FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Demonstrador de Condução Autónoma

José Júlio Areal Ferreira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores - Ramo de Automação

Orientador: Prof. Doutor Armando Jorge Sousa

Fevereiro de 2012

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2011/2012

A Dissertação intitulada

"Demonstrador de Condução Autónoma"

foi aprovada em provas realizadas em 20-02-2012

o júri

Presidente Professor Doutor Luís António Pereira de Meneses Corte-Real
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Agostinho Gil Teixeira Lopes
Investigador Auxiliar Centro Algoritmi - Escola de Engenharia da Universidade do Minho

Professor Doutor Armando Jorge Miranda de Sousa
Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - José Júlio Areal Ferreira

Faculdade de Engenharia da Universidade do Porto

Resumo

A utilização da robótica móvel para fins demonstrativos tem sempre que manter um compromisso entre a componente apelativa, capaz de cativar o público menos especializado, e a vertente pedagógica e reutilizável para fins benéficos para a sociedade. Neste projeto, tendo em consideração a robótica atual, criou-se uma plataforma robótica de locomoção diferencial com componentes para o consumidor geral ('*consumer grade*'), com custos interessantes e capazes de efetuar algumas atividades como desviar-se de obstáculos recorrendo a trajetórias inteligentes, percursos pré-definidos e identificar objetos como esferas e cilindros.

A perceção do mundo neste projeto foi realizada através de visão por profundidade, utilizando o sensor *kinect* da Microsoft e sensores de odometria para localização relativa do robô. O projeto conta ainda com um software de decisão e controlo capaz de receber dados da camada de abstração de hardware, planear tarefas e tomar decisões.

O método de identificação de objetos consiste na interpretação de nuvens de pontos 3D obtidas pelo sensor de profundidade *kinect*. Durante a análise destas nuvens é feita uma pesquisa por características que identificam esferas e cilindros baseados no algoritmo *Gauss-Newton*. Se o objeto em causa não estiver inserido em nenhuma das classes de objetos referidas, marca-se como obstáculo no mapa uma área com as dimensões 2D do objeto.

Após a realização de vários testes para a identificação de objetos obtiveram-se para um conjunto de 100 amostras possíveis valores entre os 77% e os 99% de eficácia enquanto o tempo de convergência dos métodos varia entre 1.05 e 1.65 ms para os cilindros e entre 0.25 e 1.85 ms para as esferas. Valores obtidos utilizando um computador portátil com um CPU (*Central Processing Unit*) Intel i5 de 2.27 GHz.

A localização do robô é feita através da fusão entre dois tipos de dados, os dados provenientes do sistema de odometria e os dados provenientes do sistema de localização por balizas. As balizas são objetos conhecidos e cada um deles encontram-se em posições conhecidas no mundo. Os dois sistemas juntos combinam uma forma precisa de determinar a posição e direção do robô.

Realizaram-se vários testes com diferentes percursos, uns pré-definidos e outros para desvio de obstáculos determinados pelo algoritmo A*. No conjunto global dos testes realizados, através das técnicas de localização referidas anteriormente, obtiveram-se no final dos mesmos para a posição do robô erros máximos de 12 cm enquanto para a direção do robô erros máximos de 8 graus num mapa de dimensões 6 por 4 metros.

Abstract

The use of mobile robotics for demonstration purposes must always maintain a balance between the attractive component, able to captivate an audience less specialized, and a pedagogic purpose, reusable beneficial to society. In this project, taking into account the current robot technology, was created a robotic platform with differential mobility with components for the general consumer ('consumer grade'), with interesting costs and able to perform some attractive activities such as bypass obstacles presents in the field using intelligent routes, predefined routes and identify objects such as spheres and cylinders.

The perception of the world in this project was accomplished through vision by depth using the Microsoft kinect sensor and odometry sensor to obtain relative location of the robot. The project also includes a decision and control software capable of receiving data from the hardware abstraction layer, planning tasks and make decisions.

The object detection method consists in the interpretation of 3D point clouds obtained by the depth sensor Kinect. During the analysis of these clouds is made a search for characteristics that identify spheres and cylinders based on the Gauss-Newton algorithm. If the object in question is not inserted in any of these classes of objects, brand yourself as an obstacle in the map area with the 2D dimensions of the object.

After conducting several tests for the identification of objects were obtained for a sample set of 100 possible values between 77% and 99% efficiency while the convergence time of the methods varies between 1.05 and 1.65 ms for cylinders and between 0.25 and 1.85 ms for the spheres. Values obtained using a laptop computer with an Intel Core i5 processor running at 2.27 GHz.

The location of the robot is done by merging the two data types, data from the odometer system and data from the system location beacons. The beacons are known objects and which each of them is located in known spots in the world. The two systems combine together to determine accurately the position and direction of the robot.

There were made several tests with different routes, some pre-defined and other including obstacle avoidance by using an A* algorithm. In the overall set of tests, using the localization techniques mentioned above were obtained at the end of them to the robot's position, maximum errors of 12 cm while for the direction of the robot maximum errors of 8 degrees on a map of dimensions 6 by 4 meters.

Agradecimentos

Agradeço em primeiro lugar aos meus pais pela possibilidade de frequentar este curso e por todo o apoio e motivação ao longo de todo este projeto. Agradeço todos os conselhos e devo-vos tudo o que sou como pessoa. Agradeço todo o vosso amor e preocupação. Em especial agradeço à minha mãe pela sua orientação, dedicação e presença. À minha irmã também pelo seu apoio e amizade.

Agradeço ao meu orientador Prof. Doutor Armando Jorge Miranda de Sousa por todo apoio e acompanhamento ao longo deste projeto e pela oportunidade de participar por diversas ocasiões em competições de robótica, permitindo alargar os meus horizontes nessa área.

Agradeço também aos meus colegas e amigos Nuno Miguel Baptista dos Santos, Pedro da Silva Machado, Rui Pedro Gomes Ferreira e Pedro Miguel Teixeira pelas valiosas contribuições dadas a este projeto.

Agradeço também ao André Almeida Vidal e ao Héber Miguel Placido Sobreira por estarem disponíveis sempre que precisei.

Agradeço a todos os outros que de alguma forma me ajudaram a atingir o meu objetivo.

Sem todos vocês não teria conseguido.

José Júlio Areal Ferreira

"Many of life's failures are people who did not realize how close they were to success when they gave up."

Thomas A. Edison

Conteúdo

1 Introdução **1**
- 1.1 Motivação . . . 1
- 1.2 Enquadramento . . . 2
- 1.3 Objetivos . . . 2
- 1.4 Plano de trabalho no âmbito do projeto DCA . . . 2
- 1.5 Estrutura da Dissertação . . . 4

2 Estado da Arte **5**
- 2.1 Condução Autónoma de Veículos Terrestres . . . 5
  - 2.1.1 História da Condução Autónoma . . . 5
  - 2.1.2 Competições Nacionais e Internacionais . . . 6
  - 2.1.3 Projetos de Condução Autónoma . . . 10
- 2.2 Tipos de Locomoção . . . 16
  - 2.2.1 Diferencial . . . 16
  - 2.2.2 Ackermann . . . 17
  - 2.2.3 Triciclo . . . 18
  - 2.2.4 Omnidirecional . . . 19
  - 2.2.5 Modelo Síncrono . . . 20

3 Projeto do Demonstrador de Condução Autónoma **21**
- 3.1 Requisitos . . . 21
- 3.2 Arquitetura de Hardware . . . 22
  - 3.2.1 Sistema de locomoção . . . 25
  - 3.2.2 Sistema de Controlo e Acionamento . . . 31
  - 3.2.3 *kinect* . . . 33
  - 3.2.4 Sistema de Alimentação . . . 33
  - 3.2.5 Arduino Mega 2560 . . . 34
- 3.3 Arquitetura de Software . . . 36
  - 3.3.1 Protocolos de Comunicação . . . 36
  - 3.3.2 Linguagens de Programação . . . 40
  - 3.3.3 Software de Decisão e Controlo . . . 41
  - 3.3.4 Software de Interação com o *kinect* (*APPkinect*) . . . 44
  - 3.3.5 Software da Camada de Interface . . . 48

4 Sistema Sensorial para DCA **53**
- 4.1 Sensor *kinect* . . . 53
  - 4.1.1 Principio de Funcionamento . . . 53
  - 4.1.2 Limitações do Sensor . . . 54

4.1.3 *PinHole Model* do Sensor . . . 55
4.1.4 Homografia . . . 56
4.1.5 Conclusões . . . 57
4.2 Odometria . . . 58
4.2.1 Abordagem geral . . . 58
4.2.2 Limitações e Desvantagens do sistema de Odometria . . . 58
4.2.3 Vantagens do sistema de Odometria . . . 59
4.2.4 Modelo de Odometria . . . 59

5 Deteção e Identificação de Objetos **63**
5.1 Deteção de Objetos . . . 63
5.2 Identificação de Objetos . . . 66
5.2.1 Esferas . . . 67
5.2.2 Cilindros . . . 74
5.3 Conclusões do Capítulo . . . 81

6 Localização e Movimentação do Robô **83**
6.1 Localização absoluta por balizas recorrendo a *beacons* presentes no mapa . . . 83
6.2 Seguimento de trajetórias e auto-calibração da posição e direção do robô . . . 86

7 Conclusão **93**

A Controlo de Trajetórias **97**
A.1 *Follow Line* . . . 97
A.2 *Follow Circle* . . . 100

Referências **103**

Lista de Figuras

2.1 DARPA Challenge no deserto Mojave[1] . . . 7
2.2 DARPA Challenge Urbano[1] . . . 7
2.3 Condução Autónoma[2] . . . 8
2.4 Elemento da equipa 5dpo-2000 de MSL da FEUP[4] . . . 10
2.5 Modelo de CAD (*Computer-Aided Design*) de um robô *small* da equipa 5dpo da FEUP[5] . . . 10
2.6 Robot@Factory[6] . . . 10
2.7 FEUPCar 2.0 . . . 12
2.8 Robô Versa . . . 13
2.9 Robôs Atlas2010 à esquerda e AtlasMV3 à direita [14] . . . 15
2.10 Clever Robot . . . 16
2.11 Tração Diferencial [17] . . . 17
2.12 Configuração Ackermann[18] . . . 18
2.13 Trajetória Realizada[7] . . . 18
2.14 Configuração Triciclo [18] . . . 18
2.15 Configuração Omnidirecional [17] . . . 19
2.16 Configuração Síncrona[18] . . . 20

3.1 Estrutura do robô desenvolvido . . . 22
3.2 Arquitetura do Hardware . . . 23
3.3 Vista Frontal . . . 24
3.4 Vista Traseira . . . 24
3.5 Interior . . . 25
3.6 Motor Maxon + Caixa Redutora + *Encoder* . . . 25
3.7 Configuração Planetária [23] . . . 27
3.8 Comparação de binários com caixas de velocidades de diferentes tipos de materiais[25] 29
3.9 Roda Utilizada . . . 30
3.10 Sinal de Quadratura do *Encoder* [27] . . . 30
3.11 *Encoder* MR [28] . . . 31
3.12 Lista de drivers estudados . . . 32
3.13 Driver Motores AMC DZRALTE – 012L80 [29] . . . 33
3.14 *kinect* [31] . . . 33
3.15 Bateria *Diamec* com 12 V e 4.2 Ah . . . 34
3.16 Arduino Mega 2560 [32] . . . 35
3.17 Constituição da mensagem de leitura/escrita a ser enviada pelo mestre [33] . . . 38
3.18 Constituição da mensagem de resposta do escravo [33] . . . 40
3.19 Tabela de interpretação do parâmetro *Control Byte* [33] . . . 40
3.20 Tabela de interpretação do parâmetro *Status* [33] . . . 40

3.21 Arquitetura de Software de decisão e Controlo 42
3.22 Software de decisão e controlo 42
3.23 Parâmetros de configuração do controlador PID do software de decisão e Controlo 43
3.24 Janela de criação de *Logs* do software de decisão e Controlo 44
3.25 Odometria para o software de decisão e Controlo 44
3.26 Fluxograma do funcionamento da aplicação de interação com o *kinect* 46
3.27 Fluxo grama da interação entre o módulo de estado do mundo e o *kinect* 49
3.28 Fluxo grama do funcionamento da aplicação de camada de interface 50

4.1 Principio de funcionamento do *kinect* [35] 54
4.2 Limitação por ocultação de objetos [37] 55
4.3 *PinHole Model* da câmara e do sensor *Depth* [7] 55
4.4 Testes de calibração do sensor *kinect* 57
4.5 Teste do Quadrado com o efeito de cada erro sistemático[16] 60
4.6 Robô diferencial com parâmetros de odometria (Visto de baixo) 61

5.1 Redimensionamento da Imagem *Depth* 64
5.2 Clustering[39] 65
5.3 Objeto 66
5.4 Síntese Ilustrativa do método implementado 67
5.5 Exemplificação dos parâmetros a minimizar 67
5.6 Resultados: a) Imagem RGB do *kinect* ; b) Resultado da Imagem *Depth* ; c) Reconstrução em 3D do ambiente através de pontos obtidos por um único varrimento do *kinect* ; d) Nuvens de pontos bem definidas envolvendo os objetos 71
5.7 Representação dos resultados obtidos para a deteção das esferas 73
5.8 Representação dos resultados obtidos para a deteção das esferas (Zoom In) 74
5.9 Efeito Barril [7] 74
5.10 Divisão do objeto em partes iguais *i* 75
5.11 Circunferência de um nível *i* 75
5.12 Inclinação das Retas que unem os pontos das extremidades das circunferências de cada nível do cilindro *i* 76
5.13 Resultados: a) Imagem RGB do *kinect* ; b) Resultado da Imagem *Depth* ; c) Reconstrução em 3D do ambiente através de pontos obtidos por um único varrimento do *kinect* ; d) Nuvens de pontos bem definidas envolvendo os objectos 77
5.14 Modelo Trigonométrico do Cilindro Utilizado 78
5.15 Representação de um vetor 78
5.16 Representação dos resultados obtidos na deteção dos cilindros 79
5.17 Representação dos resultados obtidos na deteção dos cilindros (Zoom In) 80
5.18 Resultados obtidos para as coordenadas do vetor *c* dos cilindros 81

6.1 Diagrama trigonométrico da localização absoluta por duas balizas 84
6.2 Cenário base em que os testes foram realizados 86
6.3 Legenda referente aos símbolos utilizados nos resultados dos percursos 87
6.4 Percurso boomerang 88
6.5 Resultado final do teste do percurso boomerang (coordenadas em metros) 88
6.6 Resultado final do teste do percurso circular em torno de um objeto (coordenadas em metros) 89
6.7 Realce da zona de aglomerado de pontos (lado esquerdo) e da posição final do robô (lado direito) 90

6.8 Resultado final do teste de desvio de um obstáculo (coordenadas em metros) . . . 91
6.9 Resultado final do teste do mapeamento completo de um objeto desconhecido (coordenadas em metros) . . . 91

A.1 Trajetória Linear . . . 98
A.2 Modelo Trigonométrico no sentido direto . . . 98
A.3 Modelo Trigonométrico no sentido inverso . . . 99
A.4 Controlador Trajetória Linear . . . 100
A.5 Modelo Trigonométrico sentido anti-horário . . . 100
A.6 Modelo Trigonométrico sentido horário . . . 101
A.7 Controlador Trajetória Circular . . . 102

Lista de Tabelas

3.1 Características do Arduino Mega 2560 35

5.1 Posição e raio das Esferas 71
5.2 Resultados Obtidos para a deteção das esferas 72
5.3 Resultados Obtidos para a deteção das esferas (cont.) 73
5.4 Informação dos Cilindros 77
5.5 Coordenadas dos vetores a, b e c 78
5.6 Resultados Obtidos para a deteção dos cilindros 79
5.7 Resultados Obtidos para a deteção dos cilindros (cont.) 80
5.8 Resultados Obtidos para a deteção dos cilindros (cont.) 80

6.1 Informação Objeto Desconhecido 92

Abreviaturas e Símbolos

CAD	*Computer-Aided Design*
CAMBADA	*Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*
CC	Corrente Contínua
COTS	*Commercial off-the-shelf*
CPU	*Central Processing Unit*
DARPA	*Defense Advanced Research Projects Agency*
DC	*Direct Current*
DCA	Demonstrador de Condução Autónoma
DEEC	Departamento de Engenharia Eletrotécnica e de Computadores
EUA	Estados Unidos da América
FEUP	Faculdade de Engenharia da Universidade do Porto
FNR	Festival Nacional de Robótica
GPS	*Global Positioning System*
GPU	*Graphics Processing Unit*
HSV	*Hue, Saturation and Value*
IDE	*Integrated Development Environment*
IP	*Internet Protocol*
IR	*Infra-Red*
LED	*Light Emitting Diode*
MIEEC	Mestrado Integrado em Engenharia Eletrotécnica e de Computadores
MR	*Magneto-Resistive*
PC	*Personal Computer*
PID	*Proportional, Integral and Derivative*
PWM	*Pulse-Width Modulation*
RGB	*Red, Green and Blue*
RMS	*Root Mean Square*
RPM	*Revolutions Per Minute*
RPS	*Revolutions Per Second*
RS-232	*Recommended Standard 232*
SPR	Sociedade Portuguesa de Robótica
TFT	*Thin-Film Transistor*
TMEL	*Tsukuba Mechanical Engineering Lab*
UDP	*User Datagram Protocol*

Capítulo 1

Introdução

1.1 Motivação

Condução Autónoma é a capacidade de reproduzir a complexa tarefa encarregue a um condutor no interior de um veículo, adquirindo dados sensoriais sobre o que o rodeia e tomar certas decisões baseadas na análise desses dados. É uma área de interesse emergente, especialmente no mundo da robótica e da indústria automóvel. Num modo geral, um sistema de condução autónoma tem como objetivo navegar de um ponto para outro sem qualquer intervenção de um operador humano, assegurando todo o tipo de segurança necessária ao veículo assim como a todos os passageiros no seu interior enquanto obedece às leis do trânsito. Esta área de interesse é largamente explorada tanto a nível académico como industrial e está em constante expansão.

Existem várias competições na área da robótica autónoma como as que se encontram no Festival Nacional de Robótica (FNR) organizado pela Sociedade Portuguesa de Robótica (SPR). Estas iniciativas adquirem um papel bastante importante quer pelo apelo visual quer pelo interesse em focalizar a investigação científica ao promover a comparação fácil de soluções numa dada área Cientifica e Tecnológica. Mais à frente neste relatório serão referidas algumas das competições mais mediáticas nesta área tanto a nível nacional como internacional assim como alguns dos seus projetos.

Para que os robôs operem em ambientes não estruturados têm de ser capazes de percecionar o mundo. Nos últimos anos tem-se percorrido um longo caminho, desde os simples sensores de distâncias baseados em sonares ou Infravermelhos ('*Infra-Red, IR*') proporcionando alguns bytes de informação acerca do mundo, passando pelas câmaras convencionais até aos sensores de varrimento laser. Estes últimos têm demonstrado resultados de elevada qualidade na representação 3D do mundo, infelizmente são demasiado caros, estando fora do alcance da maioria dos projetos de robótica. Muito recentemente foram introduzidas no mercado tecnologias de sensores 3D que pode modificar totalmente a forma como os projetos atuais estão a ser desenvolvidos. Por exemplo, o sensor *kinect* para a consola de videojogos Xbox 360 da Microsoft, baseado em tecnologia desenvolvida pela empresa *Prime Sense*, pode ser adquirido por 150 euros, proporcionando tanto nuvens de pontos em tempo real como imagens provenientes da câmara associada. Como

resultado, pode-se esperar que a maioria dos projetos robóticos possa percecionar, futuramente, o mundo em 3D. Tudo o que é preciso neste momento é de mecanismos capazes de lidar eficientemente com essas nuvens de pontos.

O desenvolvimento de uma plataforma robótica robusta e simultaneamente de baixo custo, com manutenção e instalação/utilização simples é um desafio difícil de atingir.

Neste projeto juntamente com esta ferramenta de perceção do mundo, irá ser produzido um estudo dos componentes atuais do mercado (COTS (*Commercial off-the-shelf*)) de modo a desenvolver um robô com componentes modulares e de baixo custo.

1.2 Enquadramento

Este projeto enquadra-se na área da condução autónoma de um veículo terrestre. Pretende-se um veículo capaz de concorrer no FNR e principalmente capaz de fazer demonstrações apelativas num ambiente fácil de montar e testar. O hardware do robô utilizado nesta dissertação foi desenvolvido de raiz, onde todos os equipamentos foram dimensionados com algum detalhe para cumprir os objetivos desta dissertação.

O software utilizado de alto nível utilizado para representação do mundo e decisão e controlo foi baseado num software utilizado num projeto anterior de condução autónoma que contou com a participação do autor desta dissertação. Embora a estrutura da arquitetura não tenha sido implementada durante a realização desta dissertação, foram incluídos vários módulos como um módulo de geração inteligente de trajetórias procurando caminhos mínimos num espaço complexo e módulos de comunicação para os novos equipamentos. No capítulo 3 será feita uma descrição do projeto e das ferramentas utilizadas, onde será abordado o modo de funcionamento da aplicação já desenvolvida.

1.3 Objetivos

De um modo geral, o objetivo principal desta dissertação é construir um robô capaz de efetuar demonstrações apelativas na área da robótica, baseado na experiência de condução de um cidadão comum e tirando também partido de ideias de experiências assimiladas em participações anteriores na competição de condução autónoma do festival nacional de robótica. Mais à frente neste relatório, no capítulo 3, serão colocados em detalhe os requisitos necessários a atingir o objetivo proposto.

1.4 Plano de trabalho no âmbito do projeto DCA

Os trabalhos do projeto Demonstrador de Condução Autónoma (DCA) estão organizados de acordo com o seguinte plano:

- Pesquisa bibliográfica;

- Desenvolvimento do estado da arte, abordando tecnologias utilizadas na robótica atualmente;
- Desenho da estrutura do robô;
- Escolha dos drivers dos motores;
- Escolha do sistema de deteção e identificação de objetos;
- Estudo do funcionamento dos drivers dos motores;
- Estudo do funcionamento do sistema de deteção e identificação de objetos;
- Construção do robô;
- Estudo do funcionamento de sensores de efeito de Hall. Desenvolvimento de um algoritmo de contagem de impulsos, por comutação entre estados de acordo com a sequência de sinais dos sensores de efeito de Hall. Os sensores estão fixos e consoante a posição do rotor, obtém-se combinações diferentes entre esses sinais;
- Após uma avaria nos sensores de efeito de Hall, utilizou-se *encoders*, provocando uma mudança no sistema de deteção de impulsos. Reestruturação do programa para o novo sistema sensorial;
- Teste e calibração do sistema de odometria;
- Identificação da posição e direção do robô recorrendo ao sistema de odometria;
- Desenvolvimento de uma aplicação que detete e identifique objetos;
- Criação de mapas e identificação da posição no mapa dos objetos detetados;
- Planeamento e controlo de vários tipos de trajetórias para navegação. Recorrer a algoritmos de varrimento inteligente do mapa em questão de forma a permitir uma maior inteligência e flexibilidade ao robô;
- Preparar o robô para a competição de Condução Autónoma. Estudo e aplicação do regulamento;
- Participação na competição na de Condução Autónoma;
- Implementar sistema de controlo flexível para vários tipos de trajetos;
- Implementar um sistema de localização absoluta por balizas, através do reconhecimento de objetos presentes no mapa;
- Conjugação dos dados recebidos pelo sistema de localização absoluta e do sistema de odometria de modo a obter uma posição e direção mais precisas do robô;

- Continuação de construção de novos comportamentos demonstradores;
- Testes e recolha de resultados para a deteção e reconhecimento de objetos;
- Testes e recolha de resultados para localização, movimentação e auto-calibração do robô;
- Escrita e apresentação da Dissertação;

1.5 Estrutura da Dissertação

No capítulo 1 introduz-se e contextualiza-se o trabalho realizado, apresentando assim, o enquadramento, os objetivos e o plano do projeto.

No capítulo 2 aborda-se o estado da arte e no capítulo 3 a estrutura do projeto, onde são definidas as arquiteturas de hardware e software bem como os respetivos requisitos. Todos os equipamentos utilizados serão referenciados bem como as linguagens de programação e tipos de comunicação entre as diversas camadas do sistema.

No capítulo 4 é descrito o sistema sensorial do robô. No capítulo 5 são descritos os algoritmos de deteção e identificação de objetos, bem como os resultados obtidos de forma a validar os métodos introduzidos ao longo do capítulo.

No capítulo 6 é feita uma abordagem à localização do robô em relação a *beacons* presentes no mapa durante a movimentação do mesmo. Com este processo de localização é feita a validação do sistema de odometria e a auto-calibração da posição do robô.

Por fim, no capítulo 7 são tiradas conclusões do projeto e são propostas sugestões para trabalhos futuros.

Capítulo 2

Estado da Arte

Neste capítulo é feita inicialmente uma abordagem geral sobre condução autónoma desde as suas primeiras aparições até aos dias de hoje. Depois são referidas algumas competições de sucesso tanto a nível internacional como nacional. É feita também a descrição de alguns projetos na área da condução autónoma portuguesa.

O capítulo termina referindo vários tipos de locomoção robótica e algumas vantagens na utilização de cada um.

2.1 Condução Autónoma de Veículos Terrestres

2.1.1 História da Condução Autónoma

O desenvolvimento de veículos autónomos tem evoluído bastante até aos dias atuais. Os primeiros passos dados nesse sentido requeriam uma combinação especial entre hardware e software. Antigamente a limitação da tecnologia não lhes permitia ter sistemas com respostas rápidas, necessárias para reagir no caso de ambientes desconhecidos e inóspitos. No entanto a contribuição do conhecimento científico ao longo das décadas para sistemas robustos e fiáveis atualmente é claramente um facto de enorme importância.

Para identificar a origem da condução autónoma, tem que se retroceder no tempo até 1977 quando no Japão, o TMEL (*Tsukuba Mechanical Engineering Lab*) construiu o primeiro carro inteligente no mundo. Este carro seguia linhas brancas de uma estrada e conseguia atingir até 30 Km/h.

Entre 1987 e 1995, a comissão Europeia fundou o EUREKA Prometheus Project (EPP), um programa de pesquisa e desenvolvimento dedicado a criar veículos autónomos. Em 1994 na apresentação final do projeto EUREKA em Paris, os veículos autónomos VaMP e VITA-2, desenvolvidos por uma equipa de engenheiros da Universidade Federal das Forças Armadas Alemãs em Munique em colaboração com a Mercedes-Benz, percorreram mais de 1000 km com trânsito caótico e numa autoestrada com três faixas, atingindo a velocidade de 130 km/h.

Estes veículos usaram visão dinâmica para detetar objetos em movimento e foram capazes de evitar e ultrapassar outros carros na via pública.

Os reutilizados e semiautónomos carros da Mercedes eram capazes de controlar a direção, o acelerador e os travões através de um sistema de comando computorizado. O sistema tomava as suas decisões de acordo com a evolução das sequências de imagens em tempo real obtidas pelo sistema de visão, o que requeria alguma intervenção humana.

Um ano mais tarde um modelo da Mercedes-Benz, desenvolvido pela mesma equipa, efetuou um percurso ida e volta desde Munique até Copenhaga excedendo a velocidade de 177 km/h de velocidade de ponta e completando o trajeto com 95% de condução autónoma.

Entre o ano de 1996 e 2001, o projeto italiano ARGO criou veículos que eram capazes de seguir faixas brancas numa autoestrada modificada. Um dos veículos resultantes deste projeto é o modificado Lancia Thema que atingiu uma velocidade média de 90 km/h com 94% de condução autónoma. O veículo era equipado com apenas duas câmaras a preto e branco e usava algoritmos de visão estereoscópica para seguir o trajeto pretendido.

A habilidade impressionante demonstrada na área dos veículos robóticos terrestres despoletou interesse mundial e pesquisa cientifica nessa área, incluindo os projetos "DEMO" da DARPA (*Defense Advanced Research Projects Agency*). Estes projetos focavam-se no desenvolvimento de veículos capazes de navegar por terrenos hostis e providenciaram o conhecimento e experiência inicial no campo da robótica automobilística.[1]

2.1.2 Competições Nacionais e Internacionais

Nesta secção será feita referência a atividades na área da robótica a decorrer anualmente em Portugal e no estrangeiro. Este tipo de eventos visa incentivar os mais jovens para as novas tecnologias, possuindo tanto uma vertente pedagógica como de entretenimento, promovendo também a investigação aplicada.

O festival nacional de robótica é realizado anualmente desde o ano 2001 e conta com a presença de várias competições entre as quais a Condução Autónoma, o Futebol Robótico e o Robot@Factory que serão abordadas nas secções seguintes.

2.1.2.1 DARPA *Grand Challenge*

O DARPA *Grand Challenge* é uma competição criada para incentivar a investigação e o desenvolvimento em veículos autónomos. Este desafio foi criado pela DARPA, uma organização de investigação do departamento de defesa dos EUA (Estados Unidos da América). É no entanto centralizado para a criação e desenvolvimento de tecnologias para fins militares. Para a DARPA, todas as tarefas perigosas deverão ser realizadas por uma máquina em vez de um humano de modo a proteger os soldados e permitir que os esforços humanos sejam empregues de forma mais eficiente. Esta é no entanto a razão que fundamentou todo o investimento efetuado pelo governo dos EUA.

O congresso dos EUA autorizou a DARPA a recompensar monetariamente no valor de 1 milhão de dólares para o vencedor da primeira edição da prova, em 2004 e aumentou para 2 milhões de dólares na edição seguinte, em 2005.

A competição consiste em criar um veículo capaz de efetuar um percurso de forma completamente autónoma e atingir o ponto de destino em tempo mínimo. Foi organizada no deserto Mojave nos EUA (figura 2.1) e o percurso a efetuar tinha 228.5 km de comprimento sem obstáculos na proximidade do trajeto. Na primeira edição nenhum dos participantes terminou a prova e o melhor que foi conseguido foi a distância de 11.84 km. Em 2005 cinco equipas terminaram a prova e dez delas efetuaram o percurso abaixo das 10h. De todas as equipas participantes, apenas uma não conseguiu atingir a melhor pontuação da edição anterior.

Aproveitando o sucesso do *Grand Challenge*, a DARPA organizou um evento chamado *Urban Challenge* em Novembro de 2007 (figura 2.2). O evento requer que as equipas desenvolvam um veículo autónomo capaz de se mover em tráfego congestionado, efetuando manobras complicadas como interseções, cedências de passagem e estacionamentos. Esta foi a primeira competição de condução autónoma em que veículos tivessem que interagir com outros veículos, com ou sem conduto, num ambiente urbano. A viagem envolvia um trajeto de 96 km ao longo de uma área urbana em que os veículos autónomos tinham que obedecer às regras de trânsito, evitar outros carros na estrada que partilhavam a mesma via. De todas as 35 equipas de todo o mundo, apenas 6 conseguiram terminar a prova com uma velocidade média de 20 km/h. De referir que, embora não houvesse nenhum curso predefinido, existia um mapeamento extremamente preciso com cerca de 3000 *way-points* com bastantes *way-points* por trajeto, o que permite o uso extensivo do GPS (*Global Positioning System*) por parte das equipas.

Figura 2.1: DARPA Challenge no deserto Mojave[1]

Figura 2.2: DARPA Challenge Urbano[1]

2.1.2.2 Festival Nacional de Robótica (FNR)

Condução Autónoma

A prova de Condução Autónoma é uma das competições com presença anual no festival nacional de robótica desde a sua primeira edição, que decorreu em 2001, em Guimarães. Esta prova

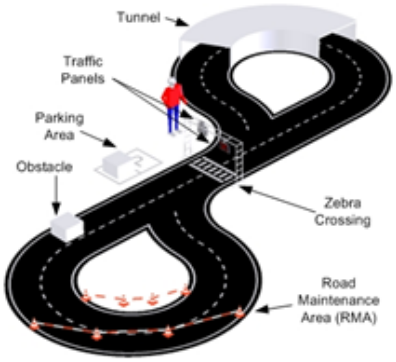


Figura 2.3: Condução Autónoma[2]

representa um desafio técnico no qual um robô móvel e autónomo deve percorrer um percurso ao longo de uma pista fechada, que apresenta semelhanças marcantes com a condução de um veículo automóvel numa estrada convencional.

A pista utilizada tenta reproduzir, em certa medida, um cenário real, embora a competição decorra num ambiente estruturado. A pista, em formato de 8, simula uma estrada com duas vias à qual foram adicionados uma passadeira com um par de painéis semafóricos (um em cada sentido), um túnel, uma zona de obras, um obstáculo e uma área de estacionamento com dois lugares em que um deles está ocupado. A posição do obstáculo na pista, a localização exata da área de estacionamento e a posição livre nessa área são dados desconhecidos para o robô no início da prova.

A competição desenvolve-se em três fases, realizadas em três dias consecutivos, com um aumento progressivo da complexidade efetuado através da adição de novos desafios. Em todas as 3 fases os robôs partem da passadeira após o reconhecimento do sinal "seguir em frente"exibido no painel semafórico e evoluem autonomamente na pista executando duas voltas completas.

Além da identificação do sinal exibido pelo painel semafórico, a primeira fase requer apenas o controlo do movimento do robô ao longo do percurso. O robô deverá executar duas voltas completas à pista o mais depressa possível.

A segunda fase exige que o robô seja capaz de identificar um de 5 sinais diferentes exibidos pelo painel semafórico e que reaja em conformidade. Os sinais, mostrados através de um ecrã TFT (*Thin-Film Transistor*) de 17", podem indicar que o robô deve parar, seguir em um de dois sentidos (virar à esquerda ou ir em frente), que a sua prova terminou, ou que deve iniciar a manobra de estacionamento. Na segunda fase os robôs também têm que lidar com um obstáculo, que ocupa uma das faixas, e que está localizado numa posição desconhecida. O obstáculo deve ser detetado e o robô deve evitá-lo seguindo pela outra faixa, mas sem sair da pista.

Finalmente, na terceira fase são adicionados mais dois problemas: um túnel que cobre uma parte do caminho e uma zona de pista não estruturada designada por zona de obras. O túnel influencia significativamente as condições de luz, o que tem como consequência uma alteração do modo como o robô navega nessa zona da pista. A zona de obras é um desvio da trajetória inicial

que é desconhecido a priori. O novo percurso é marcado através de cones coloridos (laranja e branco semelhantes aos utilizados nas estradas, mas de menor dimensão), unidos através de uma fita de plástico com listas vermelhas e brancas. Nesta zona, o robô deve deixar a faixa inicial e seguir pelo novo caminho sem tocar em qualquer dos elementos que o delimita, e reentrar na pista onde a zona de obras termina.

O desafio referido até agora é referente à prova de condução autónoma classe *Challenge*, pois existe ainda uma versão mais simplificada da anterior (classe *Roockie*) com apenas duas mangas que teve início na edição de 2011 com vista à participação dos mais jovens[2].

Futebol Robótico

A liga de robôs médios, Middle Size League (MSL) em inglês, é uma liga oficial do RoboCup. Duas equipas com 5-6 robots completamente autónomos, cujas dimensões vão até 80 cm de altura, 50 cm de diâmetro e 40 Kg de peso, defrontam-se num campo semelhante ao de futebol de 11 humano, mas com um tamanho mais reduzido (18 m x 12 m)[3]. Esta modalidade é uma das que desperta grande atenção devido à sua elevada multidisciplinaridade[3] pois permite a conjugação de diversas áreas cientificas como a eletrónica e processadores, visão por computador, processamento de imagem, sensores e atuadores, navegação e controlo em tempo-real, inteligência artificial, sistemas distribuídos e cooperativos, telecomunicações, sistemas de locomoção, etc. A investigação destas áreas científicas permite a aplicação num vasto conjunto de problemas de interesse social e económico.

Em Portugal existem bastantes equipas a participar nesta modalidade tanto a nível nacional como internacional. A equipa CAMBADA (*Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*) da Universidade de Aveiro tem-se evidenciado nestes últimos anos tendo ganho vários prémios como o penta campeonato nacional e o primeiro lugar no campeonato mundial de 2008 realizado na China. A FEUP (Faculdade de Engenharia da Universidade do Porto) apresenta uma equipa também bastante competitiva (5DPO-2000, figura 2.4) adquirindo, desde o seu início, alguns prémios a nível nacional e dois pódios no campeonato europeu.

Além da FEUP e da Universidade de Aveiro, outras equipas têm marcado presença na mais alta competição desta modalidade como o ISocRob do Instituto Superior Técnico de Lisboa (IST), Minho Team da Universidade do Minho e ISEPorto Team do Instituto Superior do Porto (ISEP).

No entanto a FEUP tem revelado excelente notoriedade na liga de robôs pequenos (figura 2.5), também liga oficial da RoboCup, com dois pódios de 2º e 3º lugar no campeonato do mundo de 2006 e 1998, respetivamente. Ainda nesta modalidade foram campeões europeus por três vezes nas edições 2001, 2006 e 2007 do RoboCup.

As atuações Portuguesas fora de portas não ficam por aqui pois na liga de simulação 2D e 3D, outra competição oficial do RoboCup, a FC Portugal, equipa formada por elementos da FEUP e da Universidade de Aveiro arrecadaram inúmeros prémios como dois campeonatos mundiais em 2000 e 2006 e dois campeonatos europeus em 2001 e 2006.

Figura 2.4: Elemento da equipa 5dpo-2000 de MSL da FEUP[4]

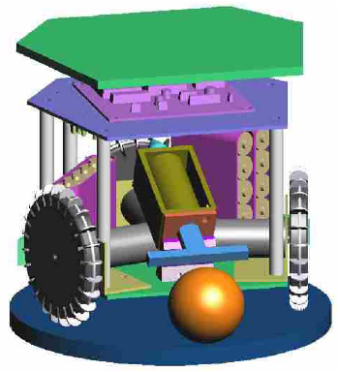


Figura 2.5: Modelo de CAD (*Computer-Aided Design*) de um robô *small* da equipa 5dpo da FEUP[5]

Robot@Factory

Nesta competição procura-se recriar um problema inspirado nos desafios que um robô autónomo terá de enfrentar durante a sua utilização numa fábrica. Um ou mais robôs deverão ser capazes de transportar material entre armazéns e máquinas que operam sobre esse material. Os robôs deverão apresentar um mínimo de capacidades que incluem recolher, transportar e posicionar os materiais, localizar-se e navegar no ambiente fornecido assim como evitar choques com paredes, obstáculos e outros robôs[6].

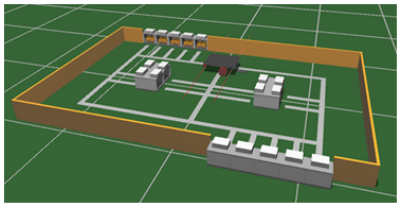


Figura 2.6: Robot@Factory[6]

2.1.3 Projetos de Condução Autónoma

2.1.3.1 FEUPCar 2.0

O FEUPCar 2.0 é um projeto de final do curso de MIEEC (Mestrado Integrado em Engenharia Eletrotécnica e de Computadores) realizado por André Almeida Vidal. Este projeto consiste num robô de locomoção ackermann com vista à participação no concurso nacional de robótica na competição de condução autónoma.

É um robô equipado com três câmaras, duas para detetar as linhas laterais da pista e outra para detetar e identificar sinais de trânsito. Possuí ainda sensores de odometria que permitem, além

de serem utilizados para fechar a malha de velocidade do motor, ser utilizados para localização relativa do robô e assim auxiliar o sistema de visão de forma a determinar a posição do robô no mundo.

As duas câmaras que identificam a pista tentam reproduzir a perceção humana, permitindo uma visão bastante alargada do mundo. Para tal foi desenvolvido um algoritmo que permita a sincronização das câmaras.

Após a câmara da esquerda estar em sincronia com a da direita, as respetivas imagens provenientes são analisadas de forma a isolar as linhas da pista e consequentemente determinar a distância relativa a cada uma dessas linhas. A localização relativa do robô é determinada de acordo com a informação extraída dessas linhas.

Deteção das linhas da pista

O procedimento de deteção de linhas é a parte mais importante do sistema de visão deste projeto. Este procedimento informa o sistema de navegação da posição do robô na pista de modo a poder atuar em conformidade. Devido a um efeito de perspetiva as linhas da pista tendem a convergir no horizonte, no entanto o mais importante é verificar o comportamento dessas linhas nos pontos mais próximos do robô. O algoritmo inicia-se com uma tarefa de pré-processamento, que filtra a imagem proveniente da fusão entre as duas câmaras de modo a extrair as zonas mais importantes para serem analisadas. Depois as linhas laterais são separadas em duas imagens distintas de forma a determinar o vetor que as define. Baseado nesse vetor e na imagem proveniente do estado de pré-processamento é possível isolar a região que contém a linha tracejada central, uma vez que as linhas são todas paralelas.

Deteção de sinais de trânsito num ecrã TFT

Os sinais de trânsito a serem detetáveis no projeto FEUPCar estão representados num TFT retangular. Na imagem no entanto irá aparecer ligeiramente distorcido, tomando a forma de um trapézio devido ao efeito de perspetiva. O primeiro passo efetuado consiste em analisar separadamente cada plano de cor, para isso a imagem foi dividida em três imagens uma para a cor vermelha, uma para a cor verde e outra para a azul. Para cada plano de cor é aplicado um threshold e em seguida um procedimento de dilatação de forma a eliminar possíveis descontinuidades. Em seguida é aplicado um procedimento de procura de contornos, que serão posteriormente aproximados por formas poligonais. O resultado da aproximação poligonal é um vetor contendo todos os polígonos detetados com o respetivo número de arestas e os pontos de cada vértice. Após a procura de polígonos é feita uma filtragem rigorosa tendo em conta a área do polígono que se pretende encontrar bem como o número das suas arestas e a sua convexidade. No espaço Euclidiano um objeto é convexo se para cada par de pontos no interior do objeto, cada ponto no segmento de linha que os une encontra-se também no interior do objeto [7]. Um objeto quadrado verifica esta propriedade o que permite colocar de parte grande parte da informação recebida. Após encontrar polígonos que se encaixem nas características pretendidas é aplicado um processo verificação de semelhança

com um padrão guardado em memória, que não é mais que uma imagem do sinal pretendido. Esse padrão é então ajustado ao polígono detetado e obtido o seu coeficiente de semelhança.

Para a deteção da passadeira é aplicado inicialmente um threshold adaptativo com um valor de offset alto. É seguido então de uma operação de aproximação para preservar a convexidade da forma e aplicado então um filtro de erosão de modo a reduzir partículas de ruído e evitar que sejam responsáveis pela união de duas formas. Após a filtragem da imagem segue-se o mesmo processo anterior para deteção de contornos e aproximação poligonal dos tracejados da passadeira. Os parâmetros de procura serão o número de arestas, a área, a convexidade e a distância que estão a ser detetadas.

Figura 2.7: FEUPCar 2.0

2.1.3.2 VERSA Robot

O robô VERSA (figura 2.8) foi um robô da FEUP que participou na competição nacional de robótica em 2005. A plataforma robótica apresenta uma configuração diferencial e está equipada com um sistema de visão composto por duas câmaras, uma para detetar as linhas no solo e outra para detetar semáforos e sinais de trânsito, sensores de codificação para medir a distância percorrida por cada roda e sensores de ultra-som distribuídos pela frente e laterais do robô para detetar obstáculos.

O sistema de processamento de imagem inicia-se com uma binarização da imagem e analisa-se o seu histograma de luminosidade para determinar o ponto de *threshold*. De forma a detetar os objetos presentes na imagem aplica-se um filtro de deteção de transições à imagem binarizada. Após localizar os objetos através do "algoritmo do pixel vizinho"e analisando os momentos dos objetos, guarda-se num array multi-dimensional os dados de cada objeto como o seu identificador, as coordenadas do seu centro de massa, a sua área e a sua orientação.

Depois de recolher os dados dos objetos presentes na imagem procede-se à análise da sua forma. O parâmetro que distingue a faixa de rodagem da linha da passadeira é a orientação do objeto, uma vez que a linha da passadeira tem a característica única de ter uma inclinação perpendicular à orientação do robô [8]. Para distinguir a faixa central tracejada da lateral, utiliza-se a informação do centro de massa e da orientação do objeto.

A posição absoluta do robô é determinada através da combinação do sistema de odometria com a medida da distância a uma das faixas detetadas.

A determinação do estado dos sinais de trânsito é igualmente feita através da análise dos objetos detetados. As características dos objetos encontrados são depois comparadas com características dos sinais que se pretende detetar para determinar qual a sua correspondência.

O sistema de deteção de obstáculos consiste na informação do sensor de ultra-som frontal que faz com que o robô comute de faixa. O retorno à faixa principal é feito de acordo com a informação dos sensores laterais. Se estes deixarem de detetar qualquer objeto, o robô retorna ao seu trajeto normal.

Figura 2.8: Robô Versa

2.1.3.3 ATLAS Robot

O projeto ATLAS também deve ser mencionado neste contexto. A sua primeira aparência foi em 2003, conseguindo o 4º lugar na competição. O sistema de visão usava uma webcam que adquiria imagens através de um espelho, de modo a obter uma visualização de toda a pista. Desde essa altura foi continuamente improvisando, chegando a um excelente 2º lugar na edição de 2005 com o robô Atlas III. A abordagem inicial era baseada num veículo de arquitetura Ackermann, em que o seu controlo era baseado puramente em análise de imagens. Em 2006 e 2007, o robô Atlas atingiu o 1º lugar na competição de condução autónoma. Na edição de 2008, a equipa introduziu um novo robô na competição, o AtlasMV, criado para ser mais pequeno, mais leve e mais rápido que o anterior. Na edição de 2009, este novo robô, esteve equipado com um laser detetor de distâncias para facilitar a tarefa de detetar e desviar de obstáculos. O AtlasMV é equipado com uma arquitetura de software distribuída, em que múltiplos programas ficam responsáveis por múltiplas tarefas[9].

Ao longo das várias edições da FNR, as abordagens usadas no projeto ATLAS foram alteradas e improvisadas. Os robôs começaram com a utilização de apenas uma câmara, que foram mais tarde substituídas por duas câmaras com maiores ângulos de abertura de modo a obter uma visão mais alargada da pista. As imagens recebidas das duas câmaras foram intersetadas numa só imagem através de várias transformações. Nesta abordagem, nem os parâmetros de modelação das

lentes nem as transformações de perspetiva foram utilizadas, devido aos custos computacionais associados. Em vez disso, foi utilizada calibração manual para os parâmetros de distorção para cada câmara, resultando numa imagem sem precisão geométrica nem consistência. Contudo, em termos de navegação, foram obtidos bons resultados[10],[11]. A abordagem usada mais recentemente, utiliza uma plataforma multi-câmaras montada numa unidade vertical e noutra horizontal, para permitir uma perceção do mundo mais eficiente. Nesta solução, todas as câmaras no sistema são corrigidas através de parâmetros de distorção pré-determinados. Baseado no modelo de cinemática da plataforma que incorpora quatro câmaras, e da transformação da perspetiva da câmara, cada ponto da imagem é mapeado num ponto do mundo real[12].

No que diz respeito à perceção do mundo, dois algoritmos principais foram desenvolvidos para a segmentação da estrada. A primeira leva vantagem da homogeneidade da estrada, ou seja, a conectividade entre as extremidades das linhas. Neste caso, uma linha horizontal virtual é demarcada definindo, desta maneira, a área permitida para a navegação. Esta abordagem permite também a facilidade em detetar a região de curvatura na estrada[11]. Outra abordagem foi desenvolvida, sendo esta uma solução mais precisa e robusta usada nos robôs mais recentes, adquirindo vantagem de sistemas integrados de multi-câmaras. A imagem analisada tirada de uma perspetiva do topo da estrada, obtendo através de transformações de perspetiva e da disposição das multi-câmaras. O algoritmo efetua a pesquisa por linhas, obtendo indicadores estatísticos que são comparados com um modelo para inferir a presença da linha atual[13].

A respeito da deteção da passadeira, o algoritmo consiste em efetuar uma pesquisa por padrões de passadeira similares, ou apenas algumas partes relevantes. Esta pesquisa está restringida pelas linhas principais da pista. Com a primeira disposição da câmara, a deteção era baseada na análise de objetos binários e posteriormente a computação de fator área/perímetro para inferir a presença da passadeira. Com a disposição por multi-câmaras, a imagem é correlacionada com o padrão da passadeira pré-obtido. De modo a evitar as limitações do método de deteção de semelhanças a um padrão conhecido (Matching), como por exemplo sensibilidade a escalas dos padrões e rotação das imagens, procede-se à rotação do padrão da passadeira de acordo com a posição do robô. O ângulo é obtido através da determinação do ângulo de uma das linhas relativamente ao fundo da imagem.

Para a deteção de obstáculos, a abordagem por visão foi usada no primeiro robô assumindo que os obstáculos eram parte integrante da linha, desde que fossem pintados de branco, como na edição de 2006. Contudo, os obstáculos foram mudados para caixas verdes nas edições seguintes. Este facto foi resolvido utilizando filtros de cor HSV (*Hue, Saturation and Value*) e mais tarde computação de centros geométricos para determinar a posição dos objetos[11].

A deteção de sinais de trânsito é feita através da sua cor e forma, através da conversão da imagem em componentes HSV. Posteriormente, o resultante objeto binário é avaliado baseado na região do centro geométrico e no retângulo envolvente.

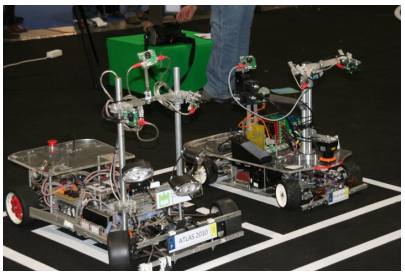


Figura 2.9: Robôs Atlas2010 à esquerda e AtlasMV3 à direita [14]

2.1.3.4 Clever Robot

Este projeto foi realizado no âmbito de uma dissertação de mestrado integrado de Héber Sobreira e tem como objetivo o desenvolvimento de um robô para fins publicitários. Para atingir esse objetivo o projeto apresenta uma plataforma móvel, autónoma, de baixo custo e de simples utilização[15].

O hardware do projeto é composto por um robô de locomoção diferencial munido de sensores de ultra-som para detetar presença de degraus e obstáculos (embora não estivessem ainda montados no robô aquando da apresentação do projeto), *encoders* para determinar a distância percorrida por cada roda e fechar a malha de velocidade dos motores e por fim um recetor infra-vermelho para detetar sinais codificados por frequência provenientes de duas balizas equidistantes externas ao robô, sistema que lhe irá permitir navegar numa área limitada pelo alcance dessas balizas. Pretende-se que este projeto seja um sistema de chave-na-mão na ótica do utilizador, sem necessidade de recalibrações sempre que o hardware mude de lugar.

A posição do robô no mundo é computada através do ângulo entre os segmentos que unem o robô a cada uma das balizas e a distância a cada uma delas[15]. No entanto o sistema de deteção de balizas implementado no robô apenas consegue obter o ângulo em relação a cada uma das balizas e apenas uma de cada vez, sendo que o robô para obter a medida relativa à outra baliza necessita de efetuar uma rotação de 360º sobre si próprio até a conseguir encontrar. A distância às balizas é determinada de forma indireta recorrendo à odometria[15].

Trata-se de uma topologia bastante dependente de um sistema de localização relativa o que leva a uma necessidade acrescida de uma calibração robusta de todos os parâmetros envolvidos. Para tal introduziu-se o método de calibração UMBMark[16] para redução dos erros sistemáticos ao longo de um determinado percurso e os resultados foram bastante satisfatórios.

Foi também introduzido um modelo estocástico do erro da odometria para lidar problemas não-sistemáticos como o resvalamento das rodas (piso escorregadio, aceleração excessiva, forças externas), por pavimentos irregulares ou devido a objetos inesperados no chão.

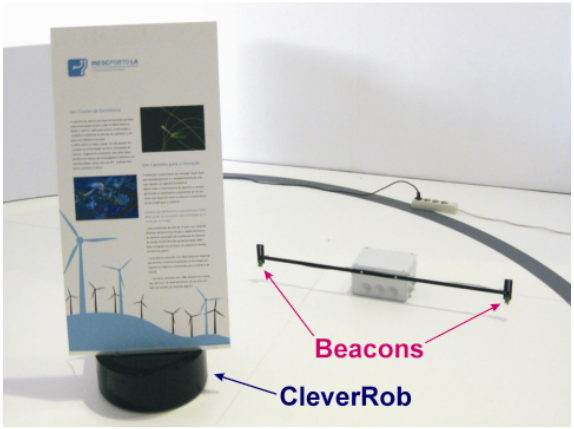


Figura 2.10: Clever Robot

2.2 Tipos de Locomoção

Para a escolha do tipo de locomoção a utilizar é necessário ter em conta vários aspetos, entre os quais:

- Capacidade de Manobra: Facilidade em mudar de direção;
- Controlabilidade: Hardware e Software utilizados para o controlo;
- Tração: aderência suficiente para evitar a derrapagem das rodas;
- Suportar irregularidades do piso;
- Estabilidade;
- Eficiência Energética;
- Manutenção;[17]

2.2.1 Diferencial

O modelo de tração de um robô diferencial consiste em duas rodas independentes responsáveis pela tração do veículo e uma roda extra livre para apoio da estrutura. Esta geometria apresenta um modelo cinemático extremamente simples uma vez que em condições ideais se as duas rodas motrizes forem idênticas, através do conhecimento das velocidades lineares de cada roda e da distância entre rodas (b) é possível determinar a posição (X_R,Y_R) e orientação (θ) do robô num espaço definido por X e Y, como se pode verificar pela figura 2.11 e pelas equações de cinemática 2.1, 2.2 e 2.3.

Este tipo de configuração tem uma excelente capacidade de manobra o que permite solucionar problemas simples num ambiente mais obstruído, possibilitando a rotação sobre si próprio. No entanto não é possível efetuar movimentos de translação segundo o eixo que passa pelos veios

dos motores, diferenciando desta forma da geometria holonômica como é o caso da locomoção omnidirecional e síncrona.

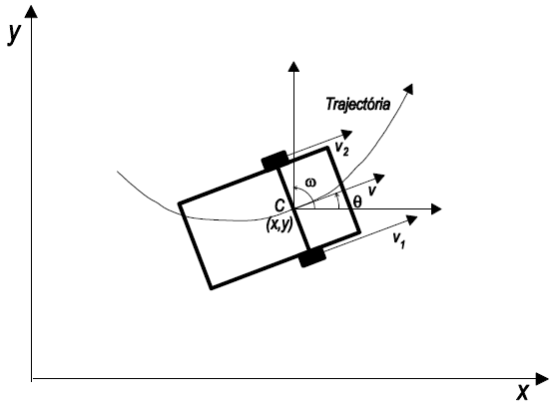


Figura 2.11: Tração Diferencial [17]

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ \Theta(t) \end{bmatrix} = \begin{bmatrix} \cos(\Theta(t)) & 0 \\ \sin(\Theta(t)) & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{2.1}$$

$$v(t) = \frac{v_1(t) + v_2(t)}{2} [17] \tag{2.2}$$

$$\omega(t) = \frac{v_1(t) - v_2(t)}{b} [17] \tag{2.3}$$

As velocidades linear e angular estão representadas por v e ω respetivamente. Na figura 2.11 o robô efetua uma trajetória no sentido positivo (ω>0), como se pode verificar pelo tamanho das setas que indicam o valor absoluto da velocidade de cada roda (v_1 e v_2). Pela equação 2.3, conclui-se que se v_1>v_2, como é o caso, ω será positivo. A velocidade linear v, pela equação 2.2 terá o valor médio das duas velocidades lineares.

2.2.2 Ackermann

Esta configuração é semelhante a um carro com quatro rodas. As rodas de trás são responsáveis pela tração do veículo e as duas da frente pela direção. A roda direcional interior apresenta um ângulo superior à exterior e percorre uma menor distância (circunferência mais pequena da figura 2.13). Esta geometria das rodas assegura que qualquer que seja o ângulo de direção

(*steering angle*), todos os centros das circunferências de cada eixo vão intersetar-se num ponto em comum, fazendo com que o robô realize uma curvatura correta como ilustrado na figura 2.13.

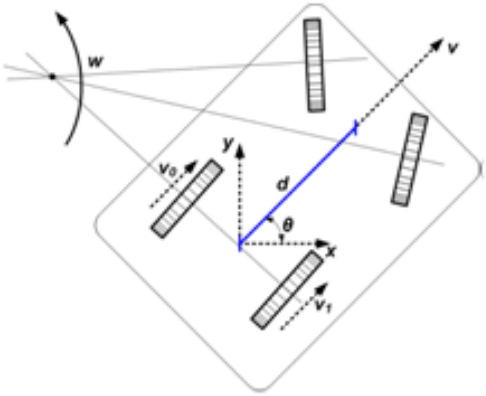


Figura 2.12: Configuração Ackermann[18]

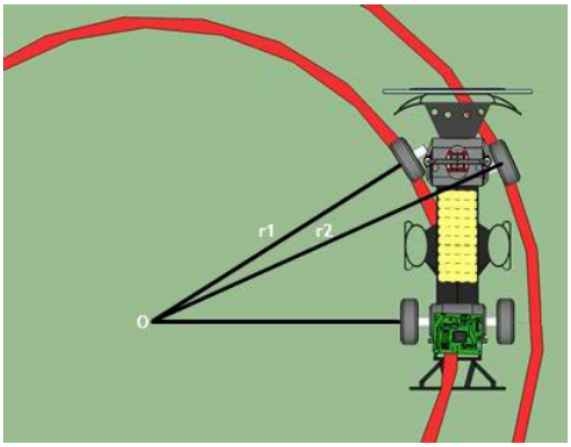


Figura 2.13: Trajetória Realizada[7]

2.2.3 Triciclo

Esta configuração é menos estável quando comparada com a configuração Ackermann. A grande vantagem competitiva é o facto de ser mecanicamente mais simples devido a possuir menos uma roda. Uma das topologias possíveis apresenta apenas uma roda responsável pela tração e direção como se pode ver pela figura 2.14[18]. O modelo cinemático desta configuração está representado pelas expressões 2.4, 2.5 e 2.6.

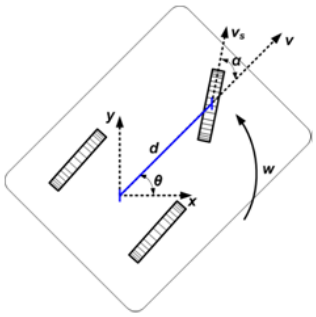


Figura 2.14: Configuração Triciclo [18]

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ \Theta(t) \end{bmatrix} = \begin{bmatrix} \cos(\Theta(t)) & 0 \\ \sin(\Theta(t)) & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{2.4}$$

$$v(t) = v_S(t)\cos\alpha(t)[19] \tag{2.5}$$

$$\omega(t) = \frac{v_S}{d}\sin\alpha(t)[19] \tag{2.6}$$

2.2.4 Omnidirecional

Os robôs de locomoção omnidirecional apresentam uma mobilidade completa no plano, sem necessidade de se orientar quando se pretende deslocar para um determinado ponto. Este conceito é amplamente explorado quando a aplicação robótica está limitada a pequenos espaços e a sua manobrabilidade é bastante reduzida, como por exemplo as cadeiras de rodas. Outra utilidade é a sua aplicação na competição de futebol robótico de robôs. Neste caso o robô é capaz de preparar o remate com a direção pretendida ao mesmo tempo que se está a deslocar para a bola.

A construção de um robô omnidirecional de três rodas requer uma complexidade e precisão elevadas de forma a não prejudicar o seu modelo cinemático.

O modelo cinemático do robô pode ser expresso pelas equações 2.7 e 2.8. Em que v_0, v_1 e v_2 são as velocidades lineares das três rodas do robô que podem ser obtidas, e vice-versa, através da velocidade linear (v), normal (v_n) e angular (w) do robô. O parâmetro d é a distância do centro de cada roda ao centro do robô.

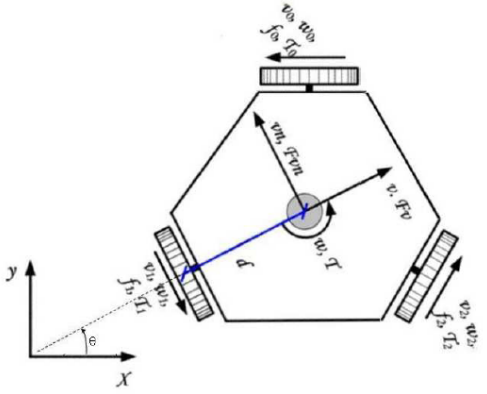


Figura 2.15: Configuração Omnidirecional [17]

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta(t)) & \sin(\theta(t)) & 0 \\ -\sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} * \begin{bmatrix} v(t) \\ v_n(t) \\ w(t) \end{bmatrix} [17] \tag{2.7}$$

$$\begin{bmatrix} v(t) \\ v_n(t) \\ w(t) \end{bmatrix} = \begin{bmatrix} -\frac{\sqrt{3}}{3} & 0 & \frac{\sqrt{3}}{3} \\ \frac{1}{3} & -\frac{2}{3} & \frac{1}{3} \\ \frac{1}{3d} & \frac{1}{3d} & \frac{1}{3d} \end{bmatrix} * \begin{bmatrix} v_0(t) \\ v_1(t) \\ v_2(t) \end{bmatrix} [17] \tag{2.8}$$

2.2.5 Modelo Síncrono

Neste tipo de configuração cada roda tem a capacidade de ser controlada e direcionada, apresentando um movimento sincronizado. A topologia típica para este tipo de configuração passa pelo uso de três rodas dispostas sobre um triângulo equilátero, ver figura 2.16[18].

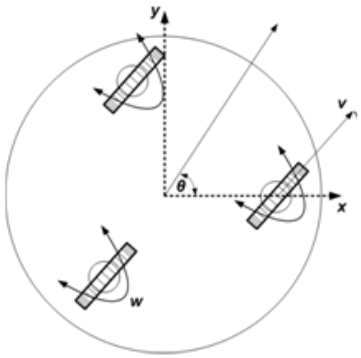


Figura 2.16: Configuração Síncrona[18]

As características mais relevantes desta configuração são:

- Todas as rodas são motoras e direcionais;
- Um conjunto de correias faz girar as rodas sincronizadamente e à mesma taxa;
- Um outro conjunto de correias dá tração às rodas;
- Pode mover-se linearmente em qualquer direção após rotação das rodas.

Capítulo 3

Projeto do Demonstrador de Condução Autónoma

Depois do estudo apresentado no capítulo anterior, procedeu-se ao desenvolvimento e montagem do robô.

Na secção 3.1 serão apresentados os requisitos gerais do sistema. Na secção 3.2 será apresentado o hardware utilizado e a sua arquitetura de funcionamento. Na secção 3.3 irá apresentar-se o software desenvolvido, as linguagens de programação, o tipo de comunicações entre as diversas camadas bem como a natureza da informação que flui entre elas.

3.1 Requisitos

O presente trabalho foi desenvolvido no âmbito de um projeto designado por Demonstrador de Condução Autónoma. De modo a cumprir os objetivos propostos no capítulo 1.3 será necessário cumprir o seguinte conjunto de requisitos:

- Plataforma móvel;
- Custo reduzido;
- Simples utilização;
- Estar munido de equipamentos que permitam percecionar o mundo bem como o estado interno do robô;
- Software e hardware terão de ser escaláveis, modulares e facilmente aplicáveis a qualquer veículo;
- Atuação tanto em ambientes apertados como amplos;

- Funcionamento em ambientes semiestruturados e portáteis;
- Possuir um sistema de deteção e identificação de objetos;
- Possuir um sistema de localização por reconhecimento de objetos;
- Efetuar manobras atrativas para o público menos especializado como efetuar dois percursos pré-definidos, um do tipo boomerang e outro circular em torno de um objeto, e um gerado de forma inteligente desviando-se de um obstáculo detetado a priori;
- Utilizar tecnologias mediáticas como o *kinect* ou *wiimote*.

3.2 Arquitetura de Hardware

Na figura 3.1, está representado o robô implementado neste projeto. Trata-se de um robô de locomoção diferencial de médias dimensões. O *kinect* está colocado a uma distância de 40 cm do solo.

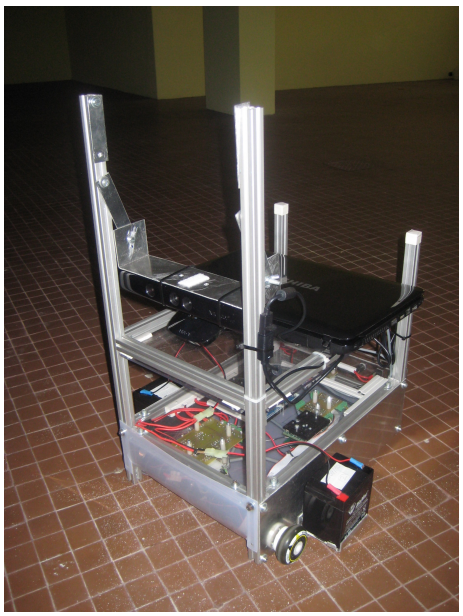


Figura 3.1: Estrutura do robô desenvolvido

Na figura 3.2 está representada a arquitetura funcional do hardware do sistema implementado. Nesta aplicação foi utilizado um robô de locomoção diferencial face à sua facilidade em controlar e manobrar em situações de algum aperto, pois permite a rotação sobre si próprio.

A arquitetura do robô está dividida em três camadas, uma de baixo nível, uma de nível intermédio (camada de interface) e outra de nível mais alto.

O computador no topo da hierarquia controla todo o sistema, permite receber informação de vários nós e atuar sobre os respetivos equipamentos de forma a cumprir uma certa tarefa imposta pelo sistema de decisão. Outro elemento da camada de alto nível é o sensor *kinect* utilizado para

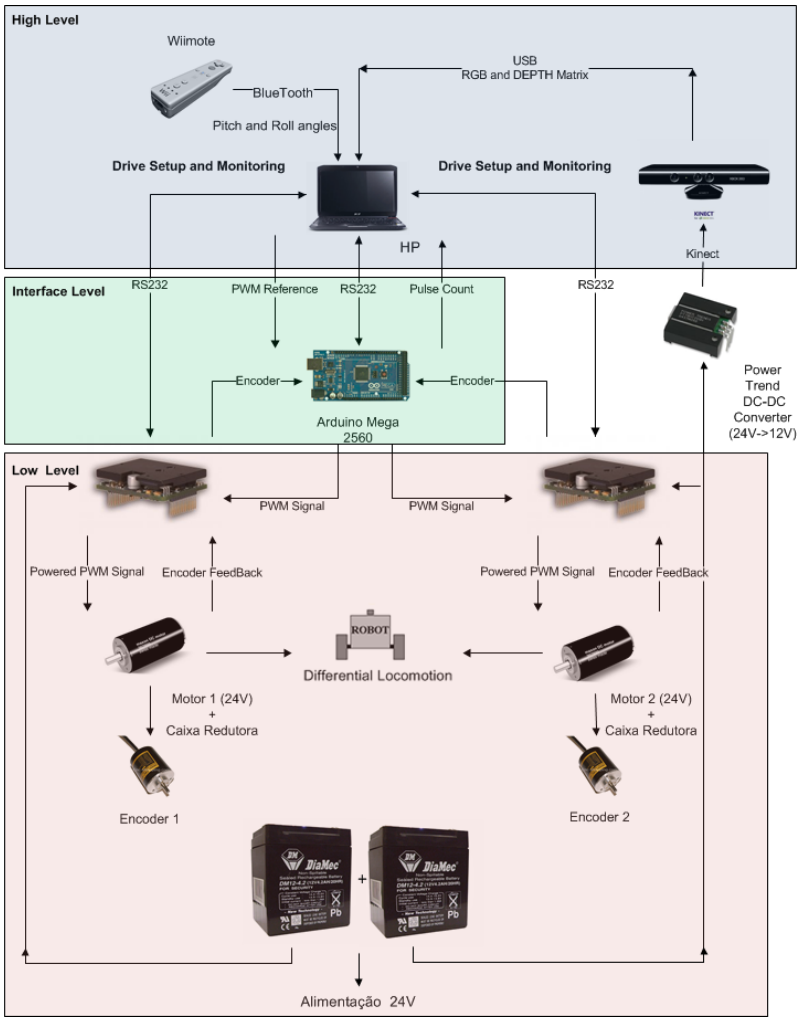


Figura 3.2: Arquitetura do Hardware

informar o computador, através da câmara de profundidade, sobre o ambiente que rodeia o robô. Ainda na camada de alto nível encontra-se o comando remoto da consola *Wii*, o *Wiimote*. Este dispositivo tem como função controlar manualmente o robô, efetuar paragens de emergência e definir alguns parâmetros como a velocidade do robô. Uma das teclas do *wiimote* foi definida como "cravelha"de segurança, quando esta é ativada o robô responde apenas ao controlo manual proveniente do comando remoto e só volta ao estado autónomo assim que a mesma "cravelha"for desativada. Esta funcionalidade é bastante importante uma vez que durante o funcionamento em regime permanente do robô podem ocorrer comportamentos inesperados que podem causar tanto danos materiais como físicos.

Na camada de interface está inserido um módulo que permite a ponte entre o sistema de alto nível com o sistema de nível mais baixo. O dispositivo é um Arduino Mega 2560, capaz de gerar dois PWM (*Pulse-Width Modulation*), um para cada motor e ainda possuir entradas digitais suficientes para receber os sinais lógicos dos *encoders*.

O sistema de alto nível envia referências de velocidade para o módulo de interface e este

encarrega-se de gerar o sinal de comando PWM com o respetivo *Duty Cycle* para os drivers no nível mais baixo. Após a receção do sinal de comando PWM, o driver reencaminha-o para a ponte em H onde se processa a amplificação da potência do sinal de modo a acionar o motor.

O movimento mecânico do motor provoca diferentes combinações entre as fases dos sinais dos *encoders*. Estas combinações permitem detetar uma sequência de operações pré-definidas para determinar tanto o sentido de rotação como a velocidade de cada motor. Sendo estas sequências de operações conhecidas e normalizadas, é possível gerar uma máquina de estados de forma a acompanhar cada alteração mecânica no motor.

Para efeitos de inicialização e monitorização dos parâmetros do motor é ainda possível comunicar diretamente entre o sistema de alto nível e o nível mais baixo da hierarquia, sendo esta uma comunicação por RS-232 (*Recommended Standard 232*) bidireccional entre o computador e os drivers dos motores, como se pode verificar na ilustração da arquitetura.

Projeto em 3D do robô desenvolvido

Antes de se proceder à construção física do robô é necessário projetar e simular o posicionamento de todos os seus componentes constituintes. As figuras 3.3 e 3.4 ilustram o modelo, desenvolvido em *Google Sketchup CAD*, do robô a ser construído enquanto que na figura 3.5 está representada uma perspetiva mais detalhada do robô através da identificação de cada componente.

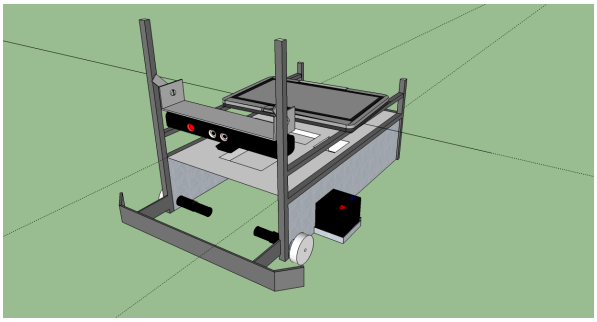


Figura 3.3: Vista Frontal

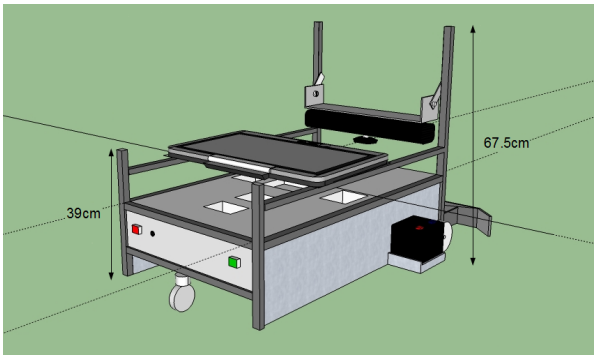


Figura 3.4: Vista Traseira

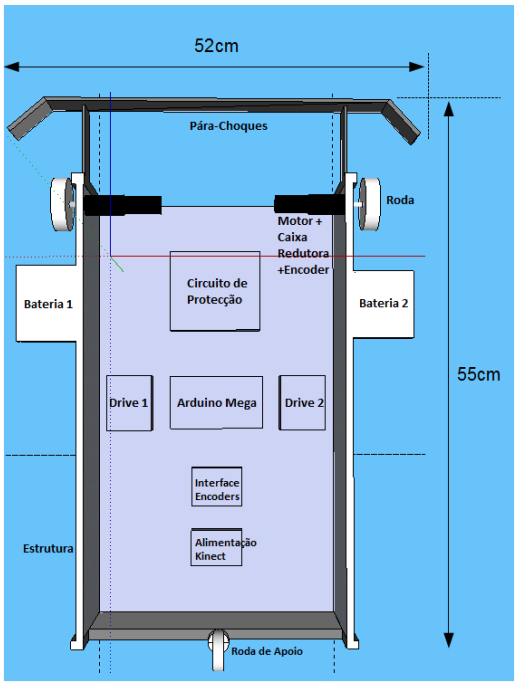


Figura 3.5: Interior

3.2.1 Sistema de locomoção

Para o sistema de movimentação foi utilizado uma configuração já existente, que se poderá ver na figura 3.6. Esta solução consiste na utilização de um motor já equipado com caixa redutora e *encoder*.

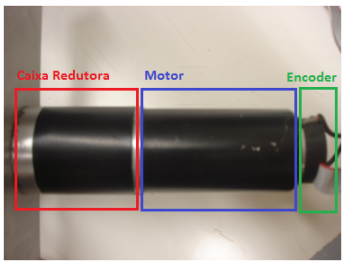


Figura 3.6: Motor Maxon + Caixa Redutora + *Encoder*

3.2.1.1 Motores

O motor disponível foi desenvolvido pela empresa *Maxon Motors*. O equipamento está inserido na série RE 40 (Ref: 148867 [20]) e necessita de uma tensão nominal CC (Corrente Contínua) de 24 V. A potência nominal do motor é de 150 W e possui uma velocidade nominal com carga de 6930 RPM (*Revolutions Per Minute*).

A necessidade de controlo por PWM

Os controladores para motores de elevada potência são normalmente criados com estados de saída pulsada. A tensão aplicada no motor comuta muito rapidamente a uma frequência de relógio de 20-60 KHz entre dois ou três níveis (normalmente +V_{cc}, 0 e -V_{cc}). Devido à elevada constante de tempo mecânica do motor apenas a tensão média é eficaz. Este valor médio é ajustado, variando o período relativo dos níveis de tensão (PWM). A vantagem principal da utilização do PWM é que as perdas e consequentemente a dissipação de calor nos transístores podem ser reduzidas para um valor mínimo. Desta forma os controladores não requerem refrigeração adicional e podem ser mantidos pequenos.

Considerações a ter com os motores Maxon da série RE 40

Os motores Maxon da série RE 40 são bastante dinâmicos (polo elétrico rápido) o que provoca grandes variações de corrente (baixa indutância) nos intervalos de comutação dos transístores (ciclo PWM), o que implica uma elevada corrente RMS (*Root Mean Square*) provocando um sobreaquecimento do motor. Para proteger o motor é necessário colocar uma indutância adicional em série com o motor, permitindo baixar os picos de corrente. Um dos benefícios da indutância adicional é melhorar a estabilidade do controlador de corrente.

Alguns controladores já incluem indutâncias para resolver este problema, evitando o acoplamento de bobines externas.

A expressão aproximada para a corrente de *ripple* encontra-se a seguir:

$$I_{Ripple} = \frac{V_{cc}}{(2 * f_{SW} * (L_{Motor} + L_{Add}))}(A)[21] \tag{3.1}$$

Em que I_{Ripple} é a variação máxima da corrente no motor num ciclo de onda PWM, que depende da tensão de alimentação aplicada ao motor (V_{cc}), da frequência de relógio PWM (f_{SW}), da indutância intrínseca ao motor (L_{Motor}) e qualquer indutância que seja colocada em série com o motor (L_{Add}).

Através da expressão 3.1 é possível extrair as seguintes conclusões para a redução de I_{Ripple}, de acordo com os parâmetros que a relacionam e também tendo em consideração a limitação dos restantes equipamentos:

- Reduzir a tensão de alimentação (V_{cc}) se possível, o que iria aumentar o *Duty Cycle* e diminuir consequentemente o T_{off} da comutação. Este procedimento também provocaria um decréscimo na potência máxima disponível ao sistema. Neste caso não será possível reduzir V_{cc} pois o restante equipamento não o permite;

- Aumentar a frequência de PWM de comutação da ponte de transístores do driver (f_{SW}). Pelo menos 20 Khz, mas o ideal seria 50 Khz ou mais. Neste caso será impossível pois o driver de controlo tem uma frequência fixa de 20 Khz, estando portanto no limite;

- Escolher um enrolamento com indutância maior. Não será possível pois não se pode abrir o motor e trocar o enrolamento;

- A solução mais viável será então colocar uma indutância mais elevada em série com o motor. Com uma tensão de entrada de 24 V, frequência de comutação de 20 Khz, uma indutância do motor de 0.0823 mH e uma indutância adicional de 0.112 mH, a corrente de *ripple* será de 3.088 A, o que é bastante mais baixa e atenua bastante o problema em causa.

3.2.1.2 Caixa de Velocidades

A configuração planetária (figura 3.7) consiste num sistema de uma ou mais engrenagens (planetas) a rodar em torno de uma engrenagem central (sol). Os planetas (*Planetary Gear*) são colocados num braço móvel (*Planet Carrier*) que, por sua vez, pode rodar relativamente ao sol. Este tipo de configuração incorpora ainda uma engrenagem em forma de anel (*Ring Gear*), permitindo a interligação entre os diversos planetas[22].

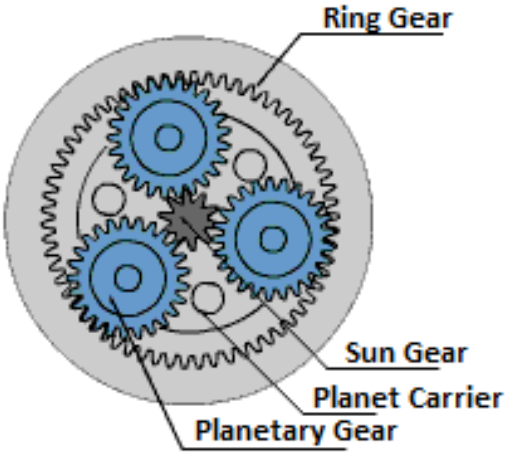


Figura 3.7: Configuração Planetária [23]

Com este tipo de configuração, além de tanto a engrenagem de entrada e de saída partilharem o mesmo eixo e todos os eixos de todas as engrenagens são paralelos entre si, é também possível efetuar diversas combinações entre os vários tipos de engrenagens de modo a obter diferentes estados de velocidade. Estes estados de velocidade estão relacionados com os rácios de velocidade de rotação entre a entrada e a saída da caixa redutora. A configuração planetária, como já foi referido anteriormente, possuí três componentes básicos que são: o sol (*Sun Gear*), o braço onde estão inseridas as engrenagens planetárias (*Planet Carrier*) e o anel (*Ring Gear*). Em muitos sistemas de configuração planetária um destes três componentes básicos apresenta um papel estacionário enquanto que dos dois restantes componentes um é considerado uma entrada, providenciando energia para o sistema (através do veio do motor), e o outro é considerado uma saída,

recebendo energia do sistema e transmitindo para o veio de saída onde se encontra uma das rodas do robô. A relação entre a rotação de entrada e a rotação de saída é dependente do número de dentes em cada engrenagem e do componente que é atribuído o estado estacionário[22].

Os vários tipos de combinações de engrenagens podem ser agrupados em três grupos, os que provocam uma redução de velocidade (velocidade de saída menor que a velocidade de entrada), os que provocam um aumento de velocidade ou *overdrive* (velocidade de saída maior que a velocidade de entrada) e os que provocam o movimento no sentido inverso[22].

A funcionalidade redutora da caixa é obtida colocando o anel como engrenagem estacionária e escolhendo qualquer umas das outras duas para entrada e saída. A atribuição das duas restantes engrenagens irá provocar apenas diferentes relações entre a velocidade de rotação da entrada e da saída, apresentando sempre a funcionalidade redutora. A funcionalidade em *overdrive* é obtida fixando a engrenagem solar e escolhendo uma das combinações entre as duas engrenagens restantes. As combinações referidas até agora foram todas relacionadas com o sentido direto do movimento. No sentido inverso existe apenas uma única combinação entre os três componentes básicos. Esta funcionalidade é obtida fixando o braço (*Planet Carrier*) e colocando o sol como entrada e o anel como saída. Com esta configuração é possível obter no total cinco estados de velocidade, quatro no sentido direto e uma no sentido inverso[22].

A caixa de velocidades utilizada neste projeto foi desenvolvida pela empresa Maxon Motor e pode ser analisada em maior detalhe na *Datasheet* do produto, disponível em [24] (Ref: 203116). A caixa de velocidades possuí uma configuração planetária com apenas duas velocidades/estados, sentido direto e inverso, e provoca à saída uma redução de 15:1 na velocidade do motor nos dois sentidos. O equipamento referido insere-se então no grupo dos redutores, com uma velocidade de saída inferior à da entrada, apresentando também a capacidade de inversão de marcha. Para pertencer ao grupo dos redutores, sabe-se que apresenta como engrenagem estacionária o anel mas não é especificado pelo fabricante qual papel desempenha qualquer um dos dois restantes componentes. Como a velocidade nominal do motor é de 6930 RPM, de acordo com o fabricante, a velocidade após a redução será de 6930/15$\simeq$462 RPM. Embora esta seja a velocidade real à saída de cada motor, será no entanto restringida pelo tipo de roda utilizada, como se verá mais à frente.

Algumas das vantagens da configuração planetária é permitir uma elevada densidade de potência em qualquer engrenagem, ser capaz de produzir reduções bastante elevadas em pequenos volumes e também permitir múltiplas combinações cinemáticas, como foi referido anteriormente.

Em relação às desvantagens desta configuração é o facto da sua inacessibilidade e a complexidade da arquitetura.

A caixa de velocidades de configuração planetária apresenta inúmeras vantagens em relação às caixas convencionais. Uma delas é o facto de apresentar uma combinação única entre a compacidade e a extraordinária eficiência na transmissão de energia. Uma perda de energia típica nesta configuração é de apenas 3% por estado. Este tipo de eficiência assegura que uma elevada porção de energia seja transmitida à saída sem que seja despendida em perdas mecânicas no interior da

caixa de velocidades. Ou vantagem é a distribuição de carga. A carga aplicada é distribuída uniformemente pelos múltiplos planetas presentes no braço (*Planet Carrier*), a capacidade de binário é aumentada consideravelmente. Quantos mais planetas existirem na configuração, maior é a carga admissível e consequentemente a densidade do binário também será maior[25].

A cerâmica é o tipo de material que constitui o interior da caixa de velocidades utilizada neste projeto. Este tipo de material apresenta um melhor desempenho face às engrenagens de aço devido ao seu baixo coeficiente de erosão, permitindo um tempo de vida mais longo, binários contínuos mais elevados e admite velocidades de entrada também mais elevadas. Na figura 3.8 está representada uma comparação entre os binários de cada caixa com diferentes tipos de material, que confirma a afirmação anterior[25].

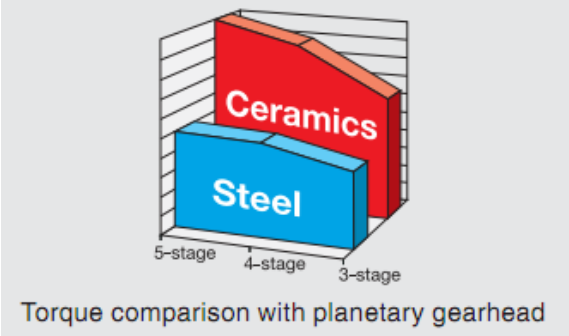


Figura 3.8: Comparação de binários com caixas de velocidades de diferentes tipos de materiais[25]

3.2.1.3 Rodas

As rodas utilizadas no projeto do robô estão ilustradas na figura 3.9. A velocidade máxima de cada motor com este tipo de roda será calculada da seguinte forma: como já foi referido anteriormente, a velocidade nominal do motor com a caixa redutora era de 462 RPM o que corresponde a 7,7 RPS (*Revolutions Per Second*). Tendo em conta que o raio da roda é aproximadamente R=0.03 m, numa rotação do motor a roda percorre $2*\pi*R \simeq 0.19$ m. Num segundo o motor efetua 7,7 rotações, o que aplicando a regra de três simples, o robô terá por cada roda uma velocidade máxima de 1,45 m/s$\simeq$5.22 km/h. Este valor é relativamente baixo, face à capacidade dos motores utilizados, mas suficiente para o que é pretendido, ou seja, um robô que circule em espaços fechados e relativamente pequenos para efeitos de demonstração.

3.2.1.4 *Encoders*

Os *encoders* utilizados neste projeto possuem um funcionamento em quadratura (figura 3.10). Este tipo de *encoders* é do tipo incremental, capazes de determinar a posição, a direção do movimento e a velocidade de um motor. Os parâmetros do *encoder* utilizado podem ser analisados em maior detalhe na sua *Datasheet*, disponível em [26] (Ref: 225783).

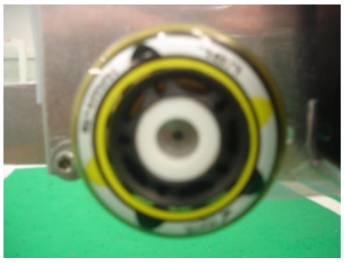


Figura 3.9: Roda Utilizada

Os *encoders* utilizados usam dois sinais iguais (A e B) em forma de onda quadrada e com um desfasamento entre eles de 90º. Este procedimento é utilizado de forma a evitar a ambiguidade na determinação do sentido de orientação de um motor, quando utilizado apenas um sinal. Estes sinais são então comparados para determinar o sentido de rotação do motor. Pela figura 3.10 é possível detetar de forma trivial quando ocorre a troca de sentido de rotação. O princípio é relativamente simples, um sinal desfasado de 90º do outro de forma fixa. Desta forma pode-se afirmar que um dos sinais, num dos sentidos de rotação, está à frente do outro. Esta verificação é feita, efetuando uma comparação lógica entre os dois sinais.

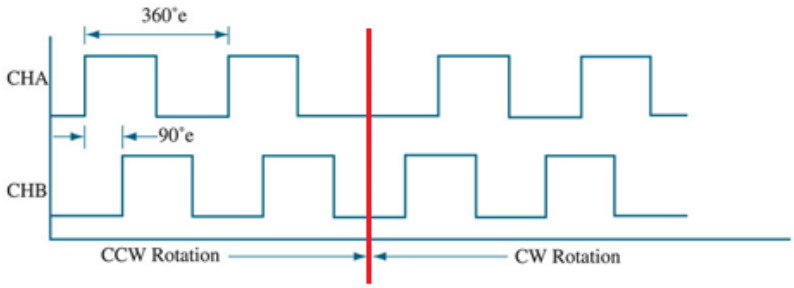


Figura 3.10: Sinal de Quadratura do *Encoder* [27]

O *encoder* utilizado neste projeto apresenta um princípio de funcionamento magneto-resistivo, que consiste num disco magnético, inserido no veio do motor que, devido ao movimento do mesmo, produz uma tensão sinusoidal no sensor MR (*Magneto-Resistive*). Os sinais digitais das fases A e B do *encoder* são determinados por interpolação e refinamento dos sinais eletrónicos. Magneto-resistência é a propriedade de um material que permite alterar o valor da sua resistência elétrica, quando um campo magnético externo é aplicado. O sensor magneto-resistivo (MR) tem como objetivo detetar estas alterações no disco magnético multipolar, como representado na figura 3.11.

Como o *encoder* encontra-se acoplado ao veio do motor, não havendo reduções nem multiplicações intermédias, a relação entre a rotação do *encoder* e do motor é de 1:1. Portanto, como o *encoder* em questão permite uma resolução de 209 impulsos por volta numa fase, é possível concluir que por cada rotação do motor, utilizando as quatro fases do sinal (quadratura completa),

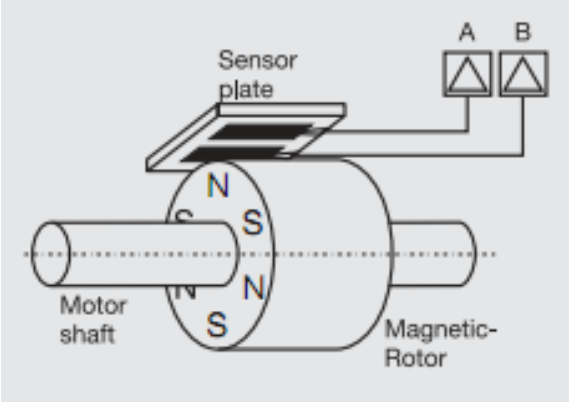


Figura 3.11: *Encoder* MR [28]

proporciona 209*4 impulsos no *encoder*. No entanto, como existe uma caixa redutora de 15:1 acoplada ao veio do motor, por cada volta completa do veio externo da caixa o número de impulsos será de 209*4*15.

De referir que embora na *Datasheet* do produto apresente o valor de 256 para o número de impulsos por volta numa fase, através de vários testes verificou-se o valor 209, daí a sua utilização.

3.2.2 Sistema de Controlo e Acionamento

Durante a fase de projeto e dimensionamento dos drivers de controlo dos motores foram analisadas várias hipóteses como as que se encontram ilustradas na figura 3.12. Pretendia-se um sistema de controlo versátil, que funcionasse tanto para motores com escovas ou sem escovas, que tivesse um controlo de velocidade por PID (*Proportional, Integral and Derivative*) e *FeedForward*, e que permitisse que a malha de controlo fosse fechada tanto por sensores de efeito de Hall (no caso dos motores sem escovas) como por *encoders* (no caso dos motores com escovas). Um sistema modular, de fácil montagem e aprendizagem também estão incluídos nos fatores de escolha. Era também fator de escolha, um driver que permitisse que o *setup*, teste e monitorização dos motores fossem processos a serem efetuados através de RS-232. Naturalmente o baixo custo será também um dos pontos a ter em consideração na escolha do componente.

O hardware mais indicado para satisfazer os requisitos referidos anteriormente será a opção que se encontra na primeira linha da lista. As suas características serão agora analisadas em maior detalhe.

Drivers AMC DZRALTE - 012L80

Para o sistema de acionamento e controlo dos motores utilizaram-se os drivers AMC DZRALTE - 012L80 da empresa *ADVANCED Motion Controls*. Trata-se de um driver extremamente versátil com funcionamento tanto com motores de escovas ou sem escovas. Possui controladores

Marca	Nome do Equipamento	Acessórios	Tensão(V)	Corrente(A)	Corrente de Pico (A)	+IVA Preço(€)	Precisa Micro + ?	Entrada Analogica	PWM	RS232	Controlador	FOTO
AMC	DZRALTE-012L080	MC1XDZ Mounting Card (Single Axis)	20..80	6	12	211	N	N	S	S	PID + FF	
AMC	AZB6A8	MC1XAZ01-HR Mounting Card	20..80	3	6	164	S	S(-10..10V)	S	N	PI	
PMD	ION 500		12..56	8	15	480	N	N	S	S	Scalable PID with Vel + Acc feedforward, integration limit, offset bias, dual biquad filter and settable derivative sampling time	
FAULHABER Drive systems	BLD 4803-SH4P		6..48	2,5	5	113 - 5%	S	S(0..5V)	S	N	Speed Regulator PI	
Maxon	Digital positioning controller EPOS2 P 24/5, 5 A, 11 - 24 VDC		11..24	5	10	471.66	N	N	S	S		
Maxon	DECV 50/5, digital 4-Q-EC Amplifier 50 V / 5 A, speed control		12..50	5	10	187.7	S	S(0..5V)	S	N		

Figura 3.12: Lista de drivers estudados

lineares PID tanto de velocidade como de corrente e também controlo por *FeedForward*. Apresenta um vasto leque de sistema de proteções, como limitadores de corrente, proteção contra curto circuitos, etc.

Está disponível ainda uma interface desenvolvida pelo fabricante que permite configurar, monitorizar e controlar todos os parâmetros do driver. A aplicação é simples e de fácil entendimento e a comunicação com o computador é feita por comunicação série RS-232 ou RS-485.

O driver permite que sejam aplicadas na entrada tensões que variam entre os 20 e os 80 V. A corrente em regime contínuo não poderá exceder os 6 A. A corrente de pico está limitada a um valor máximo de 12 A, podendo permanecer neste valor até 2 segundos. A especificação dos limites de corrente, tensão e período de tempo máximo de corrente de pico são exemplos do que é permitido fazer com o software de configuração.

Permite tanto a ligação de sensores de efeito de Hall como de *encoders* para o controlo de velocidade do motor.

Os acessórios disponíveis permitem uma fácil interface com os pinos do driver.

A limitação mais notória apresentada por este equipamento é o facto da frequência de comutação dos transístores ser fixa em 20 Khz, como se verificou anteriormente (secção 3.2.1).

O equipamento referido está representado na figura 3.13.

Como foi referido na secção 3.2.1, durante o desenvolvimento do projeto foi necessário substituir motores sem escovas por motores com escovas, o que acabou por não se tornar uma situação crítica uma vez que o driver escolhido permite os dois modos de funcionamento.

Figura 3.13: Driver Motores AMC DZRALTE – 012L80 [29]

3.2.3 *kinect*

Neste projeto foi utilizado o sensor *kinect* da Microsoft (figura 3.14) desenvolvido em parceria com a empresa *Prime Sense*. Pretendia-se adquirir um equipamento que percecionasse o mundo através da profundidade, inicialmente pensou-se numa rede de sensores IR (*Infra-Red*) *Sharp* mas o custo envolvido seria elevado, tendo em conta que cada sensor com alcance de 0.4 a 3 metros custa aproximadamente 55 euros além de permitir apenas uma abertura angular de 25º[30]. O surgimento do sensor *kinect* permitiu abandonar a ideia inicial face ao seu baixo custo ($\simeq$150 euros), boa precisão, elevado alcance (0.8 a 3.5 metros) e uma boa abertura da lente (57º horizontal e 43º vertical), além de ser uma tecnologia mediática capaz de atrair o público em geral.

Este sensor possui duas câmaras distintas de perceção do mundo, uma RGB que retorna uma matriz com três canais de cores por pixel (vermelho, verde e azul) e outra por infravermelhos para reconhecer movimento e profundidade. Estas câmaras estão separadas por um distância fixa, que através de transformações entre os dois padrões de informação é possível obter uma correspondência direta de cor e profundidade de um dado ponto no mundo. Neste projeto apenas se utilizou a informação da imagem de infravermelhos para detetar e reconhecer objetos.

Mais à frente neste relatório será abordado o princípio de funcionamento do sensor, assim como as suas principais limitações, não só em relação a interferências externas mas também a fatores intrínsecos à sua própria natureza tecnológica.

Figura 3.14: *kinect* [31]

3.2.4 Sistema de Alimentação

Para o dimensionamento do sistema de alimentação a utilizar é necessário ter em consideração os componentes que englobam todo o sistema robótico. Os equipamentos mais críticos e mais

exigentes em regime permanente são o motor e o *kinect* que consomem no máximo em regime permanente correntes de 5.77 A (condição em que a carga e velocidade do motor apresentam valores nominais) e 1.08 A, respetivamente. Uma vez que o consumo nominal do motor é de 5.77 A, ajusta-se via software o valor limite do fornecimento de corrente em regime contínuo pelo controlador que passa de 6 A para 5.77 A.

O conjunto de baterias ideal seria de LiPo (*Lithium Polymer Battery*), por serem leves, ocuparem pouco espaço, terem um tempo de vida mais longo e terem um tempo de carga mais rápido que as células de chumbo ou NiMH. Este último ponto só resulta se forem feitas algumas considerações como por exemplo, nunca deixar a tensão de uma célula baixar do valor crítico (2.7 V) ou acima de 4.3 V pois podem ficar completamente inutilizadas e até se incendiarem.

No entanto devido ao seu elevado custo utilizou-se duas baterias em série de chumbo ácido de 12 V cada e com uma autonomia de 4.2 Ah que se encontravam disponíveis no laboratório. Para estas baterias não são necessárias medidas relevantes relativas ao processo de descarga uma vez que permitem descarga completa, no entanto é necessário ter atenção ao posicionamento das mesmas. As baterias como contém ácido se estiverem deitadas durante muito tempo, o líquido escorre e acaba por verter pelas aberturas no plástico do topo da bateria.

Tendo em consideração que cada motor consome 5.77 A em regime contínuo, os dois motores do robô consomem 11.54 A. Somando a este valor o consumo do *kinect* (1.08 A), fica-se com 12.62 A. Desta forma pode-se estimar, em condições nominais, uma duração de aproximadamente 33 min com as baterias de chumbo ácido. No entanto o controlador permite sempre redefinir os limites de corrente, podendo desta forma gerir a autonomia do sistema.

O equipamento utilizado está ilustrado na figura 3.15.

Figura 3.15: Bateria *Diamec* com 12 V e 4.2 Ah

3.2.5 Arduino Mega 2560

Para a aquisição dos sinais dos *encoders* utilizou-se uma placa *Arduino*. Um *Arduino* é uma plataforma de hardware livre, projetada com um microcontrolador Atmel AVR. Esta placa física é baseada num circuito de entradas/saídas. A linguagem de programação tem origem na plataforma de prototipagem eletrónica de código aberto *Wiring*, que é essencialmente C/C++. A plataforma *Wiring* é composta por um IDE (*Integrated Development Environment*), uma placa de prototipagem eletrónica e uma documentação detalhada, com uma vasta biblioteca de funções que permite

criar uma abstração ao hardware em utilização. A plataforma possui um software capaz controlar e interagir com o hardware instalado na placa eletrónica através de uma linguagem simples e abstrata.

Neste caso foi tido em consideração um microcontrolador que permitisse a receção de pelo menos quatro sinais digitais por interrupção externa, sendo neste caso as fases A e B dos dois *encoders*. A placa *Arduino* que se enquadrava foi a que incorporava o microcontrolador ATmega2560, o *Arduino Mega 2560* (ver figura 3.16). As características do dispositivo encontram-se expostos na tabela 3.1.

Tabela 3.1: Características do Arduino Mega 2560

Características	
Micro-controlador	ATmega2560
Tensão de Alimentação	5 V
Tensão de Entrada (Recomendado)	7-12 V
Tensão de Entrada (Limites)	6-20 V
Nº de Pinos E/S Digitais	54
Nº de PWM	14
Nº de Interrupções Externas	6
Corrente CC por Pino E/S	40 mA
Freq. Cristal de Oscilação	16 MHz

Para determinar se a frequência do cristal de oscilação do dispositivo é suficiente para detetar cada impulso dos *encoders*, é necessário verificar a frequência de ocorrência dos impulsos para o caso mais crítico, ou seja, à velocidade nominal do motor, que será a velocidade máxima utilizada neste projeto para cada motor.

Como já foi referido anteriormente (secção 3.2.1), para cada rotação do motor ocorre uma rotação do *encoder*. Em cada rotação do *encoder* ocorrem aproximadamente 209 impulsos para um quarto da quadratura do sinal. Neste projeto utilizou-se apenas metade da quadratura do sinal (duas fases), portanto à velocidade nominal de 6930 RPM$\simeq$116 RPS ocorrem 116*209*2=48488 impulsos por segundo, o que implicitamente significa que no máximo os impulsos ocorrem a uma frequência de 48488 Hz. Incluindo uma margem de segurança para a deteção dos impulsos de pelo menos o dobro da frequência máxima e tendo em consideração que se trata de dois *encoders*, o microcontrolador tem de ser capaz de detetar alterações nas portas de interrupção externa a uma frequência de 48488*4$\simeq$194 KHz. Uma vez que o cristal de oscilação é de 16 MHz, é possível afirmar que o sistema irá detetar com segurança cada impulso dos *encoders*.

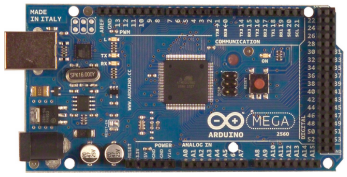


Figura 3.16: Arduino Mega 2560 [32]

O tipo de abordagem por interrupção externa permite oscilações na carga de processamento do microcontrolador, apresentando maior carga à velocidade nominal do motor. No entanto existe outro método mais estável do ponto de vista da carga de processamento, em que consiste num temporizador a amostrar o sinal a uma cadência fixa. Este último método permite uma carga constante mas é mais suscetível a perda de impulsos.

3.3 Arquitetura de Software

Nesta secção será apresentado o software desenvolvido.

No que diz respeito ao software, já foi referido anteriormente que o sistema implementado está dividido em várias camadas. A camada de mais alto nível é composta por duas aplicações diferentes, a aplicação de decisão que toma todas as decisões do sistema e a aplicação que recolhe e trata as informações fornecidas pelo *kinect*. Ambas as aplicações estão alojadas no mesmo equipamento, o computador.

Na camada de nível de interface encontra-se a aplicação que permite a ponte de fluxo de dados entre o sistema de decisão e a camada de nível baixo onde se encontram os motores, os drivers que controlam os motores e os *encoders*.

Na secção 3.3.1 serão apresentados os diferentes protocolos de comunicação utilizados no projeto e na secção 3.3.2 as linguagens de programação utilizadas.

Na secção 3.3.3 será apresentado a arquitetura da aplicação de decisão e na secção 3.3.4 será explicada por via de um fluxo grama a sequência de operações a serem efetuadas pelo software de decisão e controlo após a receção de informação proveniente do *kinect*.

Na secção 3.3.5 encontra-se a sequência de operações desenvolvidas na aplicação da camada de interface.

3.3.1 Protocolos de Comunicação

Neste projeto foram utilizados dois tipos de protocolos de comunicação, RS-232 e UDP (*User Datagram Protocol*).

Comunicação entre diferentes camadas (Alto Nível - Camada de Interface)

O protocolo RS-232 é utilizado na troca de dados entre a camada de alto nível e a camada de interface e entre os drivers dos motores (baixo nível) e o computador (alto nível) para efeitos de monitorização, configuração e controlo manual dos motores. No primeiro caso o fluxo de dados, como já foi referido anteriormente, é bidireccional. O conteúdo da informação no sentido 'nível alto-camada de interface' é referente à referência de velocidade (0-255) a ser atribuída a cada um dos motores, sendo que 0 é o máximo valor da velocidade do motor no sentido inverso e 255 o máximo valor para a velocidade no sentido direto, uma vez que 127 é o valor estacionário. A camada de interface apenas aguarda mensagem de um nó (computador), o que será apenas necessário efetuar um encapsulamento simples da trama (caractere de inicio e fim delimitando os

dados) de forma a evitar possível ruído pela porta série e de fácil identificação e extração de dados por parte do nó de destino. Como caractere de início foi utilizado o símbolo '<' e como caractere de fim foi utilizado o símbolo '>'. O nó de destino após detetar estes dois símbolos, verifica se o tamanho da trama entre esses mesmos símbolos corresponde ao esperado. De referir que, para uma simples e rápida identificação, o tamanho da trama é sempre o mesmo, cada referência de velocidade é representada por três caracteres (000..255) e a referência para o motor esquerdo é separada da referência do motor direito através de uma vírgula mas sempre pela mesma ordem (<EEE,DDD>). No sentido inverso (camada de interface-nível alto) o conteúdo altera-se mas o princípio é o mesmo. Os dados a serem enviados neste sentido serão as diferenças da contagem dos impulsos dos *encoders* em cada roda a cada 40 ms, que apenas serão enviados caso se verifique alguma alteração. Neste caso o tamanho da trama será maior uma vez que a resolução dos *encoders* é de 16 bits, o que terá o valor máximo de 65535. Cada trama terá então a seguinte configuração: '<EEEEE,DDDDD>'.

Comunicação no interior da mesma camada (PC - *kinect*)

A comunicação entre a aplicação de decisão (nível mais elevado) e a aplicação do *kinect* realiza-se no interior da mesma máquina/equipamento. Sendo assim, como as perdas de informação durante a comunicação são desprezáveis e pode-se dispensar todas as verificações de fiabilidade da mensagem, evitando assim o congestionamento ao nível da interface de rede e aumentando a velocidade de processamento. Desta forma utilizou-se o protocolo UDP, cumprindo então os requisitos mencionados.

A aplicação de comunicação com o *kinect* informa a aplicação de nível mais elevado se o robô está diante de um conjunto de balizas conhecidas (para efeitos de localização), esferas, cilindros ou qualquer tipo de objeto desconhecido de relevância. A trama base criada para o fluxo de dados entre estas duas aplicações é '<Opt{dx:value;dy:value;}>'. O parâmetro 'Opt' é um abreviatura para 'Option' que significa que neste campo estará presente a natureza da informação enquanto que 'dx' e 'dy' correspondem às coordenadas X e Y do objeto no mundo, respetivamente. Em vez de 'Opt' irá aparecer 'bc1', 'bc2', 'sph', 'cyl' ou 'blk' para receber informações sobre o conjunto de balizas/*beacons* número 1, número 2, esferas, cilindros ou objetos desconhecidos, respetivamente. No caso dos conjuntos de balizas a trama será: '<bc1/bc2{dx:value;dy:value;dx:value;dy:value;}>' uma vez que se trata de dois objetos por conjunto. No caso das esferas ou dos cilindros a única diferença é a troca de 'Opt' para 'sph' ou 'cyl'. Em relação aos restantes objetos além da posição serão enviados também as dimensões 2D detetadas, portanto a trama será:

'<blk{dx:value;dy:value;dimX:value;dimY:value;}>'.

Esta abordagem utiliza o conceito de '*String Explode*', ou seja, o algoritmo ao analisar a trama utiliza como referência caracteres especiais como '<' ou '>' para delimitar a mensagem ou ';' para separar cada um dos parâmetros.

Comunicação série entre o PC e o Sistema de Acionamento e Controlo

A comunicação entre o PC e a unidade de controlo de baixo nível que controla os motores, como já foi referida anteriormente (3.2.2), é apenas utilizada quando é necessário efetuar configurações, testes e calibrações dos motores. Quando um novo motor é associado à unidade de acionamento e controlo, é procedido a uma configuração inicial de todos os parâmetros necessários sendo que a partir desse momento o dispositivo encontra-se apto para o funcionamento em regime permanente. A conexão só volta a ser restabelecida quando for detetada qualquer anomalia ao funcionamento normal do driver, uma vez que o dispositivo contém uma vasta lista de possíveis anomalias detetáveis pelos vários sensores que o constituem que são passíveis de serem detetados e circunscritos via software.

A comunicação é ponto-a-ponto do tipo mestre-escravo, em que o papel de mestre é desenvolvido pela aplicação criada pelo fabricante do dispositivo de controlo, adequada para a interação com esse dispositivo (escravo). Após a conexão entre mestre e escravo ter sido estabelecida, o mestre pode solicitar ao escravo informações sobre o driver ou sobre o motor e ainda permite escrever ou reescrever dados no escravo. As funções referidas anteriormente são de leitura e escrita, respetivamente. O escravo envia dados para o mestre se e só se receber um pedido de leitura ou escrita e se a mensagem estiver corretamente identificada e direcionada. Na figura 3.17 estão ilustrados os vários constituintes da trama de leitura/escrita. A informação relativa à função da respetiva trama está presente no parâmetro *Control Byte*.

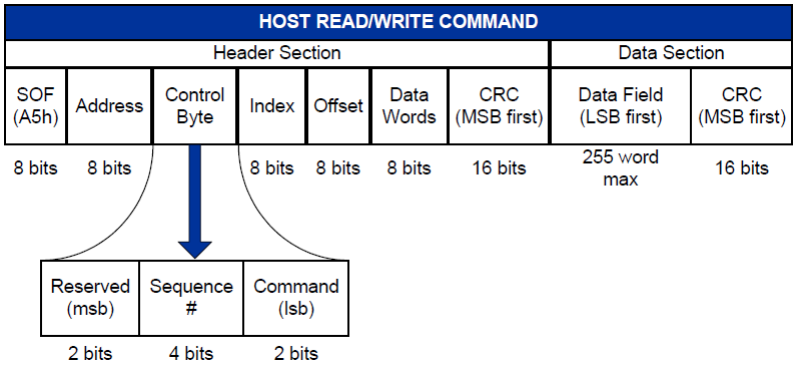


Figura 3.17: Constituição da mensagem de leitura/escrita a ser enviada pelo mestre [33]

Nos tópicos seguintes será explicado o significado de cada parcela da mensagem da figura 3.17.

Start of Frame (S.O.F.)

Qualquer mensagem entre mestre e escravo inicia-se com um byte indicando o inicio da trama. O valor é sempre A5h (hexadecimal), qualquer que seja o remetente da mensagem, mestre ou escravo.

Address

Corresponde ao endereço de destino da mensagem. Cada escravo tem um número de identificação único que pode apresentar desde o valor 1 até 63. O valor de fábrica é 63, portanto como a aplicação apenas comunica com um dos drivers apenas quando um deles está desconectado, pode-se atribuir este mesmo valor a cada um dos escravos.

Control Byte

Este parâmetro, como já foi referido anteriormente, identifica a função a ser enviada, ou seja, tratando-se de uma mensagem de leitura ou escrita.

Index e Offset Byte

O dispositivo utilizado, para que o acesso aos dados presentes em memória seja mais rápido, apresenta o conceito de *Look Up Table* para aceder aos dados. Desta forma é necessário especificar o índice e o valor de *offset* nesse índice. O valor *offset* é necessário, porque cada índice possui mais que um parâmetro, tornando-se então necessário definir a sua posição.

Data Words

O parâmetro *Data Words*, trata-se de uma valor de 8 bits contendo informação do número de *words* (2 bytes) presentes no campo de dados.

Caso seja um comando de escrita, este parâmetro apresenta o número de *Data Words* presentes na mensagem do mestre enquanto que numa mensagem de leitura este parâmetro corresponde ao número de *Data Words* presentes na mensagem de resposta do escravo.

Header CRC Value

Header Section e *Data Section* devem ter uma valor de CRC incluídos. CRC é um detetor de erros, que permite detetar anomalias nas mensagens a serem transmitidas. Se não existirem quaisquer dados, os parâmetros relativos ao *Data Section* serão descartados.

Se o escravo não identificar o endereço ou não concordar com o valor do CRC do *Header Section* proveniente da mensagem do mestre, a mensagem será ignorada até detetar um novo S.O.F. Caso passe os testes anteriores e falhe na verificação do CRC do *Data Section*, será enviada, pelo escravo, uma mensagem de erro a informar o sucedido.

A resposta do escravo encontra-se ilustrada na figura 3.18.

Na mensagem de resposta do escravo (driver de controlo) o parâmetro *Control Byte* adquire outro significado, ou seja, permite informar o mestre se a mensagem que ele está a receber contém ou não informação (dados) (figura 3.19).

Outra diferença detetada é o facto de se incluir o parâmetro *Status*, que informa o mestre da ocorrência de alguma anomalia na mensagem que este anteriormente enviou.

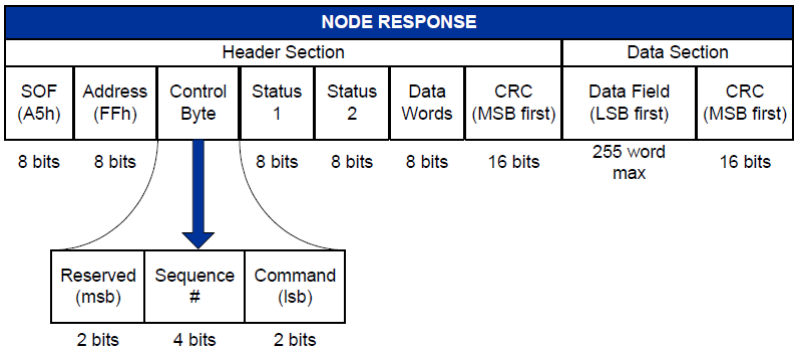


Figura 3.18: Constituição da mensagem de resposta do escravo [33]

Command Bits 0 & 1	Sequence Bits 2 - 5	Reserved Bits 6 & 7	Description
0	User specified	X	This message contains no data.
1	User specified	X	Reserved for future use.
2	User specified	X	This message contains Data as specified by Data Words in the Header section of the Response message.
3	User specified	X	Reserved for future use.
	Example: Node responds to a Host 1 command containing a sequence value of 5. Node Response Control byte value = 00010110 or 16h; this indicates node is sending the requested data to host.		

Figura 3.19: Tabela de interpretação do parâmetro *Control Byte* [33]

Value	Description
1h	Command complete
2h	Command incomplete
4h	Invalid command
6h	Do not have write access. See index "Access Control" for obtaining write access.
8h	Frame or CRC error

Figura 3.20: Tabela de interpretação do parâmetro *Status* [33]

3.3.2 Linguagens de Programação

Neste projeto foram utilizadas três linguagens de programação diferentes. Para a aplicação de decisão foi utilizado *Lazarus* como ambiente de desenvolvimento. Este ambiente de desenvolvimento é livre e permite compilar em qualquer sistema operativo. Utiliza o *FreePascal* como linguagem de programação.

Para a aplicação do *kinect* utilizou-se C++, pois já existem bibliotecas desenvolvidas para iteração com o *kinect* ('libfreenect' para linux) e bibliotecas de processamento de imagem 'OpenCV' que se revelaram bastante úteis para a iteração com imagens e cálculo entre matrizes.

Na aplicação da camada de interface utilizou-se o software desenvolvido em Java pelo fabricante para programação em Arduinos. A aplicação em Java permite programar em linguagem C com funções bastante específicas para a programação de micro controladores. Esta aplicação permite uma abstração quase total para o utilizador na iteração com o hardware.

3.3.3 Software de Decisão e Controlo

No caso da aplicação de decisão, esta deverá cumprir os seguintes requisitos:

- Receber dados do estado mundo acerca de qualquer informação que condicione a atual trajetória do robô, como a sua posição atual ou a presença de objetos no percurso;
- Recalcular novas trajetórias na presença de obstáculos no mundo de modo a contorna-los;
- Informar o estado do mundo dos comandos aplicados ao sistema físico de modo a este acompanhar a evolução do mundo real;

Como se pode verificar pela figura 3.21, a aplicação de alto nível foi desenvolvida de forma a permitir a interação com diversos equipamentos independentemente do seu tipo e quantidade. Para isso criou-se uma camada de abstração que comunica diretamente com os equipamentos, que se encontram na camada de baixo nível, proporcionando a ponte de informação com a camada de alto nível. A camada de abstração (HAL) trata-se de N aplicações que recebem e processam a informação de N sensores e atuadores, cuja função é ajustar a informação a parâmetros interpretáveis pelos respetivos destinatários, permitindo que tanto a camada e alto e baixo nível não tenham conhecimento da sua existência.

A arquitetura de software implementada possuí um sistema de configuração acessível tanto pelo módulo de decisão e controlo como pelo módulo de localização e fusão de informação. O sistema de configuração permite estabelecer parâmetros como ganhos de PID para controladores de trajetórias, velocidade nominal do robô, etc. A interação com o utilizador permite o ajuste e alteração desses parâmetros.

O estado do mundo contém toda a informação do mundo através dos equipamentos de perceção do mundo instalados no robô. Esta informação é então partilhada com o módulo de localização de modo a determinar qual a posição e direção atual do robô e com o módulo de decisão e controlo de modo a definir tarefas de acordo com o novo estado do mundo. Todas as decisões são posteriormente reencaminhadas para as aplicações da camada de abstração de forma a interpretar a informação e aplicar o comando adequado ao respetivo equipamento.

A arquitetura de software desenvolvida permite ainda operar tanto como modo de simulação como modo real. Esta funcionalidade é de uma utilidade bastante importante já que é necessário manter a integridade do equipamento, assim todos os algoritmos serão inicialmente testados num ambiente modelizado do sistema e após as respetivas validações poderá então ser colocado em prática no mundo real.

3.3.3.1 Descrição das Funcionalidades da Aplicação de Decisão de Controlo

A aplicação de decisão e controlo foi, como já foi referido anteriormente, desenvolvida em *Lazarus FreePascal*. O ambiente gráfico está ilustrado na figura 3.22

Em *A* encontra-se a representação do mapa do mundo em 2D, toda a informação relativa à perceção do mundo é projetada neste espaço que é atualizado a cada 40 ms. Em *B* encontram-se um conjunto de opções que permite ao utilizador definir novas posições para o robô e para o

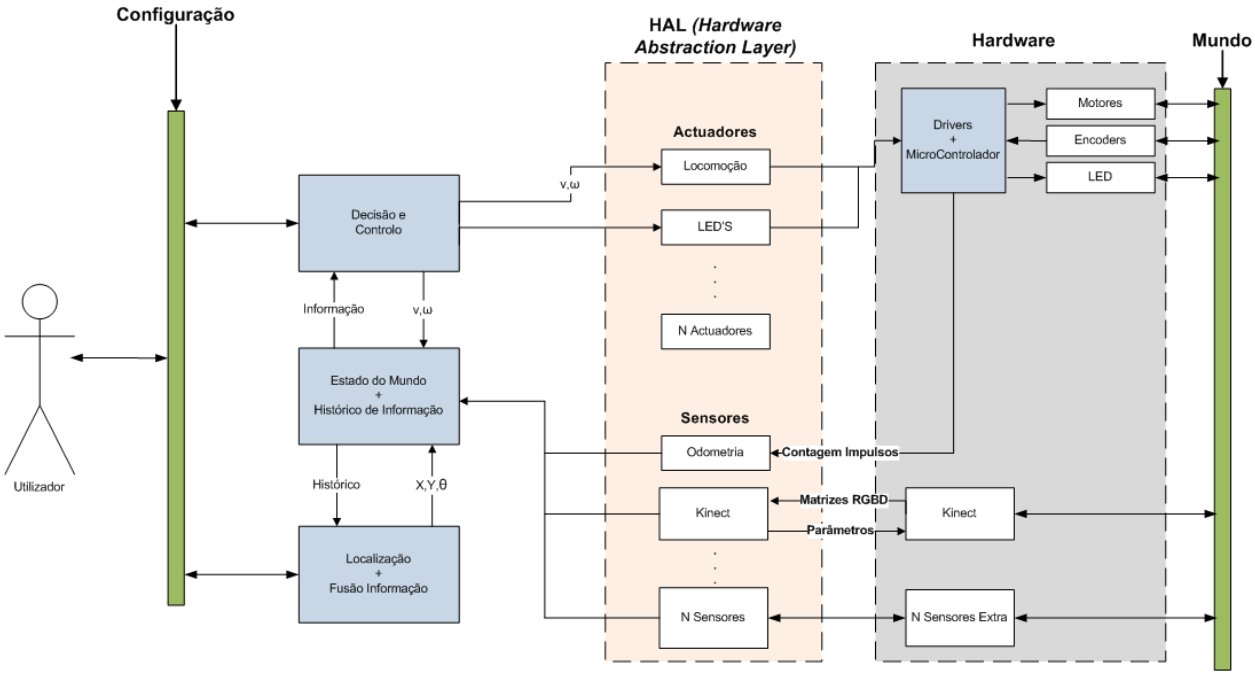


Figura 3.21: Arquitetura de Software de decisão e Controlo

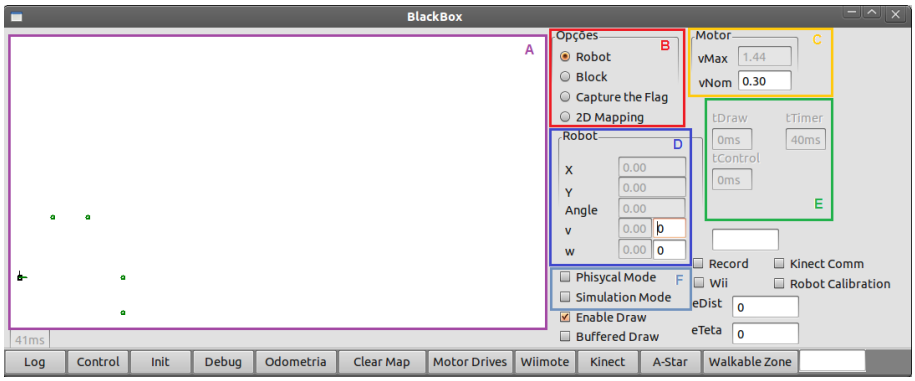


Figura 3.22: Software de decisão e controlo

bloco através do *click* do rato no mapa, caso estes estiverem selecionados. Também é permitido definir uma *Flag* em qualquer ponto do mapa como ponto de destino para o robô gerando desta forma, no momento que é criada, uma trajetória de caminho mínimo, baseado no algoritmo A*[34] até esse ponto de destino desviando-se de possíveis obstáculos no caminho. Quando a opção de representação 2D está selecionada é feito um mapeamento do mundo através mecanismos de perceção do mundo presentes no robô.

Em *C* é possível definir a velocidade máxima e nominal de cada roda do robô. Em *D* encontra-se a secção de monitorização da posição e direção do robô bem como a sua velocidade linear e angular atuais. É possível ainda definir referências destas mesmas velocidades, que tem como

destinatário os motores, passando em primeiro lugar pela camada de abstração de hardware que se encarrega da comunicação com os atuadores.

O software de iteração com o utilizador permite ainda informar o tempo atual de cada ciclo de controlo, bem como o tempo a ser despendido na atualização do mapa.

Em *F* é possível definir o modo de operação do software. O modo físico permite que a posição e orientação do robô no mapa seja atualizada através da informação do sistema de odometria, enquanto no modo de simulação a sua posição e orientação são definidas diretamente pela velocidade pretendida. A seleção de um destes modos permite o acionamento do controlo de trajetórias caso estas estejam presentes na lista de tarefas do robô. Os tipos de trajetórias presentes, passíveis de serem percorridas pelo robô, são do tipo linear, circular, pontual e angular. Para o controlo de trajetórias é ainda possível definir os parâmetros do controlador linear PID, como ilustrado na figura 3.23.

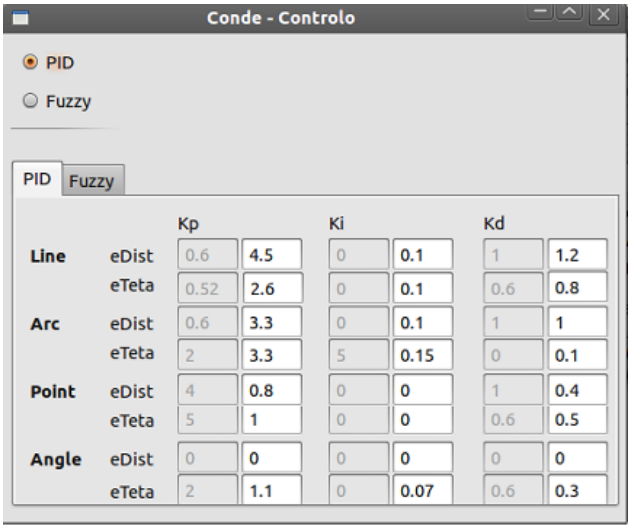


Figura 3.23: Parâmetros de configuração do controlador PID do software de decisão e Controlo

É possível ainda controlar o robô através do comando wiimote. O emparelhamento entre o PC e o comando é feito através do botão 'Wiimote' e o modo de funcionamento com o wiimote é ativado validando a caixa 'wii'. Neste momento o robô responde apenas às ordens do comando remoto.

A perceção do mundo através do *kinect* pode ser acionado a qualquer instante (Validando caixa 'Kinect Comm' na figura 3.22) mas apenas quando o modo físico estiver ativo. A atualização da posição do robô através do sistema de localização por balizas é ativada validando a opção 'Robot Calibration', caso esta esteja inativa o robô apenas utiliza a odometria para se localizar no mundo.

O trajeto percorrido pelo robô pode ainda ser gravado, através da opção 'Record', de forma a verificar o comportamento do robô ao longo de uma trajetória. Dependendo do resultado é possível atuar em conformidade afinando os parâmetros dos controladores de trajetórias como ilustrado na figura 3.23.

A criação de ficheiros com registos temporais (*Logs*) de um dado parâmetro pode ser realizado recorrendo à janela ilustrada na figura 3.24.

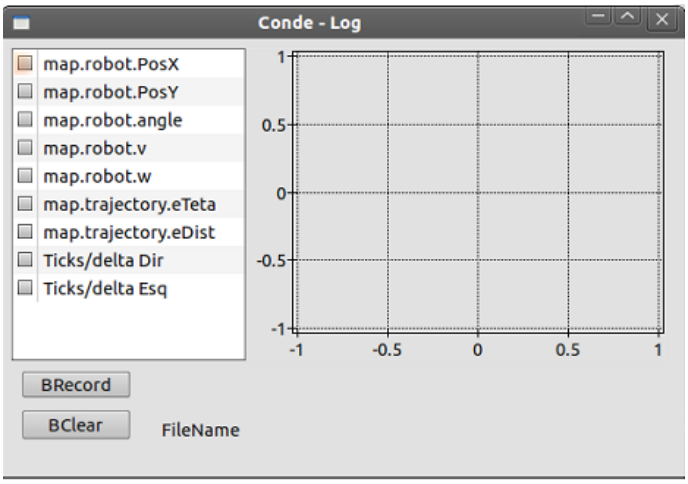


Figura 3.24: Janela de criação de *Logs* do software de decisão e Controlo

É também permitido estabelecer uma comunicação direta com o software que interage com os sensores de codificação e motores, para monitorização dos impulsos do *encoder* de cada roda e atuação nos motores. A utilização destas opções é exclusivamente para efeitos de calibração e teste do sistema de odometria (figura 3.25).

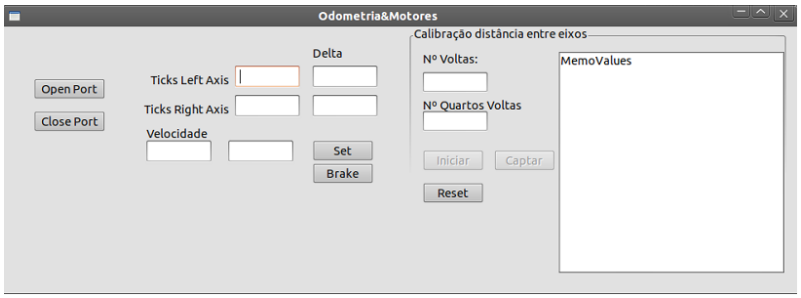


Figura 3.25: Odometria para o software de decisão e Controlo

A rotina de pintura no ecrã pode ser interrompida através da desativação da opção 'Enable Draw' (figura 3.22). Esta opção é útil de forma a tornar mais rápido o processamento da aplicação mas impede o acompanhamento da evolução do sistema.

3.3.4 Software de Interação com o *kinect* (*APPkinect*)

De modo a cumprir as necessidades do sistema, a aplicação que comunica com o *kinect* deverá cumprir os seguintes requisitos:

- Ter a capacidade de comunicar com o sensor RGBDepth e receber dados provenientes da imagem RGB e Depth;

- Ter a capacidade de detetar objetos, retornando a sua posição em coordenadas cartesianas xy em relação ao referencial do robô, bem como as suas dimensões estimadas;
- Reconhecer esferas e cilindros como balizas, retornando a sua posição em coordenadas cartesianas xy em relação ao referencial do robô, bem como as suas dimensões;
- Enviar por UDP as informações devidamente etiquetadas relativamente aos objetos encontrados;
- Ter um processamento em tempo real para permitir ao software de decisão atuar no robô em conformidade.

A aplicação que comunica com o *kinect* é desenvolvida em C++ e comunica com a aplicação de decisão através de sockets UDP. O fluxo de dados trocados entre as duas aplicações é unidirecional. O tipo de dados depende do que for detetado pelo *kinect*. Caso se trate de um objeto desconhecido é enviada a sua posição em relação ao robô bem como as suas dimensões 2D. Caso se tratem de esferas ou cilindros (objetos conhecidos), podem ser enviados dois tipos de informações:

- Informação composta - Cada mensagem contém informação de dois objetos (podendo ser esferas, cilindros ou mistos). Desta forma a posição deles no mundo já é conhecida a priori, apenas é necessário referir a que conjunto pertencem e as suas posições relativas ao robô. Este tipo de informação é utilizada para efeitos de localização;
- Informação singular - Cada mensagem contém o tipo de objeto (esfera ou cilindro), a sua posição relativa e as suas características (centro e raio para as esferas e centro, raio e altura para os cilindros). Este tipo de informação pode servir futuramente para definir um ponto de destino num determinado percurso do robô.

O diagrama correspondente à aplicação que interage com o *kinect* está ilustrado na figura 3.26.

Como se pode verificar pela figura 3.26, após o arranque da aplicação é feita uma inicialização do sistema, em que os parâmetros intrínsecos e extrínsecos do da câmara de profundidade do *kinect* são inicializados, assim como todas as variáveis relevantes para o desenvolvimento do processo. Durante o processo de inicialização é também efetuada a importação de uma imagem padrão previamente gravada numa diretoria à escolha do utilizador. A ciência envolvida na utilização de uma imagem padrão será analisada mais à frente neste relatório na secção de deteção de objetos (capítulo 5).

Após a inicialização do sistema, o processo entra num ciclo infinito, permanecendo neste estado até que seja dado ordem pelo utilizador para sair do programa.

Durante o processo cíclico o programa fica à espera que haja uma chamada do sistema operativo (*CallBack*) a informar que existe uma nova imagem a ser analisada. Após a chegada de uma nova imagem é verifica-se se existe um pedido do utilizador para gravar essa nova imagem como uma imagem padrão, substituindo a anterior. Se não então procede-se de imediato à análise da imagem.

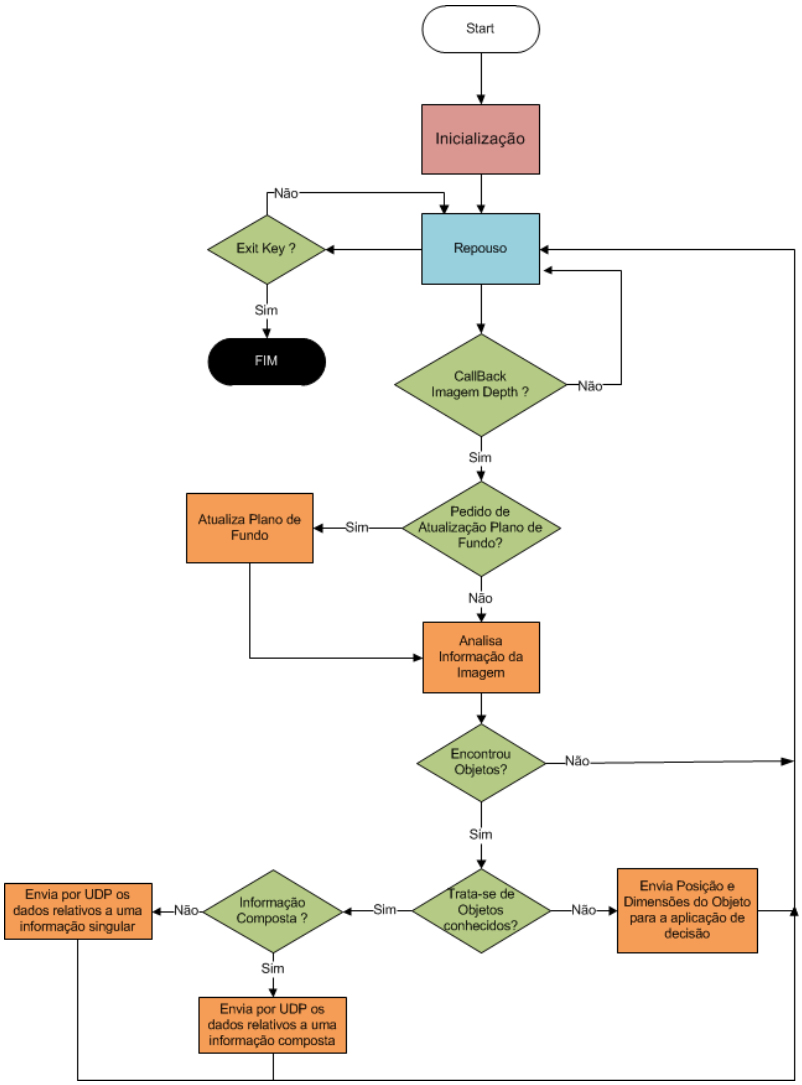


Figura 3.26: Fluxograma do funcionamento da aplicação de interação com o *kinect*

O foco da aplicação insere-se no bloco onde é analisada toda a informação da imagem. Neste bloco procede-se a todo um processo de filtragem para evitar ruídos na imagem até que se obtenha grupos de informação bem definidos. Todos os pixels da imagem são submetidos a uma transformação de coordenadas para coordenadas do mundo (X,Y,Z). Esses grupos são depois submetidos a um tratamento rigoroso de modo a verificar se algum deles correspondem a objetos conhecidos (esferas ou cilindros) ou um simples objeto desconhecido/obstáculo. Ambos são enviados para o sistema de decisão devidamente etiquetados. O processo de separação e agregação dos grupos de pontos 3D são abordado no capítulo 5, enquanto que os algoritmos de identificação de esferas e cilindros estão explicados no capítulo 6.

A aplicação depois de enviar as informações para o sistema de decisão volta à etapa "Repouso"onde aguarda a receção de uma nova imagem.

3.3.4.1 Sequência de Operações no software de Decisão após a receção de informação proveniente do módulo de abstração de hardware *APPkinect*

Nesta secção será demonstrado o processo implementado na aplicação de decisão referente à fase de receção da informação proveniente do *kinect*, da análise dessa informação e atualização das tarefas do robô.

Na figura 3.27 está representado o diagrama de sequência de operações desde o momento que a informação é recebida até ao momento em que as tarefas são atualizadas.

Após o arranque da aplicação de decisão, é feita a seguinte inicialização dos parâmetros do sistema:

- Parâmetros PID dos controladores de trajetórias;
- Limites de velocidade do robô;
- Velocidade Inicial do robô (0 m/s);
- Nº de objetos (obstáculos e *beacons*/balizas) presentes no mapa (0 no início);
- Valor da confiança da presença de objetos no mapa (inicia-se a 0, sendo que a confiança incrementa sempre que chegar informação de um objeto na proximidade dos existentes);
- Nº trajetórias a serem percorridas pelo robô (0 no início);
- Inicialização do algoritmo A* com cada elemento da grelha a 0 (cada elemento da grelha/mapa possui informação da ocupação por parte do objeto assim como o gradiente de suavização na sua proximidade (valor de 0 a 255, sendo que 0 corresponde a um espaço/célula livre e 255 a uma espaço/célula ocupado));
- Posição inicial do robô;
- Parâmetros da comunicação UDP, como número da porta e IP (*Internet Protocol*) (neste caso é *localhost* porque se trata de comunicação entre aplicações no interior da mesma máquina/PC);
- Todas as restantes variáveis relevantes ao processo ao correto funcionamento do sistema.

Após a inicialização do sistema o módulo só desperta novamente na chegada de um pacote UDP. Após a receção dessa mensagem procede-se à sua descodificação para verificar tratar-se de um par de objetos conhecidos para efeitos de localização, um objeto conhecido para ser utilizado como ponto de destino ou simplesmente de um objeto desconhecido. Dependendo do conteúdo da informação o módulo aplica diferentes ações. Caso se trate de um objeto desconhecido, analisa-se a sua posição e as suas dimensões e coloca-se na grelha/mapa esse espaço como ocupado. Neste momento o robô sabe que existem zonas do mapa que não pode percorrer mas permanece imóvel pois ainda não sabe se existe algum ponto de destino para o qual ele se possa deslocar. Após o programa ter conhecimento que existe um ponto de destino no mundo é gerada uma trajetória

através do algoritmo A* até esse ponto tendo em consideração as células ocupadas do mapa. Os objetos para efeitos de localização são encontrados aos pares podendo ser constituídos por uma esfera e um cilindro ou até serem os dois da mesma categoria. Estes objetos têm posições conhecidas a priori, sendo assim a única informação necessária será saber a que par de objetos pertencem e quais as suas posições relativamente ao robô.

O percurso gerado pelo algoritmo A* é composto por um conjunto de células/pontos interligados numa grelha/mapa desde um ponto de início até um ponto final. Depois de obtidos esses pontos é possível definir o melhor tipo de trajetória que os una. Utilizou-se trajetórias lineares, pelo baixo peso computacional (comparativamente com trajetórias circulares) e pela facilidade em definir o tamanho do segmento de reta que unem os pontos.

Desta forma o robô está pronto a seguir o percurso gerado. A única verificação que o sistema de decisão tem que efetuar é o modo de operação atual. Se o sistema estiver em modo de simulação a posição atual é definida tendo em consideração as velocidades linear e angular pretendidas e um ciclo de controlo bem definido, enquanto se o sistema se encontrar em modo físico a sua posição é obtida recorrendo à informação recebida pelos *encoders*.

O controlador de trajetória linear utilizado consiste em corrigir a distância e o ângulo do robô face a cada segmento de reta em que ele se encontra. As expressões desses erros bem como o fluxo grama do controlador encontram-se explicados no anexo A.

3.3.5 Software da Camada de Interface

Esta aplicação tem como objetivo enviar para a aplicação de decisão a quantidade de impulsos obtidos pelos *encoders* ocorridos entre a medida anterior e atual de modo a permitir ao sistema de representação, atualizar a posição do robô no mundo com base no sistema de odometria. Estes dados são enviados a uma cadência de 40 ms que coincide também com a frequência de controlo da aplicação de decisão. A aplicação da camada de interface tem também como objetivo aplicar, nos drivers de controlo dos motores, o sinal de comando (PWM) necessário para produzir um determinado movimento do robô pretendido pelo sistema de decisão e controlo.

De modo a cumprir as necessidades e os objetivos do sistema, a aplicação que comunica com o módulo da camada de interface deverá cumprir os seguintes requisitos:

- Ter a capacidade de receber dados do sistema de decisão relativamente às referências de velocidade a aplicar a cada motor e aplicar o respetivo PWM a cada driver;
- Ter a capacidade de receber a informação dos *encoders* e efetuar a contagem dos impulsos e enviar, a cada ciclo de programa, o resultado da diferença entre a contagem enviada anteriormente e a atual para o sistema de decisão no mesmo meio de comunicação;
- Ser capaz de não perder nenhuma informação relativamente às transições das fases do *encoder*.

A aplicação de nível de interface está ilustrada na figura 3.28, onde é explicada a sua sequência funcional.

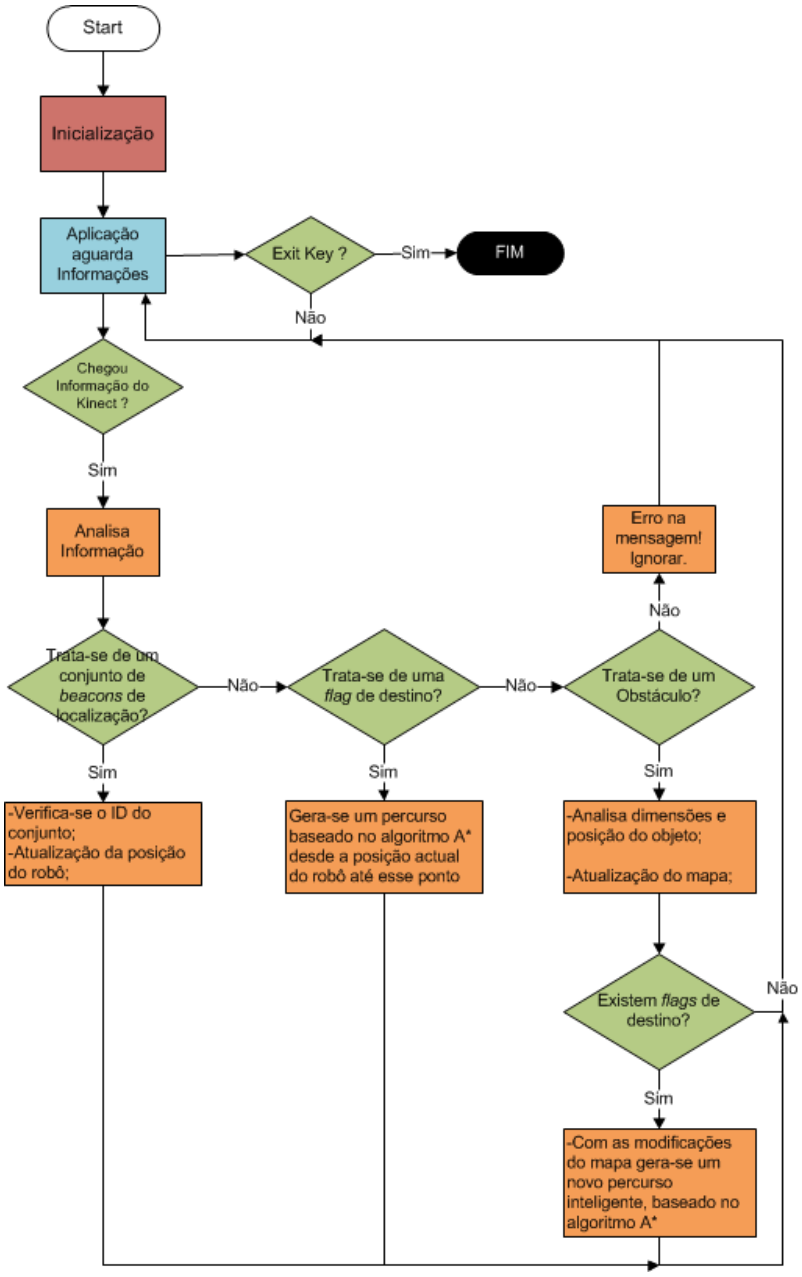


Figura 3.27: Fluxo grama da interação entre o módulo de estado do mundo e o *kinect*

Após o arranque da aplicação, procede-se a uma inicialização de todos os componentes relevantes para o seu funcionamento. Neste estado são definidos os pinos digitais do micro controlador a serem utilizados bem como especificar se são pinos de entrada ou saída. Os sinais lógicos dos *encoders* estão conectados às portas de interrupção externa, uma vez que é necessário monitorizar cada alteração da sequência dos seus sinais. Estes pinos serão inicializados como entrada e os sinais de comando (PWM) para os drivers de controlo dos motores serão atribuídos como saída. Para cada porta de saída do sinal PWM é definido a frequência da portadora. A frequência a definir será a frequência de comutação admissível para os transístores presentes nos drivers de

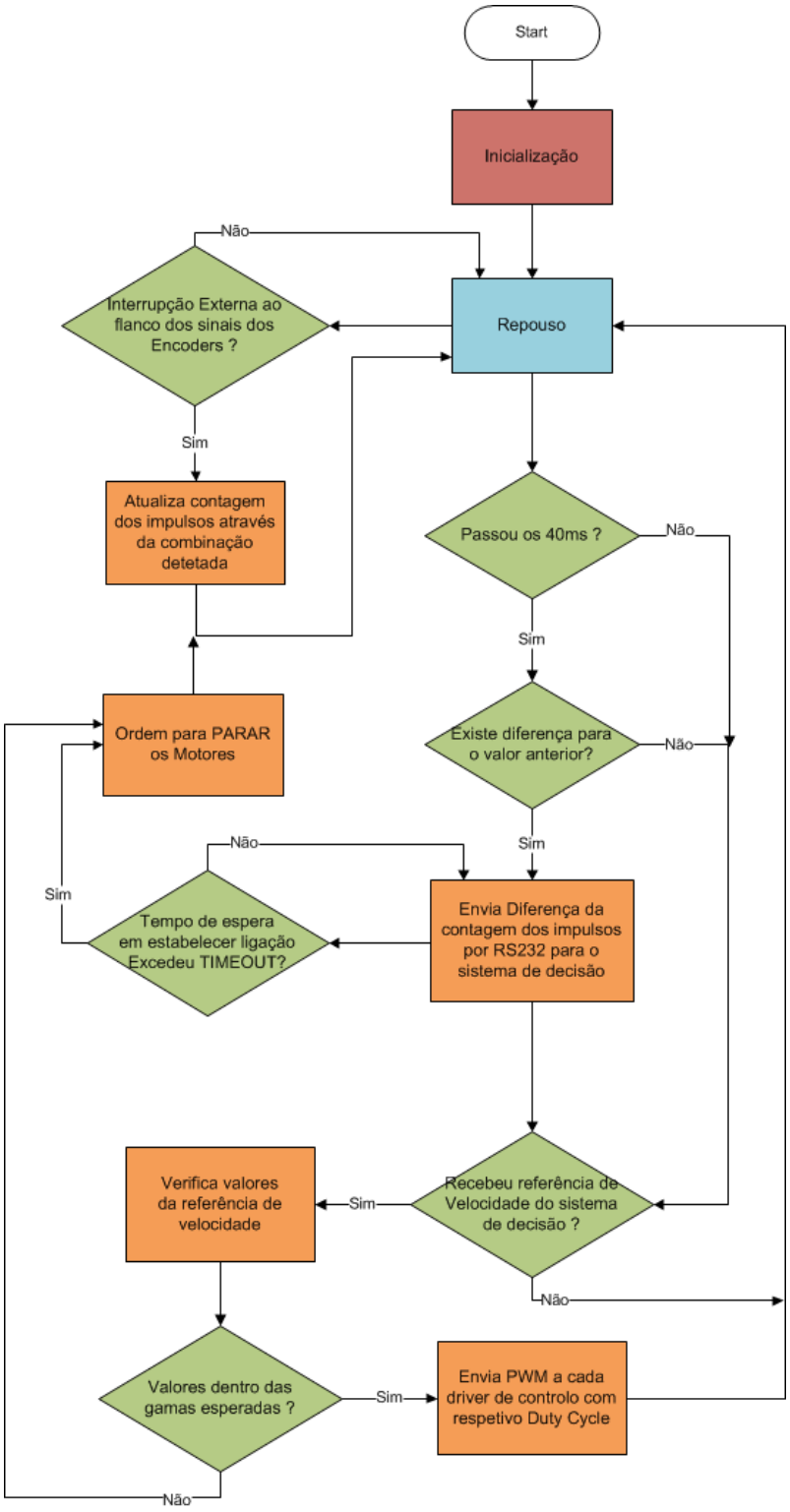


Figura 3.28: Fluxo grama do funcionamento da aplicação de camada de interface

controlo. Variáveis auxiliares, como variáveis de acumulação de impulsos dos *encoders* ou auxiliares de estado de uma sequência de operações também será serão inicializadas assim como a

definição do *BaudRate* da porta série (RS-232) para uma comunicação bidireccional com o PC (*Personal Computer*). Por questões de segurança, sempre que a aplicação é iniciada, é aplicada referência de PWM com o valor 0 de modo a imobilizar por completo o robô.

No estado 'Repouso', aguarda-se que qualquer um dos eventos do sistema seja atuado.

Podem ocorrer três situações, como por exemplo, interrupções externas provenientes dos sinais dos *encoders*, o temporizador disparar a uma cadência de 40 ms para enviar periodicamente dados para o PC ou a chegada de informação proveniente do PC contendo a referência de PWM a aplicar a cada um dos drivers de controlo dos motores.

Definição de Prioridades entre eventos

Naturalmente quando existe a possibilidade de ocorrência de eventos concorrentes, como é o caso, surge a necessidade de se estabelecer prioridades entre os eventos. Nesta aplicação utilizam-se dois tipos de interrupções, uma externa e outra por temporização. Se estes dois tipos de eventos ocorrerem ao mesmo tempo, o sistema por imposição atribui maior prioridade à interrupção externa. O seguimento da sequência de impulsos dos *encoders* é de extrema importância e naturalmente prioritário em relação aos restantes eventos de modo a não prejudicar a localização do robô com base na odometria.

Em relação à mensagem a ser recebida proveniente do PC é atribuída a menor prioridade dos três eventos, uma vez que esta pode ser retida no buffer da porta série e ser utilizada mais tarde. Para que não se perca a validade da mensagem, esta tem que ser considerada antes da receção da próxima.

Análise do processo quando ocorrem Interrupções Externas

Sempre que ocorra uma interrupção externa, é feita uma análise do estado dos dois sinais lógicos das fases A e B dos *encoders* de modo a determinar a sua sequência de movimento atual.

A chamada de interrupção é feita à subida do flanco tanto na fase A, quando se pretende detetar o movimento no sentido dos ponteiros do relógio, como na fase B do *encoder*, quando se pretende detetar o movimento no sentido contrário aos ponteiros do relógio. Quando ocorrer uma interrupção de uma das fases, é analisado o estado da outra fase de acordo com a figura 3.10. Este procedimento permite detetar o sentido de rotação do motor.

Analisando em maior detalhe a figura 3.10, quando se está perante a situação ilustrada do lado esquerdo (sentido contrário aos ponteiros do relógio), deteta-se a subida do flanco da fase A e analisa-se, posteriormente, o estado da fase B, se esta estiver a nível lógico baixo, como é o caso, é feito o decremento da contagem global dos impulsos, o que significa que o motor está a rodar no sentido contrário aos ponteiros do relógio. Quando se está perante a situação ilustrada no lado direito da figura (sentido dos ponteiros do relógio) inverte-se os papéis e deteta-se a subida do flanco da fase B e analisa-se, posteriormente, o estado da fase A, se estiver a nível lógico baixo, como é o caso, é feito o incremento da contagem global dos impulsos, o que significa que o motor está a rodar no sentido dos ponteiros do relógio.

Análise do processo quando ocorre Interrupção por Temporização

Quando o sistema detetar uma interrupção por temporização, concedida a cada 40 ms, é feita uma verificação de cada variável global que armazena os impulsos totais de cada roda e verificar se existe alguma alteração face à verificação anterior, caso haja o resultado é enviado para o PC através do protocolo RS-232 (3.3.1) para ser processada pela aplicação de decisão. Caso não haja qualquer alteração, poderá significar que o robô se encontra parado e nesse caso evita-se enviar informação redundante.

Análise do processo após receção de mensagens do PC

Quando o sistema deteta a receção de uma mensagem por RS-232 proveniente do PC, esta é descodificada e processada. Os valores apenas serão considerados se e só se estiverem dentro das gamas esperadas, ou seja, referências de PWM entre 0 e 255 (8 bits). Caso esteja tudo em conformidade então é gerado o sinal de comando com o *Duty Cycle* adequado, com base na referência recebida, para o driver de controlo respetivo (nível baixo) de forma a produzir o movimento diferencial pretendido pela unidade de alto nível.

O driver de controlo por seu lado é responsável por manter o motor à velocidade de referência pretendida.

Capítulo 4

Sistema Sensorial para DCA

4.1 Sensor *kinect*

Nesta secção será apresentado em maior detalhe o sensor *kinect*. Na secção 4.1.1 é explicado o seu principio de funcionamento, na secção 4.1.2 serão apresentadas algumas das limitações do sensor e na secção 4.1.3 será demonstrado um método de determinar o *PinHole Model* da câmara. Por fim na secção 4.1.4 será desenvolvido um método de conversão do sinal recebido pela câmara *depth* para metros.

4.1.1 Principio de Funcionamento

O sensor *Depth* do *kinect* foi durante algum tempo especulado sobre o seu real funcionamento.

Inicialmente a Microsoft revelou que o seu funcionamento era baseado no principio das câmaras ToF (*Time of Flight*), que corresponderia ao envio de um feixe IR e a sua distância calculada através do tempo que este demorava a ser refletido para o recetor. No entanto o seu real funcionamento é um pouco diferente, até porque os semicondutores utilizados no *kinect* são produtos COTS que não têm a capacidade extrair o tempo de voo de uma luz modulada.

O principio de funcionamento é baseado numa patente publicada pela *Prime Sense*, empresa que desenvolveu o equipamento. Essa patente revela que a tecnologia consiste no mapeamento de profundidade utilizando padrões projetados. Este método consiste na emissão de um feixe de luz IR através de um material transparente com a capacidade de a difundir (lente), que produz um padrão de pontos fixo (guardado em memória) que será projetado num determinado objeto. A distorção criada no padrão pelo objeto é então enviada para o recetor (câmara *Depth*) que computacionalmente determina qualquer diferença relativa ao padrão. Esta distorção é detetada comparando cada intensidade de luz do padrão com a que é devolvida, consistindo assim numa técnica de codificação por intensidade luz[35].

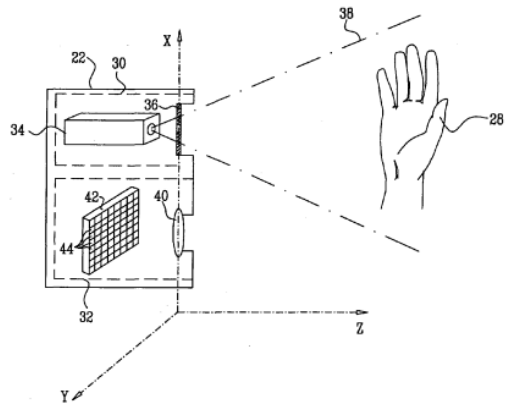


Figura 4.1: Principio de funcionamento do *kinect* [35]

4.1.2 Limitações do Sensor

O sensor *kinect*, face à natureza do seu funcionamento possui algumas limitações ao seu manuseamento.

Não é possível utiliza-lo em ambientes onde esteja presente a luz solar, pois a luz infravermelha proveniente do sol iria causar a saturação do sensor IR. Neste caso a luz visível pelo recetor não seria apenas do sensor IR mas também de outra fonte desconhecia, dificultando a técnica de descodificação por luz referida anteriormente. As interferências com a luz IR também estão presentes quando existem objetos refletores, transparentes ou objetos que absorvem sinais infravermelhos[36].

As limitações referidas anteriormente diminuiriam consideravelmente caso o ambiente que o sensor esteja inserido seja controlado, no que diz respeito à luminosidade e face à natureza do próprio material presente.

Outra limitação própria do sensor é o facto de um objeto causar ocultação a um outro objeto. Analisando a figura 4.2, quando existem 2 ou mais objetos no ambiente e um deles estiver na zona a sombreado indicada na figura, este não é detetado. Embora a câmara de profundidade consiga visualizar o objeto ocultado, este não reflete o sinal do emissor IR. Este fenómeno, assim como os referidos anteriormente, provocam na imagem zonas mortas, denominados como buracos negros[36], sem qualquer informação, dificultando a aquisição dos dados.

Uma outra limitação é o facto das imagem *Depth* e RGB não coincidirem uma com a outra, estando sempre sujeitas a uma calibração de forma a realizar uma correspondência entre as duas imagens.

Para o preenchimento dos buracos presentes na imagem do recetor IR existem tecnologias, baseadas em GPU (*Graphics Processing Unit*), utilizando estatísticas temporais preditivas capazes de o realizar em tempo real[36].

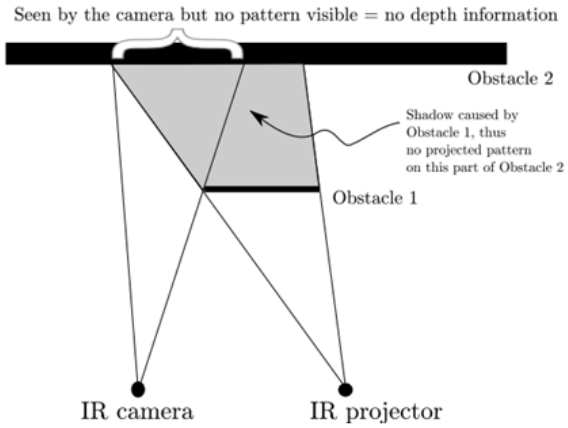


Figura 4.2: Limitação por ocultação de objetos [37]

4.1.3 *PinHole Model* do Sensor

O *PinHole Model* de uma câmara é simplesmente o modelo da câmara. Neste modelo é descrita a relação matemática entre um ponto 3D do mundo e a sua projeção num plano de imagem. Consiste num buraco muito pequeno por onde todos os raios de luz passam antes de serem invertidos pela lente e posteriormente projetados no plano de imagem. A relação entre a sua posição real e a posição projetada num plano de imagem é dada pela distância focal. Para mapear um ponto 3D do mundo real para um ponto 2D no plano de imagem é realizada uma projeção de acordo com a figura 4.3.

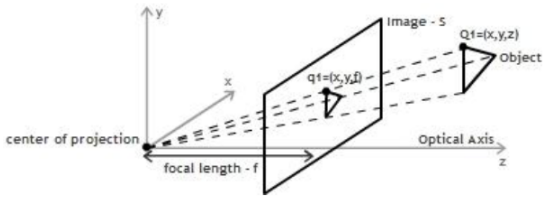


Figura 4.3: *PinHole Model* da câmara e do sensor *Depth* [7]

Na figura 4.3, um ponto Q_1 nas coordenadas do mundo é projetado no plano de imagem passando pelo centro de projeção, resultando no ponto q_1 no plano de imagem S. Como se pode verificar, a relação entre os dois triângulos similares pode ser usada para derivar a relação entre os pontos Q_1 e q_1 como se pode verificar pelas equações 4.1 e 4.2.

$$\frac{x}{f} = \frac{X}{Z} \tag{4.1}$$

$$\frac{y}{f} = \frac{Y}{Z} \tag{4.2}$$

De facto o centro de imagem do sensor não é normalmente no eixo ótico da câmara, o que leva à introdução de dois novos parâmetros, c_x e c_y para modelizar essa diferença. Existe também dois valores diferentes para a distância focal, uma por cada eixo de coordenadas. Este procedimento deve-se ao facto de os pixels numa câmara são normalmente de forma retangular em vez de quadrangular[7]. Estes novos parâmetros introduzem alterações nas equações 4.1 e 4.2, como se pode verificar nas equações 4.3 e 4.4.

$$x = f_x \frac{X}{Z} + c_x \tag{4.3}$$

$$y = f_y \frac{Y}{Z} + c_y \tag{4.4}$$

Neste projeto utilizou-se uma câmara de profundidade em que cada valor da imagem, $f(x,y)$, tem correspondência direta com a coordenada Z do objeto (distância da câmara a um ponto do objeto). Conhecendo x *e* y, coordenadas do plano de imagem do ponto $f(x,y)$, determina-se facilmente as coordenadas X e Y no mundo de um ponto do objeto utilizando as equações 4.3 e 4.4, respetivamente.

Os valores utilizados para os parâmetros intrínsecos da câmara de profundidade estão representados em [38].

É necessário ter também atenção aos parâmetros extrínsecos da câmara, neste caso a sua disposição relativamente ao robô. Neste projeto como referido no capítulo 3 o *kinect* está colocado a 40 cm do solo e alinhado com o eixo dos Y. Desta forma como o *kinect* se encontra deslocado apenas de h=0.40 m no sentido do eixo dos Y face à origem do referência do robô, o parâmetro h terá de ser incluído na equação 4.4, sendo o valor real de y dado pela equação 4.5.

$$y = f_y \frac{Y + h}{Z} + c_y \tag{4.5}$$

4.1.4 Homografia

A câmara de profundidade do sensor *kinect* produz uma saída de gama 0 a 2047 por cada pixel de imagem. Este é então o valor que é necessário converter, de forma a obter a informação da distância na unidade de sistema internacional, metros. Após vários testes foi possível verificar que a relação entre os valores da gama original e a distância real é linear mas inversamente proporcional.

Na figura 4.4 é possível verificar os testes realizados, variando linearmente a distância. Como esta é uma relação inversa entre estes dois parâmetros, de forma a obter o valor real em metros em cada pixel, dado um valor entre 0 e 2047 será feito de acordo com a equação 4.6.

$$dist_{metros} = \frac{1}{f(x)} \tag{4.6}$$

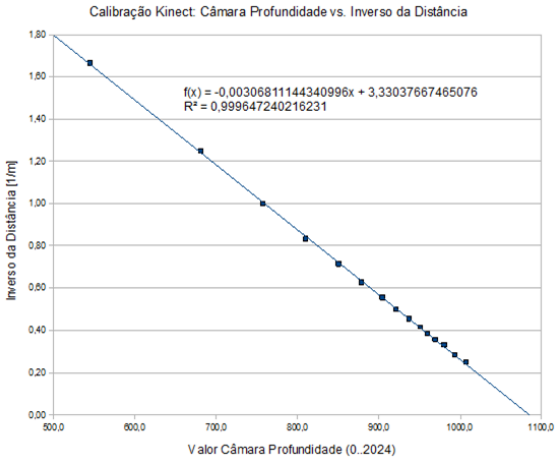


Figura 4.4: Testes de calibração do sensor *kinect*

4.1.5 Conclusões

Apesar de todas as limitações, o *kinect* trata-se de boa solução para quem procura uma ótima relação qualidade/preço. A chegada do *kinect* veio quebrar um ciclo que até agora pertencia a equipamentos de elevado custo, interessantes embora ao alcance de poucos projetos científicos. A utilização de lasers range finder para mapeamento 3D do mundo era um mito até à chegada do sensor da Microsoft.

Os mais críticos podem apontar inúmeras falhas ao seu manuseamento mas por vezes esquecem-se que este sensor foi concebido para operar em zonas onde certos parâmetros estão bastante controlados, como a luminosidade e o espaço.

A precisão nas suas medições são bastante razoáveis e a correspondência entre o sinal devolvido pelo recetor e a distância real são de simples obtenção. Consequentemente a trigonometria envolvida na conversão de um ponto da imagem para coordenadas XYZ proporciona uma boa e simples abordagem no manuseamento do equipamento.

O sensor *kinect* atualiza as imagens *Depth* e RGB a uma frequência de 30 Hz e a resolução das imagens é de 640 x 480. Em particular a câmara de profundidade providencia a cada elemento da matriz uma sensibilidade de 2048 níveis (11-bits) de profundidade. O sensor tem um ângulo de visão de 57º na horizontal e 43º na vertical, podendo obter uma variação na vertical até 27º quando utilizado o pequeno servo motor acoplado. Este sensor aplicado à consola Xbox apresenta

um alcance prático de 1.2 m até 3.5 m o que limita a ação a uma área de 6 m^2, embora ele permita um rastreamento mais alargado (0.7 m até 6 m) mas também com um maior ruído e dificuldade nas medidas. À distância mínima o sensor tem um alcance máximo de 87 cm na horizontal e 63 cm na vertical, o que faz como que o sensor apresente uma resolução de 1.3 mm por pixel.

4.2 Odometria

4.2.1 Abordagem geral

Odometria é um dos métodos mais utilizados na determinação momentânea da posição de um robô. Consiste na integração da informação incremental do movimento linear das rodas ao longo do tempo, de forma a medir o deslocamento efetuado pelo robô.

Na secção 4.1 será apresentado o modelo da odometria; na secção 4.2 será discutida a questão dos erros sistemáticos que afetam a odometria, e será ainda descrito um método de calibração da mesma; a secção 4.3 abordada os erros aleatórios e apresenta uma forma de quantificar a certeza da medida da pose, através de um modelo do erro da odometria.

4.2.2 Limitações e Desvantagens do sistema de Odometria

A principal limitação deste método deve-se à acumulação de erro ao longo da trajetória fazendo a medida da posição divergir do seu valor real. Isto porque, o deslocamento do robô é obtido a partir da integração das medidas do deslocamento linear de cada roda do robô, medidas essas afetadas por erros que serão também integrados ao longo do tempo. Os erros que afetam a odometria podem ser tanto de natureza sistemática como aleatória.

Erros não sistemáticos

Erros não sistemáticos são erros provocados por características imprevisíveis no ambiente que o robô está inserido, como pavimentos irregulares e escorregadios. Esta natureza de situações pode traduzir complicações no cálculo final da posição global do robô, visto que a odometria parte do pressuposto que as rotações das rodas podem ser traduzidas em movimentos lineares em relação ao solo[15].

Movimentos verticais provocados pelos pavimentos irregulares podem informar o programa de controlo que o robô andou mais do que na realidade, assim como os pisos escorregadios, uma vez que o robô permanece no mesmo sítio embora a roda continue a rodar.

Erros sistemáticos

Embora os erros não sistemáticos possam de alguma forma ser controlados com ambientes bem estruturados, sem imprevisibilidades, existem erros sistemáticos fruto de uma construção mais descuidada do robô, com rodas mal alinhadas, rodas com diâmetros diferentes (desgaste ou

defeito), folgas entre as engrenagens dos motores ou dificuldade em detetar o ponto de contacto das rodas no solo.

Geralmente este tipo de erros não se alteram significativamente ao longo do tempo, mas alterações da carga que o robô transporta, ou a sua distribuição, podem alterar a deformação dos pneus e alterar tanto o diâmetro das rodas como o ponto de contacto das rodas com o chão, e consequentemente alterar o valor da distância entre eixos (parâmetro b da figura 4.6)[16].

O efeito dos erros sistemáticos é particularmente importante, pois, sendo estes mais ou menos constantes e estando integrados juntamente com as medidas ao longo do tempo, o seu efeito está a cada período de amostragem a ser adicionado à medida da posição do robô[15]. Tratando-se de erros sistemáticos é possível reduzir o seu efeito recorrendo a métodos de calibração.

Efeito de cada tipo de erros ao longo de uma trajetória

Na figura 4.5 é possível identificar individualmente o efeito provocado por cada um dos erros sistemáticos referidos anteriormente. No quadrado que sofre uma ligeira curvatura corresponde ao erro provocado por rodas com diferentes diâmetros, visto que ao longo de uma trajetória retilínea se uma roda possuir um diâmetro maior que o diâmetro da outra roda, o robô terá tendência a curvar, afastando-se do percurso definido. O efeito provocado pela incerteza no ponto de contacto das rodas é apenas quando o robô efetua uma rotação. Por exemplo num robô diferencial em que a expressão da velocidade angular do mesmo depende diretamente da distância entre eixos, se for em seguida enviada uma ordem para o mesmo baseado nesta informação e se não estiver bem calibrado, o robô irá rodar mais ou menos dependendo do valor.

Para evitar o problema da incerteza do ponto de contacto de cada uma das rodas, deve-se utilizar rodas com pneus finos. Estes tipos de erros são acumulativos ao longo de uma trajetória, podendo ao final de 10 metros, ou até menos, o robô apresentar-se numa posição completamente diferente. A certeza na posição do robô obtida depende do grau de precisão que o sistema de odometria foi submetido. Sendo um método dependente de medidas anteriores poderá inserir erros sistemáticos nas equações de cinemática do robô.

4.2.3 Vantagens do sistema de Odometria

Apesar das limitações associadas ao sistema de odometria, esta proporciona uma boa precisão a curto prazo, é económica de implementar e permite taxas de amostragem muito altas. Muitas vezes, na prática, o método utilizado consiste em ter a odometria a medir a posição atual do robô e existir outro método de localização, como uma câmara, que periodicamente corrige a posição obtida pela odometria do robô.

4.2.4 Modelo de Odometria

Como se pode verificar pela figura 4.6, a posição de um robô é definida pela sua posição (x, y) e orientação (θ) em relação ao um referencial cartesiano. U_e, U_d, D_e e D_d são o deslocamento e o

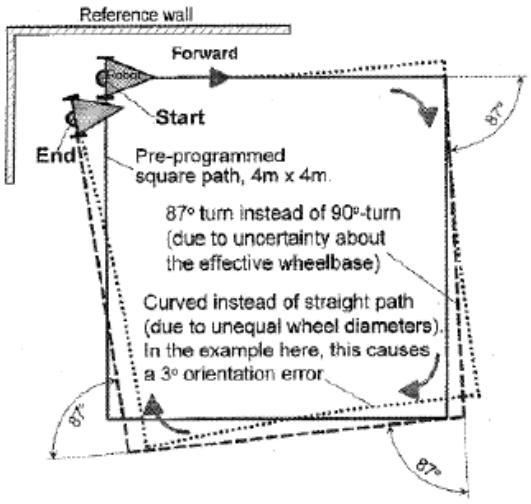


Figura 4.5: Teste do Quadrado com o efeito de cada erro sistemático[16]

diâmetro da roda esquerda e direita respetivamente (em metros), U o deslocamento linear efetuado pelo robô.

O modelo da odometria permite, a partir da informação dada periodicamente pelos *encoders* de cada roda, $N_{e/d}$ (i), medir a posição do robô entre intervalos de amostragem, i. A relação entre $N_{e/d}$ (i) e o deslocamento de cada roda efetuado no período de amostragem i, $\Delta U_{e/d}$ (i), é dada pela seguintes expressões:

$$c_{e/d} = \frac{\pi * D_{e/d}}{n * R_e} [15] \tag{4.7}$$

$$\Delta U_{e/d}(i) = c_{e/d} * N_{e/d}(i) [15] \tag{4.8}$$

Em que $c_{e/d}$ é o fator de conversão entre os impulsos dos *encoders* e o deslocamento horizontal da respetiva roda, n é a relação da caixa redutora e R_e a resolução dos *encoders*.

A variação de deslocamento linear efetuado no período de amostragem i pelo robô, U (i), está representada na expressão 4.9 e a variação da rotação na expressão 4.10, em que b é a distância entre cada ponto de apoio das rodas no solo.

$$\Delta U(i) = \frac{\Delta U_d(i) + \Delta U_e(i)}{2} [15] \tag{4.9}$$

$$\Delta \theta(i) = \frac{\Delta U_d(i) - \Delta U_e(i)}{b} [15] \tag{4.10}$$

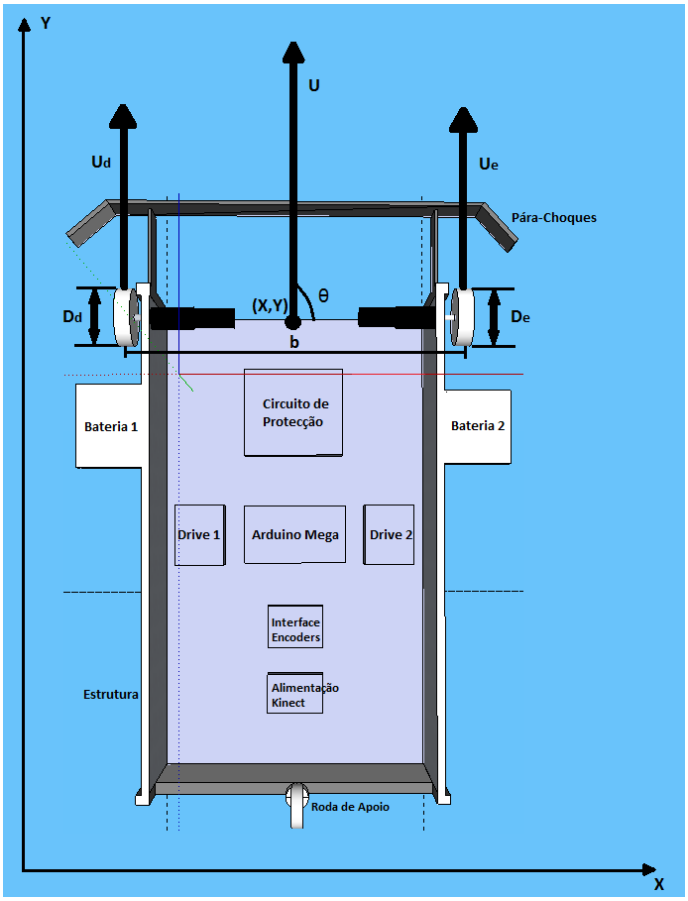


Figura 4.6: Robô diferencial com parâmetros de odometria (Visto de baixo)

Para obter uma correspondência entre o número de impulsos efetuados por cada roda e os metros percorridos pela mesma é necessário determinar o número de impulsos por metro. Para tal colocou-se o robô a movimentar-se em linha reta até uma distância de 4,66 m e guardou-se os valores dos impulsos de cada roda. Este teste foi feito 5 vezes e após uma média desses valores obteve-se para a roda esquerda 31096 impulsos por metro e para a roda direita 31099 impulsos por metro. Através destes valores é possível determinar o diâmetro de cada roda recorrendo às expressões 4.7 e 4.8, em que para a roda esquerda obtém-se 6,4223 cm e para a roda direita, 6,4217 cm.

Para a obtenção do valor da distância entre os pontos de contacto das duas rodas, b, colocou-se o robô a rodar 5 vezes sob si próprio e recolhidos os valores finais dos impulsos de cada roda. Através da expressão 4.10 verifica-se que o valor de b é de 36,137 cm.

$$x(i+1) = x(i) + \Delta U(i)\cos(\theta(i) + \frac{\Delta\theta(i)}{2}) \tag{4.11}$$

$$y(i+1) = y(i) + \Delta U(i)\sin(\theta(i) + \frac{\Delta\theta(i)}{2}) \tag{4.12}$$

$$\theta(i+1) = \theta(i) + \Delta\theta(i)) \tag{4.13}$$

Após serem obtidos os valores de conversão entre o número de impulsos e o número de metros de cada roda e o valor de b é possível obter através das expressões 4.11, 4.12 e 4.13 a posição e a direção do robô.

4.2.4.1 Conclusões

A odometria é um método de localização relativa com boa precisão a curto prazo e de baixo custo. Apresenta naturalmente algumas limitações como a adulteração das medidas da posição do robô no mundo devido a imprevisibilidades do terreno e mecânica descuidada e não compensada. No entanto tendo conhecimento das principais fontes de erro esses problemas podem ser minimamente controlados.

Os testes de calibração utilizados neste capítulo são efetuados de forma empírica. Esta abordagem é necessária porque embora seja conhecida a resolução dos *encoders*, a multiplicação da caixa de velocidades e o tipo de roda utilizada, é sempre necessário não desprezar o desgaste das rodas (que afeta o diâmetro) e outros parâmetros importantes como a distância entre rodas (dificuldade em detetar o ponto de contacto das rodas no solo). Desta forma, a melhor abordagem para a conversão para metros do número de impulsos efetuados pelos *encoders* será deslocar a plataforma robótica x metros e realizar esse teste várias vezes. O resultado do valor médio desses testes será utilizado como valor base para a conversão dos impulsos para metros. Este tipo de testes é afetado consoante o tipo de piso, sendo conveniente que se realize num pavimento plano e com boa aderência de forma a tornar os testes mais fidedignos.

Capítulo 5

Deteção e Identificação de Objetos

Neste capítulo serão introduzidos métodos de isolamento e interpretação dos pontos obtidos pelo sistema de captação sensorial por profundidade (*kinect*), bem como técnicas de identificação de objetos como esferas e cilindros.

Na secção 5.1 é feita uma abordagem que permite detetar em que zona do mapa se encontram os objetos e determinar as suas dimensões nos três eixos cartesianos. Este processo é possível recorrendo inicialmente a um isolamento dos pontos da imagem. Após o processo de isolamento dos pontos 3D, na secção 5.2 serão então introduzidas técnicas que permitem a identificação dos objetos referidos anteriormente.

5.1 Deteção de Objetos

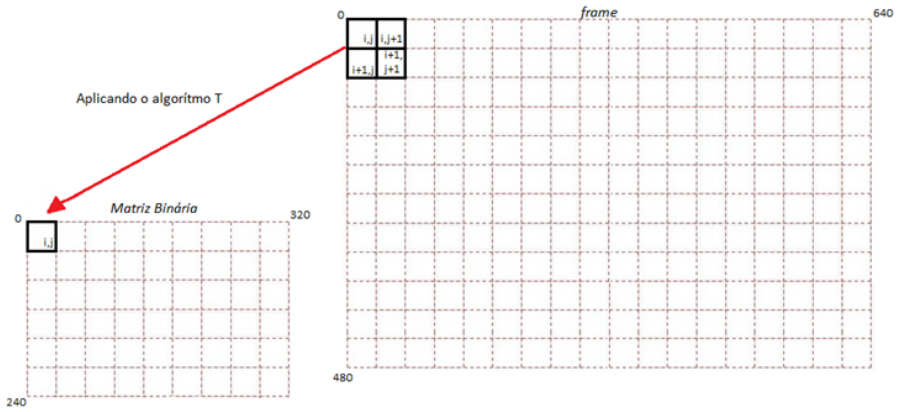
O método de deteção de objetos implementado consiste na subtração entre dois planos de imagem (*background subtraction*) em que um deles é definido como um padrão, idealmente uma imagem captada de uma cena vazia sem objetos presentes. A outra imagem é obtida e atualizada a uma determinada cadência (sempre que ocorra uma chamada de interrupção (*CallBack*) do sistema operativo a informar que existe uma nova frame) e tem o objetivo de informar o software de qualquer alteração relativa à imagem padrão. São estas alterações que se pretende analisar para posteriormente ser possível atribuir-lhes um significado.

De referir que o algoritmo desenvolvido apenas analisa a imagem mais recente, sendo assim quando estiver a ser processada uma imagem de profundidade e a determinado momento estiver uma nova imagem disponível, esta é descartada para não ser colocada em espera. Quando terminar o processamento, o algoritmo aguarda a chegada da imagem mais atual. Foi utilizada esta abordagem porque se existissem informações em fila de espera, estas, para situações de tempo real, não serviriam de referência para retratar o estado atual do mundo.

Numa fase inicial de modo a reduzir o capacidade de processamento em operações futuras aplica-se o algoritmo ilustrado na figura 5.1 que permite fazer um redimensionamento para metade da imagem original e ainda aplicar um pequeno filtro de ruído. O algoritmo consiste em analisar a imagem de quatro em quatro píxeis e contabilizar quantos desses píxeis apresentam uma diferença

para a imagem padrão superior a um dado nível threshold. Se essa condição se verificar pelo menos três vezes então na nova imagem o pixel correspondente receberá informação de ocupado (1), caso contrário recebe 0, como estando livre. Como resultado deste algoritmo a matriz redimensionada será totalmente binária em que cada pixel corresponderá a um espaço ocupado (1) ou desocupado (0).

Na figura 5.1 poderá ser possível verificar o mapeamento referido anteriormente.



Algoritmo T:

$$if \sum_{\substack{0\le i\le 1\\ 0\le j\le 1}} [BackGround(i,j) - frame(i,j)] \ge threshold_{level} \;\; then\; k = k+1$$

$$if\; k \ge 3\; then\; Binary(i,j) = 1$$

$$else\;\; Binary(i,j) = 0$$

Figura 5.1: Redimensionamento da Imagem *Depth*

Após o passo anterior procede-se neste momento a uma segunda fase de verificação, nova filtragem de ruído agora sobre a imagem binária.

De modo a não sobrecarregar em demasia o processamento do sistema, é analisada a continuidade de uma região ao longo de uma linha, ou seja, para cada pixel analisado, são verificados k píxeis na sua frente. Se esses píxeis tiverem todos o valor 1, a mediana das posições analisadas é guardada num vetor já com as coordenadas cartesianas da frame original. O algoritmo recomeça k píxeis à frente de modo a não possuir informação redundante. Todos os valores guardados neste momento formam nuvens de pontos com concentrações bem definidas mas para todos os efeitos não pertencem a nenhum grupo, não estão etiquetadas.

De referir que o filtro aplicado na verificação da continuidade está diretamente relacionado com o número de pontos que irá constituir o grupo associado ao objeto, que poderá ser sempre um parâmetro ajustável caso seja necessário ter mais resolução na aquisição da informação. Este parâmetro permite definir de quantos em quantos píxeis é percorrida a imagem binária, que naturalmente tendo um valor grande o processamento acelera, mas perde-se precisão (menos pontos

detetados). A perda de precisão é crítica no sentido em que o cluster de pontos pode ser descartado por se tratar de ruído espontâneo ou mesmo não possuir informação suficiente para garantir a convergência dos métodos de identificação de objetos abordados mais à frente neste relatório. No entanto se o parâmetro de verificação for pequeno, a resolução aumenta significativamente mas pode vir a complicar a aquisição de dados com o aumento de ruído. No entanto a filtragem e redimensionamento inicial permite ter uma maior flexibilidade no manuseamento desse parâmetro.

A técnica de Clustering consiste em procurar uma estrutura numa coleção de dados sem rótulo que poderá ser caracterizada. É um processo de organização de objetos em grupos cujos membros são similares de alguma maneira[39]. A definição de cluster pode ser demonstrada como na figura 5.2.

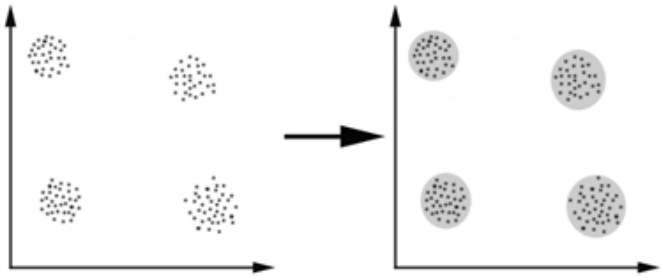


Figura 5.2: Clustering[39]

No caso da figura 5.2, consegue-se distinguir quatros clusters distintos onde o critério de semelhança é a distância, ou seja, um ou mais objetos pertencem ao mesmo cluster se estiverem perto consoante uma dada distância pré-definida. Neste caso o método de agregação é baseado em distâncias.

Nesta dissertação será usado igualmente o método de agregação baseado em distâncias e posteriormente quando se obtiver uma nuvem pontos agrupada, é calculado o centro de massa do seu centroide. Com esta informação poderá afirmar-se com alguma certeza o centro onde estará concentrada a maior informação do objeto mas não os seus contornos.

Para um conjunto de dados de grande dimensão, a métrica mais adequada para o critério de distância é a métrica de *Minkowski* (equação 5.1).

$$d_p(x_i,x_j) = \sum_{k=0}^{p} |x_{i,k} - x_{j,k}|^{\frac{1}{p}} [39] \tag{5.1}$$

Em que, d é a dimensão dos dados e p o índice para o tipo de métrica pretendida. Com p=2 tem-se a distância *Euclidiana*. Como neste projeto todos os pontos do conjunto/grupo estão nas mesmas unidades físicas, a simples métrica de distâncias *Euclidiana* é o suficiente.

O algoritmo de agrupamento inicia-se escolhendo um ponto qualquer (normalmente o primeiro dos detetados anteriormente) para ser a semente do método de Clustering, formando então desta

forma o primeiro cluster. Se o ponto seguinte na lista não estiver próximo desse cluster (distância definida na configuração) então adiciona-se um novo cluster formado apenas por esse ponto. Caso o ponto esteja nas imediações do cluster então este é adicionado e consequentemente o seu peso ajusta o centro de massa do grupo.

O algoritmo percorre todos os pontos e verifica em todos os clusters qual deles se ajusta melhor ao ponto em questão. Cada ponto só pode pertencer a um e só um cluster (Exclusive Clustering).

Após a criação dos grupos procede-se à caracterização dos mesmos. Tendo em consideração que estes grupos correspondem a objetos presentes no mundo, estes podem ser representados por uma caixa retangular (figura 5.3) em que as suas dimensões (*dx*, *dy* e *dz*) podem ser estimadas através da diferença máxima entre os pontos de cada eixo e a sua posição (*x,y,z*) obtida através do centro de massa do grupo.

No entanto este método poderá ser em alguns casos, ainda sem apresentar uma base de resultados, pouco preciso no que diz respeito ao cálculo da posição real do objeto, já que o alcance do *kinect*, face às suas limitações naturais, não permite o mapeamento de pontos total sobre o objeto. A componente mais afetada é do eixo dos XX (em profundidade). No entanto essa é uma situação esperada e que pode ser facilmente contornada, ou seja, à medida que o robô percorre o seu trajeto começa a ter em linha de vista o que falta do objeto e através do mesmo processo abordado neste capítulo envia o centro de massa e as dimensões da área analisada de modo a ajustar a sua posição.

Uma desvantagem deste método é o facto de por vezes, consoante o percurso tomado, o robô não tenha visualização do que falta do objeto. Trata-se de uma situação natural pois não se conhece o que não se consegue ver a não ser que esses objetos apresentem certas características que permitam, através de certos processos, serem identificáveis. Nas secções seguintes será possível verificar casos em que isso é possível em esferas e cilindros.

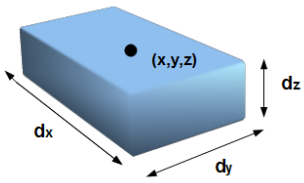


Figura 5.3: Objeto

Na figura 5.4 está ilustrado, em forma de síntese, o método implementado.

5.2 Identificação de Objetos

Na secção anterior implementou-se um método para agrupar conjuntos de pontos detetados pelo sensor *kinect*. Nesta secção o objetivo será tentar encontrar neles objetos conhecidos, como esferas e cilindros. Inicialmente será explicado o método utilizado e em seguida os seus resultados associados.

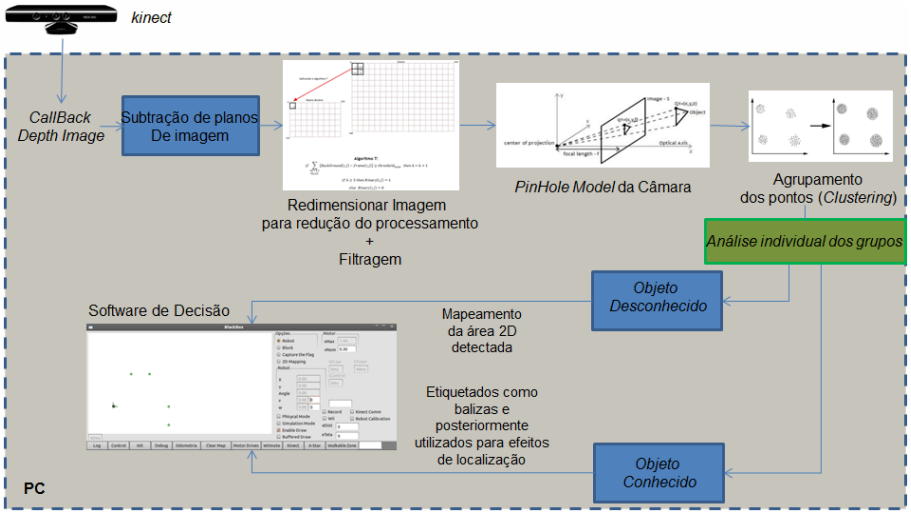


Figura 5.4: Síntese Ilustrativa do método implementado

5.2.1 Esferas

Nesta secção será introduzido um método de mínimos quadrados para a deteção de esferas. Tratando-se de um problema de mínimos quadrados não linear, será implementado o método *Gauss-Newton* para determinar a aproximação final da superfície à nuvem de pontos que envolvem o objeto.

O método explicado nesta secção é baseado no documento *Least Squares Best-Fit geometric elements* referenciado em [40].

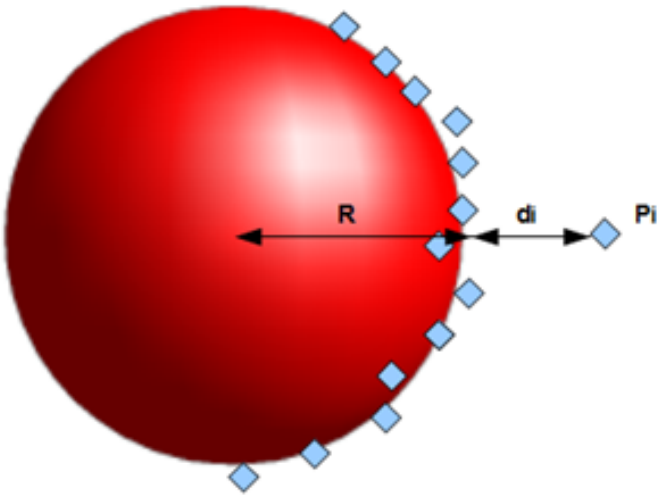


Figura 5.5: Exemplificação dos parâmetros a minimizar

Uma esfera é caracterizada pelo seu centro (X_o,Y_o,Z_o) e o seu raio r_o. Qualquer ponto na superfície da esfera terá que satisfazer a seguinte equação 5.2.

$$(X - X_o)^2 + (Y - Y_o)^2 + (Z - Z_o)^2 = r^2 \tag{5.2}$$

Para proceder à aproximação dos pontos à melhor esfera é necessário determinar uma estimativa inicial que sirva de ponto de partida ao método dos mínimos quadrados. Existem várias maneiras de determinar a estimativa inicial, que são aplicadas dependendo da robustez do algoritmo de aproximação final. O método mais simples é determinar o centro de gravidade do centro e do raio. Nesta dissertação será feita uma abordagem pelo seguro e utilizar método dos mínimos quadrados logo na estimativa inicial e posteriormente, como se trata de um sistema não linear, será utilizado o método iterativo de *Gauss-Newton* para obter o resultado final.

Existem métodos mais robustos para determinar o resultado final, como o método de Levenberg-Marquardt[41], este é capaz de convergir para o resultado final mesmo com uma estimativa inicial bastante afastada da final. Trata-se de um método mais pesado computacionalmente. O método de estimativa inicial utilizado neste projeto é robusto o suficiente para permitir ao método *Gauss-Newton* garantir a convergência rápida para o resultado final.

Considerando a função de diferenças de raio $f_1 = r_i - r$, onde r_i é dado pela equação 5.3.

$$r_i = \sqrt{(X_i - X_o)^2 + (Y_i - Y_o)^2 + (Z_i - Z_o)^2} \tag{5.3}$$

Diferenciando f_1 em função de X_o, Y_o, Z_o e r_o iria resultar em equações bastante complicadas e difíceis de resolver. Contudo, considerando $f_2 = r_i^2 - r^2$ é possível verificar que esta função pode ser escrita como $f_2 = (r_i + r)(r_i - r) \simeq 2r(r_i - r)$ já que $r_i + r \simeq 2$r.

Expandindo a função de minimização $f = r_i^2 - r^2$:

$$f = (X_i - X_o)^2 + (Y_i - Y_o)^2 + (Z_i - Z_o)^2 - r^2 = -(2X_iX_o + 2Y_iY_o + 2Z_iZ_o) + \rho + (X_i^2 + Y_i^2 + Z_i^2) \tag{5.4}$$

Em que,

$$\rho = (X_o^2 + Y_o^2 + Z_o^2) - r^2 \tag{5.5}$$

A variável ρ é introduzida para tornar a equação f linear.

A equação f anterior pode ser escrita na forma matricial para um conjunto de n pontos, como se pode verificar na expressão 5.6.

$$\begin{bmatrix} -2X_1 & -2Y_1 & -2Z_1 & 1 \\ -2X_2 & -2Y_2 & -2Z_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ -2X_n & -2Y_n & -2Z_n & 1 \end{bmatrix} \begin{bmatrix} X_o \\ Y_o \\ Z_o \\ \rho \end{bmatrix} - \begin{bmatrix} X_1^2+Y_1^2+Z_1^2 \\ X_2^2+Y_2^2+Z_2^2 \\ \vdots \\ X_n^2+Y_n^2+Z_n^2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \tag{5.6}$$

Para uma equação de mínimos quadrados, f terá que ser igualado a 0. Utilizando a notação anterior, tem-se uma equação do tipo AP-B=0

Para uma equação de mínimos quadrados, f terá que ser igualado a 0. Utilizando a notação anterior, tem-se que AP-B=0. O sistema anterior é impossível e indeterminado. Uma solução alternativa permite resolver este sistema, baseando na seguinte propriedade:

Seja A $\in M_{mxn}$ (R) e $b \in R^m$. Uma coluna $x \in R^n$ é uma solução no sentido dos mínimos quadrados do sistema Ax=b se e só se satisfaz $A^T Ax = A^T$ b. Neste caso P=x, portanto pela equação 5.7.

$$P = (A^T A)^{-1} A^T b \tag{5.7}$$

As estimativas iniciais (X_o,Y_o,Z_o) e ρ são obtidas como solução de P. A estimativa inicial de ro pode ser obtida através da equação de ρ, como se verificou anteriormente.

Depois de obter as estimativas iniciais para o centro e o raio da esfera, procede-se à utilização do método *Gauss-Newton* para determinar os seus valores finais.

A função a ser minimizada será $d = r_i - r$, como exemplificado na figura 5.5.

A matriz 'jacobiana' representa a melhor aproximação linear de uma função diferencial a um ponto na sua proximidade. Dada a função de minimização, a matriz 'jacobiana' (J) será constituída da seguinte forma:

$$J = \begin{bmatrix} \frac{\partial d_1}{\partial X_o} & \frac{\partial d_1}{\partial Y_o} & \frac{\partial d_1}{\partial Z_o} & \frac{\partial d_1}{\partial r_o} \\ \frac{\partial d_2}{\partial X_o} & \frac{\partial d_2}{\partial Y_o} & \frac{\partial d_2}{\partial Z_o} & \frac{\partial d_2}{\partial r_o} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial d_n}{\partial X_o} & \frac{\partial d_n}{\partial Y_o} & \frac{\partial d_n}{\partial Z_o} & \frac{\partial d_n}{\partial r_o} \end{bmatrix} = \begin{bmatrix} \frac{-(X_1-X_o)}{r_1} & \frac{-(Y_1-Y_o)}{r_1} & \frac{-(Z_1-Z_o)}{r_1} & -1 \\ \frac{-(X_2-X_o)}{r_2} & \frac{-(Y_2-Y_o)}{r_2} & \frac{-(Z_2-Z_o)}{r_2} & -1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{-(X_n-X_o)}{r_n} & \frac{-(Y_n-Y_o)}{r_n} & \frac{-(Z_n-Z_o)}{r_n} & -1 \end{bmatrix} \tag{5.8}$$

Resolvendo a equação de mínimos quadrados 5.9,

$$JP = -d \tag{5.9}$$

Em que,

$$P = \begin{bmatrix} \Delta X_o \\ \Delta Y_o \\ \Delta Z_o \\ \Delta r_o \end{bmatrix} \tag{5.10}$$

e

$$d = \begin{bmatrix} r_1 - r_o \\ r_2 - r_o \\ \vdots \\ r_n - r_o \end{bmatrix} \tag{5.11}$$

Os parâmetros finais serão ajustados de acordo com as expressões 5.12 , 5.13 , 5.14 e 5.15.

$$X_o = X_o + \Delta X_o \tag{5.12}$$

$$Y_o = Y_o + \Delta Y_o \tag{5.13}$$

$$Z_o = Z_o + \Delta Z_o \tag{5.14}$$

$$r_o = r_o + \Delta r_o \tag{5.15}$$

O algoritmo só termina quando houver convergência, ou seja, todos os procedimentos envolvendo o método *Gauss-Newton* são repetidos até que a variação dos valores finais for $\simeq 0$.

5.2.1.1 Resultados da Identificação de Esferas

De forma a validar o algoritmo descrito em 5.2.1 foram realizados testes com vários tipos de esferas. Na figura 5.6 *a)* encontram-se os vários tipos de esferas que se pretende detectar. Em *b)* ilustra a matriz de profundidade que se pretende analisar para a deteção das esferas. Em *c)* encontra-se a reconstrução 3D do ambiente visualizado recorrendo ao software *Meshlab*, que utilizando as informações da posição de cada ponto (obtido em 4.1.3) e a sua cor correspondente

(recorrendo à câmara RGB do *kinect*), permite obter a cena apresentada. Por fim em *d)* encontram-se conjuntos de pontos bem definidos (a verde) envolvendo as esferas. Estes pontos são obtidos após a realização de um processo de filtragem e Clustering (secção 5.1). Embora os objetos estejam todos presentes na mesma imagem, os testes foram realizados com um objeto de cada vez.

De referir que devido ao tipo de material que constitui a esfera 2 (figura 5.6), que provocava o desvio por reflexão dos sinais IR, tornou-se necessário revestir o objeto com uma película branca não refletora.

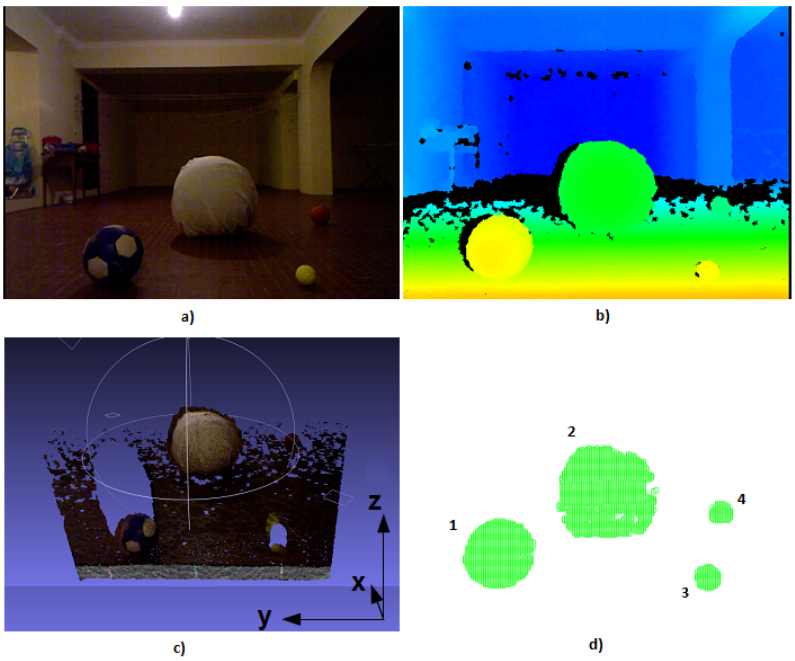


Figura 5.6: Resultados: a) Imagem RGB do *kinect* ; b) Resultado da Imagem *Depth* ; c) Reconstrução em 3D do ambiente através de pontos obtidos por um único varrimento do *kinect* ; d) Nuvens de pontos bem definidas envolvendo os objetos

Toda a informação relativa a cada uma das esferas como o raio e a posição real no mundo está representada na tabela 5.1.

Tabela 5.1: Posição e raio das Esferas

	Coordenadas (metros)		
Esfera	**X**	**Y**	**Raio**
1	1,20	0,32	0,10
2	2,0	0	0,25
3	1,20	-0,59	0,032
4	2,45	-0,85	0,065

Pelo método de deteção de esferas descrito em 5.2.1, é possível verificar que em primeiro lugar é feita uma estimativa inicial do centro e do raio da esfera baseado no método dos mínimos

quadrados e só depois é aplicado o método iterativo de *Gauss-Newton* de modo a conduzir o conjunto de pontos a um resultado final para os parâmetros do objeto.

Para a consolidação e validação da informação, procedeu-se à extração de 100 amostras para cada valor relevante à caracterização das esferas, como o centro e raio, o número de iterações efetuadas pelo método iterativo de *Gauss-Newton*, o número de pontos que compõem o grupo associado ao objeto, a percentagem de amostras válidas (eficácia) e ainda o tempo que decorre durante o procedimento de identificação de esferas e o tempo total desde a captação da imagem até à identificação do objeto. A exposição e o detalhe da informação será maioritariamente sob a forma de tabelas essencialmente utilizando para demonstração dos resultados, o valor médio e desvio padrão para cada parâmetro. No caso do centro e raio da esfera será feita ainda a comparação com o seu valor real.

Nas figuras 5.7 e 5.8 e na tabela 5.2 estão representados os resultados para deteção de esferas, focando parâmetros como as coordenadas XY, o raio e a eficiência de amostras válidas. Através da informação da tabela 5.2 é possível verificar que o desvio padrão dos parâmetros analisados é relativamente pequeno e a sua perceção insignificante para a maioria das esferas, ao contrário da esfera 4 em que a elipse do erro de posição é bastante visível assim como a variação do raio. Comparando os valores médios das posições obtidas com os valores reais, verifica-se que estão bastante próximos sendo que apenas no eixo dos YY se nota uma maior disparidade, podendo ser explicado por um possível erro sistemático que através de uma pequena calibração possa ser corrigido. No entanto não se pode garantir com a máxima precisão a colocação do objeto assim como a medição real da posição. Também é possível detetar com notoriedade nos resultados obtidos a diferença de dimensões que apresentam os quatro objetos, permitindo a identificação trivial de cada um.

A inconsistência para a esfera 4 pode ser explicada pela presença de uma maior quantidade de ruído para distâncias mais elevadas provocando perturbações nas medidas. Outra causa para este efeito é o facto de o objeto estar próximo dos limites da imagem da câmara *Depth* apresentando, uma vez que não foram corrigidos, efeitos provocados pelas distorções das lentes. O efeito barril ou distorção tangencial provoca a deformação dos píxeis nos cantos da imagem. A distorção é zero no centro ótico da imagem, mas cada vez maior quando se desloca até à periferia. Este fenómeno pode ser verificado na figura 5.9.

Tabela 5.2: Resultados Obtidos para a deteção das esferas

	Resultados Obtidos num Conjunto de 100 Amostras									
	X_c **(m)**			Y_c **(m)**			*Raio* **(m)**			**Eficiência**
Obj.	**Real**	$\langle a \rangle$	σ	**Real**	$\langle a \rangle$	σ	**Real**	$\langle a \rangle$	σ	%
1	1,200	1,200	$1x10^{-3}$	0,320	0,338	0	0,100	0,119	$4x10^{-3}$	99
2	2,000	2,011	$3x10^{-3}$	0,000	0,036	$4x10^{-3}$	0,250	0,265	$1x10^{-3}$	98
3	1,200	1,199	$6x10^{-3}$	-0,590	-0,535	$2x10^{-3}$	0,032	0,036	$2x10^{-3}$	77
4	2,450	2,410	$6x10^{-2}$	-0,850	-0,741	$2,6x10^{-2}$	0,065	0,080	$6,1x10^{-2}$	89

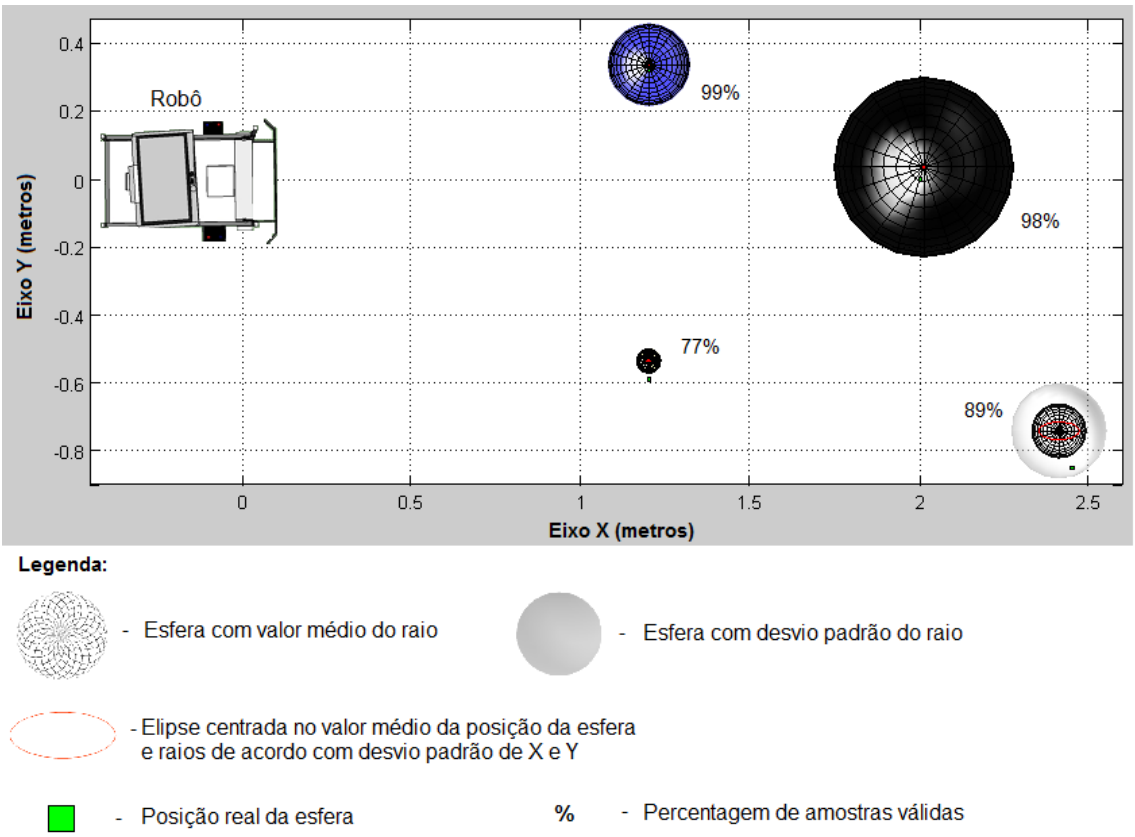


Figura 5.7: Representação dos resultados obtidos para a deteção das esferas

Tabela 5.3: Resultados Obtidos para a deteção das esferas (cont.)

	Resultados Obtidos num Conjunto de 100 Amostras							
	Nº Pontos		***Nº Iterações***		***Tempo Convergência (ms)***		***Tempo Total (ms)***	
Obj.	$\langle a \rangle$	σ	$\langle a \rangle$	σ	$\langle a \rangle$	σ	$\langle a \rangle$	σ
1	1084	12	3	0	1,194	0,284	41,06	4,82
2	1719	18	3	0	1,863	0,369	43,97	3,68
3	132	3	7	2	0,301	0,087	38,74	5,26
4	92	3	9	2	0,273	0,071	38,43	4,48

As temporizações obtidas (tabela 5.3) neste método estão naturalmente diretamente relacionadas com o número de iterações ocorridas no método *Gauss-Newton* e o número de pontos que constituí o objeto, embora seja este último que mais contribuí para diferenças acentuadas nos valores das temporizações. De referir que é notória a dificuldade em convergir quando o objeto possuí poucos pontos detetados, chegando mesmo a não convergir em algumas amostras (baixa eficácia). No método *Gauss-Newton* definiu-se que o método atingiu a convergência quando a diferença entre o valor obtido na iteração anterior e a atual for inferior a $5\text{x}10^{-8}$. Estabeleceu-se também um limite de 30 iterações que se for ultrapassado admite-se que naquela amostra não houve convergência.

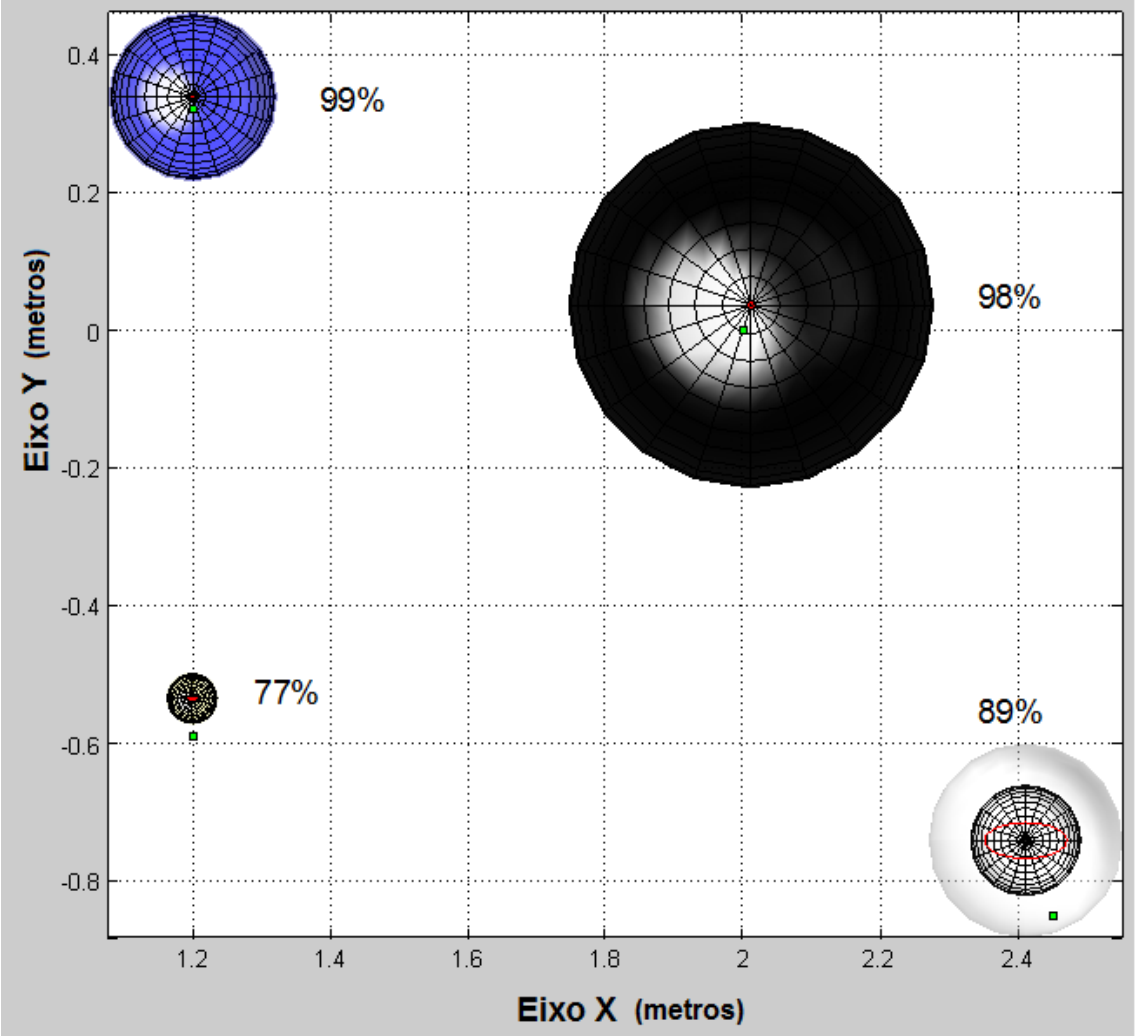


Figura 5.8: Representação dos resultados obtidos para a deteção das esferas (Zoom In)

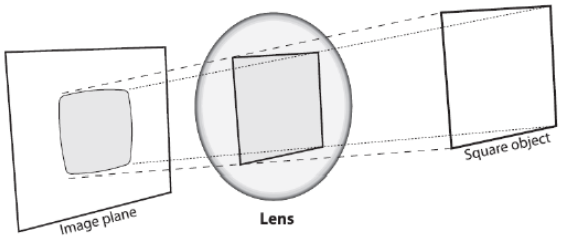


Figura 5.9: Efeito Barril [7]

5.2.2 Cilindros

Nesta secção será abordado um método de deteção de cilindros num espaço a três dimensões.

Este método consiste em dividir o objeto em vários níveis na direção do eixo do *ZZ* (figura 5.10) e analisar cada fatia circular resultante em coordenadas *XY*.

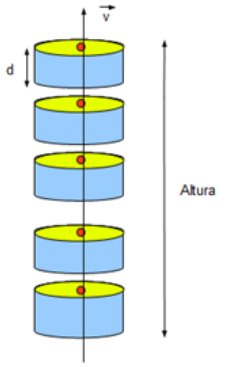


Figura 5.10: Divisão do objeto em partes iguais *i*

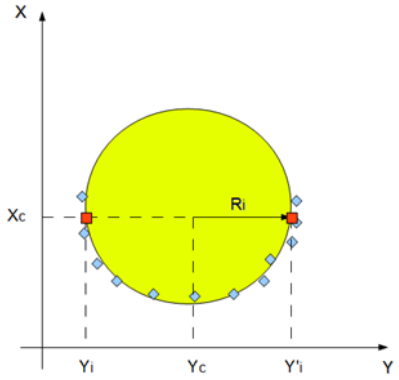


Figura 5.11: Circunferência de um nível *i*

Todos os níveis do objeto estão equidistantes de uma distância *d*, que é obtida dividindo a altura máxima do objeto pelo número de níveis pretendido.

A fatia associada a cada nível é então parametrizada, utilizando uma aproximação circular dos pontos detetados ao longo desse nível, como se pode verificar na figura 5.11.

O método de identificação de cilindros consiste, como já foi referido, na divisão do objeto em *n* níveis equidistantes. O critério de procura de pontos ao longo de um desses níveis, de forma a obter um boa aproximação circular, tem naturalmente um peso importante no resultado final. Cada nível corresponderá a um determinado valor de *Z* e o que se pretende definir é o desvio em torno desse valor, formando dessa forma uma janela de procura. A definição deste parâmetro influência o sucesso do algoritmo de aproximação circular, uma vez que o número de pontos utilizado no algoritmo de aproximação está diretamente relacionado com esse parâmetro. A ausência de uma boa consistência de pontos ou a interferência de ruído, pode levar à não convergência do método de aproximação circular ou mesmo levar a valores totalmente errados. Este método de deteção de cilindros é interrompido assim que pelo menos num dos níveis não tenha havido convergência. A essas amostras são chamadas de *OutSiders* e não são obviamente considerados.

A primeira condição de identificação do objeto é verificar se este se encontra na vertical. Esta condição é verificada pelo alinhamento de todos os centros das fatias circulares detetadas como se pode verificar pelo vetor *v* na figura 5.10. Se o vetor *v* tiver uma inclinação paralela ao eixo dos *ZZ* então potencialmente trata-se do objeto pretendido.

Neste momento resta verificar a inclinação das retas em duas das extremidades do cilindro. Fazendo passar uma reta pelo centro de uma circunferência *i* e paralelamente ao eixo dos *YY* tem-se os dois pontos indicados na figura 5.11. Cada nível terá dois pontos como os da figura 5.11. As coordenadas dos pontos a vermelho são obtidas pelas seguintes equações 5.16, 5.17 e 5.18.

$$Y_i = Y_c - R_i \tag{5.16}$$

$$Y_i' = Y_c + R_i \tag{5.17}$$

$$X_i = X_i' = X_c \tag{5.18}$$

A ligação entre os pontos dos diversos níveis deverá apresentar a configuração da figura 5.12, onde os vetores a e b são iguais em módulo e paralelos ao eixo dos ZZ no que diz respeito à sua direção. Ambas as retas definidas pelos vetores a e b são obtidas por aproximação linear dos pontos a azul.

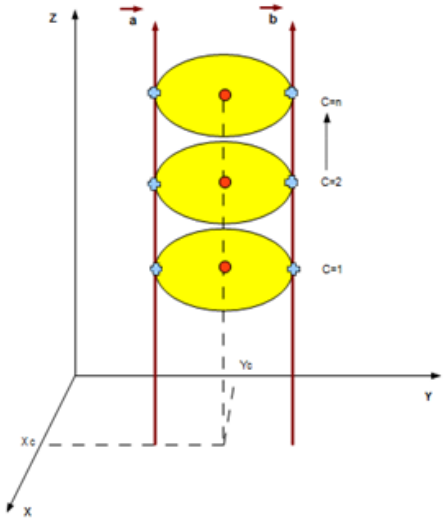


Figura 5.12: Inclinação das Retas que unem os pontos das extremidades das circunferências de cada nível do cilindro i

As verificações anteriores consistiam na elaboração de um método identificação de um cilindro qualquer, desde que mantivesse o tipo de características que definem esse tipo de objeto. Para verificar se realmente se trata do cilindro pretendido é necessário ter em consideração as suas dimensões. Para tal será utilizado o método descrito na secção 5.1.

5.2.2.1 Resultados da Identificação de Cilindros

De forma a validar o algoritmo descrito em 5.2.2 foram realizados testes com vários tipos de cilindros.

Na figura 5.13 *a)* encontram-se os vários tipos de cilindros que se pretende detetar. Em *b)* ilustra a matriz de profundidade que se pretende analisar para a deteção dos cilindros. Em *c)* encontra-se a reconstrução 3D do ambiente visualizado recorrendo ao software *Meshlab*, que utilizando as informações da posição de cada ponto (obtido em 4.1.3) e a sua cor correspondente

(recorrendo à câmara RGB do *kinect*), permite obter a cena apresentada. Por fim em *d)* encontram-se conjuntos de pontos bem definidos (a verde) envolvendo os cilindros. Estes pontos são obtidos após a realização de um processo de filtragem e Clustering (secção 5.1).

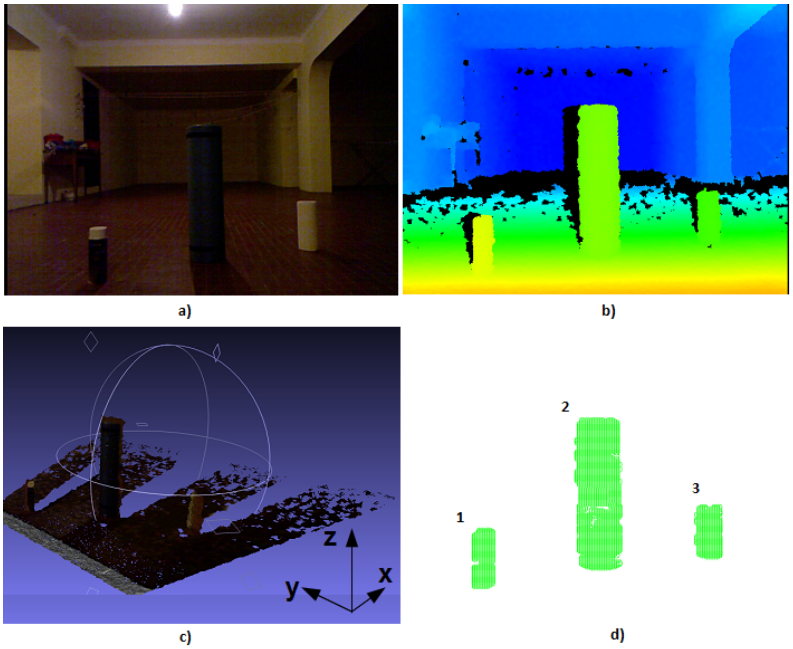


Figura 5.13: Resultados: a) Imagem RGB do *kinect* ; b) Resultado da Imagem *Depth* ; c) Reconstrução em 3D do ambiente através de pontos obtidos por um único varrimento do *kinect* ; d) Nuvens de pontos bem definidas envolvendo os objectos

A informação relativa a cada um dos cilindros como a posição no mundo, o raio e a altura está representada na tabela 5.4.

Tabela 5.4: Informação dos Cilindros

	Informação Geral (metros)			
Obj.	**X**	**Y**	**Raio**	**Altura**
1	1,2	0,4	0,035	0,19
2	1,5	0	0,08	0,60
3	1,6	-0,48	0,045	0,23

O cilindro apresenta um modelo trigonométrico como o ilustrado na figura 5.14.

Cada vetor pode ser representado pelos cossenos dos ângulos entre esse vetor e os três eixos de coordenadas (figura 5.15). Esta será então a nomenclatura utilizada na discussão dos resultados dos vetores definidos na figura 5.14.

Na tabela 5.5 encontram-se os valores reais de cada componente vectorial que se pretende comparar com os resultados obtidos. De referir que estes valores são iguais para todos os cilindros em teste, uma vez que se encontram todos na direção vertical.

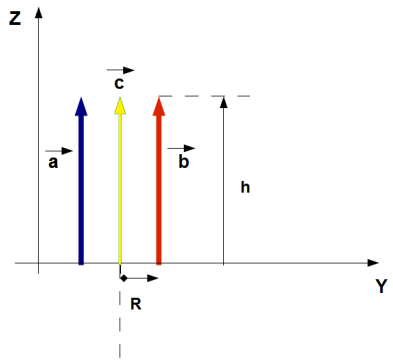


Figura 5.14: Modelo Trigonométrico do Cilindro Utilizado

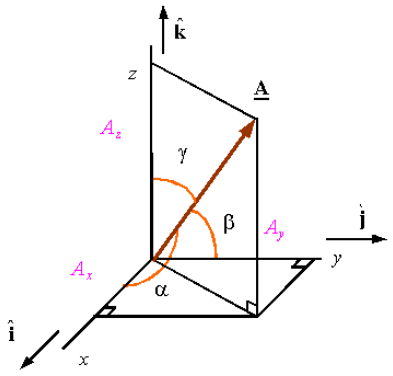


Figura 5.15: Representação de um vetor

Tabela 5.5: Coordenadas dos vetores a, b e c

	Coordenadas dos Vetores		
Vectores	**X**	**Y**	**Z**
a	0	0	1
b	0	0	1
c	0	0	1

Cada valor obtido terá a contribuição de um conjunto de 100 amostras para consolidação e validação da sua informação.

O método de identificação de cilindros consiste (como referido em 5.2.2) na divisão do objeto em *n* níveis equidistantes. Neste caso foram criados 10 níveis para cada cilindro com uma janela de procura de 3cm.

Nas figuras 5.16 e 5.17 e na tabela 5.6 estão representados os resultados na deteção de cilindros através dos seus valores médios e respetivos desvios padrões de um conjunto de amostras. É possível verificar que a variação entre os valores das amostras é muito pouco significativa apresentando, no conjunto geral dos três cilindros, uma gama de valores para o desvio padrão entre os 0.1 cm e os 0.6 cm. Comparando os valores médios das posições obtidas com os valores reais, verifica-se que estão bastante próximos sendo que apenas no eixo dos YY se nota uma maior disparidade, podendo ser explicado por um possível erro sistemático que através de uma pequena calibração possa ser corrigido. No entanto não se pode garantir com a máxima precisão a colocação do objeto assim como a medição real da posição. Também é possível detetar com notoriedade nos resultados obtidos a diferença de dimensões que apresentam os três objetos, permitindo a identificação trivial de cada um. De referir também que embora os objetos estejam todos presentes na mesma imagem, os testes foram realizados com um objeto de cada vez.

Na tabela 5.6 estão representados os valores obtidos para a posição, raio e altura de cada cilindro. Os parâmetros obtidos para o cilindro são apresentados com um valor médio e respetivo desvio padrão do conjunto de amostras analisado. De referir que a obtenção da altura do cilindro não está diretamente relacionada com o método de identificação de cilindros descrito em 5.2.2,

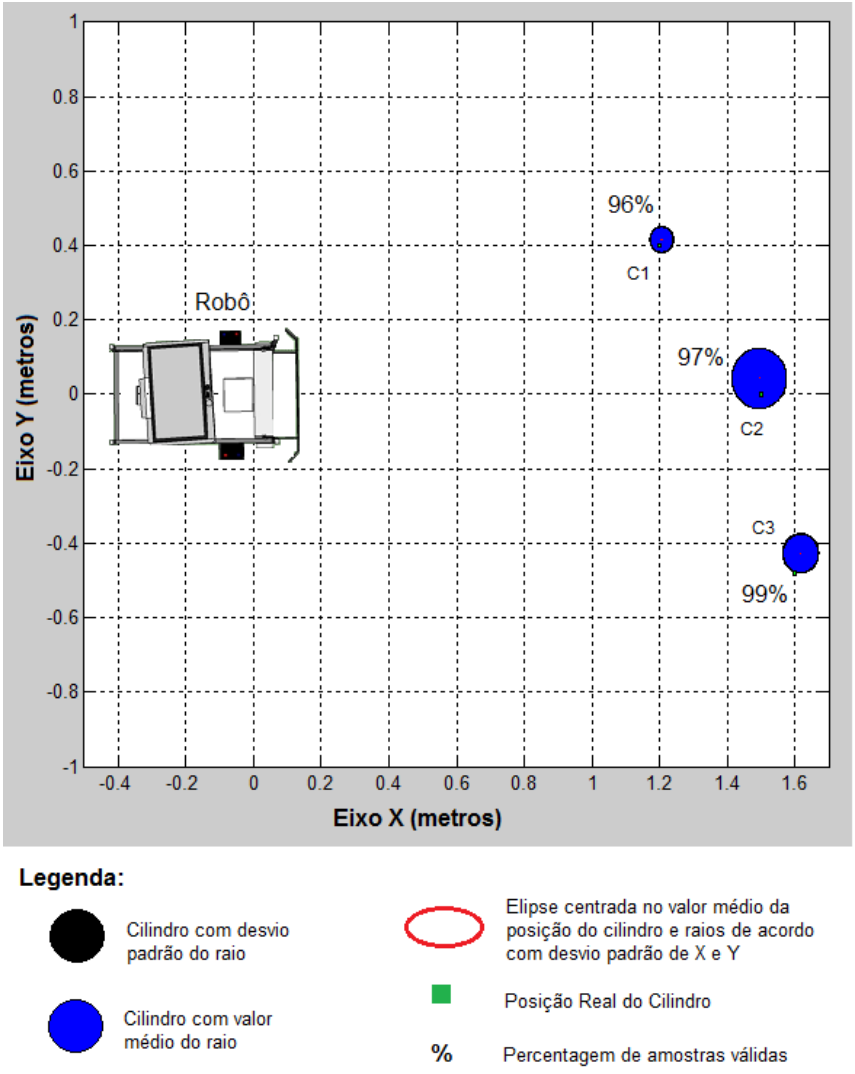


Figura 5.16: Representação dos resultados obtidos na deteção dos cilindros

Tabela 5.6: Resultados Obtidos para a deteção dos cilindros

	Resultados Obtidos num Conjunto de 100 Amostras											
	X (m)			**Y (m)**			**Raio (m)**			**Altura (m)**		
O	**Real**	$\langle a \rangle$	σ	**Real**	$\langle a \rangle$	σ	**Real**	$\langle a \rangle$	σ	**Real**	$\langle a \rangle$	σ
1	1,20	1,206	$3x10^{-3}$	0,40	0,416	$1x10^{-3}$	0,035	0,034	$1x10^{-3}$	0,19	0,176	$2x10^{-3}$
2	1,50	1,494	$1x10^{-3}$	0,00	0,043	0	0,080	0,081	$1x10^{-3}$	0,60	0,573	$2x10^{-3}$
3	1,60	1,617	$6x10^{-3}$	-0,48	-0,427	$2x10^{-3}$	0.045	0,052	$1x10^{-3}$	0,23	0,212	$4x10^{-3}$

mas tendo apenas em consideração o valor mínimo e máximo segundo o eixo dos ZZ do conjunto de pontos que envolve o objecto (5.13 *d)*).

Para demonstrar os resultados obtidos para as coordenadas do eixo de cada cilindro criou-se o gráfico que está representado na figura 5.18 em que todos as amostras realizadas podem ser visualizadas de uma perspetiva segundo o eixo dos ZZ (vista de cima), o que permite ter uma

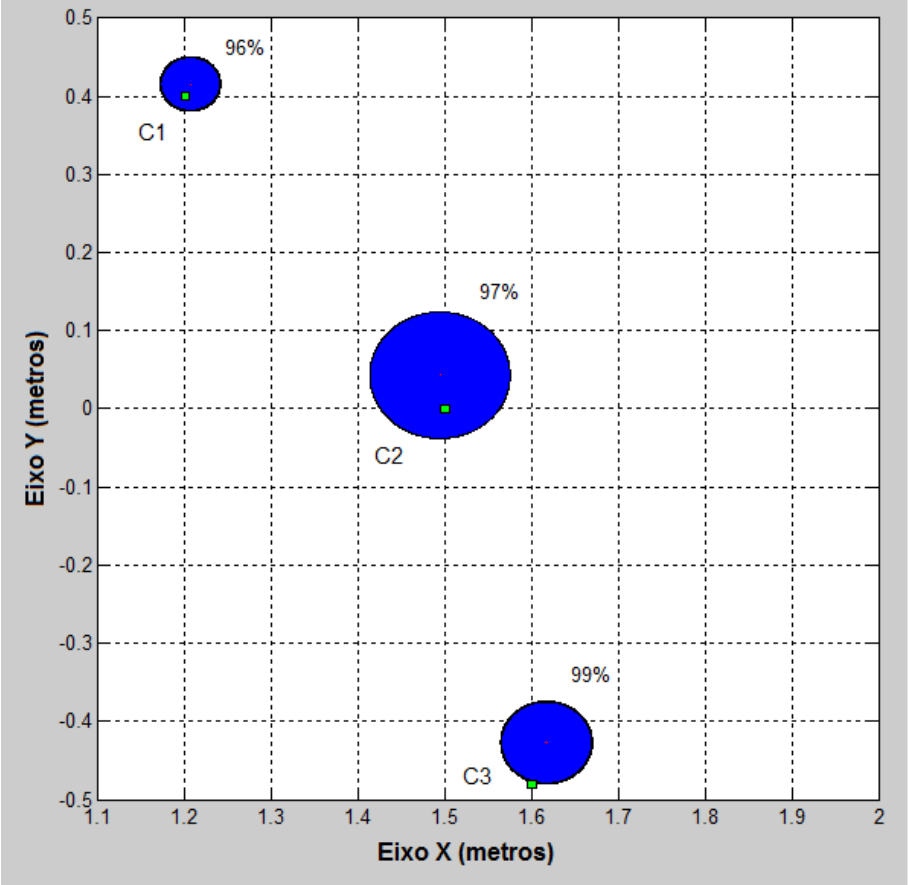


Figura 5.17: Representação dos resultados obtidos na deteção dos cilindros (Zoom In)

Tabela 5.7: Resultados Obtidos para a deteção dos cilindros (cont.)

	Ângulo dos Vectores em Z (º)						
		Vector *a*		**Vector *b***		**Vector *c***	
Obj.	**Real**	$\langle a \rangle$	σ ($\simeq$º)	$\langle a \rangle$	σ ($\simeq$º)	$\langle a \rangle$	σ ($\simeq$º)
1	90	86,11	0,62	86,16	0,66	86.82	1,15
2	90	89,34	0,20	88,17	0,09	88,81	0,09
3	90	88,03	0,88	87,77	1,24	88,37	0,96

Tabela 5.8: Resultados Obtidos para a deteção dos cilindros (cont.)

	Resultados Obtidos num Conjunto de 100 Amostras						
	Nº Pontos		***Tempo Convergência (ms)***		***Tempo Total (ms)***		***Eficácia***
Obj.	$\langle a \rangle$	σ	$\langle a \rangle$	σ	$\langle a \rangle$	σ	%
1	360	13	1,040	0,093	43,87	4,23	96
2	1435	11	1,642	0,140	44,92	4,07	97
3	328	4	1,198	0,139	43,55	4,15	99

visão global do desfasamento de cada amostra realizada. O objetivo desta comparação surge do facto de que o eixo de cada cilindro, estando na posição vertical, deverá ser paralelo ao eixo dos

ZZ. Todas os dados relativos aos três cilindros estão representadas no mesmo gráfico, uma vez que é esperado que se obtenha o mesmo resultado para o eixo de cada cilindro (ver valores para vetor c na figura 5.14 e tabela 5.5). Estes dados representados graficamente apresentam apenas um caráter informativo, sendo possível também identificar em que quadrante se encontra inclinado o objeto. Para o objetivo deste projeto a informação do quadrante é irrelevante uma vez que o mais importante é a magnitude dessa inclinação de forma a concluir tratar-se ou não do objeto pretendido. A magnitude da inclinação dos objetos encontram-se representados na tabela 5.7 pelos valores da componente em Z do vetor c (eixo do objeto).

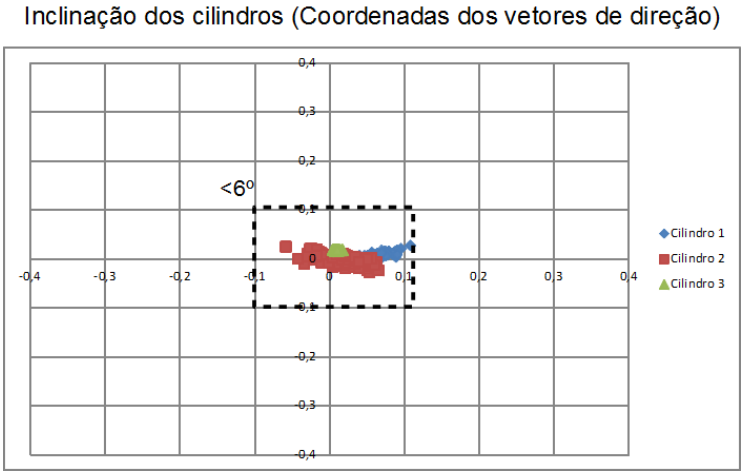


Figura 5.18: Resultados obtidos para as coordenadas do vetor *c* dos cilindros

A duração média do método utilizado para a deteção de cada cilindro tendo em consideração o número de pontos analisados em cada cilindro está representada na tabela 5.8. Como seria de esperar o cilindro 2 apresenta o maior tempo, já que possuí uma maior quantidade de pontos a serem analisados.

5.3 Conclusões do Capítulo

Durante o desenvolvimento de procedimentos de deteção e identificação de objetos 3D, pretendeu-se introduzir conceitos simples e com um baixo peso computacional. Todos os procedimentos são passíveis de serem customizáveis, podendo sempre introduzir uma maior ou menor precisão ao sistema, alterando a janela de pesquisa e filtragem da imagem adquirida pelo *kinect*.

Existem naturalmente limitações neste método analisado pois nem sempre é possível eliminar completamente o ruído da imagem e é necessário estar frequentemente a atualizar a imagem padrão (imagem de fundo) devido à trepidação do robô que provoca alterações na posição so sensor *kinect* e consequentemente mais ruído na imagem. Este problema pode ser atenuado com uma estrutura sólida que permita uma imobilização resistente. O facto do método implementado para a deteção de objetos não identificar a forma do objeto mas apenas aproximá-lo por um objeto em forma de paralelepípedo de dimensões máximas obtidas para cada eixo cartesiano permite uma

baixo aproveitamento do espaço livre no mapa mas por seu lado permite um contorno mais seguro do objeto aquando da passagem do robô.

No que diz respeito à identificação de objetos, o conceito iterativo para aproximação de um conjunto de pontos é bastante utilizado. Os métodos iterativos necessitam de uma boa estimativa inicial para que a convergência para o resultado final seja a mais rápida possível. Neste projeto introduziu-se um método de mínimos quadrados na determinação das estimativas iniciais, o que se revelou bastante eficaz, tanto para a determinação de superfícies esféricas como para a aproximação circular nos cilindros. Desta forma, a convergência do método *Gauss-Newton* utilizava muito poucas iterações e o tempo também muito reduzido. A limitação mais notória neste método de identificação de objetos será então a dificuldade de os identificar quando um outro objeto se encontra muito próximo como por exemplo uma esfera encostada a uma parede. Neste caso os pontos dos dois objetos fundem-se formando um novo objeto na perspetiva do método implementado.

Capítulo 6

Localização e Movimentação do Robô

A capacidade de um robô móvel conhecer a sua localização em relação a um referencial externo é um dos requisitos fundamentais para a autonomia do mesmo.

Neste capítulo será apresentado o sistema de auto-localização desenvolvido recorrendo a duas balizas de posição conhecida no mapa. O cenário é composto por dois conjuntos de balizas, sendo que o robô só se localiza por um conjunto de cada vez. Cada conjunto é composto por dois *beacons* espaçados em diferentes posições e naturalmente com diferentes características para um reconhecimento seletivo dos mesmos. O reconhecimento dos *beacons* e respetiva posição relativa ao robô serão feitos recorrendo aos métodos de identificação de objetos referidos no capítulo 5.

Na secção 6.1 será introduzido o método de localização por duas balizas com os respetivos cálculos da posição e direção do robô. Na secção 6.2 são apresentados e discutidos os resultados de percursos efetuados pelo robô, combinando o sistema de odometria com o sistema de localização absoluta.

6.1 Localização absoluta por balizas recorrendo a *beacons* presentes no mapa

Nesta secção será introduzido o modelo matemático para a determinação da posição e direção do robô recorrendo a dois objetos presentes no mapa. Na figura 6.1 está representado o diagrama trigonométrico do modelo referido. A utilização de pelo menos duas balizas é necessária para diminuir situações de ambiguidade. Se em vez de duas fosse utilizada apenas uma baliza, o robô poderia se encontrar em qualquer ponto a uma determinada distância (D_{B1} ou D_{B2}) da baliza. Ao serem utilizadas duas balizas a situação de ambiguidade fica reduzida à indecisão entre dois pontos (P_{R1} e P_{R2} da figura 6.1). Neste momento surge então a necessidade de utilizar a fusão de informação proveniente da odometria com a informação do sistema de localização absoluta de modo a selecionar um dos pontos a vermelho da figura 6.1. Naturalmente o ponto escolhido é o que estiver a menor distância da posição atual do robô dado pelo sistema de odometria.

Nas equações 6.1 e 6.2 então representadas as equações da circunferência em torno das balizas 1 e 2, respetivamente. Uma vez que o sistema de deteção de objetos fornece a distância

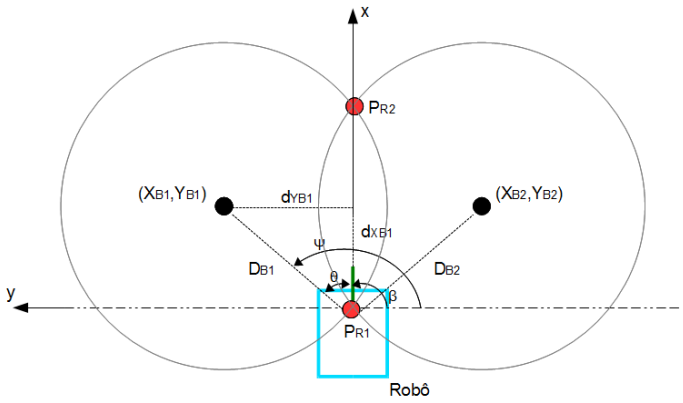


Figura 6.1: Diagrama trigonométrico da localização absoluta por duas balizas

a cada uma das balizas nas coordenadas cartesianas X e Y, será necessário determinar D_{B1} e D_{B2} recorrendo às expressões 6.3 e 6.4, respetivamente.

$$(X - X_{B1})^2 + (Y - Y_{B1})^2 = D_{B1}^2 \tag{6.1}$$

$$(X - X_{B2})^2 + (Y - Y_{B2})^2 = D_{B2}^2 \tag{6.2}$$

Em que,

$$D_{B1} = \sqrt{d_{XB1}^2 + d_{YB1}^2} \tag{6.3}$$

$$D_{B2} = \sqrt{d_{XB2}^2 + d_{YB2}^2} \tag{6.4}$$

Os ângulos ψ e θ da figura 6.1 são obtidos através das equações 6.5 e 6.6, respetivamente.

$$\psi = arctan_2(\frac{Y_{B1} - Y_{R1}}{X_{B1} - X_{R1}}) \tag{6.5}$$

$$\theta = arctan_2(\frac{d_{YB1}}{d_{XB1}}) \tag{6.6}$$

Os pontos P_{R1} e P_{R2} são obtidos resolvendo o sistema 6.7.

$$\begin{cases} (X - X_{B1})^2 + (Y - Y_{B1})^2 = D_{B1}^2 \\ (X - X_{B2})^2 + (Y - Y_{B2})^2 = D_{B2}^2 \end{cases} \tag{6.7}$$

Resolvendo o sistema 6.7 é possível obter os dois pontos através das expressões 6.8 e 6.9.

$$Y_{R1,2} = \frac{-F \pm \sqrt{F^2 - 4EG}}{2E} \tag{6.8}$$

$$X_{R1,2} = \frac{Y_{R1,2}D + A}{C} \tag{6.9}$$

Em que,

$$\begin{cases} A = X_{B1}^2 - D_{B1}^2 + Y_{B1}^2 - X_{B2}^2 - Y_{B2}^2 + D_{B2}^2 \\ B = D_{B1}^2 - X_{B1}^2 - Y_{B1}^2 \\ C = 2X_{B1} - 2X_{B2} \\ D = 2Y_{B2} - 2Y_{B1} \\ E = D^2 + C^2 \\ F = 2DA - 2X_{B1}DC - 2Y_{B1}C^2 \\ G = A^2 - 2X_{B1}AC - BC \end{cases} \tag{6.10}$$

6.2 Seguimento de trajetórias e auto-calibração da posição e direção do robô

Nesta secção pretende-se testar e validar os sistemas desenvolvidos ao longo desta dissertação. Pretende-se combinar o sistema de identificação de objetos com o sistema de localização de balizas e este último com o sistema de odometria. O cenário para os testes está montado de acordo com a figura 6.2. Para uma melhor e mais fiável aquisição de dados, todos os testes foram realizados a uma velocidade de 0.3 m/s.

Figura 6.2: Cenário base em que os testes foram realizados

Como se pode verificar pela figura 6.2, existem dois pares de balizas em que cada um é composto por uma esfera e um cilindro. Embora os dois conjuntos de balizas apresentem objetos da mesma classe, estes apresentam características distintas como diferentes alturas no caso dos cilindros e diferentes diâmetros no caso das esferas. Para o conjunto de balizas 1 foi utilizado uma esfera com 10 cm de raio e um cilindro com uma altura de 30 cm. Para o conjunto de balizas 2 foi utilizado uma esfera com 6.5 cm de raio e um cilindro com uma altura de 40 cm. Ambos os cilindros apresentam o mesmo diâmetro de 7.5 cm.

Após alguns testes realizados para a deteção das balizas, com o robô a uma distância de 1.5 m e numa posição e direção frontal a cada um dos conjuntos individualmente, verificou-se que em média, em cada 1000 imagens analisadas pelo *kinect* obtém-se, para o conjunto 1, 484 vezes a esfera, 460 vezes o cilindro e 317 vezes ambos os objetos na mesma imagem. Para o conjunto 2 os valores são ligeiramente diferentes, 459 vezes a esfera, 542 vezes o cilindro e 346 vezes ambos os objetos na mesma imagem. Este baixo número deve-se à restrição rigorosa na deteção dos objetos em causa. No caso das esferas possui para o método *Gauss-Newton*, um fator de erro de convergência de $5x10^{-9}$. Para o critério de identificação dos cilindros consideraram-se, para cada

nível do objeto, todos os pontos contidos numa gama ± 2 cm e a aproximação circular com um critério de convergência de $5x10^{-9}$. Além disso não são aceites cilindros com inclinações abaixo dos 80^{o}. Com os testes realizados através da análise de 1000 imagens verificou-se que em média cada imagem demora 46 ms a ser totalmente analisada e processada. Todas os critérios de teste estabelecidos são referentes aos métodos de identificação de objetos abordados no capítulo 5. Em contrapartida quando estes são detetados pode-se considerar que se tratam de facto dos objetos pretendidos.

A figura 6.3 corresponde à legenda dos símbolos a serem utilizados na demonstração dos resultados obtidos mais à frente.

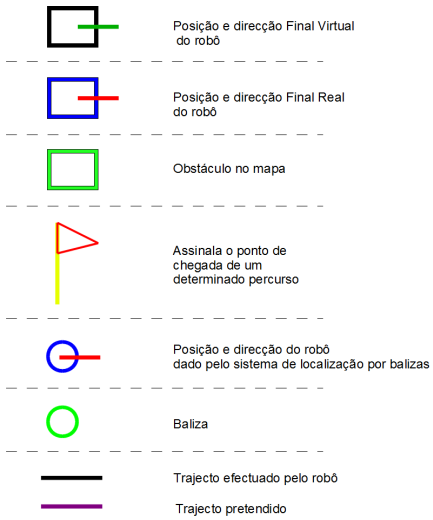


Figura 6.3: Legenda referente aos símbolos utilizados nos resultados dos percursos

De referir que todos os testes desenvolvidos para a localização absoluta do robô foi dada uma confiança fixa ao sistema de localização por odometria e ao sistema de localização por balizas, sendo que para o primeiro atribuiu-se 30% de confiança na sua medição e para o segundo 70%.

Percurso Boomerang Pré-definido

Este teste tem o nome de percurso boomerang porque é um percurso criado para se iniciar e terminar no mesmo ponto, como se pode ver pela figura 6.4. Com este teste o mais importante não é de facto constatar se chegou exatamente ao ponto pretendido mas sim verificar se a posição virtual, ou seja, a posição do robô que o software assume como sendo a verdadeira, coincide com a posição real (medida pelo operador humano).

Na figura 6.5 ilustra o resultado final do percurso boomerang. Como se pode verificar o robô corrigiu a sua posição e direção por diversas vezes nas duas zonas de calibração. O resultado final é bastante satisfatório uma vez que o robô ainda percorreu uma certa distância sem qualquer calibração pelo sistema de localização por balizas. A diferença entre a posição real e virtual do robô é de aproximadamente 12 cm enquanto a diferença de direção é praticamente inexistente com apenas 5^{o}.

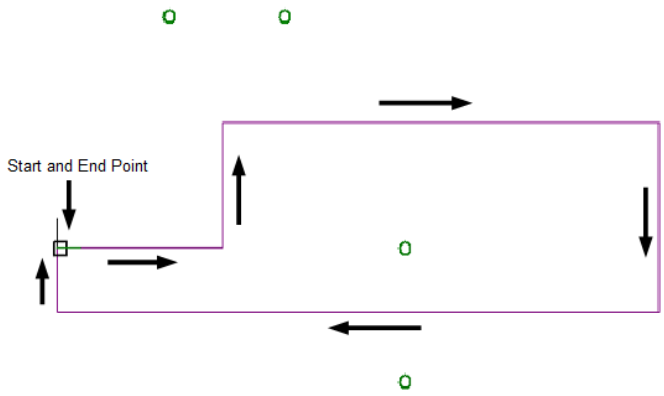


Figura 6.4: Percurso boomerang

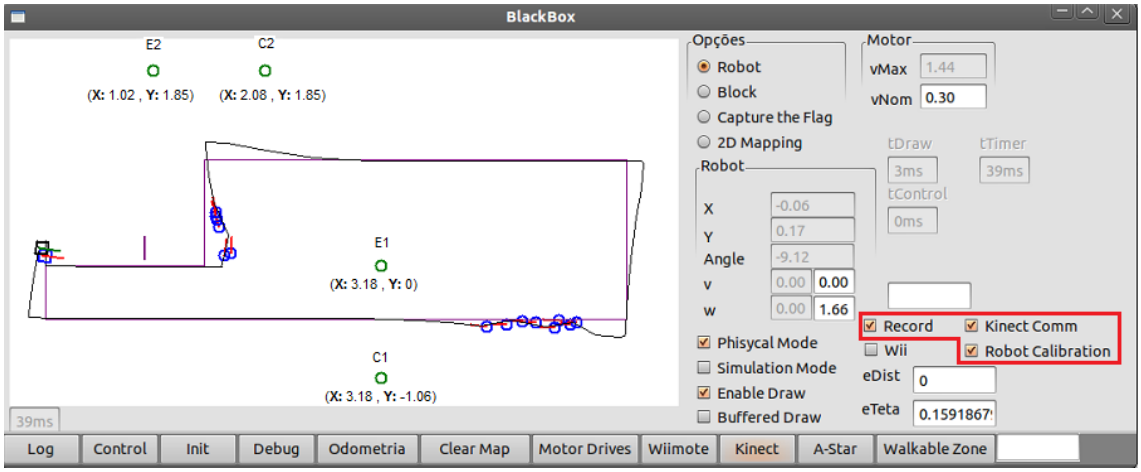


Figura 6.5: Resultado final do teste do percurso boomerang (coordenadas em metros)

Do lado esquerdo da figura 6.5 é possível verificar uma zona a vermelho que englobam as opções necessárias para que o software apresente esta informação. A opção 'Record' permite gravar todo o percurso virtual do robô. A opção 'Kinect Comm' ativa a comunicação por UDP, permitindo a troca de dados entre a aplicação de alto nível com a aplicação que comunica com o *kinect*. Por fim a opção 'Robot Calibration' permite que o software corrija a posição e direção do robô através do sistema de localização por balizas. Se esta opção estiver inativa a localização fica encarregue apenas pelo sistema de odometria. No entanto o sistema não funciona sem antes estar ativo o modo físico do sistema nem o modo de pintura no ecrã.

De referir que no final de cada segmento de trajetória linear o robô tem de parar, rodar e posteriormente iniciar o seguimento do próximo segmento de reta. No momento de paragem, como o robô possuí apenas um sistema de aplicação de velocidade de referência com valor 0 nos motores e nenhum sistema de travagem, faz com que, devido à sua inércia, demore um pouco a imobilizar-se (como se pode verificar na figura 6.5). Esta situação piora quanto maior for a

velocidade, sendo que uma das soluções seria abrandar gradualmente à medida que se aproxima do final da trajetória.

Percurso Circular Pré-definido

Neste teste pretende-se que o robô detete um objeto e que o represente no mapa e ainda criar um percurso circular de raio 70 cm com centro igual ao centro do objeto. O objetivo será então fazer o robô circular em torno da circunferência e autocalibrando a sua posição e direção quando passar nas zonas de calibração dos dois conjuntos de balizas. A figura 6.6 representa o resultado final do teste após o robô ter percorrido pouco mais de duas voltas e meia. Devido à dificuldade do robô detetar o conjunto de balizas nº2, a componente da localização por balizas obteve-se apenas pelo conjunto nº1. No final a posição virtual encontra-se a aproximadamente 4 cm da posição real enquanto a direção apenas apresenta um desvio de 3º. Uma perspetiva mais detalhada do resultado final pode ser vista na figura 6.7.

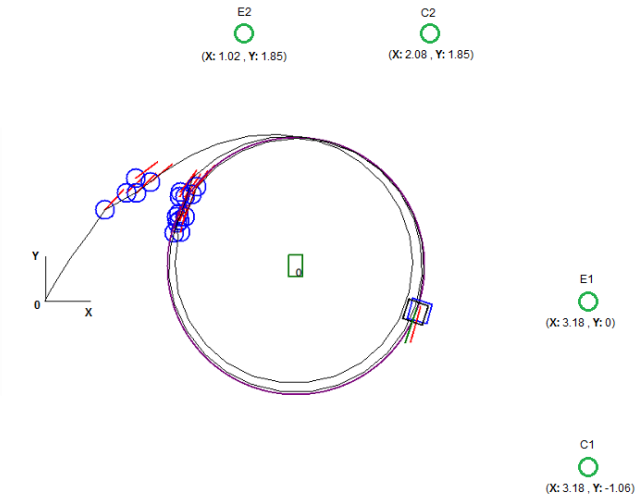


Figura 6.6: Resultado final do teste do percurso circular em torno de um objeto (coordenadas em metros)

A deteção do objeto apresentou uma diferença de 5 cm face à sua real posição no mundo, portanto esse será o erro introduzido na criação da circunferência. Esta situação poderia ser considerada crítica se a circunferência apresentasse um raio mais pequeno ou se o objeto fosse de maiores dimensões. Neste caso foram dadas algumas margens a contar com estes possíveis erros.

De referir que as técnicas desenvolvidas para o controlo e seguimento de uma trajetória circular estão explicadas no anexo A.

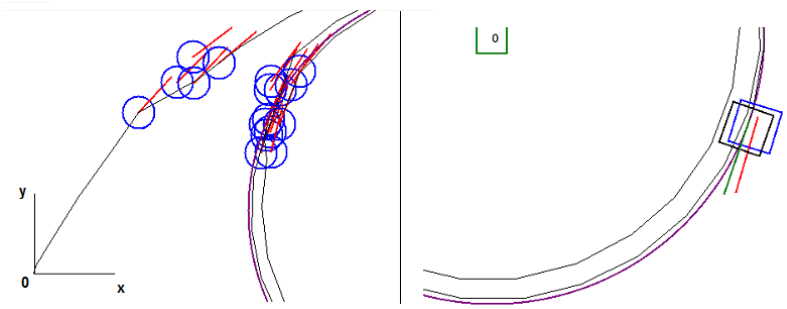


Figura 6.7: Realce da zona de aglomerado de pontos (lado esquerdo) e da posição final do robô (lado direito)

Desvio de Obstáculo

Para testar o desvio de obstáculos utilizou-se um objeto em forma de paralelepípedo com dimensões da base em dX,dY de 25, 34 cm, respetivamente. O objeto foi detetado apenas considerando uma perspetiva frontal, ainda assim conseguiu-se um mapeamento com dX$\simeq$24 cm e dY$\simeq$32 cm. A posição (X,Y) obtida para o bloco foi de (1.37,0.07) enquanto a sua posição real é de (1.46,0.03). Esta diferença deve-se ao facto dos pontos sobre objeto estarem mais concentrados na parte frontal, uma vez que a sua posição é calculada com base no centro de massa dos pontos detetados. Uma solução seria efetuar um mapeamento de todas as perspetivas do objeto, colocando o robô a navegar livremente pelo espaço. De referir que o objeto só é considerado e consequentemente incluído no mapa se apresentar 20 medições semelhantes. A sua posição e tamanho vão sendo ajustados sempre que receba objetos com posições inseridas num círculo de raio 10 cm. No entanto quanto mais medições válidas forem recebidas, maior se torna a confiança do objeto naquela posição sendo cada vez mais difícil movê-lo.

Após o mapeamento do objeto no mapa escolheu-se um ponto de destino, identificado na figura 6.8 pela bandeira. No momento da escolha do ponto de destino é gerado uma trajetória, recorrendo ao algoritmo A*[34], até esse ponto tendo em consideração as áreas ocupadas. O algoritmo A* demorou 16 ms até obter a trajetória representada no mapa da figura 6.8. A trajetória é composta por pequenos segmentos de reta interligados entre si desde o ponto inicial até ao ponto de destino. Os segmentos estão ligados de 10 em 10 células de uma grelha/mapa de dimensão 640x480.

De referir que as técnicas desenvolvidas para o controlo e seguimento dos segmentos de reta (trajetória linear/*Follow Line*) estão explicadas no anexo A

A diferença entre a perspetiva virtual e real da posição e direção do robô foi de $\simeq 5cm$ e $\simeq 8^{o}$, respetivamente.

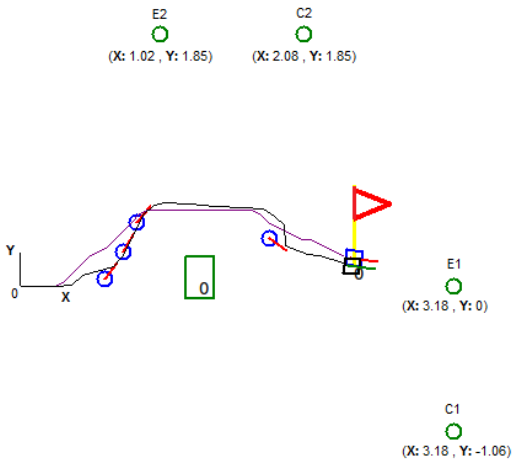


Figura 6.8: Resultado final do teste de desvio de um obstáculo (coordenadas em metros)

Mapeamento completo de um objeto desconhecido

Neste teste pretende-se encontrar a dimensão e posição 2D de um objeto desconhecido, para tal retirou-se informações desse objeto em quatro perspetivas. O resultado obtido está ilustrado na figura 6.9, em que os casos P_i (i=0,1,2,3) correspondem às quatro perspetivas de perceção do objeto. As posições de cada perspetiva bem como a média e o erro face à posição real do objeto, estão representadas na tabela 6.1.

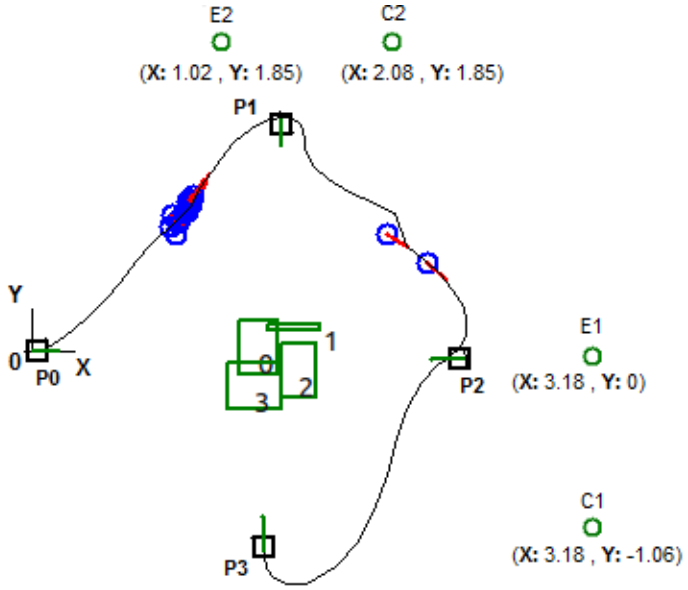


Figura 6.9: Resultado final do teste do mapeamento completo de um objeto desconhecido (coordenadas em metros)

Tabela 6.1: Informação Objeto Desconhecido

	Posição Objeto	
Perspetiva	**X (m)**	**Y (m)**
P0	1.284	0.024
P1	1.685	0.130
P2	1.513	-0.113
P3	1.262	-0.189
Média	1.436	-0.037
Real	1.400	-0.020
Erro ($\|\varepsilon\|$)	0.036$\simeq$3.6 cm	0.017$\simeq$1.7 cm

Capítulo 7

Conclusão

Neste trabalho desenvolveu-se uma plataforma robótica capaz de realizar atividades apelativas na área da robótica baseada em competições de condução autónoma anteriores. Para cumprir esses objetivos construiu-se um robô com componentes de baixo custo, utilizando uma tecnologia mediática como o *Kinect* da Microsoft.

Com o *Kinect* é possível identificar esferas e cilindros utilizando para isso métodos baseados no algoritmo *Gauss-Newton*. Os resultados obtidos neste processo são bastante satisfatórios para diferentes tipos da mesma classe de objetos, permitindo identificar com clareza cada tipo de esferas e cilindros.

A escolha para a identificação de esferas e cilindros deve-se ao facto de serem objetos que apresentam características homogéneas, desta forma basta apenas uma perspetiva para conseguir estimar o que falta do objeto.

O uso de processos iterativos para a deteção de objetos mostrou ser de grande importância, uma vez que a única informação de entrada é um conjunto incompleto de pontos tridimensionais.

Após a realização de vários testes para a identificação de objetos obtiveram-se para um conjunto de 100 amostras possíveis valores entre os 77 e os 99% de eficácia para as esferas e valores entre 96 e 99% para os cilindros, enquanto o tempo de convergência dos métodos varia entre 1.05 e 1.65 ms para os cilindros e entre 0.25 e 1.85 ms para as esferas (valores obtidos utilizando um computador portátil com um CPU Intel i5 de 2.27 GHz), sendo que o maior objeto de cada classe foi o que se obteve maior tempo de processamento por apresentarem maior quantidade de pontos. Na identificação das esferas o caso mais crítico foi a identificação de uma bola de ténis em que a quantidade de pontos era menor, conduzindo a um maior número de iterações no método *Gauss-Newton* (o mais elevado de todos os testes efetuados com média de 9 iterações) sendo também o caso em que se obteve menor eficácia com 77%. Os testes anteriores foram realizados analisando um objeto de cada vez.

Numa altura mais avançada do projeto procedeu-se ao mesmo teste de identificação de objetos, mas agora a pesquisa recaiu sobre uma imagem composta por uma esfera e um cilindro, uma vez que o robô durante o seu percurso apenas se localiza quando tiver em linha de vista um objeto de cada tipo. Efetuou-se o teste 1000 vezes com o robô parado a uma distância frontal de 1.5

m em relação ao conjunto de balizas. A eficácia diminuiu consideravelmente, adquirindo uma eficácia de 31,7% e 34,6% para os conjuntos de balizas 1 e 2, respetivamente. Este efeito pode ser justificado por uma procura mais restrita pelas características dos objetos pretendidos, enquanto os testes anteriores apenas procuravam objetos com características esféricas ou cilíndricas. Neste teste, o tempo médio de processamento de uma imagem foi de 46 ms.

Com o sistema de identificação de objetos com recurso ao *Kinect* implementou-se um sistema de localização por balizas baseado no conhecimento a priori das suas posições no mundo. Com o auxílio desta informação a posição e direção do robô torna-se mais precisa. No entanto nem sempre era possível corrigir os dados do robô recorrendo a este método sendo que na maioria do tempo o robô apenas se baseava nos dados obtidos pelo sistema de odometria, que se revelaram bastante aceitáveis.

O sistema além de permitir identificar objetos também é capaz de detetar qualquer tipo de objeto e reservar no mapa o seu espaço máximo em 2D. No entanto para detetar com a máxima precisão o objeto em causa é necessário circundá-lo e determinar os seus limites, como foi realizado no teste 'Mapeamento completo de um objeto desconhecido' da secção 6.2.

O sistema implementado permite ainda recalcular trajetórias em tempo real recorrendo ao algoritmo A* com um baixo tempo de procura de 16ms para o teste realizado em 6.2.

Realizaram-se vários testes com diferentes percursos, uns pré-definidos e outros determinados pelo algoritmo A*, de forma a testar e validar em conjunto todos os sistemas implementados. No conjunto global dos testes realizados, através das técnicas de localização referidas anteriormente, obtiveram-se no final dos mesmos para a posição do robô erros máximos de 12 cm enquanto para a direção do robô erros máximos de 8 graus. O teste boomerang foi o que se obteve um maior erro de posição com 12 cm face à posição real, sendo também o teste com um percurso mais longo.

Trabalhos Futuros

Como trabalho futuros pretende-se criar um sistema de localização mais robusto com mais balizas que permita ao robô se desloque em espaços amplos com uma menor dependência do sistema de odometria.

Para além de uma localização mais robusta, também se propunha a identificação de um leque mais alargado de objetos e também a identificação quando existe obstrução parcial do objeto.

Além dos trabalhos sugeridos propõe-se ainda:

- Acrescentar mais missões ao robô como fazer um percurso em 8, seguir objetos ou humanos;
- Acrescentar missão de mapeamento;
- Testar com ambientes dinâmicos difíceis;
- Portar implementação para um computador embarcado de menor poder de cálculo.

Ainda para trabalhos futuros mas mais vocacionado para a competição de condução autónoma seria a utilização de um sistema que permite ao *Kinect* ter um campo de visão mais abrangente,

podendo ainda ter uma visão próxima do robô. Esta abordagem é importante uma vez que o equipamento, por razões impostas pelo regulamento da competição, só pode ser colocado até uma certa altura do solo.

Anexo A

Controlo de Trajetórias

A.1 *Follow Line*

Este tipo de trajetória consiste em fazer o robô diferencial seguir uma linha. Na figura A.1, está representada as várias fases deste controlador.

Na posição inicial, o robô encontra-se em sentido oposto à linha pretendida, tendo desta forma que compensar um ângulo α antes de proceder ao seguimento da trajetória. Após a orientação estar em conformidade com a linha, torna-se necessário corrigir o erro de distância à linha. Como se pode verificar à medida que o controlador evolui no sentido de compensar a distância à linha, a orientação naturalmente sofre distorções.

Para corrigir os erros de distância e de orientação à linha ao longo da trajetória, em cada ciclo de controlo do controlador são calculadas estas diferenças e posteriormente, baseado nestes resultados, são determinados os comandos adequados para o robô de forma a compensar dinamicamente e em simultâneos os erros obtidos.

A figura A.2 corresponde ao esquema trigonométrico do controlador de trajetória linear no sentido direto. Como se pode verificar, nas expressões seguintes dos erros, tendo conhecimento da inclinação de uma reta, neste caso definida por dois pontos (P1 e P0), e da orientação do robô num dado instante é possível calcular a compensação de orientação necessária (eTeta) para o robô se manter alinhado com a reta. Recorrendo ao teorema de Pitágoras, retira-se o erro de distância à linha (eDist).

$$e_{Dist} = h * \sin\alpha \tag{A.1}$$

$$\alpha = arctan_2(\frac{y_{robot} - y_1}{x_{robot} - x_1}) - arctan_2(\frac{y_1 - y_0}{x_1 - x_0}) \tag{A.2}$$

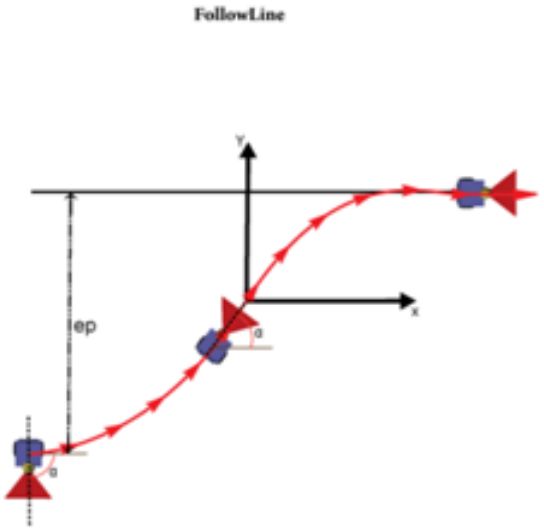


Figura A.1: Trajetória Linear

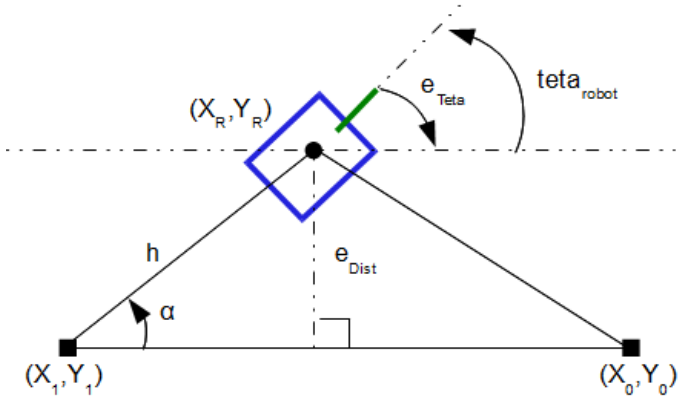


Figura A.2: Modelo Trigonométrico no sentido direto

$$e_{Teta} = teta_{robot} - arctan_2(\frac{y_1 - y_0}{x_1 - x_0}) \tag{A.3}$$

As expressões anteriores são referidas ao movimento do robô no sentido direto do seguimento da reta, podendo no entanto sofrer alguns ajustes face ao sentido de rotação do robô de forma a compensar os erros obtidos. No entanto estes ajustes serão da natureza de inversão de sinal da expressão de forma a ajustar o sentido de rotação correto. Em seguida será mostrado um exemplo de como são calculados os erros obtidos, bem como as considerações necessárias para o sentido de rotação do robô.

Por exemplo, se $teta_{robot}$=30º, se a inclinação da recta for 0º, se h=2metros e se α=30º então,

pelas equações A.1 e A.3, eTeta e eDist terão o valor de 30º e 1 respetivamente. Na figura A.2 é possível verificar-se que o robô necessita de rodar no sentido negativo para compensar os erros eTeta e eDist. Neste caso terá de se multiplicar por -1 em cada uma das expressões dos erros.

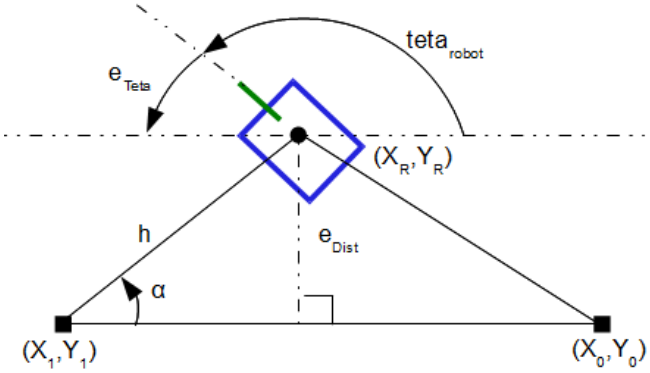


Figura A.3: Modelo Trigonométrico no sentido inverso

A figura A.3 corresponde ao esquema trigonométrico do controlador de trajetória linear no sentido direto. Neste caso a expressão do erro de distância mantém-se, mas a expressão do erro eTeta terá de alterado segundo a expressão A.4.

$$e_{Teta} = 180^{o} + teta_{robot} - atan_2(\frac{y_1 - y_0}{x_1 - x_0}) \tag{A.4}$$

Nas mesmas condições do exemplo anterior, exceto $teta_{Robot}$ que neste caso seria de 150º, o erro eTeta, pela equação A.4, teria o valor de 300º e eDist tomaria o mesmo valor de 1, já que a alteração efetuada não influenciou o seu valor. De referir que o valor de 300º terá de ser normalizado, tomando o valor de -60º. O sinal do valor obtido indica uma rotação no sentido negativo do eixo dos zz, o que pela figura percebe-se que o robô necessita de uma rotação no sentido positivo. Neste caso terá então de se fazer um ajuste no sinal do erro, invertendo o sinal do mesmo.

De notar que durante o sentido inverso do percurso o erro de distância eDist tem o valor positivo, não tendo neste caso que inverter o sinal ao contrário do sentido direto.

Na figura A.4 está ilustrada a máquina de estados do controlador de trajetória linear. No estado inicial, o robô calcula a sua orientação em relação à reta naquele instante e inicia então o ajuste dessa diferença até esta ser nula. De seguida comuta para um estado cíclico. Este estado a cada ciclo de controlo calcula o erro de distância eDist e de orientação à reta e atua sobre o robô diferencial de acordo com os valores de velocidade linear (v) e angular (w). Este estado é sempre chamado enquanto eDist e eTeta forem diferentes de zero, o que tende para o instante infinito. Normalmente este controlo termina por ordem externa ao controlador.

Os parâmetros 'kp' correspondem aos ganhos do controlador.

Figura A.4: Controlador Trajetória Linear

A.2 *Follow Circle*

Para o robô seguir um arco, e necessário ter em consideração o sentido do mesmo, ou seja, se o movimento é feito no sentido horário ou anti-horário. Esta abordagem é necessária pois para o cálculo do erro de ângulo que o robô tem de compensar e diferente para os dois casos. A figura A.5 representa o modelo trigonométrico num dado instante do seguimento de uma trajetória circular no sentido anti-horário. Como se pode verificar na figura, estão indicados os vários parâmetros necessários para o cálculo dos erros de distância e de orientação.

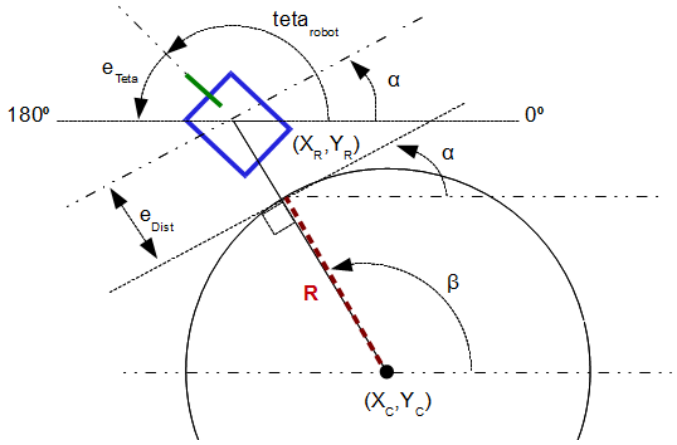


Figura A.5: Modelo Trigonométrico sentido anti-horário

Pela figura e possível verificar que para o robô tender para a Circunferência, e necessário corrigir a sua direção (e_{Teta}) e a sua posição (e_{Dist}). As equações necessárias para a obtenção destes mesmos parâmetros de erro são obtidas pelas seguintes equações:

$$e_{Teta} = teta_{robot} - (\beta + 90^{o}) \tag{A.5}$$

$$e_{Dist} = \sqrt{(x_{robot} - x_{center})^2 + (y_{robot} - y_{center})^2} - R \tag{A.6}$$

O parâmetro (β +90^{o}) na expressão do erro e_{Teta} corresponde à inclinação da recta tangente à circunferência.

Por exemplo, se $teta_{robot}$=150^{o} e β=100^{o} o valor de e_{Teta}, pela equação A.5, será -40^{o}. Com este valor o robô iria afastar-se cada vez mais da orientação pretendida, portanto terá de se trocar o sinal do e_{Teta} para fazer a rotação no sentido positivo.

A figura A.6 representa o modelo trigonométrico num dado instante do seguimento de uma trajetória circular no sentido horário. Como se pode verificar na figura, estão indicados os vários parâmetros necessários para o cálculo dos erros de distância e de orientação.

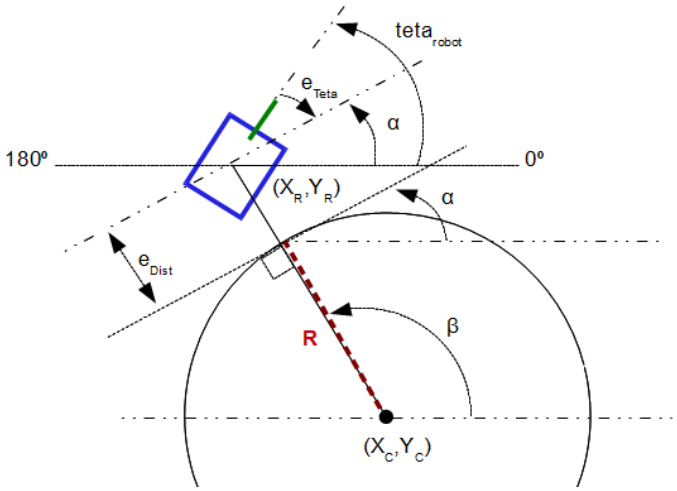


Figura A.6: Modelo Trigonométrico sentido horário

Pela figura A.6 é possível verificar que para o robô tender para a circunferência, e necessário corrigir a sua direção (e_{Teta}) e a sua posição (e_{Dist}), como foi abordado anteriormente para o sentido contrário. As equações necessárias para a obtenção destes mesmos parâmetros de erro são obtidas pelas seguintes equações:

$$e_{Teta} = 180^{o} + teta_{robot} - (\beta + 90^{o}) \tag{A.7}$$

$$e_{Dist} = \sqrt{(x_{robot} - x_{center})^2 + (y_{robot} - y_{center})^2} - R \tag{A.8}$$

Como se pode verificar, apenas o erro e_{Teta} difere do caso anterior pois o robô encontra-se exatamente à mesma distância, não havendo necessidade de alterar o erro de distância à linha e_{Dist}.

De referir que neste caso, o robô tenta seguir o ângulo 0^o da tangente à circunferência, havendo a necessidade de retirar 180^o à orientação do robô, daí a importância da introdução do ângulo 180^o na expressão do e_{Teta}. No exemplo seguinte será demonstrada a importância referida anteriormente.

Por exemplo, recorrendo às equações dos erros A.7 e A.8, se $teta_robot$=80^o e β=100^o, e_{Teta} teria o valor de 70^o. Como se pode verificar pela figura A.6, a compensação terá de ser no sentido negativo de rotação portanto terá de se multiplicar a expressão de e_{Teta} por -1. De notar que se não fosse considerado os 180^o na expressão do e_{Teta}, este teria o valor de -110^o, que seria o ângulo a ser compensado para seguir no sentido anti-horário.

Na figura A.7 está ilustrada a máquina de estados do controlador de trajetória circular. No estado inicial, o robô calcula a sua orientação em relação à reta tangente à circunferência naquele instante e inicia então o ajuste dessa diferença até esta ser nula. De seguida comuta para um estado cíclico. Este estado a cada ciclo de controlo calcula o erro de distância e_{Dist} e de orientação à reta tangente e atua sobre o robô diferencial de acordo com os valores de velocidade linear (v) e angular (ω). Este estado é sempre chamado enquanto e_{Dist} e e_{Teta} forem diferentes de zero, o que tende para o instante infinito. Normalmente este controlo termina por ordem externa ao controlador.

O parâmetro 'kp' corresponde aos ganhos do controlador e $V_{nominal}$/R é a velocidade angular necessária para o robô efetuar uma trajetória circular.

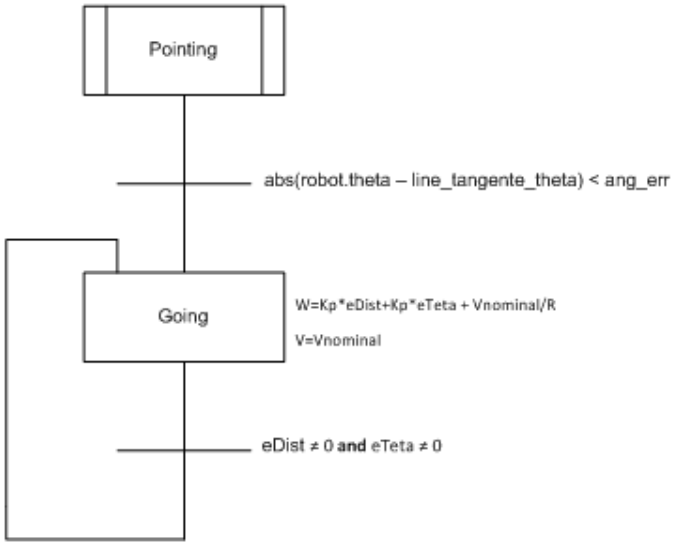


Figura A.7: Controlador Trajetória Circular

Referências

[1] Sarah Gingichashvili. Darpa's urban challenge, 2007. Disponível em http://thefutureofthings.com/articles/1001/darpas-urban-challenge-2007.html, acedido a última vez em 24 de Junho de 2011.

[2] Festival Nacional de Robótica. Condução autónoma, 2012. Disponível em http://www.robotica2012.org/12/index.php?option=com_content&view=category&layout=blog&id=72&Itemid=96, acedido a última vez em 18 de Janeiro de 2012.

[3] Festival Nacional de Robótica. Liga de robôs médios, 2012. Disponível em http://www.robotica2012.org/12/index.php?option=com_content&view=category&layout=blog&id=73&Itemid=97, acedido a última vez em 18 de Janeiro de 2012.

[4] 5dpo - robotics. Disponível em http://paginas.fe.up.pt/~robosoc/en/doku.php, acedido a última vez em 24 de Janeiro de 2012.

[5] Armando J. Sousa. *Arquitecturas de Sistemas Robóticos e Localização em Tempo Real Através de Visão*. Tese de doutoramento, FEUP, 2003.

[6] Festival Nacional de Robótica. Robot@factory, 2012. Disponível em http://www.robotica2012.org/12/index.php?option=com_content&view=category&layout=blog&id=75&Itemid=99, acedido a última vez em 18 de Janeiro de 2012.

[7] André Vidal. Feupcar 2.0:condução autónoma no festival nacional de robótica. Tese de mestrado, FEUP, 2011.

[8] A. M. F. Carvalhosa e T. L. B. Leite. Versa robot:robô móvel versátil para competições em provas de robótica. Relatório té, FEUP, 2006. Relatório Interno - DEEC.

[9] A. J. e V. Santos M. Oliveira, P. Stein. Modular scalable architecture for the navigation of the atlas autonomous robots. Em *Festival Nacional de Robótica*, Castelo Branco, Portugal, 2009.

[10] Oliveira e M. Santos R. Cancela, M. Neta. Atlas III: Um robô com visão orientada para provas em condução autónoma. Em *Festival Nacional de Robótica*, páginas 32–40, Coimbra, Portugal, 2005.

[11] M. Oliveira e V. Santos. A vision-based solution for the navigation of a mobile robot in a road-like environment. Em *Festival Nacional de Robótica*, página 8, Albufeira, Portugal, 2007.

[12] M. Oliveira e V. Santos. Multi-camera active perception system with variable image perspective for mobile robot navigation. Em *8th Conference on Autonomous Robot Systems and Competitions*, Aveiro, Portugal, 2008.

[13] M. Oliveira e V. Santos. Real time road line extraction with simple statistical descriptors. Em *Conferência Internacional (IEEE) em Sistemas de Robôs Inteligentes (IROS 2008)*, Nice, França, 2008.

[14] Projeto atlas, 2012. Disponível em http://atlas.web.ua.pt/, acedido a última vez em 17 de Janeiro de 2012.

[15] Héber Sobreira. Clever robot. Tese de mestrado, FEUP, 2009.

[16] Johann Borenstein e Liqiang Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 12(6), December 1996.

[17] António P. Moreira. Acetatos das aulas teóricas da disciplina sistemas robóticos autónomos, 2009. Disponível em https://www.fe.up.pt/si/conteudos_service.conteudos_cont?pct_id=92084&pv_cod=50P9a1CaBzaa, acedido a última vez em 24 de Janeiro de 2012.

[18] Hélder F. Oliveira. Análise do desempenho e da dinâmica de robôs omnidireccionais de três e quatro rodas. Tese de mestrado, FEUP, 2007.

[19] Luiz Chaimowicz. Robótica móvel - modelo cinemático. Relatório té, Universidade Federal de Minas Gerais, 2008. Disponível em http://homepages.dcc.ufmg.br/~chaimo/cursos/robotica/ModeloCinematico.pdf, acedido a última vez em 26 de Janeiro de 2012.

[20] Maxon Motor. Re 40 - 40 mm, graphite brushes, 150 watt. Disponível em http://www.maxonmotor.com/medias/sys_master/8796762210334/RE-40-148866_11_EN_082.pdf, acedido a última vez em 15 de Janeiro de 2012.

[21] Maxon Motor. Pwm-scheme and current ripple of switching power amplifiers. Disponível em http://www.electromate.com/db_support/attachments/PWM%20technical%20information.pdf, acedido a última vez em 25 de Janeiro de 2012.

[22] P. A. Simionescu. A unified approach to the assembly condition of epicyclic gears. *Power Transmission and Gearing*, 120(3), September 1998.

[23] Oriental Motor USA CORP. Planetary (pn) gears, 2006. Disponível em http://www.orientalmotor.com/newsletter/PN-Geared.htm, acedido a última vez em 10 de Agosto de 2011.

[24] Maxon Motor. Planetary gearhead gp 42c - 42 mm, 3 - 15 nm. Disponível em http://www.maxonmotor.com/medias/sys_master/8796903866398/GP-42-C-203113_11_EN_237-238.pdf, acedido a última vez em 15 de Janeiro de 2012.

[25] Maxon Motor. Technology – short and to the point : Planetary gearheads. Disponível em https://downloads.maxonmotor.com/Katalog_neu/eshop/Downloads/allgemeine_informationen/Technik_kurz_und_buendig/newpdf_11/

gear-Technik-kurz-und-buendig_11_EN_030-031.pdf, acedido a última vez em 10 de Agosto de 2011.

[26] Maxon Motor. Encoder mr type l, 256-1024 cpt, 3 channels, with line driver. Disponível em http://www.maxonmotor.com/medias/sys_master/8796912517150/ENC-MR-256imp-225783_11_EN_263.pdf, acedido a última vez em 15 de Janeiro de 2012.

[27] Aaron Moore. Understanding quadrature encoding. Disponível em http://prototalk.net/forums/showthread.php?t=78, acedido a última vez em 10 de Agosto de 2011.

[28] Maxon Motor. Technology – short and to the point : Digital incremental encoder. Disponível em https://downloads.maxonmotor.com/Katalog_neu/eshop/Downloads/allgemeine_informationen/Technik_kurz_und_buendig/newpdf_11/sensor-Technik_kurz_und_buendig_11_EN_032-033.pdf, acedido a última vez em 10 de Agosto de 2011.

[29] Advanced Motion Controls. Datasheet drives motores : Dzralte-012l080. Disponível em http://www.a-m-c.com/download/datasheet/dzralte-012l080.pdf, acedido a última vez em 24 de Junho de 2011.

[30] Sharp. Infrared sensor datasheet, 2011. Disponível em http://sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y3a003k_e.pdf, acedido a última vez em 12 de Outubro de 2011.

[31] cnet - asia. Disponível em http://asia.cnet.com/crave/just-how-much-of-a-game-changer-is-microsofts-kinect-62111519.htm, acedido a última vez em 12 de Agosto de 2011.

[32] Arduino. Disponível em http://arduino.cc/en/Main/ArduinoBoardMega2560, acedido a última vez em 12 de Agosto de 2011.

[33] Advanced Motion Controls. Serial communication - reference manual. Disponível em http://www.a-m-c.com/products/dzr.html?tab=2, acedido a última vez em 12 de Agosto de 2011.

[34] Patrick Lester. A* pathfinding for beginners, 2005. Disponível em http://www.policyalmanac.org/games/aStarTutorial.htm, acedido a última vez em 24 de Junho de 2011.

[35] Meir Machline Barak Freedman, Alexander Shpunt e Yoel Arieli. Depth mapping using projected patterns, 2010. United States Patent Application Publication, Disponível em http://www.freepatentsonline.com/20100118123.pdf, acedido a última vez em 24 de Junho de 2011.

[36] OpticalFlow. Kinect - insanely interesting, 2011. Disponível em http://opticalflow.wordpress.com/2011/02/27/kinect-insanely-interesting/, acedido a última vez em 25 de Janeiro de 2012.

[37] Kinect shadow, 2011. Disponível em http://media.zero997.com/kinect_shadow.pdf, acedido a última vez em 2 de Outubro de 2011.

[38] Nicolas Burrus. Kinect calibration. Disponível em http://nicolas.burrus.name/index.php/Research/KinectCalibration, acedido a última vez em 24 de Junho de 2011.

[39] A tutorial on clustering algorithms, 2011. Disponível em http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/index.html, acedido a última vez em 2 de Outubro de 2011.

[40] Jennifer Hicks e James Richards. Least squares best-fit geometric elements. Relatório té, University of Delaware, Newark, USA, 2009. Disponível em http://www.udel.edu/HNES/HESC427/Sphere%20Fitting/LeastSquares.pdf, acedido a última vez em 24 de Janeiro de 2012.

[41] Henri Gavin. The levenberg-marquardt method for nonlinear least squares curve-fitting problems. Relatório té, Duke University, USA, 2011.