada pelo processamento de imagem (capítulo 2) é usada para obter posições no espaço dos obstáculos, como está representado na secção 2.4.2. Para aplicação do algoritmo é necessário definir as dimensões da malha a sobrepor no mapa. Dado que a vertente do projeto ligada a visão computacional apresenta por si só um peso elevado

computacionalmente, resolveu-se aplicar uma malha mais grosseira, de modo a aliviar o processo computacional e com isso diminuir o tempo de processamento. A malha criada tem a uma dimensão de 10x10 e cada elemento da malha tem dimensão 40x40cm.

Sendo cada elemento da malha definido por dois números inteiros (i,j), as posições dos obstáculos determinadas anteriormente teriam que ser adaptadas para ocuparem posições certas na malha. Desta forma, reduziu-se a escala e arredondou-se ao número mais próximo cada valor de x e y correspondentes às posições dos obstáculos encontrados. Por exemplo, o obstáculo cujas coordenadas calculadas foram (203.4,240.9), depois de aplicadas as transformações de mudança de escala e arredondamento, as coordenadas do mesmo, na malha, são (5,6).

Usando como exemplo os dados da secção 2.4.2, consegue comparar-se visualmente a distribuição dos obstáculos encontrados, antes e depois da transformação para a malha.

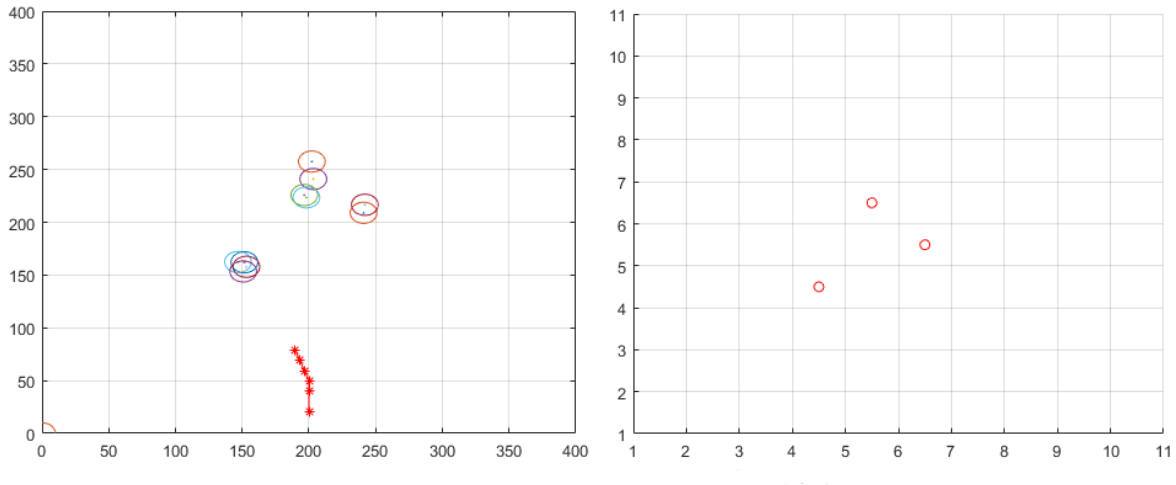

Figura 31 - Posições calculadas de obstáculos (à esquerda); normalização para a malha (à direita)

3.1.7 DEFINIÇÃO DOS PONTOS INICIAIS E FINAIS E APLICAÇÃO

Depois de definidos os obstáculos no mapa, resta definir a posição inicial e final. A posição inicial é definida no ponto (200,200) e introduzida através de uma redução à escala e arredondamento, à semelhança do que acontece no caso dos obstáculos. À medida que o RC se vai movendo, a posição é atualizada e transformada. A posição final é um input do utilizador, pelo que é introduzida inicialmente, em coordenadas e posteriormente transformada também para uma posição na malha. No caso deste trabalho, a posição final será sempre na posição (5,10) da malha, e caso essa posição da malha esteja ocupada com um obstáculo, a posição final altera-se para (5,9) e assim sucessivamente.

O algoritmo aplicado foi desenvolvido a partir de um *template* disponível na página da *mathworks* [24], onde se adaptou às necessidades inerentes ao trabalho. O algoritmo recebe como inputs as posições dos obstáculos, posição inicial e posição objetivo do RC.

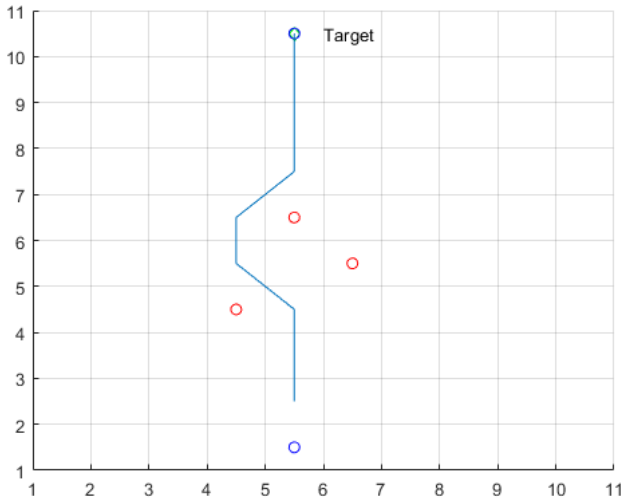

Figura 32 - Aplicação do algoritmo A* ao problema

O caminho calculado é traduzido em comandos que serão interpretados pelo sistema e levam à progressão do mesmo no mapa. Enquanto o mapa não é atualizado, a posição do sistema vai se movendo sobre a malha do mapa, e aquando da atualização do mapa, a posição atual do sistema passa a ser a posição inicial (5,6) novamente.

3.2 ALGORITMO GLOBAL

3.2.1 MOVIMENTAÇÃO NO MAPA

O algoritmo desenvolvido para o movimento autónomo do RC, evitando colisões, teve como base a ideia de ter um sistema que pudesse movimentar-se "para sempre" sem serem necessários inputs externos, criando assim um ciclo fechado infinito. Para isto, o mapa de obstáculos teria que ser inicializado e constantemente atualizado, reajustando com isso o planeamento de rota também.

De acordo com o que foi formulado no início do capítulo, o sistema move-se sobre uma malha, pelo que, encontrando-se em qualquer posição do mapa o movimento seguinte será para outra posição nessa mesma malha que esteja na linha superior à linha de posição em que se encontra o sistema no momento. O RC pode ter orientações na malha que variam no intervalo $\theta = [-45,45]^o$, o que influencia diretamente as possibilidades de movimentação descritas de seguida.

Nota: os ângulos de orientações são medidos em relação ao eixo dos yy.

Caso a orientação do sistema seja $\theta = 0^o$, existem três possíveis posições para o sistema após o comando seguinte, movendo-se então para a linha imediatamente superior e, para a mesma coluna

(movimento em frente), para a coluna imediatamente à esquerda (movimento na diagonal para a esquerda) ou para a coluna imediatamente à direita (movimento diagonal à direita).

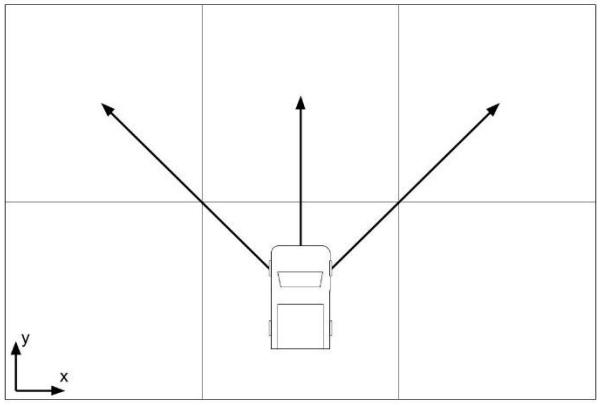

Figura 33 – Movimentos possíveis do sistema quando θ = 0º

Os casos em que a orientação do sistema é θ = -45º (imediatamente após virar à esquerda) ou θ = 45º (imediatamente após virar à direita) deixam apenas 2 possibilidades de movimento de seguida: ou manter o movimento na orientação em que o sistema se encontra, ou virar 45º na direção contrária à que se tomou antes, voltando a ficar θ = 0º.

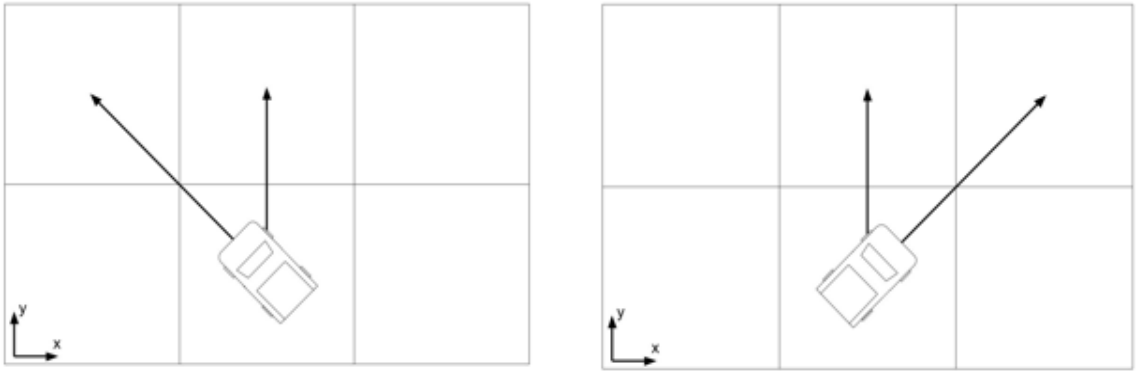

Figura 34 – Movimentos possíveis quando: θ = -45º (esquerda); θ = 45º (direita)

3.2.2 INICIALIZAÇÃO DO ALGORITMO

O algoritmo tem duas divisões principais. A inicialização do algoritmo, que só é executada, tal como o nome indica, quando o sistema é iniciado, e uma segunda fase em que o sistema entra num ciclo fechado de movimento e atualização do mapa.

Como está explicado no segundo capítulo, o método utilizado para localizar obstáculos envolve a captação de duas imagens em posições diferentes após movimento linear da câmara, como tal, inicialmente não havendo ainda mapa definido, não existe também rota definida. Assim, a inicialização do algoritmo trata-se da primeira criação do mapa, que consiste na captação de uma imagem,

movimento linear do sistema até uma distância previamente introduzida, e nova captação de imagem. Esta série de 3 processos só são executados uma vez, ao contrário dos restantes processos, que são executados em ciclo até o sistema ser quebrado pelo utilizador.

3.2.3 CICLO

Depois da inicialização do algoritmo, calculam-se as posições dos obstáculos e aplica-se o método de planeamento de rota desenvolvido. A rota é definida por um conjunto de comandos que guiam o RC entre as posições da malha do mapa.

O primeiro comando da rota é executado e é avaliado o comando seguinte. Neste ponto, o sistema tem duas opções. Caso o comando seguinte seja na mesma direção do comando executado, pode ser feita atualização do mapa, então é captada uma nova imagem e em seguida, o comando avaliado é executado, fazendo mover o sistema linearmente. Neste momento mais nenhum dos comandos da rota anterior será executado. Depois do movimento é captada nova imagem e o sistema volta ao processo de mapeamento de obstáculos, reiniciando o ciclo.

A outra opção, quando o comando a seguir ao comando atual é avaliado, é esse comando ter uma direção diferente da direção atual, impossibilitando assim a captação de duas imagens seguidas linearmente. O sistema continua então na rota previamente calculada, iterando o número do comando a executar e voltando ao passo de execução de comandos.

Deste modo, depois de iniciado o sistema, se não houver ordem do utilizador, o sistema continua a mover-se e a atualizar o mapa até que se acabem as baterias.

3.3 HARDWARE E COMUNICAÇÃO

3.3.1 PI CAMERA

A câmara utilizada neste trabalho foi a *Pi NoIR Camera V2*, que se trata da segunda versão de câmaras específicas para RPi, lançada em Abril de 2016. Esta câmara tem um sensor Sony IMX219 de 8-megapixels, promovendo assim um melhoramento em relação ao modelo anterior que tinha um sensor OmniVision OV5647 de 5-megapixels, tendo melhorado as capacidades ao nível de qualidade de imagem, fidelidade de cor e performance em condições de iluminação fraca. Permite, para além de captação de fotografias isoladas, gravação de vídeo em 180p30fps, 720p60fps e VGA90 e conecta-se através de uma fita de 15 cm à porta CSI na RPi.

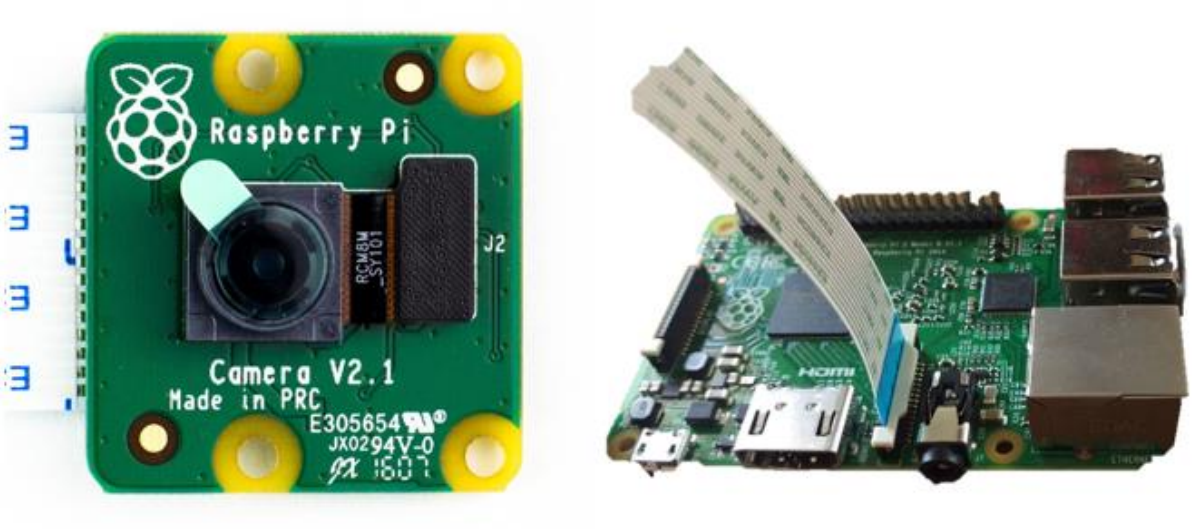

Figura 35 – *Pi Camera* (esquerda); Ligação da câmara à Rpi

3.3.2 LIMITAÇÕES

Todo o sistema é controlado remotamente através de um computador. O algoritmo foi implementado em *Matlab* e toda a informação que o RC recebe e transmite (através da câmara) passa pelo computador. A comunicação entre os vários subsistemas é feita através de *wi-fi*, e ainda que seja um sistema fechado de transmissão direta para o computador, o que melhora a velocidade de transmissão de informação, existe atraso no envio e receção de dados.

Como está explicado no capítulo 2, o sistema de reconhecimento e estimação da posição de obstáculos está dependente do conhecimento da distância percorrida entre a captação da primeira e segunda imagem. Assim, é importante que a câmara capte imagens nas posições esperadas pelo algoritmo para que o mapeamento seja feito corretamente.

Como se verá no capítulo seguinte, devido à comunicação, o RC tem respostas diferentes a um mesmo comando, verificando-se que existe algum erro de posição. Só por si, estes erros não influenciam o funcionamento do sistema, mas na captação de imagens também se verifica que a câmara responde ao comando enviado pelo computador em tempos diferentes de cada vez que se repete o comando e, associando os erros de posição com o erro de *timing* de captação de imagens, leva a que os cálculos de mapeamento de obstáculos não sejam adequados.

De maneira a resolver o problema e conseguir implementar e validar o método, procedeu-se de forma a eliminar o erro proveniente do *timing* de captação de imagens pela câmara. Isto foi conseguido através do movimento do RC seguido de paragem e captação de imagem aquando da paragem. Assim as imagens obtidas comtemplam apenas o erro associado diretamente ao movimento do RC e o tempo exato em que as imagens são captadas já não introduzem erro, pelo que os cálculos de posições e mapeamento serão melhores.

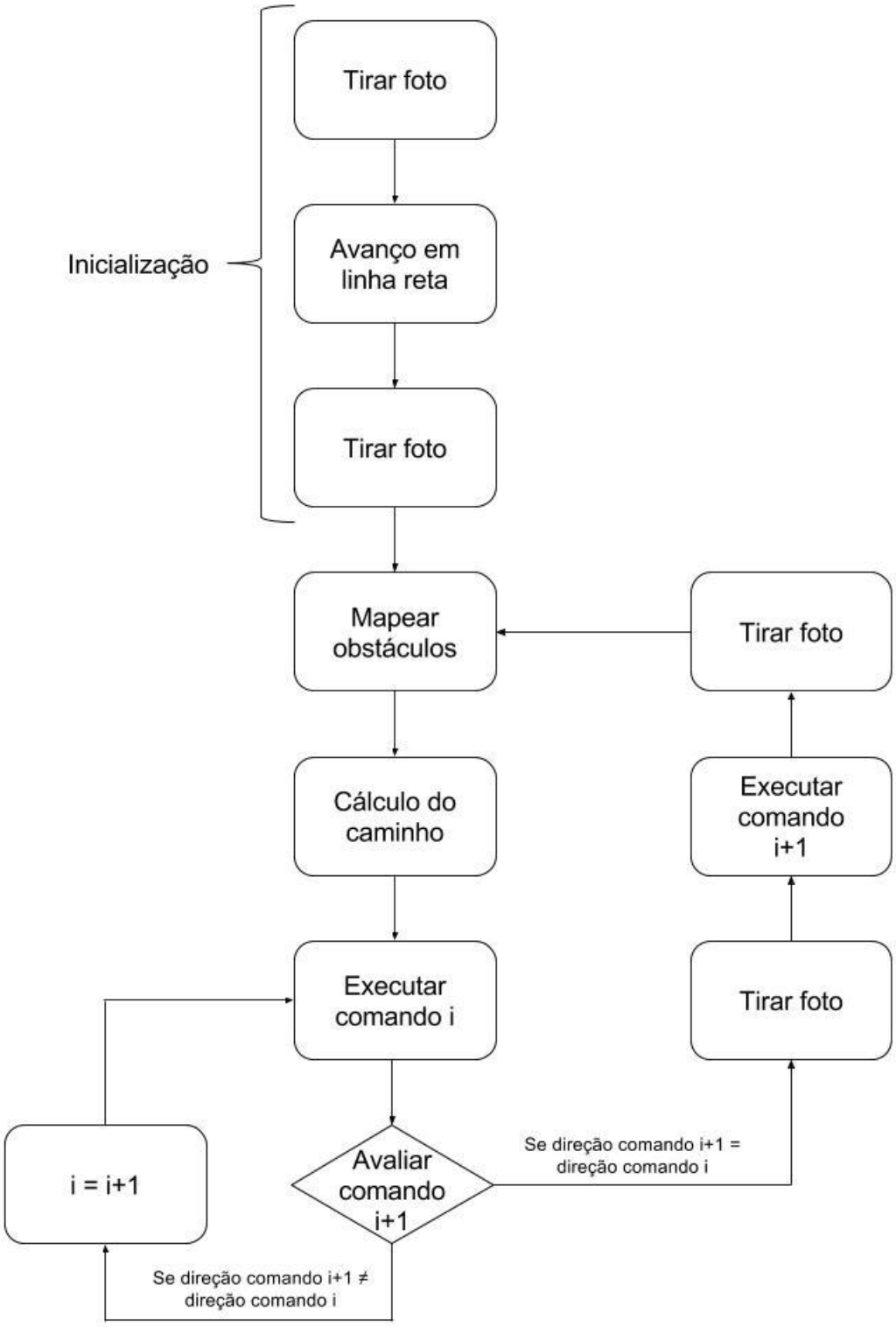

Figura 36 - Esquema global do algoritmo

Capítulo 4 – Identificação do Sistema

De modo a formular um modelo que permitisse implementar o algoritmo pretendido foi necessário executar testes de identificação de sistemas. Com isto, pretendia-se obter um modelo que relacionasse os sinais de potência e direção enviados para o RC e as respostas obtidas pelo mesmo. Também se pretendia avaliar os dados recebidos pela IMU acoplada de maneira a perceber se seria possível extrair dados que permitissem deduzir distâncias percorridas, pois nesse caso, poder-se-ia fazer realimentação no sistema dessas distâncias calculadas e ajustar a posição do carrinho. Caso contrário, seria apenas possível confiar no modelo encontrado e aplicar o algoritmo em anel aberto.

Figura 37 -Laboratórios da Área Científica de Controlo, Automação e Informática Industrial (ACCAII) – Pavilhão de Mecânica III - IST

Este capítulo começa por introduzir os componentes de hardware utilizados para a realização dos testes e em seguida apresenta os modelos deduzidos e uma avaliação dos resultados que permitiram chegar a esses mesmos modelos.

Nota: Todos os dados obtidos foram tirados com as baterias do RC totalmente carregadas. A placa MD25 indica a tensão das baterias. O sistema deve fazer a leitura deste valor ao longo do processo de forma a que os resultados esperados não sejam afetados por diminuição de potência devida à descarga da bateria.

4.1 HARDWARE E SETUP

O modelo RC usado neste projeto foi o LaTrax SST [25]. Trata-se de um veículo à escala de 1/18 baseado num jipe/*American Truck* e que foi desenvolvido e comercializado pela empresa Traxxas. O veículo apresenta um motor elétrico alimentado por baterias recarregáveis (6-cell 2/3A NiMH) que atinge uma velocidade máxima de 12 metros por segundo. Este modelo apresenta uma suspensão com mola e amortecedor independentes para cada roda e a direção é controlada por um servo.

Figura 38 - RC

O *software* desenvolvido [3], nomeadamente a interface que permite a interação entre *Matlab* e a *Raspberry Pi 3 model B* acoplada ao RC foram usados neste capítulo juntamente com as câmaras da arena do laboratório de robótica (figura 37) e a IMU instalada no RC em conjunto com a RPi. No software, os impulsos de potência e direção direcionados para o RC foram normalizados. Assim, o software aceita valores entre -1 e 1 para potência e direção, sendo que os valores negativos para a potência apenas travam, não produzindo movimento para trás. No caso da direção, valores negativos correspondem a virar à esquerda e valores positivos, a virar à direita. Os valores apresentados para potência e direção, neste capítulo, seguem esta norma de valores normalizados igualmente.

Figura 39 – RC com câmara acoplada

A carcaça do RC foi pintada de preto e aplicado um marcador vermelho na forma de uma semiesfera para ser identificado pelas câmaras da arena e transmitir as coordenadas x e y do RC.

4.2 ENSAIOS REALIZADOS

4.2.1 DADOS RECEBIDOS PELA IMU (GY-80)

Através do acelerómetro incluído na IMU acoplada no RC, pretendia-se obter dados de aceleração em linha reta para vários valores de potência introduzidos no sistema, de maneira a chegar a uma equação de aceleração, fazendo depois a integração para obter valores de velocidade e nova integração para valores de posição, ao longo do tempo, comparando também com os valores obtidos de posição, obtidos através das câmaras da arena.

Inicialmente foi imposto um valor de potência de 0.3 e os dados obtidos foram os seguintes:

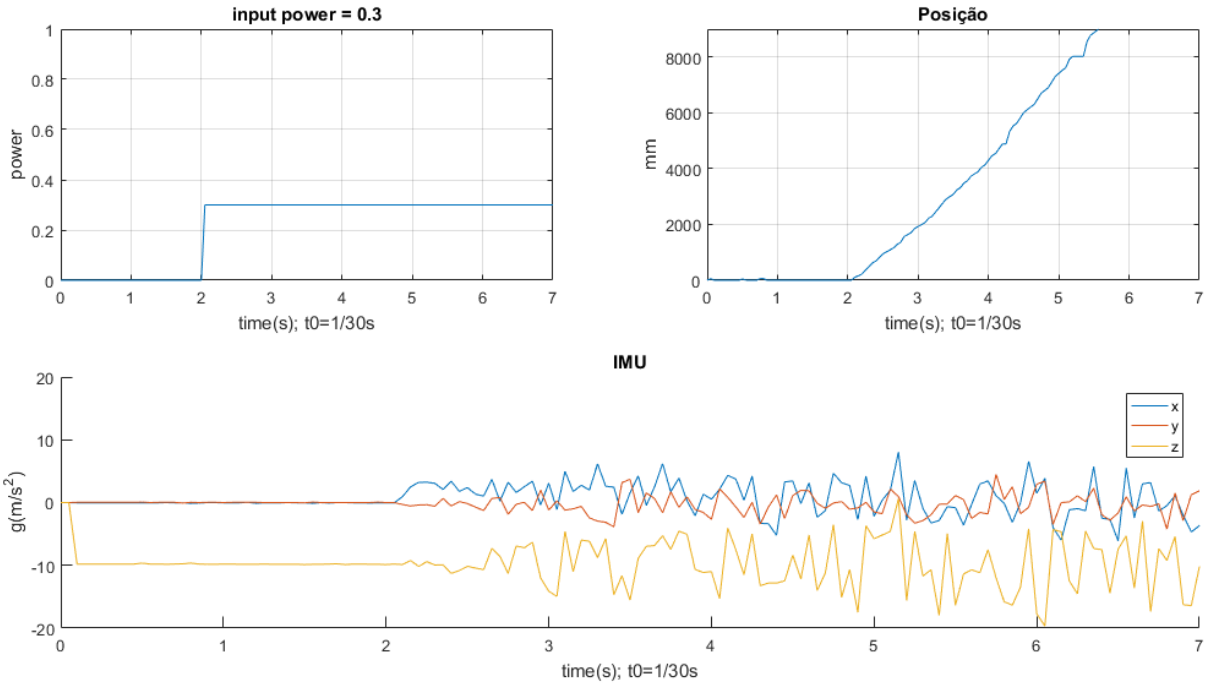

Figura 40 - Resposta a um degrau (power = 0.3)

O tempo de amostragem para a realização do teste foi de 1/30s.

Analisando os resultados obtidos na resposta ao degrau representada nos gráficos anteriores, percebe-se que, até aos 2s, sensivelmente, o sistema está em repouso, o que permite verificar que os dados recebidos pelo acelerómetro da IMU estão dentro do esperado, já que as acelerações nos eixos dos xx e yy são iguais a zero e no eixo dos zz a aceleração é negativa e igual à aceleração gravítica (9.8 m/s^2). Também os dados de posição recebidos pelo sistema de câmaras são verificados no terceiro

gráfico, onde a posição de repouso é igual a zero. Era de esperar que após a entrada do degrau no sistema se verificasse um pico de aceleração no eixo dos xx e uma descida dessa mesma aceleração até zero, fazendo com que a velocidade atingisse ao fim de algum tempo um valor constante e a posição aumentasse linearmente (MLU). No entanto, no gráfico obtido relativos aos valores de aceleração provenientes da IMU não se consegue tirar conclusões, dado que os valores das acelerações em todos os eixos ordenados oscilam de forma quase aleatória quando pelo menos as acelerações em y e em z deviam ser constantes, ou com pequenas variações devido a perturbações, e a aceleração em x devia ter um pico inicial, mas descer também para zero após algum tempo. No gráfico de posição verifica-se que, tal como esperado, o sistema atinge um ponto em que a posição aumenta linearmente com o tempo, no entanto não é possível perceber a partir de que ponto isto se verifica. Este facto pode ser explicado por o tempo de amostragem não ser adequado a este nível de potência introduzida no sistema. Como se sabe, os motores elétricos transmitem toda a potência disponível quase instantaneamente, pelo que para um valor de potência máxima de 0.3/1 as variações de aceleração que se pretendia verificar estão "escondidas" entre as primeiras décimas de segundo após a introdução do input no sistema.

Nota: as variações abruptas no gráfico de posição que se verificam nos segundos 4.1 e 5.2 são erros devido a má leitura de posição das câmaras e como tal devem ser desprezados.

De maneira a tentar resolver os problemas encontrados no ensaio anterior, resolveu fazer-se testes com potências mais elevadas. Com isto, pensou-se que seria possível observar claramente um pico de aceleração proveniente do acelerómetro e ver claramente a progressão dos valores de posição até atingir MLU.

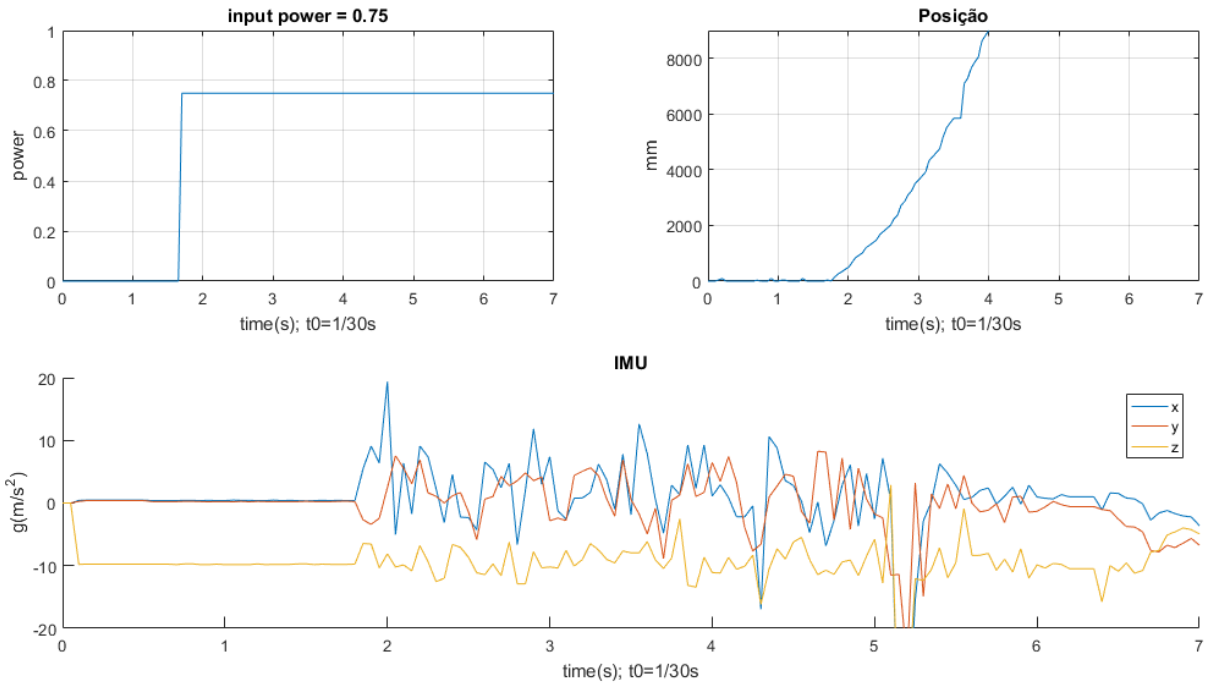

Figura 41 - Resposta a um degrau (power = 0.75)

Mais uma vez, os dados relativos às acelerações nos 3 eixos ordenados estão de acordo com o esperado. Neste ensaio, o degrau foi introduzido no sistema cerca de 0.2s mais cedo que no ensaio anterior.

Como foi dito anteriormente, os motores elétricos transmitem toda a potência disponível de uma só vez, e verificou-se que com este aumento de potência, para tentar visualizar a altura em que se passava a ter movimento linear uniforme, havia escorregamento das rodas. Este facto é suportado pelo máximo relativo seguido de mínimo relativo e máximo absoluto obtidos na curva de aceleração do eixo dos xx no intervalo [1.8;2]s. Para além da validação do escorregamento visualizado, nada mais pode ser concluído da informação obtida através da IMU, visto que à semelhança do que se passou no primeiro ensaio, os dados apresentam variações que podem ser consideradas aleatórias e que de nada servem para avaliar o sistema.

Observando a curva de posição em relação ao tempo consegue perceber-se visualmente o efeito a aceleração na posição a cada instante. Por exemplo, é fácil notar que os primeiros 4000 mm percorridos após a introdução do input demoram mais de um segundo a percorrer, enquanto que dos 4000 mm aos 8000 mm se passa claramente menos de um segundo. Não é, no entanto, possível diferenciar o tempo em que se obtém linearidade no progresso, o que pode ser devido a não ter atingido ainda velocidade estacionária aquando do fim do ensaio. De qualquer forma o escorregamento verificado nas rodas impõe não linearidades que não são desejáveis, pelo que se teve que tomar outra abordagem ao problema.

Nota: Mais uma vez, a variação abrupta no gráfico de posição que se verifica no segundo 3.5 é um erro devido a má leitura de posição das câmaras e como tal deve ser desprezado.

Foram realizados mais alguns ensaios com outros valores de potência, mas não obtendo resultados satisfatórios, o que levou a concluir que os dados recolhidos pelo acelerómetro acoplado na IMU não seriam viáveis para realizar *feedback* num possível anel de controlo e como tal, a determinação da posição do RC a cada instante teria que ser feita em anel aberto.

4.2.2 LINHA RETA

Não havendo forma de reajustar a posição à medida que o sistema avança, não pode haver escorregamento nas rodas, caso contrário os dados relativos à distância percorrida introduzidos no subsistema de visão computacional não serão adequados, o que se traduz numa avaliação errada da posição de obstáculos e inviabiliza o sistema. Como tal, optou-se por averiguar qual a potência mínima para a qual o sistema se movia, evitando assim o escorregamento das rodas e o erro que daí advém, e usar esse valor de potência para todos os movimentos realizados pelo sistema.

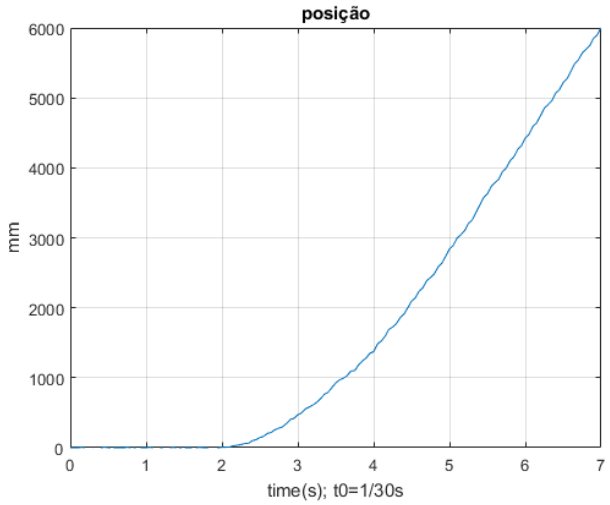

Figura 42 – Variação da posição em linha reta com o tempo

Depois de realizadas algumas experiências com valores de potência no intervalo [0.05;0.25] decidiu-se usar o valor de 0.2 para a implementação no sistema.

Assim, assegurou-se que não existia escorregamento das rodas nem no arranque, nem no progresso do sistema ao avançar em linha reta. Na figura 42 apresenta-se o gráfico de variação da distância percorrida em função do tempo para o valor de 0.2 de potência.

Admitindo que para este valor de potência, no último segundo de leitura (tempo = [6,7]s), o sistema se encontra em regime estacionário, aproximou-se essa zona da curva linearmente.

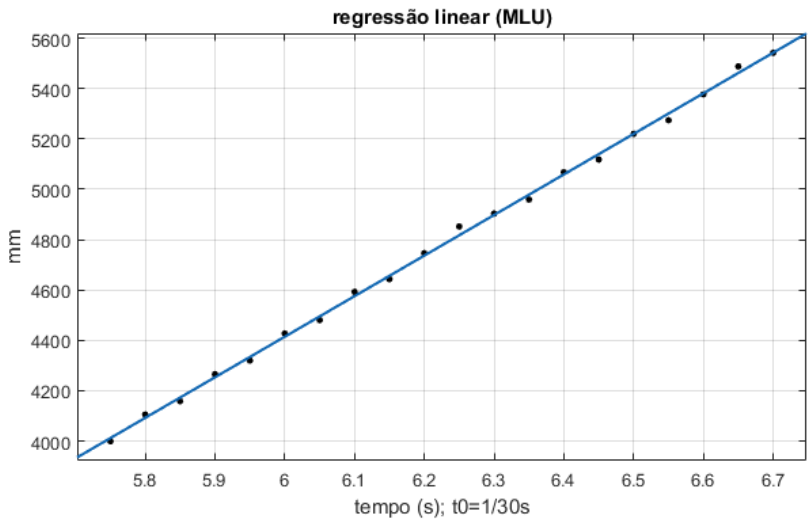

Figura 43 – Regressão linear

A função que dita a regressão linear representada no gráfico da figura anterior é:

$$p(t) = 1611t - 5250\ mm \tag{16}$$

Derivando a função de posição encontrada anteriormente, encontra-se a velocidade de estacionariedade do sistema, $v(t) = 1611\ mm/s$. Posto isto, e admitindo que a função de posição, até

atingir estacionariedade pode ser aproximada por uma função polinomial de terceira ordem, aproximou-se toda a curva por um polinómio de terceiro grau.

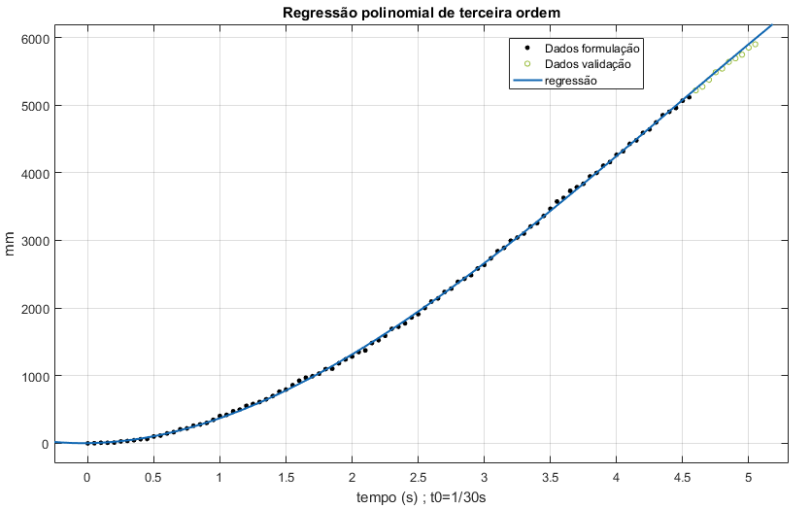

Figura 44 – Regressão polinomial de 3ªordem

A função encontrada que descreve a regressão representada no gráfico é a seguinte:

$$p(t) = -30.44t^3 + 385.6t^2 + 3.835t \tag{17}$$

A primeira derivada de p(t) determina a função de velocidade v(t) e a segunda derivada determina a função de aceleração a(t) do sistema.

$$v(t) = -91.32t^2 + 771.2t + 3.835 \tag{18}$$

$$a(t) = -182.64t + 771.2 \tag{19}$$

Sabendo que em MLU a velocidade é constante e a aceleração é nula, iguala-se a função de aceleração a zero de modo a encontrar o ponto em que é atingido esse regime.

$$0 = -182.64t + 771.2 \;\rightarrow\; t = 4.2s \tag{20}$$

Substituindo t = 4.2s na equação de velocidade, determina-se a velocidade terminal estacionária.

$$v(4.2) = -91.32\times4.2^2 + 771.2\times4.2 + 3.835 \rightarrow\; v_{final} = 1632\, mm/s \tag{21}$$

Comparando este valor de velocidade final com o valor encontrado na regressão linear anterior, percebe-se que este valor é um pouco mais elevado, o que pode querer dizer que a regressão polinomial de terceira ordem não é adequada.

Analisando o gráfico da figura 43 e o complementar gráfico de resíduos da figura 44, podem tirar-se algumas conclusões.

Tabela 7

Conjunto de dados	Dados de formulação	Dados de validação
RMSD (mm)	23.07	47.95
Módulo do Desvio Máximo (mm)	60.9	83.28

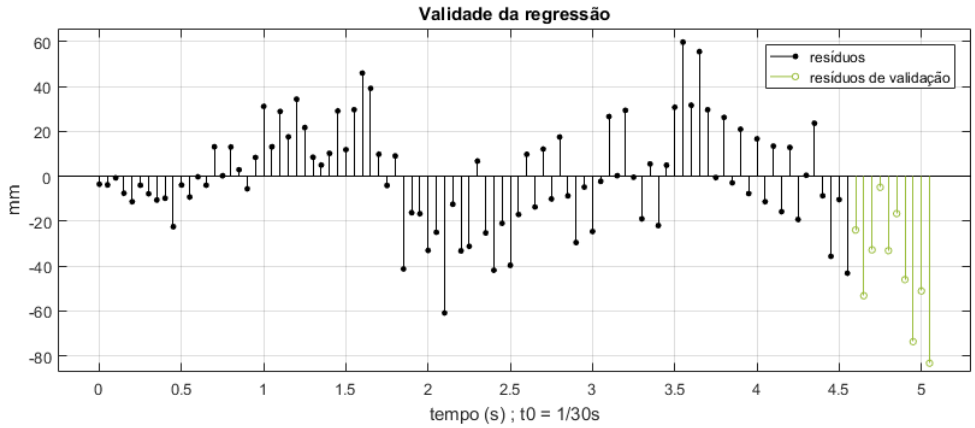

Figura 45 – Desvio da regressão polinomial de terceira ordem

Os dados no gráfico de resíduos mostram que a regressão polinomial encontrada anteriormente não se ajusta perfeitamente para os dados de validação introduzidos. Este facto é suportado pelos valores de RMSD apresentados na tabela 7, onde o valor para os dados de validação é muito superior do que o valor dos dados de formulação da regressão. Mais do que isso, pode observar-se que o desvio da regressão em relação aos pontos de validação é sempre para valores superiores aos esperados, atingindo se mesmo desvios máximos superiores ao desvio máximo obtido na definição da regressão. A regressão linear efetuada anteriormente determinou uma velocidade máxima v_{final} = 1611 mm/s, que se sabe ser adequada à zona estacionária da curva de posição. O que é proposto é conjugar as duas regressões para as diferentes posições da curva no ponto em que a regressão de terceiro grau atinge aceleração nula, t=4.2s, de maneira a ter um modelo final da progressão da posição ao longo do tempo.

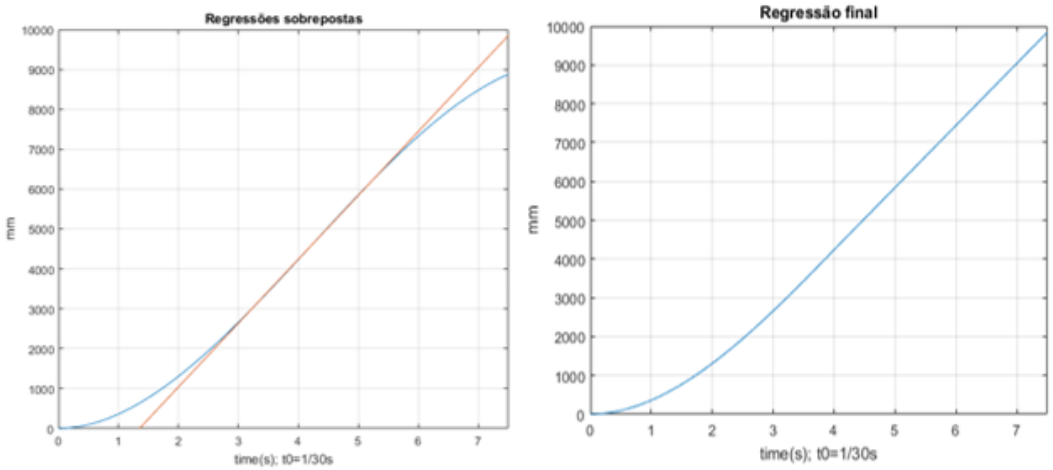

Figura 46 - Regressões sobrepostas (à esquerda) e regressão final composta (à direita)

Os gráficos anteriores representam a curva final de progressão em linha reta, cuja função é a seguinte:

$$p(t) = \begin{cases} -30.44t^3 + 385.6t^2 + 3.835t\,, t \leq 4.2 \\ 1611t - 5250\,, t > 4.2 \end{cases} \quad (22)$$

4.2.3 EM CURVA

O modelo implementado faz com que o sistema se mova de posição em posição do mapa, parando e reiniciando o movimento em cada uma das posições, para assegurar que as imagens são captadas à distância pretendida. Assim, a identificação do movimento em curva foi realizada sempre a uma potência constante igual à implementada para os ensaios em linha reta e, o que interessava perceber era a posição e orientação do sistema consoante o tempo e o valor de direção imposto ao sistema. Tal como para o caso da potência introduzida no sistema, também os valores de direção estão normalizados. A extensão de valores aceites para direção varia entre -1 e 1, sendo que os valores negativos correspondem a virar para o lado esquerdo, zero faz o sistema mover-se em frente e valores positivos fazem o sistema mover-se para a direita.

Foram realizados ensaios para os valores de direção máximos para as duas direções, esquerda e direita, sendo que o que importava modelar era o movimento do RC na extensão de 0 a 45º e 0 a -45º de orientação, visto que no algoritmo implementado, em nenhuma posição o sistema assume orientações diferentes deste intervalo.

Impôs-se então a potência e direção definidas constantes e deixou-se o RC mover-se com essas características durante algum tempo, obtendo valores de posição semelhantes aos representados no gráfico da figura 46.

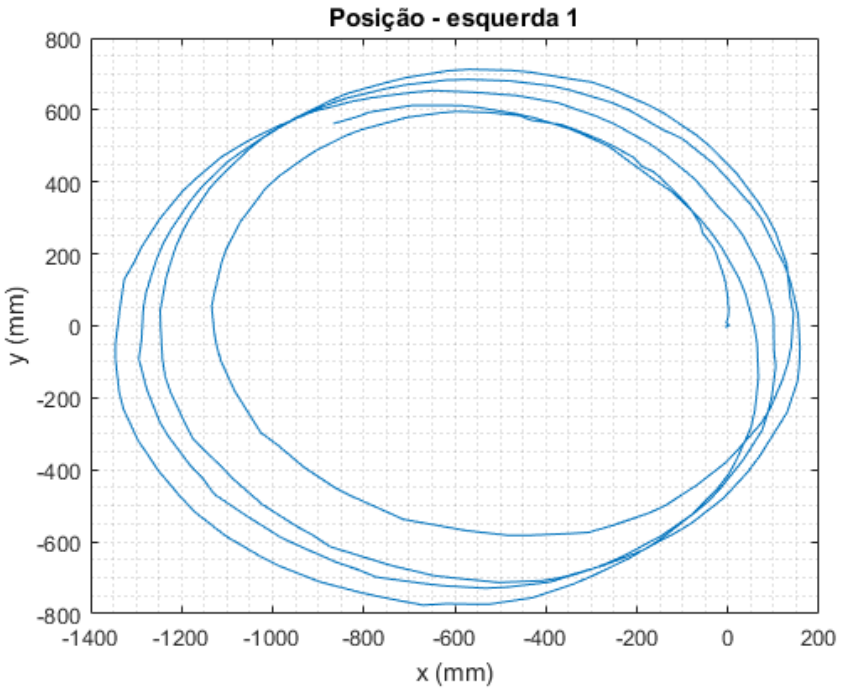

Figura 47 – Movimento circular com direção e potência constante

Através do gráfico anterior percebe-se claramente que o movimento não é regular, no entanto, como foi referido, apenas o movimento até se atingir 45º para ambas as direções tem que ser modelado. Como tal, na figura seguinte apresentam-se os pontos extraídos desde o início do movimento até ao ponto em que o RC com uma orientação de -90º, também para o caso do movimento para a esquerda com valor de direção igual a 1.

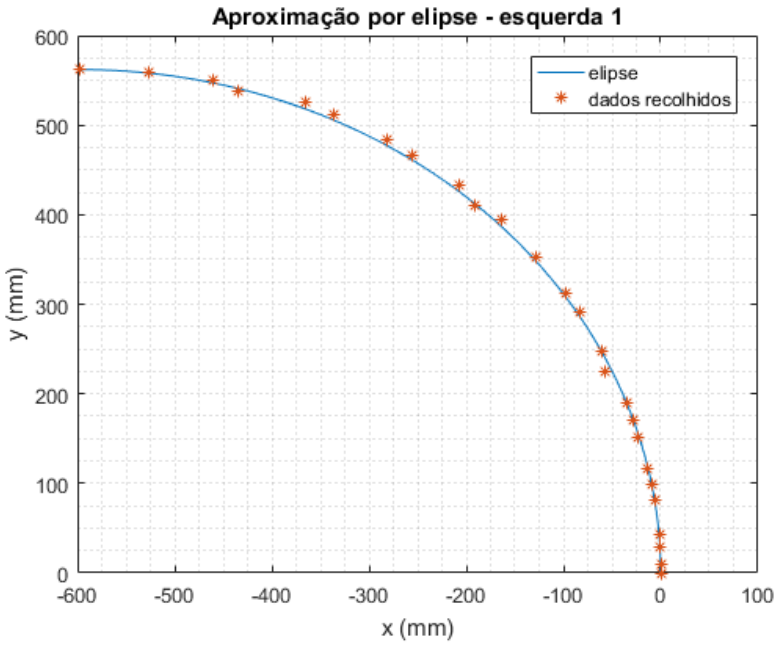

Figura 48 – Aproximação de elipse ao movimento em curva à esquerda até orientação igual a 90º

Na figura 48 para além dos pontos adquiridos no ensaio relativos à curva realizada pelo sistema, está desenhada uma elipse aproximada do movimento. A equação que descreve a elipse é a seguinte:

$$\frac{(x-h)^2}{a^2}+\frac{(y-k)^2}{b^2}=1 \tag{23}$$

Onde, a e b são os valores dos raios maior e menor, respetivamente, da elipse, e C(h,k) é o centro da elipse. Todos estes valores são constantes:

$$a=597\,;b=562\,;C(h,k)=(-597{,}0) \tag{24}$$

Depois de feita a aproximação à curva, tem que se saber em que ponto da mesma, a orientação do RC é igual a -45º, para se saber qual a posição no espaço em que o sistema se encontra e quanto tempo demorou a lá chegar. Assim, sabendo que a orientação do RC é igual ao ângulo formado pelo declive da tangente da curva e o eixo dos yy, é necessário calcular as tangentes ao longo da curva anterior e encontrar o ponto em que a tangente é igual a -45º.

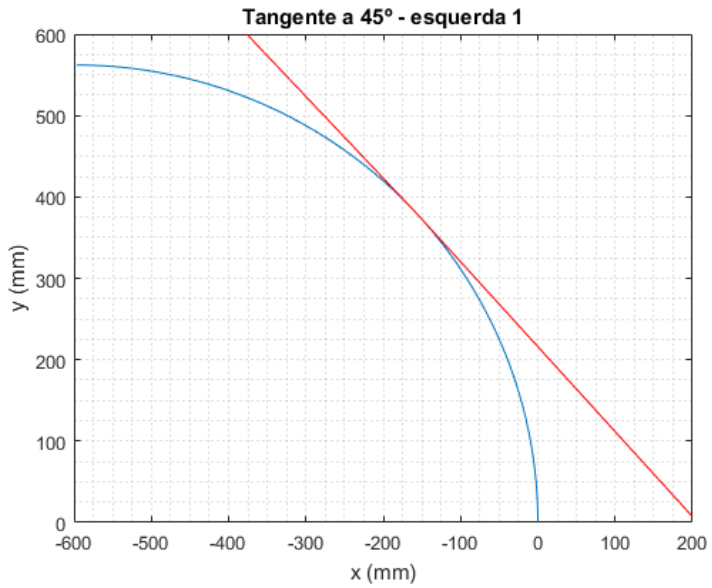

Figura 49 – Tangente à elipse quando a orientação do RC é igual a -45º

No gráfico da figura 49 está desenhada, mais uma vez a curva efetuada pelo RC e também a tangente com declive igual a -45º que permite descobrir que a posição no espaço em que o sistema se encontra quando tem esta orientação é (-161.8,384.7) mm. Sabendo a posição ocupada, resta saber o tempo demorado a atingir esta posição, e isso consegue-se através da comparação das coordenadas de posição com uma curva de coordenadas e tempo recolhida no ensaio. Como os valores recolhidos no ensaio são de natureza discreta, a posição exata calculada não é contemplada em nenhum dos pontos recolhidos, pelo que podia fazer-se uma interpolação entre os pontos mais próximos, mas visto as diferenças de ponto para ponto são de apenas meio segundo, e que um dos pontos recolhidos p = (-164,394) se encontra muito próximo do ponto pretendido, aproxima-se o tempo pretendido ao tempo que se demorou a chegar a este ponto, chegando a t(p) = 0.75s.

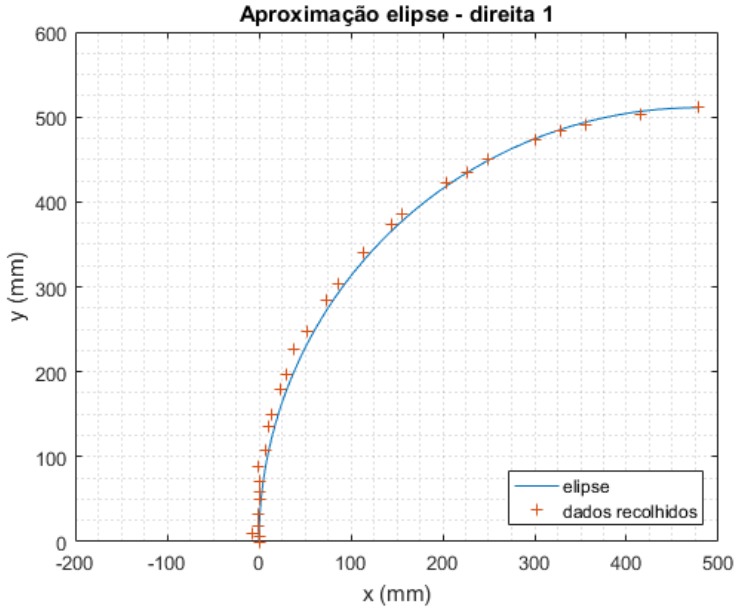

Figura 50 - Aproximação de elipse ao movimento em curva à direita até orientação igual a 90º

À semelhança da aproximação feita para curva à esquerda, aproximou-se uma equação de elipse à curva efetuada para o lado direito em condições inversas às dispostas para a curva à esquerda. A elipse representada é descrita pela função 23 e as constantes são as seguintes:

$$a = 478\,;b = 511\,;C(h,k) = (478{,}0) \tag{25}$$

Analisando visualmente os gráficos das posições recolhidas é possível perceber que a curva para a direita é uma curva mais fechada que a curva à esquerda, pelo que é de esperar que o ponto em que o sistema atinge 45º seja numa posição com coordenada x, em módulo, menor do que no caso da curva à esquerda, e vice-versa para a coordenada y. Para verificar isso, procedeu-se da mesma forma que anteriormente, descobrindo a tangente no ponto em que a orientação do RC é igual a 45º.

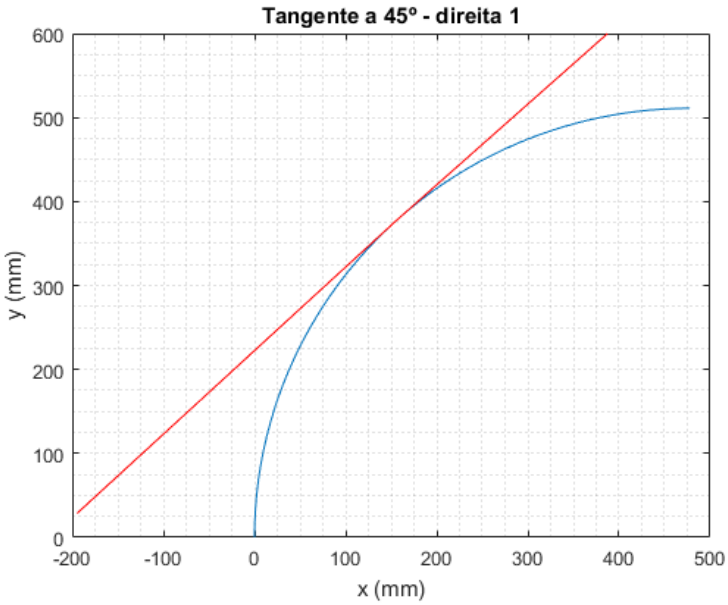

Figura 51 - Tangente à elipse quando a orientação do RC é igual a 45º

A posição no espaço que o sistema ocupa no ponto em que a tangente desenhada toca na elipse é (155.4,377.1). Este ponto corrobora o que foi dito anteriormente, confirmando a ideia visual de que a curva para o lado direito, para os valores de potência e direção impostos, é mais apertada que a curva para o lado esquerdo.

Tal como para a curva para o lado esquerdo, averiguou-se o tempo que o sistema demorava a atingir o ponto em que a sua orientação era igual a 45º. Mais uma vez, devido ao facto dos dados recolhidos serem discretos, o ponto exato em que se pretendia medir o tempo não correspondia a um ponto existente nos dados, pelo que, se aproximou ao ponto mais próximo, que era na posição p = (157,368). O tempo que o sistema demorou a chegar a este ponto foi então t(p)=0.85s.

Capítulo 5 – Testes e Resultados

Este capítulo tem três secções principais. Uma primeira onde se descreve os princípios por detrás da escolha das dimensões da malha usada para a formulação dos mapas de obstáculos pelos algoritmos, que envolvem algumas alterações de movimentos definidos no capítulo 4. Uma segunda parte onde se apresentam três testes para o seguimento de caminhos diversos pelo RC. E uma secção final onde se apresentam dois testes do sistema global a funcionar em dois ambientes com disposições de obstáculos diferentes.

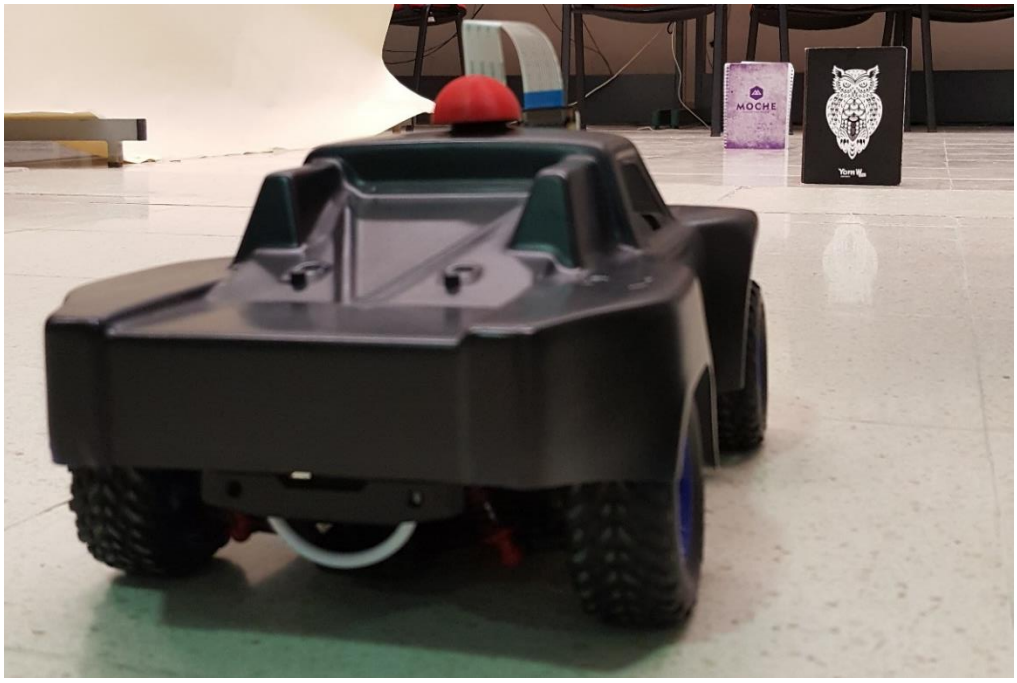

Figura 52 – testes ao sistema

5.1 Definição das dimensões da malha

Antes da realização dos testes definiu-se o tamanho da malha na qual o mapa seria desenhado e do qual os parâmetros, a introduzir no algoritmo, dependem. Para a definição do tamanho da malha é importante ter em conta que quanto mais reduzida for, maior precisão o sistema terá, mas também será mais demorado o processo de planeamento de rota, no entanto, se for uma malha muito grosseira há uma diminuição de possibilidades para o sistema se movimentar, pelo que era importante encontrar dimensões adequadas. Para isso, levou-se em linha de conta os dados recolhidos no primeiro teste de validação do método de visão computacional, no capítulo 2, em que se percebe que distâncias entre imagens de 10 cm apresentam erros maiores quando comparando com distâncias calculadas com imagens obtidas a distâncias maiores, pelo que o tamanho da malha deve ser maior que 10 cm. Para

distâncias superiores a 40 cm entre imagens não foram feitos ensaios, dado que pequenos desvios na orientação do RC ficam mais evidentes, obviamente, com maiores distâncias percorridas, pelo que se consideraram tamanhos de malhas inferiores a 40 cm. Mais uma vez, recorrendo à tabela dos dados recolhidos na primeira validação do método de visão computacional, e atendendo aos erros obtidos, optou-se por usar uma malha com 30 cm de lado de quadrado, e tamanho total 10x10 quadrados, cobrindo assim uma área de 3x3 metros.

Com a definição das dimensões da malha para o mapa, surge um problema relacionado com o raio de curvatura do RC e o pressuposto de movimentações possíveis propostas no algoritmo.

Analisando o caso em que o RC se encontra numa qualquer posição da malha com orientação igual a 0º, dos três movimentos possíveis a partir desta posição, dois deles envolvem curvar, seja para a esquerda ou para a direita. Nestes dois casos, segundo o formulado anteriormente, o RC após um movimento tem que se encontrar 30 cm deslocado no eixo dos xx e 30 cm no eixo dos yy e com orientação igual a ±45º. É aqui que surge o problema, visto que tanto para um lado como para o outro, o RC precisa de avançar mais do que 30 cm no eixo dos yy para atingir 45º de orientação.

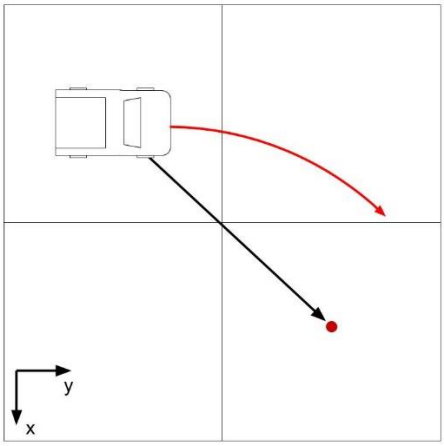

Figura 53 – Ilustração do movimento de curva à direita

Na figura 53 está representado o problema acima descrito. A seta vermelha representa o movimento real do RC, enquanto que a seta preta representa o movimento que o algoritmo esperava que fosse feito pelo sistema. É possível perceber que da maneira que o algoritmo está formulado não se consegue chegar às posições esperadas nesta malha, pelo que teve que se pensar num método alternativo para atingir as posições pretendidas.

Segundo o método definido, o algoritmo avalia apenas o comando i+1 para decidir que decisão tomar. A solução pensada para contornar o problema envolve a avaliação de mais comandos. A ideia básica é começar o movimento de viragem mais cedo para que haja mais espaço para o RC atingir a posição e orientação pretendidas. No entanto, para se aproveitar mais espaço, o sistema precisa de ter mais informação sobre os comandos planeados na rota. O que se pretende é que antes de um movimento em linha reta seguido de uma curva, o sistema possa antecipar o movimento de curva e começar a curvar antes de atingir o ponto médio do quadrado após o movimento em linha reta.

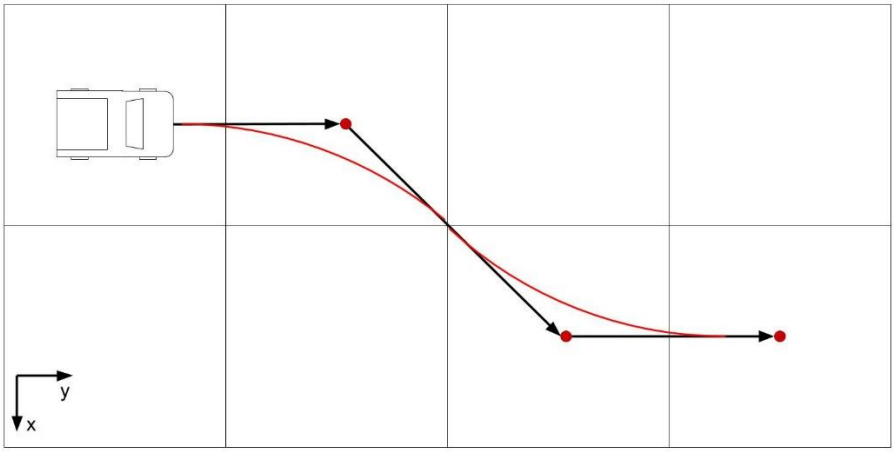

Figura 54 – Movimento alternativo ao definido no algoritmo

Na figura 54 está representado o que se pretende, no caso de uma curva e contracurva. Algumas posições não chegam a ser atingidas, mas este procedimento permite que se consiga percorrer o espaço na malha definida. É de notar que se pretende que a posição do sistema em que a sua orientação é 45º seja sensivelmente sobre a interseção entre quatro secções do mapa.

Pseudo-código:

se o comando (i+1) == reta
captar imagem;
avançar 30cm em linha reta;
captar imagem;
reiniciar ciclo;
se o comando (i+1) == direita/esquerda && comando (i+2) == esquerda/direita
avançar em linha reta distância necessária até ao inicio da curva;
avançar em curva;
avançar em contracurva;
avançar em linha reta a distância restante até à posição desejada;
i = i+2;
se o comando (i+1) == direita/esquerda && comando (i+2) == reta
avançar em linha reta distância necessária até ao inicio da curva;
avançar em curva;
captar imagem;
avançar 30cm em linha reta;
captar imagem;
reiniciar ciclo;
fim

A definição das distâncias a percorrer nas secções em linha reta, antes e depois das curvas, é feita através das curvas encontradas no capítulo de identificação do sistema. Analisando o caso da curva para a direita, o ponto em que se atinge 45º é nas coordenadas (155.4,377.1) mm, pelo que, em 450mm

(1.5 quadrados da malha) o sistema deve estar na posição (150,450) mm, considerando a posição inicial (0,0). Para isso, tem que se percorrer 450-377.1=72.9 mm em linha reta antes de começar a curvar. Para percorrer esta distância, recorre-se à função encontrada no capítulo anterior que relaciona a distância percorrida em linha reta com o tempo, igualando p(t) a 72.9 mm.

$$72.9 = -30.44t^3 + 385.6t^2 + 3.835t \quad (26)$$

$$t = -0.43s\ ; t = 0.44s\ ; t = 12.7s \quad (27)$$

A única opção dentro dos parâmetros é t = 0.44s, pelo que este é o tempo que o sinal de potência igual a 0.2 e orientação igual a 0 tem que ser enviado para o sistema percorrer 72.9 mm.

O mesmo processo foi usado para os tempos que os sinais dos outros movimentos necessitam e os resultados que foram depois usados no modelo estão dispostos na tabela seguinte.

Tabela 8

		Reta	Esquerda e direita	Direita e esquerda	Esquerda e reta	Direita e reta
Tempo (s)	linha reta	0,91	0,43	0,44	0,43	0,44
	curva esquerda	-	0,75	0,75	0,75	0,85
	curva direita	-	0,85	0,85	-	-
	linha reta	-	0,43	0,43	0,91	0,91

5.2 TESTES AO MOVIMENTO E PARÂMETROS

De forma a testar os parâmetros recolhidos no capítulo anterior e a sua aplicação, começou-se por testar apenas a efetividade com que o sistema percorria um caminho predefinido. Este teste servia para avaliar a evolução do erro de posição causado pelos atrasos de comunicação mencionados no capítulo 4, ao fim de vários comandos seguidos. Ainda que o acumular de erro na posição não fosse extremamente importante, visto que para a grande maioria dos casos, após uma série pequena de comandos o mapa seria atualizado e como tal a posição atual seria considerada posição inicial novamente. Para os casos em que a série de comandos imposta pelo algoritmo não permitisse a atualização do mapa tão rapidamente, era necessário perceber se os erros de posição se iam acumulando e prejudicando a validade do caminho a ser percorrido. Sem o sensor ligado, realizaram-se então três testes com obstáculos virtuais a obrigar o sistema a criar uma rota para os evitar e de seguida a percorrer a rota calculada.

O primeiro e segundo teste foram realizados para testar separadamente os movimentos de contornar obstáculos pelo lado esquerdo e pelo lado direito, respetivamente. Para o primeiro teste foram colocados obstáculos nas posições [(3,4);(4,4);(5,6);(6,6)].

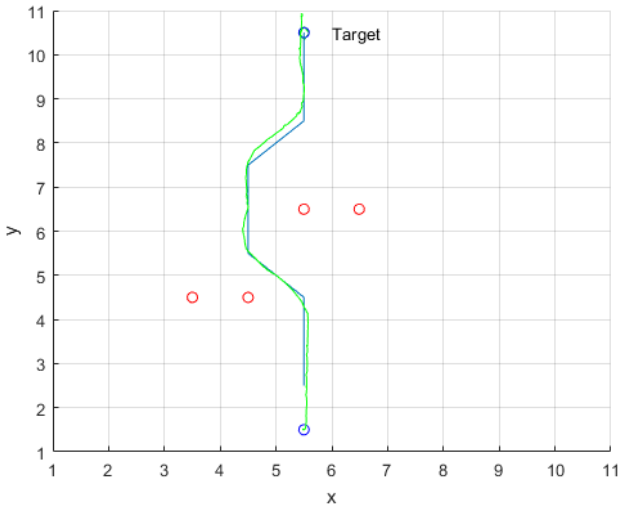

Figura 55 – 1ºteste

Na figura 55 está representado o mapa com os obstáculos referidos e a rota planeada pelo algoritmo, a azul. A verde está representado o percurso efetuado pelo RC. Com uma observação à vista desarmada, o percurso parece ter sido percorrido com boa precisão, no entanto é pertinente avaliar o erro pelo menos nos pontos centrais das secções da malha por onde a rota passou. Avaliou-se o erro nas duas coordenadas separadamente.

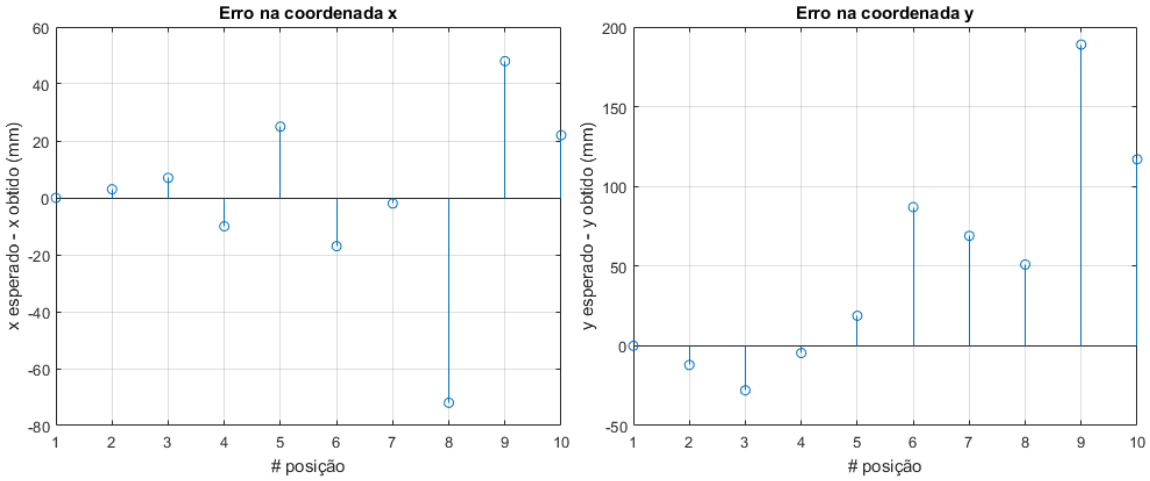

Figura 56 - Erro na coordenada x (à esquerda) e erro na coordenada y (à direita)

Os gráficos de erros estão de acordo com o que se esperava. Na coordenada x, havendo apenas duas variações de posição na malha, aquando da curva para a esquerda e mais tarde na curva para a direita de volta à orientação inicial, pelo que seria de esperar que as variações de erro fossem pequenas e que o erro máximo, em módulo, fosse também pequeno. É isso que se verifica, visto que o erro máximo que se verifica é de 72 mm e que o erro médio é de 21 mm. Já na coordenada y, existem 10 mudanças de posição, pelo que se esperava que os erros fossem superiores, o que é corroborado pelo gráfico da figura 53, onde o erro máximo verificado é de 189 mm e o erro médio é de 58 mm. Para além disto, o erro também tem tendência a ser maior à medida que se vai avançando. Se se analisar, o erro médio

nas primeiras 5 posições é de 13 mm, enquanto que para as últimas 5 posições, o erro médio é de 89 mm, no entanto 10 posições poderão não representar uma amostra significativa.

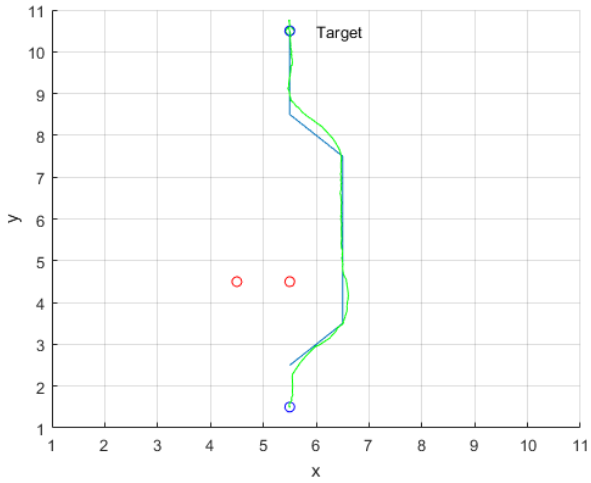

Figura 57 – 2º teste

Tal como no caso do primeiro teste, na figura 57 estão representados os obstáculos previamente definidos, a rota planeada pelo algoritmo e o percurso efetuado pelo sistema. Como se pretendia, o algoritmo contornou os obstáculos seguindo pela direita dos mesmos.

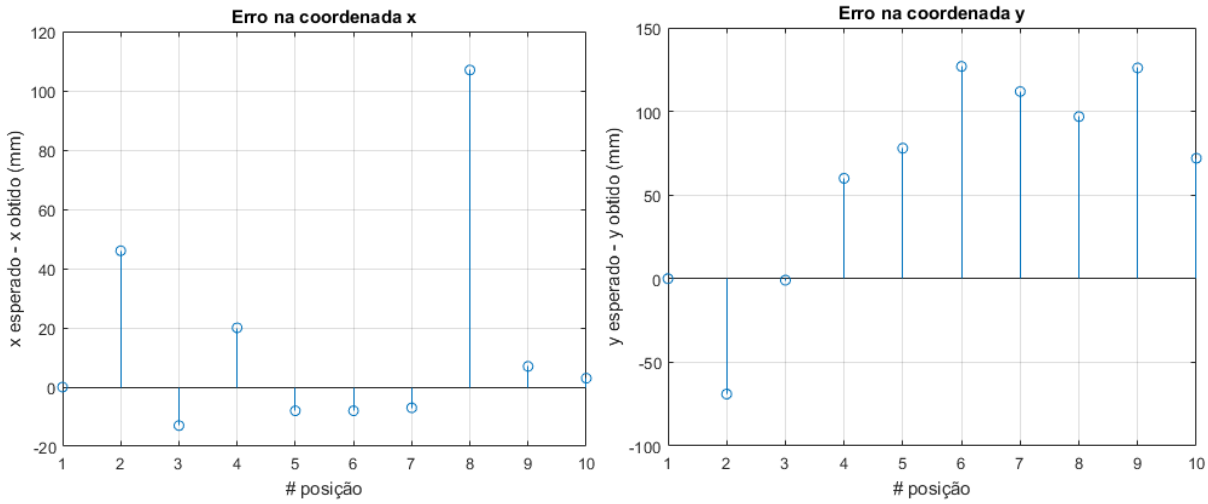

Figura 58 - Erro na coordenada x (à esquerda) e erro na coordenada y (à direita)

Mais uma vez, os gráficos relativos aos erros de posição vão de encontro ao esperado. São efetuadas apenas duas variações na coordenada x e o gráfico do erro é semelhante ao gráfico do primeiro teste, apresentado erros, no geral, pequenos. Já os erros na coordenada y, ao contrário do que se verificou no primeiro teste, apesar de serem mais elevados nas últimas 5 posições, parecem elevar-se a partir da primeira mudança de direção e desde esse momento mantêm-se relativamente constantes.

No terceiro teste foram colocados mais obstáculos e alterado o ponto objetivo para que houvesse três mudanças de direção, e como tal, três valores diferentes para a coordenada x no mesmo percurso. Os obstáculos foram colocados então nas posições [4,2;4,3;5,3;5,4;5,5;4,6;4,7;4,8].

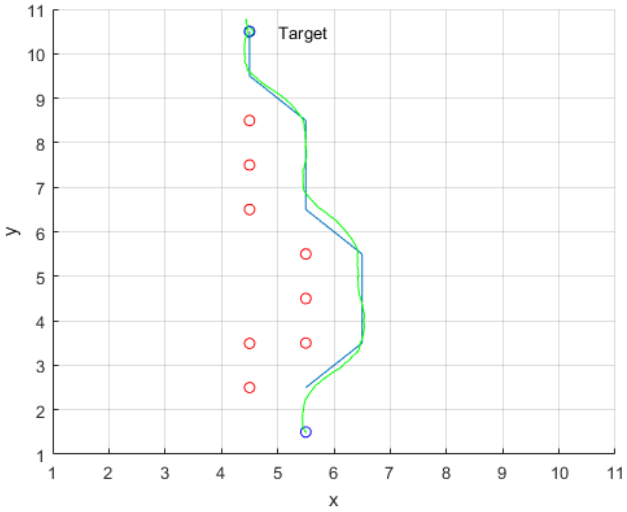

Figura 59 - 3º teste

Na figura 59 pode observar-se a maior quantidade de obstáculos presentes neste teste e a forma como o algoritmo planeou a rota, apresentando um caminho ligeiramente mais complexo do que os anteriores.

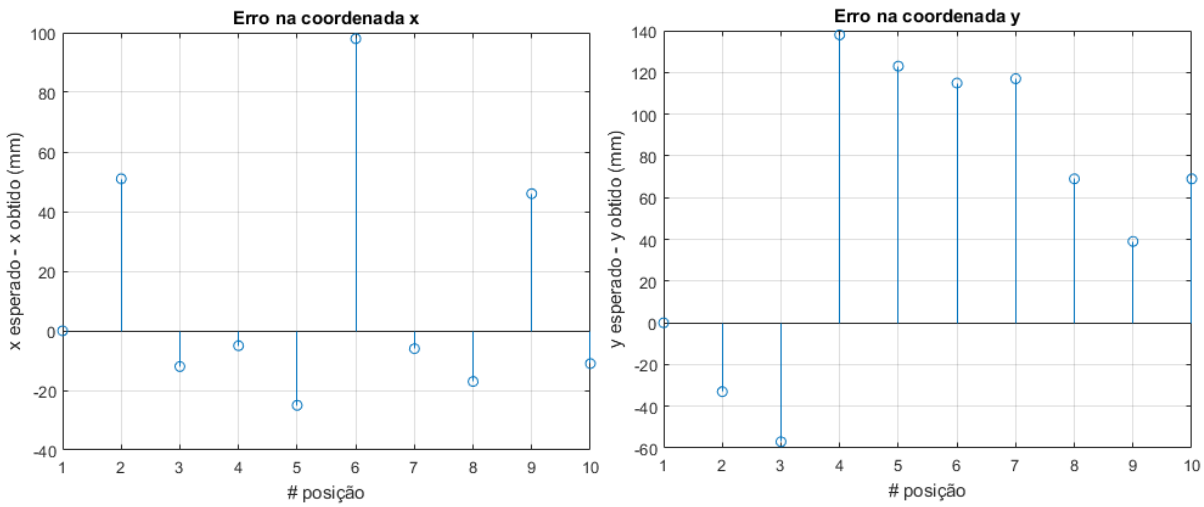

Figura 60 - Erro na coordenada x (à esquerda) e erro na coordenada y (à direita)

Os gráficos dos erros neste percurso apresentam muitas semelhanças com os erros do segundo teste, o que se justifica pela semelhança no percurso, em grande parte. Particularmente, no gráfico relativo à coordenada x os erros em cada posição têm pequenas variações, à exceção da sexta posição, que corresponde a curvar à esquerda, onde se encontra um pico de erro discrepante. Se se notar, já no segundo teste, a variação mais abrupta no erro na coordenada x verifica-se no ponto em que o sistema curva à esquerda. Já na coordenada y, também se verificam semelhanças com o segundo teste, visto que a maior variação de erro ocorre aquando da mudança de direção à direita, mantendo-se de aí em diante as variações de erro pequenas. Para além disto, a última alteração de trajetória para atingir o ponto objetivo, que é a principal alteração neste caminho em relação ao segundo teste, não parece suscitar variações no erro relevantes.

Depois de executados os três testes acima descritos, criou-se a tabela seguinte. Todos os dados apresentados estão em mm.

Tabela 9

	1º teste		2º teste		3º teste		média total	
	x	y	x	y	x	y	x	y
direita-esquerda	50,0	120,0	-26,0	129,0	-56,0	171,0	-10,7	140,0
esquerda-direita	-7,0	91,5	14,0	14,1	-3,0	12,5	1,3	39,4
reta	-6,3	-33,3	-10,7	4,3	-15,5	-31,5	-10,8	-20,2
média total	12,2	59,4	-7,6	49,1	-24,8	50,7	-	-

Da tabela anterior pode concluir-se que os movimentos relativos à sequência de curvas direita/esquerda são os únicos que revelam erros mais elevados e que podem levar a que o sistema não seja viável, no entanto, espera-se que o processo de atualização de mapa contorne este problema.

5.3 TESTES AO SISTEMA

Depois de testados os parâmetros relacionados com o movimento do sistema e de verificada a validade dos mesmos para a implementação no sistema, procedeu-se à introdução do método de identificação e cálculo de distâncias até aos obstáculos no sistema, testando assim todas as vertentes introduzidas nesta dissertação.

Os testes realizados tinham como objetivo averiguar de que modo os erros no movimento do RC influenciariam a leitura por parte do sensor do ambiente em frente do sistema e se, com isso, o funcionamento do sistema completo seria válido. Foram realizados trinta testes com vários obstáculos e várias disposições desses mesmo obstáculos no ambiente em frente do RC. De seguida ilustram-se dois dos testes realizados.

5.3.1 1º TESTE

No primeiro teste, foram colocados dois obstáculos no ambiente em frente do sistema, sendo que o objeto mais próximo do RC está diretamente no caminho do mesmo e o segundo um pouco mais afastado e deslocado para a esquerda.

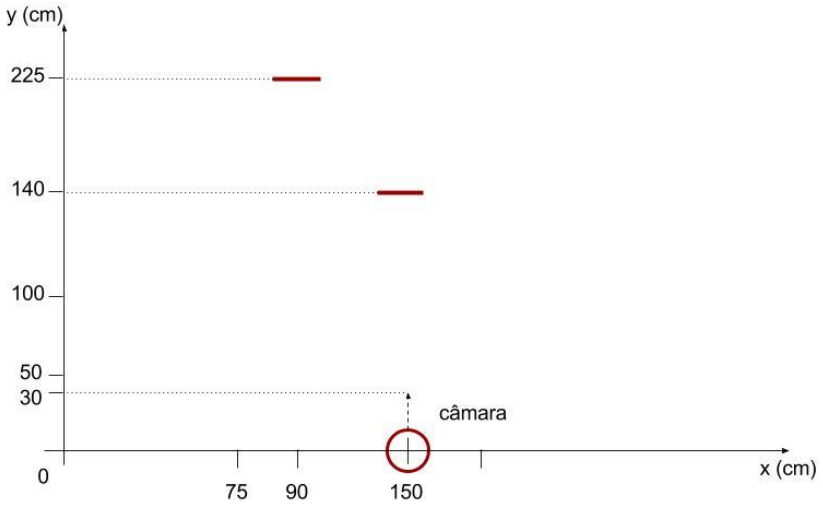

Figura 61 - Disposição do ambiente no 1º teste

Na figura 61 está esquematizada a posição dos obstáculos no ambiente do primeiro teste. O objetivo desta disposição de objetos no espaço prendia-se com a identificação e cálculo de posições corretas dos obstáculos na inicialização do sistema e com a capacidade de execução do caminho calculado pelo algoritmo, evitando os obstáculos mapeados.

Após a inicialização, o emparelhamento dos pontos de interesse encontrados na imagem captada no ponto inicial e após o movimento de inicialização estão representados na figura seguinte.

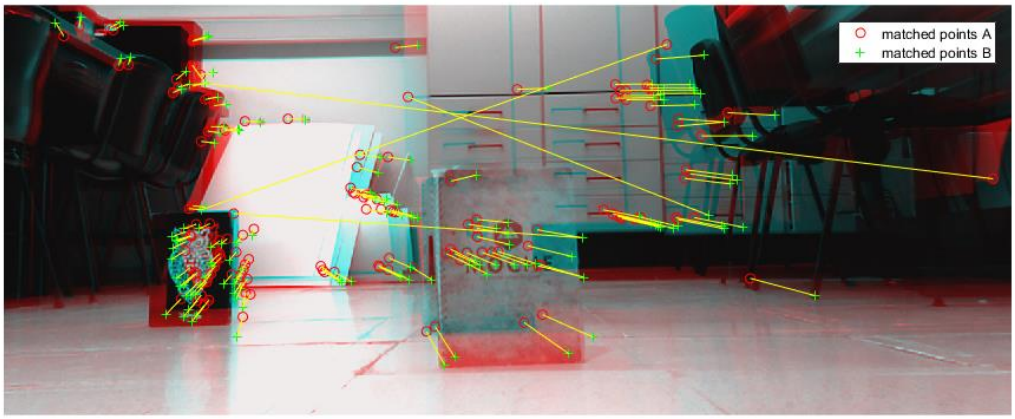

Figura 62 - Emparelhamento de figuras na inicialização do sistema

Na figura 62 é possível ver a disposição dos obstáculos e os pontos emparelhados. Olhando para o fundo da imagem, é importante notar que as linhas do móvel junto à parede sofreram deslocamento para o lado direito, o que indica que a segunda imagem foi captada numa posição mais à esquerda, ou com a orientação do sistema ligeiramente deslocada para a esquerda. Este efeito é causado pelo erro de posição na coordenada x verificado na primeira parte deste capítulo. É possível também ver muitos pontos emparelhados erradamente que são eliminados apenas posteriormente na fase de *clustering* e cálculo de distâncias.

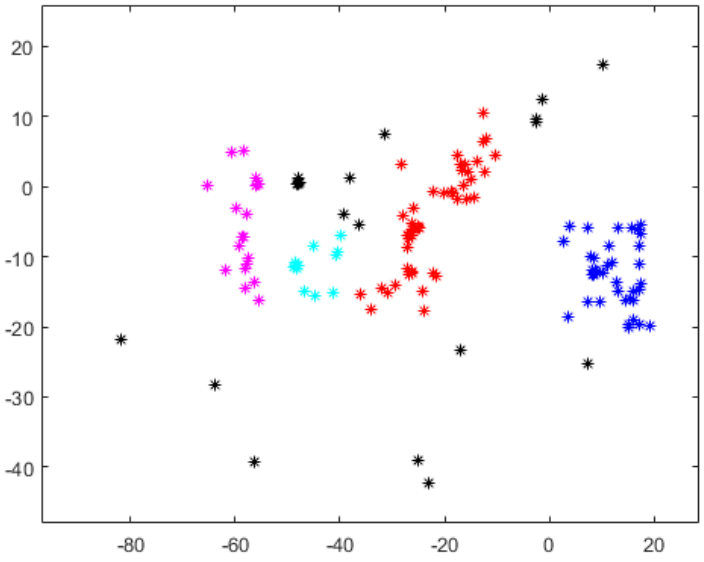

Figura 63 - *Clustering* da fase da inicialização

Como se pode ver na figura anterior, foram criados quatro clusters para os quais foram calculadas distâncias longitudinais e laterais. Estes grupos pertencem a outras partes do ambiente que não os obstáculos colocados propositadamente no caminho do sistema, pelo que muitas vezes não surgem sequer na janela de 3x3m onde são inseridos os obstáculos. Noutras situações pode surgir o caso de, para um só objeto, vários grupos de pontos serem classificados, e nesse caso é esperado que os cálculos das distâncias coloquem vários obstáculos no mesmo local. No primeiro teste, os cálculos relativos aos grupos de pontos do gráfico de *clustering* deram origem às seguintes coordenadas de obstáculos no espaço.

Tabela 10

		1º cluster	2º cluster	3º cluster	4º cluster
posição	**x (cm)**	95	86	-7	153
	y (cm)	326	203	389	114

Dos quatro grupos de pontos para os quais foram calculadas distâncias, apenas o segundo e o quarto grupo originaram pontos no espaço que se encontram no mapa. Os obstáculos localizados no espaço do mapa correspondem aos objetos colocados previamente. O quarto cluster corresponde ao objeto mais próximo do RC e diretamente no caminho do mesmo, enquanto que o segundo cluster corresponde ao objeto à esquerda e mais afastado. Verifica-se que as posições estimadas, relativas aos dois objetos, foram calculadas com elevada precisão e exatidão, dado que somando a distância percorrida na inicialização (30 cm) às coordenadas y de cada uma das posições dos objetos, obtêm-se posições muito próximas das posições efetivas reais.

Tabela 11

	posição efetiva		posição calculada		erro absoluto		erro relativo	
	x	y	x	y	x	y	x	y
obstáculo 1	150	140	153	144	3	4	0,020	0,029
obstáculo 2	90	225	86	233	-4	8	0,044	0,036

Nota: Os valores presentes na tabela estão em cm.

Os valores presentes na tabela 11 verificam o que foi dito anteriormente, como tal, os obstáculos foram colocados em posições do mapa adequadas e o sistema planeou o caminho a percorrer em volta do obstáculo que tinha em frente e executou esse caminho.

5.3.2 2ºTESTE

No segundo teste pretendia-se que o primeiro caminho iniciado não fosse possível de executar desde o inicio ao fim, necessitando obrigatoriamente de uma atualização de mapa nalguma parte do percurso para evitar colisões. Para este efeito, usaram-se os mesmos dois objetos utilizados no primeiro teste e juntou-se um terceiro, sendo que desta vez, um dos objetos não era visível na inicialização do algoritmo, porque se encontrava colocado na mesma direção do primeiro obstáculo, apenas mais afastado.

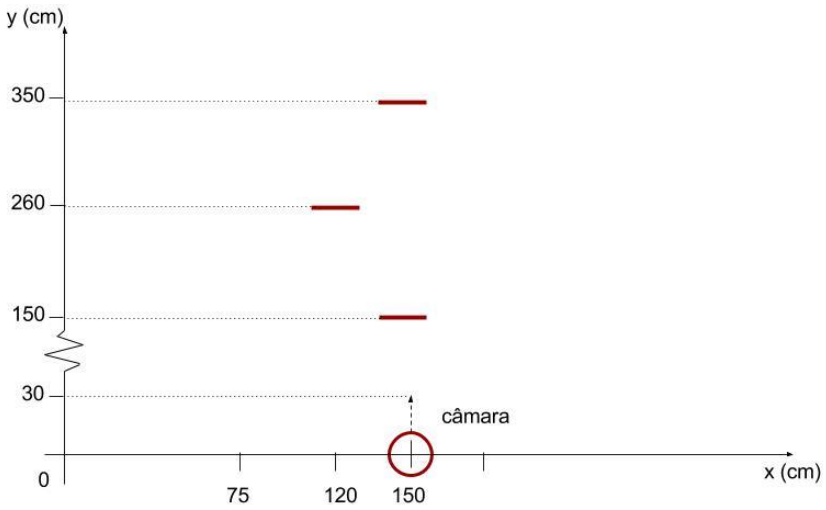

Figura 64 – Disposição do ambiente no segundo teste

Na figura anterior está esquematizado o que foi referido anteriormente, sendo possível verificar que na posição inicial do sistema, o sensor apenas consegue visualizar dois dos obstáculos expostos no ambiente. Assim, o primeiro caminho calculado pelo algoritmo levaria a colisão com o terceiro obstáculo mais afastado da posição inicial depois de evitar os primeiros dois obstáculos.

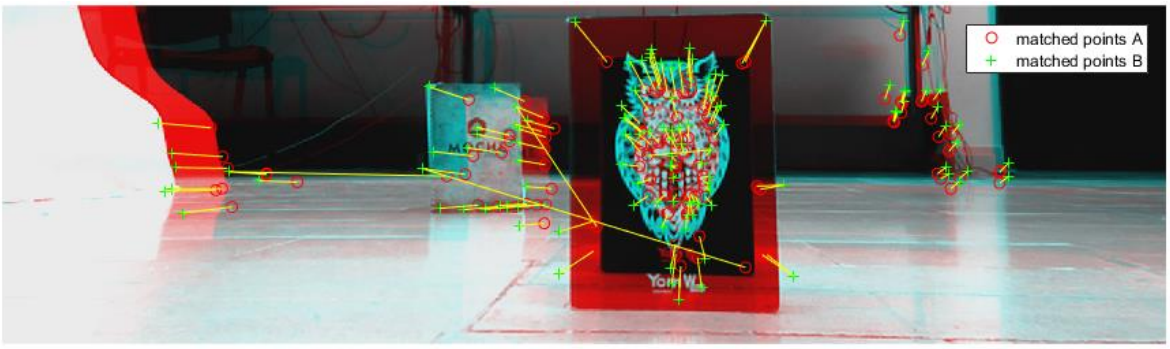

Figura 65 - Emparelhamento de pontos de interesse no 2º teste

Na figura 63 apresenta-se a visão do sistema com as duas primeiras imagens captadas sobrepostas, evidenciando apenas dois dos obstáculos colocados no ambiente em frente ao sistema, mostrando assim a impossibilidade de ver o terceiro obstáculo na inicialização do processo.

Na inicialização do sistema foram criados quatro grupos de pontos, sendo que dois deles pertenciam ao primeiro objeto, diretamente em frente do sistema, um outro ao segundo objeto, à esquerda na imagem, e por último um grupo de pontos pertencentes a pontos de interesse encontrados no fundo do ambiente que não se encontram no mapa.

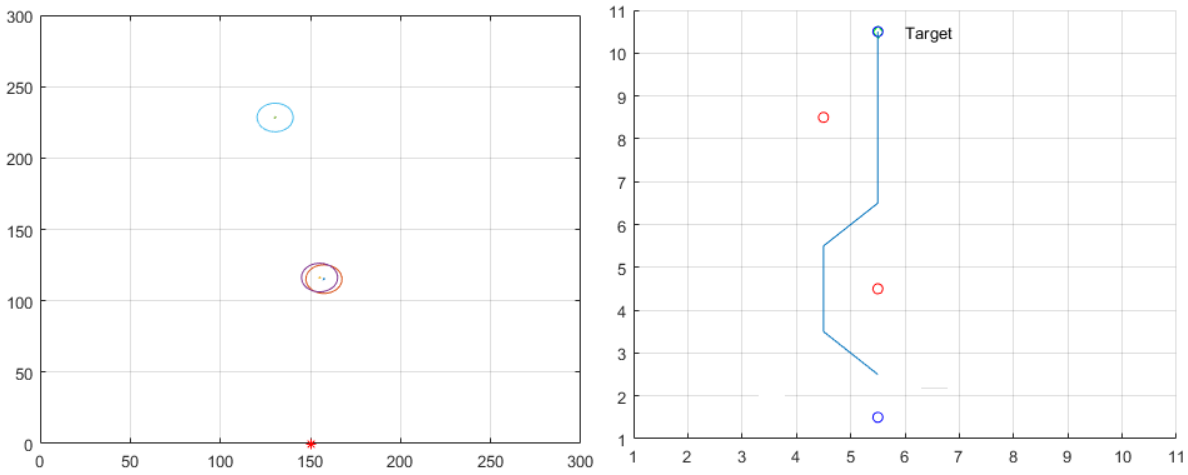

Figura 66 - Posições calculadas de obstáculos (à esquerda) e caminho estimado pelo algoritmo (à direita) na inicialização

No gráfico da esquerda da figura 64 estão esquematizadas as posições dos obstáculos encontrados pelo algoritmo, onde pode verificar-se que existem três posições assinaladas, estando duas das posições muito próximas uma da outra, dado que foram originadas por clusters diferentes de pontos emparelhados, de um mesmo objeto, nas imagens obtidas. No gráfico da direita estão representados os obstáculos encontrados previamente depois de feita a normalização e arredondamento para as posições da malha no mapa, e também o caminho desenhado pelo algoritmo que permite a movimentação do RC no mapa. O sistema contorna os dois primeiros obstáculos executando os comandos do planeamento de rota da figura 64 até atingir a posição (5,8), onde atualiza o mapa e visualiza o terceiro obstáculo.

Figura 67 - Emparelhamento de pontos de interesse no 2º teste (atualização do mapa)

Finalmente o sistema visualiza o último objeto no caminho e executa o algoritmo de forma a contorna-lo. Na figura 67 estão assinalados os pontos de interesse encontrados nas imagens que foram usadas para a atualização do mapa, e neste caso formaram-se três grupos de clusters. Dois deles pertencentes ao objeto que se pretendia identificar e um outro a elementos do fundo do ambiente visualizado.

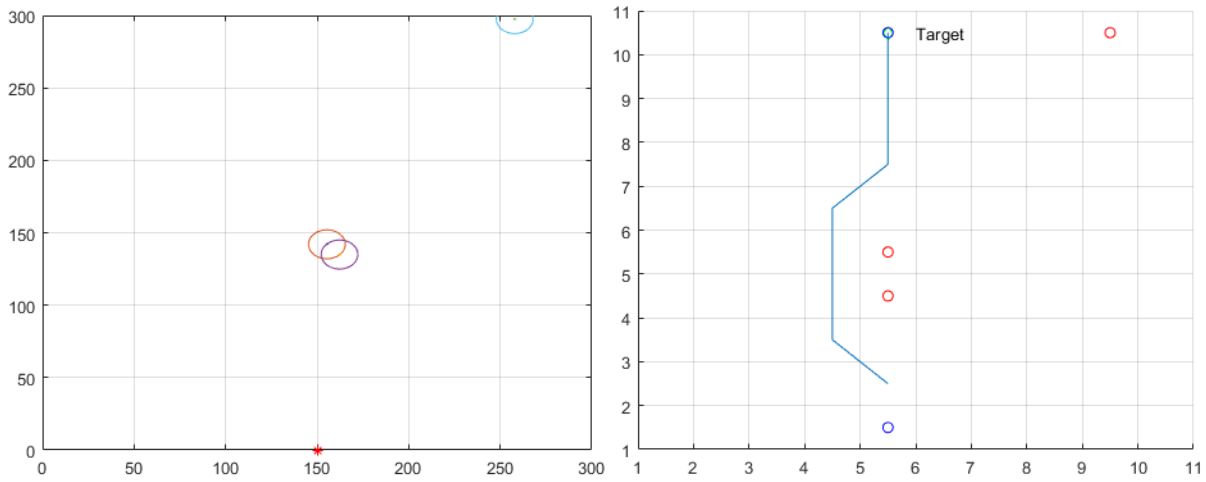

Figura 68 - Posições calculadas de obstáculos (à esquerda) e caminho estimado pelo algoritmo (à direita) na atualização do mapa

É de notar que na atualização do mapa o ponto inicial e ponto objetivo reais do sistema são alterados, passando a posição atual do sistema a ser o ponto inicial que se visualiza no gráfico da direita na figura 68. Analisando os dois gráficos presentes na figura 68, percebe-se que apesar de os dois grupos de pontos, que dão origem aos obstáculos mais próximos um do outro no gráfico da esquerda, serem provenientes do mesmo objeto, dão origem a dois obstáculos diferentes na malha do gráfico da direita. Pode ainda observar-se que, ao contrário do que aconteceu no primeiro teste e na inicialização deste, o cluster de pontos proveniente de pontos pertencentes ao fundo do ambiente das imagens originou um obstáculo que ainda se encontra dentro do mapa.

5.4 RESULTADOS DOS TESTES AO SISTEMA

Nos trinta testes realizados, semelhantes aos testes ilustrados anteriormente, foram colocados obstáculos, em número e posições diferentes, estando sempre presente no ambiente pelo menos um objeto e no máximo três. O deslocamento do sistema foi de cerca de três metros e meio em cada teste.

5.4.1 TEMPOS DE PROCESSOS

Durante os testes realizados, foram retirados tempos aos processos de cálculo de posições de obstáculos e de planeamento de rotas dado que estes são os processos do algoritmo mais pesados computacionalmente. Na tabela seguinte estão representados os tempos médios na totalidade dos testes, em cada processo, consoante o número de clusters identificados.

Tabela 12

Nº de clusters classificados	Nº de vezes verificadas	Cálculo de posição de obstáculos	Planeamento de rota	Total
1	18	1,2	0,09	1,29
2	31	2,3	0,11	2,41
3	29	3,3	0,1	3,4
4	13	4,4	0,12	4,52
5	7	5,4	0,13	5,53

Somando os valores da segunda coluna da tabela anterior chega-se ao número total de vezes que o mapa foi atualizado nos trinta testes realizados (98). Dividindo este valor pelo número total de testes, chega-se a uma proporção média de atualizações do mapa por teste igual a 3.27, o que perfaz um valor de 0.93 atualizações de mapa por metro.

Avaliando as restantes colunas da tabela, é possível concluir-se que o número de clusters identificados pelo processo, como seria de esperar, apenas afeta o tempo despendido a calcular as posições dos obstáculos, aumentando com o aumento do número de clusters identificados, não afetando o tempo de processamento do algoritmo de planeamento de rota.

5.4.2 AVALIAÇÃO DO SISTEMA

Nas 98 atualizações de mapa verificadas nos testes, múltiplas vezes ocorreram problemas que levaram a que o algoritmo de planeamento de rota calculasse um caminho que entrava em rota de colisão com os obstáculos dispostos no ambiente, no entanto, muitas das vezes, os erros nalgumas partes da disposição do mapa que levavam a que o sistema estivesse em rota de colisão, eram dissipados por

uma nova atualização do mapa e correção do que tinha sido calculado anteriormente. Estes problemas derivam de movimentos do sistema não completamente lineares ou *outliers* no método de *clustering*.

Tabela 13

	Posição errada de um obstáculo	**Obstáculo não identificado**	**Obstáculo inexistente identificado**
Número de vezes verificadas	15	7	4
Frequência por teste (%)	50,0	23,3	13,3
Frequência por mapa (%)	21,7	10,1	5,8

Na tabela 13 estão discriminados os problemas referidos anteriormente e que levaram a erros nos cálculos das rotas pelo sistema. Analisando a tabela, a frequência com que um obstáculo não é identificado ou que por outro lado, é identificado um obstáculo inexistente não parece elevada, mas não tendo referência de comparação, nada se pode concluir. Ao contrário das duas colunas à direita, os valores na primeira coluna saltam à vista como aparentemente elevados e proibitivos para que o sistema possa funcionar, no entanto, estes valores podem ser um pouco enganadores, visto que se considerou que uma posição errada de um obstáculo será um desvio em qualquer direção de uma posição na malha do mapa em relação à posição esperada e, como foi referido anteriormente neste trabalho, o algoritmo de cálculo de distâncias aos obstáculos apresenta menor precisão no cálculo a objetos mais afastados. Assim, a frequência com que se verificaram erros no cálculo da posição dos obstáculos é insuficiente para fazer uma avaliação do sistema.

Tabela 14

	Caminhos que levam a colisões	**Colisões verificadas**
Número de vezes verificadas	7	2
Frequência por teste (%)	23,3	6,7
Frequência por mapa (%)	10,1	2,9

Os problemas enumerados na tabela 13, por si só não são elucidativos dos efeitos de mapeamento errado de obstáculos. Na tabela 14 estão descritos os resultados deste mapeamento defeituoso. Desde logo é importante verificar, relacionando as duas tabelas anteriores, que do total de 26 defeitos verificados no mapeamento apenas 7 caminhos calculados resultariam em colisão com algum dos obstáculos, o que corresponde a sensivelmente um caminho em rota de colisão a cada dez atualizações

de mapa. Devido às atualizações do mapa, apenas se verificaram duas colisões ao longo dos 30 testes efetuados, no entanto o objetivo era não existirem colisões, de todo.

Capítulo 6 – Conclusões e Trabalho Futuro

6.1 Conclusões

O objetivo deste trabalho foi desenvolver um sistema de visão monocular que pudesse ser instalado num RC e permitisse ao mesmo movimentar-se autonomamente. Este processo implicava o reconhecimento e estimação de posições de obstáculos para realizar um mapeamento do ambiente. Os métodos já utilizados para a resolução deste tipo de problema baseiam-se maioritariamente na implementação de visão binocular ou introdução de um sensor auxiliar (sensor laser, por exemplo). O método aqui proposto tinha como desafio aproveitar o movimento do sistema global para resolver as ambiguidades no cálculo de profundidades associadas à captação de imagens através de uma câmara só.

Numa primeira fase, este trabalho tem uma forte componente de visão computacional, visto que é sobre a câmara que recai o reconhecimento do espaço envolvente. E numa segunda fase, estuda-se a adaptação de um algoritmo de planeamento de rota a um algoritmo global de funcionamento do sistema, bem como os parâmetros de movimentação do RC.

A ideia principal por detrás do cálculo de distâncias a obstáculos de formas e tamanhos desconhecidos no espaço baseia-se na premissa de que estes obstáculos não alteram essas características ao longo do tempo, e como tal é possível obter relações entre as dimensões dos obstáculos captando imagens após movimento linear da câmara, mesmo sem conhecimento prévio das características dos objetos. Estas diferenças nas dimensões são suficientes, em conjunto com o valor da distância focal da câmara para calcular distâncias desde a câmara até aos obstáculos. Posto isto, abordou-se a forma como obter informação sobre os objetos através da câmara. De maneira a retirar informação de uma imagem e comparar essa mesma informação com outra imagem em sequência recorreu-se a métodos de deteção de características como os métodos *SIFT* e *SURF,* tendo-se implementado no algoritmo do sistema o método *SURF* por apresentar resultados semelhantes ao método *SIFT* e ser um método de processamento mais rápido.

Um dos pontos cruciais, e também mais difíceis de implementar com validade, para o sistema funcionar prende-se com a segmentação dos obstáculos nas imagens captadas. Para esta parte do algoritmo, ao invés de se segmentar cada imagem individualmente, usando a perceção de que objetos mais próximos da câmara apresentam deslocamentos maiores e, objetos mais afastados, deslocamentos menores entre duas imagens, formaram-se vetores que unem os pontos de interesse emparelhados nas imagens, segmentando os vetores. Depois de formados os vetores, utilizou-se um método de

clustering (DBSCAN), que faria então a segmentação, sem que seja necessário dar como input um número definido de grupos. Pala além disso, este método é importante, dado que é robusto ao aparecimento de *outliers,* eliminando assim os vetores formados por pontos emparelhados erradamente.

Depois de calculadas as distâncias aos obstáculos através dos métodos referidos, era necessário recorrer a um método que decidisse que direção tomar a cada momento. Para isto recorreu-se a um algoritmo de planeamento de rota que tivesse em conta o mapeamento dos obstáculos encontrados. Optou-se pelo algoritmo A*, pois trata-se de um algoritmo relativamente rápido, em termos computacionais, e versátil, visto que é se baseia num princípio de *grid-based search*, onde se pode definir o tamanho da grelha a dispor sobre o mapa e os pontos inicial e objetivo. Este método cria um caminho entre o ponto inicial e o ponto objetivo, evitando os obstáculos no mapa. Como se pretendia que o sistema funcionasse autonomamente, os caminhos calculados pelo algoritmo teriam que ser atualizados à medida que o sistema se ia movendo. Assim, definiu-se que à medida que o sistema ia avançando, a câmara ia captando duas imagens, em posições diferentes do mapa, após movimento linear, e o algoritmo criava um novo mapa com os obstáculos visualizados, processando depois um novo caminho a percorrer, reiniciando o ciclo.

Antes de se realizarem os testes ao funcionamento do sistema foram feitos testes de identificação dos parâmetros de movimento do sistema. Para se movimentar na grelha definida no mapa, o sistema apenas podia tomar três orientações diferentes em cada posição e teria no máximo três posições diferentes para onde se mover de seguida, pelo que era necessário perceber a movimentação do RC no espaço. Depois deste processo, teve que adaptar-se o movimento na grelha definida, visto que os movimentos do RC não permitiam atingir as posições e orientações, definidas anteriormente, da maneira como estava definido o mapa.

Por fim foram realizados testes ao sistema. Inicialmente verificou-se a validade dos parâmetros resultantes da identificação do sistema, onde se percebeu que havia algumas discrepâncias entre as posições esperadas e as posições efetivas, e que estas podiam influenciar os cálculos das posições dos obstáculos no sistema global. Para testar o sistema, foram realizados 30 testes, em que se expôs o sistema a vários cenários diferentes. Destes testes verificou-se que em dois deles o sistema colidiu com algum dos obstáculos no seu caminho, devido a uma identificação deficiente do ambiente, o que resultou num caminho definido que levou a colisões.

O objetivo do trabalho era conseguir que o sistema se movimentasse autonomamente evitando colisões. Apesar de acontecerem colisões em número reduzido, verificou-se que em 2.9% das vezes que o mapa é atualizado e uma nova rota planeada, o sistema acaba por colidir com um obstáculo. Assim, conclui-se que o sistema não é efetivo a evitar colisões, e como tal o objetivo não foi completamente conseguido. Contudo, o método introduzido relativamente ao cálculo e posicionamento dos obstáculos no espaço é eficaz, apesar de algo pesado computacionalmente, bem como o método de planeamento de rota, sendo que os maiores problemas para o funcionamento correto do sistema no seu todo prendem-se com o processo de *clustering* e com as pequenas diferenças entre posições esperadas e posições efetivas do sistema.

6.2 TRABALHO FUTURO

Tendo como ponto de partida o trabalho aqui efetuado, e percebendo do último parágrafo das conclusões que os problemas encontrados no funcionamento do sistema se concentram maioritariamente no controlo de movimento do sistema e no processo de segmentação, propõem-se os seguintes pontos de seguimento para trabalhos de melhoramento desta tese:

- Estudo de novas abordagens de segmentação de imagem individual, identificando apenas de seguida pontos de interesse na imagem para os cálculos das distâncias aos obstáculos;
- Possibilidade de introdução de uma câmara que permita a variação da distância focal para conseguir ter perceção de profundidade, facilitando o processo;
- Estudo de novos processos que permitam retirar informação de sequência de imagens, por exemplo registo de imagens;
- Introdução de controlo para a posição do RC no espaço para que os movimentos sejam mais precisos, melhorando a perceção do sistema do ambiente;

Referências

[1] J. Lutin, A. Kornhauser e E. Lerner-Lam, “The Revolutionary Development of Self-Driving Vehicles and Implications for the Transportation Engineering Profession,” *Institute of Transportation Engineers. ITE Journal; Washington,* pp. 28-32, 2013.

[2] M. Bertozzi, A. Broggi e A. Fascioli, “Vision-based intelligent vehicles: State of the art and perspectives,” *Robotics and Autonomous Systems,* vol. 32, pp. 1-16, 2000.

[3] N. Martins, “Integration of RC Vehicles in a Robotic Arena,” MSc Thesis, Mestrado Integrado em Engenharia Aeroespacial, Lisboa, Jan 2017.

[4] “RaspberryPi Camera Module,” [Online]. Available: https://www.raspberrypi.org/products/camera-module/. [Acedido em 16 Fevereiro 2017].

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

[6] A. Saxena, M. Sun e A. Ng, “Make3D: Depth Perception from a Single Still Image,” Chicago, USA, In Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, July 13–17, 2008, pp. 1571-1576.

[7] R. Hartley e A. Zisserman, em *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2004, pp. 237-279.

[8] P. Alizadeh, “Object Distance Measurement Using a Single Camera for Robotic Applications,” Ontario, Canada, 2015.

[9] J. Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1991.

[10] S. Yang e Y. Hu, “Robot Path Planning in Unstructured Environments Using a Knowledge-Based Genetic Algorithm,” School of Engineering, University of Guelph, Canada, 2005.

[11] S. M. LaValle, Planning Algorithms, Cambridge: Cambridge University Press, 2006, pp. 185-248.

[12] S. M. LaValle, Planning algorithms, Cambridge: Cambridge University Press, 2006, pp. 27-74.

[13] H. Bay, T. Tuytelaars e L. Van Gool, “SURF: Speeded Up Robust Features,” Katholike Universiteit Leuven, Zurich.

[14] C. Harris e M. Stephens, A combined corner and edge detector, United Kingdom: The Plessey Company plc., 1988.

[15] [Online]. Available: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html#sift-intro. [Acedido em 18 Janeiro 2017].

[16] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” University of British Columbia, Vancouver, B.C., Canada, 2004.

[17] P. Burt e E. Adelson, “The Laplacian pyramid as a compact image code,” IEEE Transactions on Communications, 1983, p. 532–540.

[18] J. Crowley e R. Stern, “Fast computation of the difference of low pass transform,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 1984, pp. 212-222.

[19] [Online]. Available: http://lijiancheng0614.github.io/scikit-learn/modules/clustering.html. [Acedido em 13 Fevereiro 2017].

[20] M. Ester, H.-P. Kriegel, J. Sander e X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996, pp. 221-231.

[21] “wikipedia\DBSCAN,” [Online]. Available: https://en.wikipedia.org/wiki/DBSCAN. [Acedido em 13 Fevereiro 2017].

[22] P. E. Hart, N. J. Nilsson e B. Raphael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics, pp. 100-107.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest e C. Stein, Introduction to Algorithms, MIT Press and McGraw–Hill, 2001, p. Section 24.3: Dijkstra's algorithm.

[24] “Mathworks,” [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/26248-a---a-star--search-for-path-planning-tutorial. [Acedido em 10 Janeiro 2017].

[25] “LaTrax SST Manual Guide,” [Online]. Available: https://latrax.com/sites/default/files/KC2116-R06-76044-1-SST-Quick-Start-150406.pdf. [Acedido em 26 Março 2017].

[26] P. Premakumar, “https://www.mathworks.com/matlabcentral/fileexchange/26248-a---a-star--search-for-path-planning-tutorial,” [Online]. [Acedido em Março 2017].

[27] S. O. E. Kourosh Khoshelham, “Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications,” 2012.

[28] “Scholarpedia\SIFT,” [Online]. Available: http://www.scholarpedia.org/article/SIFT. [Acedido em 7 Nov 2016].

[29] D. Scharstein e R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *International Journal of Computer Vision,* pp. 7-42, 2002.

[30] J. Ponce e D. Forsyth, em *Computer Vision A modern approach*, pp. 321-345.