Faculdade de Engenharia da Universidade do Porto

Navegação e Controlo de um Veículo Móvel

Hugo Sérgio Pinto Alves

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Automação

Orientador: Prof. Dr. António Paulo Gomes Mendes Moreira
Co-orientador: Prof. Dr. Paulo José Cerqueira Gomes da Costa

Março de 2010

© Hugo Sérgio Pinto Alves, 2010

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES **2009/2010**

A Dissertação intitulada

"NAVEGAÇÃO E CONTROLO DE UM VEÍCULO MÓVEL"

foi aprovada em provas realizadas em 02/Março/2010

o júri

Presidente Professor Doutor Adriano da Silva Carvalho
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 do Centro ALGORITMI da Escola de Engenharia da Universidade do Minho

Professor Doutor António Paulo Gomes Mendes Moreira
Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Paulo José Cerqueira Gomes da Costa
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 projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Hugo Sérgio Pinto Alves

AUTOR: **HUGO SÉRGIO PINTO ALVES**

Faculdade de Engenharia da Universidade do Porto

Resumo

A constante necessidade de reduzir custos de produção e eliminar a presença humana em ambientes potencialmente perigosos e hostis para a saúde tem possibilitado a utilização de veículos guiados autonomamente, *AGV*'s, em diversas aplicações. A utilização destes veículos teria um crescimento ainda mais significativo se estes sistemas se inserissem num mercado de custo reduzido, possibilitando o seu acesso às pequenas e médias empresas.

Este documento descreve a navegação e controlo de um veículo móvel aplicado a um caso concreto da robótica móvel, os *AGV's*.

A autonomia e flexibilidade são necessidades cada vez mais desejadas em sistemas robóticos autónomos com o intuito de fornecer liberdade de movimentos a um robô. Associado a estas duas características surge o problema da localização. Para o presente trabalho, um dos objectivos gerais prende-se no desenvolvimento de um sistema de localização de custo acessível, adequado aos novos mercados. Assim, expõe-se a técnica de localização baseada em marcadores passivos artificiais desenvolvida com o objectivo de detectá-los através de visão artificial.

É apresentada a síntese de uma pesquisa bibliográfica que descreve as diferentes soluções utilizadas nos *AGV's* em relação à temática da localização e da detecção de obstáculos.

Introduzem-se alguns conceitos teóricos pertinentes para a realização e o desenvolvimento deste veículo autónomo, nomeadamente, os métodos de medição do estado do robô, os métodos estocásticos para fusão sensorial e a análise de imagem. Dentro dos métodos estocásticos introduz-se o conceito dos filtros *Kalman* e da sua versão estendida (*EKF*) que fornece ao sistema a capacidade de estimar a posição do *AGV* relativamente a referencial global.

Outro dos objectivos do trabalho que se expõe é o desenvolvimento de um *software* que permite criar uma interface de controlo e decisão da movimentação do robô.

É simulado o robô em estudo através da plataforma de simulação *SimTwo*. Com esta simulação pretende-se verificar o comportamento do robô, testar diferentes estratégias de decisão e controlo e aperfeiçoar o *software* desenvolvido. Este simulador é um sistema de simulação realista onde é possível implementar diferentes tipos de robôs.

Apresentam-se os resultados do comportamento do robô em simulação, recorrendo ao sistema de localização baseado em detecção de parede vertical.

O trabalho realizado enquadra-se nas actividades de investigação do grupo ROBIS – Robótica e Sistemas Inteligentes do INESC Porto, LA em colaboração com a empresa DELTAMATIC – Engenharia e Automação Industrial, S.A.

Abstract

The constant need to reduce costs in production/manufacture and eliminate human presence in environments which are potentially dangerous and hostile for our health, has made possible the usage of automated guided vehicles, AGV's in various applications. The usage of these vehicles would have a much larger potential for growth if these systems were implemented in a low cost market, making it possible for small and medium sized firms to have access to it.

This paper describes the navigation and control of a mobile vehicle when applied to a specific case in mobile robotics, the AGV's.

Autonomy and flexibility are requirements which are more and more desirable in robotic systems, so as to enable the freedom of movements in a robot. Associated to these two characteristics comes the problem of localisation. For the present paper, one of the main aims is linked to the development of a system of location at an affordable price, suitable for new markets. Thus, the location technique, based on artificial passive landmarks and developed with the objective of detecting them through artificial vision will be presented here.

The summary of a bibliographic research which describes the different solutions used in AGV's, in relation to the topic of location and the detection of obstacles will be presented.

Some relevant theoretical terms will be introduced for the making and development of this automated guided vehicle, namely the methods of measuring the state of the robot, the stochastic methods for the sensory fusion and the image analysis. Among the stochastic methods, it will be introduced the concept of the Kalman filters and their extended version (EKF), which gives the system the capability of estimating the position of the AGV.

Another aim is the development of software that allows the making of a control interface and the decision of the robot's movements.

There is a simulation of the robot under study through the simulation platform *SimTwo*. The aim of this simulation is to verify the robot's behaviour, test the different strategies of control and decision making and to perfect the software developed. This simulator is a realistic simulation system where it is possible to implement different kinds of robots.

There is a presentation of the results of the robot's behaviour, using the location system based on vertical wall detection.

The work undertaken is part of the research activities of the ROBIS group – Robótica e Sistemas Inteligentes of INESC Porto, LA in cooperation with the firm DELTAMATIC – Engenharia e Automação Industrial, S.A.

Agradecimentos

É com muita satisfação que expresso aqui o mais profundo agradecimento a todos aqueles que tornaram a realização deste trabalho possível.

Ao Prof. Dr. António Paulo Moreira, meu orientador, e ao Prof. Dr. Paulo Gomes da Costa, meu co-orientador, agradeço pela oportunidade proporcionada em trabalhar no grupo ROBIS – Robótica e Sistemas Inteligentes da FEUP/INESCPorto, e por toda a disponibilidade no apoio e cooperação para o sucesso no desenvolvimento desta dissertação.

Gostaria também de agradecer aos meus colegas de laboratório do Departamento de Engenharia Electrotécnica e de Computadores, Héber Sobreira, Marcos Ferreira, Paulo Malheiros e Alexandre Castro, pela ajuda demonstrada em determinadas fases desta etapa.

Aos meus grandes amigos, um muito obrigado por todos os bons momentos proporcionados ao longo da minha vida, pela amizade compartilhada e pela compreensão e força disponibilizada durante esta dissertação.

Agradeço, de uma forma muito especial, à minha família, à Ana Francisca e à sua família, pelo sentido crítico, pela dedicação, motivação e perseverança demonstrada nos bons e maus momentos passados, que se revelou fundamental para a conclusão desta jornada.

Hugo Sérgio Pinto Alves

x

Índice

Capítulo 1 ..... **1**

Introdução **1**

1.1 Enquadramento Geral 1

1.2 Objectivos 3

1.3 Organização do documento 3

Capítulo 2 ..... **5**

Estado da Arte **5**

2.1 Métodos de localização de *AGV's* 5

2.1.1 Soluções envolvendo filo-guiado 6

2.1.2 Soluções envolvendo trilateração/triangulação laser 7

2.1.3 Soluções envolvendo visão artificial 7

2.1.4 Soluções envolvendo *GPS* 8

2.1.5 Soluções envolvendo sensores ópticos 8

2.1.6 Soluções envolvendo giroscópios 9

2.2 Soluções para detecção de obstáculos 10

2.2.1 Soluções envolvendo pára-choques mecânicos 10

2.2.2 Soluções envolvendo laser 10

2.2.3 Soluções envolvendo sensores ultra-sons 11

Capítulo 3 ..... **13**

Enquadramento Teórico **13**

3.1 Métodos de medição do estado de um robô móvel 13

3.1.1 Medição de posição e orientação relativa 14

3.1.1.1 Odometria 14

3.1.2 Medição de posição e orientação absoluta 15

3.1.2.1 Marcadores Artificiais 15

3.2 Métodos Estocásticos para fusão sensorial 16

3.2.1 Filtro de *Kalman* 17

3.2.1.1 Equações em tempo discreto 17
3.2.2 Filtro de *Kalman* Estendido 19
3.3 Análise de imagem 20
3.3.1 Pixéis e níveis de uma imagem 20
3.3.2 Processamento de imagem binária 21
3.3.2.1 Aquisição de imagem binária 21
3.3.3 Técnicas de iluminação 22

Capítulo 4 23

Mini-AGV – Concepção Estrutural 23
4.1 Descrição Geral 23
4.2 Plataforma de *Hardware* 24
4.2.1 Motores 26
4.2.2 *Encoders* 26
4.2.3 Sistema de alimentação 27
4.2.4 Microcomputador e *LCD* Táctil 28
4.2.5 Sensores de ultra-sons 28
4.2.6 Sistema autónomo de visão artificial 29
4.2.7 Sensores de fim-de-curso 30
4.2.8 Controladores dos motores 30
4.2.9 Conversores de tensão *DC-DC* 31
4.3 Plataforma de *Software* 32
4.3.1 Comunicação assente no protocolo série *RS-232* 32
4.3.2 Comunicação assente no protocolo *UDP* 34
4.3.3 Comunicação assente no protocolo *TCP* 35
4.3.4 Descrição da camada superior 36
4.3.5 Descrição da camada inferior 39
4.3.6 Controladores de velocidade dos motores 39
4.3.6.1 Modelo dinâmico do robô 40
4.3.6.2 Dimensionamento do controlador 42

Capítulo 5 45

Plataforma de Simulação 45
5.1 Descrição Geral 45
5.2 Configuração do simulador 49
5.3 Configuração do corpo do robô, paredes e pista 51
5.3.1 Modelação do robô simulado 51
5.3.2 Modelação dos obstáculos 53
5.3.3 Modelação da pista 53
5.4 Controlo do robô simulado 54

Capítulo 6 ..... **57**

Localização e Navegação **57**

6.1 Descrição Geral 57

6.2 Odometria – Localização relativa 57

6.2.1 Modelo de odometria 58

6.2.2 Calibração da odometria 59

6.3 Sistema de detecção de marcadores - Localização absoluta 61

6.3.1 Visão Artificial 62

6.3.1.1 Calibração da câmara 63

6.3.1.2 Análise/Processamento da imagem 64

6.3.2 Formato das mensagens enviadas por *TCP* 65

6.4 Controlo e Planeamento de Trajectórias 66

6.4.1 Procedimentos de baixo nível 66

6.4.1.1 Goto_Target 66

6.4.1.2 Follow_Line 68

6.4.1.3 Follow_Circle 69

6.4.1.4 Stop 70

6.4.2 Procedimento de mais alto nível 70

6.4.2.1 Exemplo de um possível percurso 72

6.5 Algoritmo de localização absoluta 72

6.5.1 Modelo matemático do método de auto-localização por linhas 73

6.5.2 Estrutura e descrição do algoritmo de localização 74

6.6 Fusão Sensorial 76

6.6.1 Média Pesada 76

6.6.2 Filtro de *Kalman* 76

Capítulo 7 ..... **77**

Resultados **77**

7.1 Robô em simulação 77

7.1.1 Trajectória baseada em localização relativa sem ruído 77

7.1.2 Trajectória baseada em localização relativa com ruído 79

7.1.3 Trajectória baseada em odometria com correcção da orientação do robô através de marcadores 80

7.1.4 Trajectória baseada em odometria com correcção do estado do robô através de marcadores 81

7.1.5 Trajectória baseada em odometria com correcção do estado do robô através de marcadores e ruído aplicado aos *encoders* 82

Capítulo 8 ..... **83**

Conclusões e Trabalhos Futuros **83**

8.1 Conclusões 83

8.2 Trabalhos Futuros 84

Referências 85

Anexo A.....89

Distância de um ponto a uma recta 89

Anexo B.....91

Mapa de marcadores 91

Lista de figuras

Figura 1.1 – Aplicação de *AGV's* no porto marítimo de Roterdão 3

Figura 2.1 – Sistema de localização por filo-guiado [8] 6

Figura 2.2 – *AGV* com tecnologia filo-guiado [2] 6

Figura 2.3 – Sistema de localização por laser [8] 7

Figura 2.4 – *AGV* com tecnologia laser [11] 7

Figura 2.5 – Sistema de localização por *GPS* [13] 8

Figura 2.6 – Sistema de localização baseado em sensores ópticos [8] e [12] 9

Figura 2.7 – *AGV* com sistema de localização baseado em sensores ópticos [2] 9

Figura 2.8 – Sistema de localização por giroscópio [8] 9

Figura 2.9 – Detecção de obstáculos através de pára-choques mecânicos [14] 10

Figura 2.10 – Detecção de obstáculos por laser [15] 11

Figura 2.11 – Detecção de obstáculos através de ultra-sons para a indústria automóvel [17] 11

Figura 2.12 – Detecção de obstáculos em *AGV's* através de sensores de ultra-sons [18] 12

Figura 3.1 – Elementos constituintes de um sistema de fusão sensorial 16

Figura 3.2 – Resumo do processo de estimação e actualização do *KF* em tempo discreto 19

Figura 3.3 – Imagem em escala de cinzentos e imagem binária 21

Figura 3.4 – Esquemas básicos de iluminação 22

Figura 4.1 – Robô desenvolvido para a realização do trabalho em estudo 24

Figura 4.2 – Sistema de tracção diferencial com duas rodas "livres" 24

Figura 4.3 – Arquitectura funcional da plataforma de *hardware* 25

Figura 4.4 – Motores para o movimento da plataforma deslizante [28] 26

Figura 4.5 – Esquema de motor, caixa redutora e *encoder* utilizado para a locomoção 27

Figura 4.6 – *Encoder* Omron E6A2-CW5C [28] 27

Figura 4.7 – Sistema de alimentação..... 27

Figura 4.8 – Microcomputador instalado 28

Figura 4.9 – Ecrã Táctil 28

Figura 4.10 – Sonar SRF01 28

Figura 4.11 – Sistema de aplicação para visão artificial [30] 29

Figura 4.12 – Câmara jAi CV-A11 [30]..... 29

Figura 4.13 – Sensor fotoeléctrico Omron E3F2 [28] 30

Figura 4.14 – *Driver* dos motores de locomoção e deslocamento vertical da plataforma deslizante..... 31

Figura 4.15 – Controlador de velocidade do motor Maxon EC-45 31

Figura 4.16 – Módulo generic atmega8 31

Figura 4.17 – Esquema de ligações *serial* Daisy Chain 33

Figura 4.18 – Esquema sequencial das acções a tomar no ciclo de controlo 36

Figura 4.19 – Máquina de estados do robô 37

Figura 4.20 – Esquema da acção a tomar para eventos externos 38

Figura 4.21 – Interface gráfica da aplicação de controlo do robô 38

Figura 4.22 – Interface gráfica para visualização do movimento do robô 38

Figura 4.23 – Diagrama de blocos do sistema 39

Figura 4.24 – Folha de cálculo para construção gráfica da evolução temporal das velocidades em ambos os motores 40

Figura 4.25 – Evolução temporal das velocidades dos motores 41

Figura 4.26 – Resultados da aproximação do modelo dinâmico do robô a um sistema de 1ª ordem 42

Figura 4.27 – Diagrama de blocos do sistema controlado 42

Figura 4.28 – Sistema simulado com filtro de entrada..... 43

Figura 4.29 – Resposta ao sistema simulado com filtro de entrada 44

Figura 4.30 – Resposta ao degrau da velocidade aplicado às duas rodas motrizes do robô 44

Figura 5.1 – Janela de visualização da interface gráfica *3D* do *SimTwo*..... 47

Figura 5.2 – Janela de configuração do *SimTwo* 47

Figura 5.3 – Janela do editor de código do *SimTwo* 48

Figura 5.4 – Janela de gráficos do *SimTwo* 48

Figura 5.5 – Janela de edição em *XML* dos objectos do *SimTwo* 48

Figura 5.6 – Robô construído numa primeira fase 51

Figura 5.7 – Estrutura final do robô com visualização da distribuição de massa pelo corpo do robô 52

Figura 5.8 – Estrutura final do corpo do robô desenvolvido para a simulação 52

Figura 5.9 – Ambiente final do robô simulado no *SimTwo* 54

Figura 6.1 – Robô diferencial 58

Figura 6.2 – Movimento linear para determinar a relação de impulsos através do deslocamento 60

Figura 6.3 – Movimento circular para determinar a distância entre rodas 60

Figura 6.4 – Esquema representativo da aquisição da imagem pela câmara 62

Figura 6.5 – Imagem original capturada pela câmara instalada no robô 63

Figura 6.6 – Imagem utilizada para método de calibração 64

Figura 6.7 – Imagem original delimitada a verde pela *ROI* 65

Figura 6.8 – Imagem após pré-processamento e segmentação 65

Figura 6.9 – Esquema representativo do procedimento *Goto_Target* 67

Figura 6.10 – Máquina de estados do procedimento *Goto_Target* 67

Figura 6.11 – Representação esquemática do procedimento *Follow_Line* 68

Figura 6.12 – Máquina de estados do procedimento *Follow_Line* 68

Figura 6.14 – Máquina de estados para o procedimento *Follow_Circle* 69

Figura 6.13 – Esquema representativo do procedimento *Follow_Circle* 69

Figura 6.15 – Imagem representativa dos percursos e sentidos a realizar pelo robô ao longo da trajectória em "H" 70

Figura 6.16 – Máquina de estados para o procedimento *Follow_Path* 71

Figura 6.17 – Exemplo de um percurso a realizar pelo robô 72

Figura 6.18 – Esquema representativo da medição da distância do robô a uma parede lateral 73

Figura 6.19 – Esquema representativo da medição da distância do robô a um marcador artificial 74

Figura 6.20 – Diagrama de estados do processo de localização 75

Figura 7.1 – Trajectória baseada em odometria 78

Figura 7.2 – Trajectória baseada em odometria com ruído aplicado aos *encoders*..... 79

Figura 7.3 – Trajectória baseada em odometria com correcção da orientação do robô através de marcadores 80

Figura 7.4 – Trajectória baseada em odometria com correcção da orientação e posição do robô através de marcadores 81

Figura 7.5 – Trajectória baseada em odometria com correcção do estado do robô através de marcadores e ruído aplicado aos *encoders* 82

Lista de tabelas

Tabela 4.1 – Formato da trama do protocolo Daisy Bin 33

Tabela 4.2 – Resumo dos campos que constituem a trama do protocolo Daisy Bin 34

Tabela 4.3 – Cabeçalho das mensagens do protocolo *UDP* 35

Tabela 4.4 – Definição de *IPs* e Máscara de sub-rede para a comunicação baseada no protocolo *TCP* 35

Tabela 4.5 – Parâmetros do modelo dinâmico do robô 42

Tabela 4.6 – Parâmetros do *PI* utilizados nos motores 43

Tabela 5.1 – Classes abrangidas pela estrutura hierarquia mais elevada do *SimTwo* 49

Tabela 5.2 – Objectos básicos do *SimTwo* e suas propriedades 49

Tabela 5.3 – Objectos mais complexos suportados pelo *SimTwo* 50

Tabela 5.4 – Propriedades dos objectos mais complexos abrangidos pelo *SimTwo* 50

Tabela 5.5 – Formato da mensagem enviada pelo *SimTwo* 55

Tabela 5.6 – Formato da mensagem recebida pelo *SimTwo* 55

Tabela 6.1 – Parâmetros da odometria obtidos para a plataforma em estudo 61

Tabela 6.2 – Formato da mensagem enviada por *TCP* 65

Tabela 6.3 – Transições da máquina de estados do procedimento *Follow_Path* 71

Tabela 6.4 – Percursos efectuados em cada estado do procedimento *Follow_Path* 71

Tabela 6.5 – Transições da máquina de estados do processo de localização 75

Abreviaturas e Símbolos

Lista de abreviaturas

3D	*Three Dimensional*
5DPO	Grupo de Futebol Robótico da FEUP
AGV	*Automated Guided Vehicle*
CPU	*Central Processing Unit*
DC	*Direct Current*
DEEC	Departamento de Engenharia Electrotécnica e de Computadores
DPI	*Dots per Inch*
EKF	*Extended Kalman Filter*
FEUP	Faculdade de Engenharia da Universidade do Porto
GPS	*Global Positioning System*
IP	*Internet Protocol*
ISO	*International Organization for Standardization*
KF	*Kalman Filter*
LCD	*Liquid Crystal Display*
ODE	*Open Dynamics Engine*
PI	*Proportional, integral*
PID	*Proportional, integral, derivativo*
PLS	*Proximity Laser Scanner*
PPI	*Pixels per Inch*
PWM	*Pulse Width Modulation*
ROI	*Region of Interest*
RF	*Radio Frequency*
TCP	*Transmission Control Protocol*
XML	*Extensible Markup Language*

Capítulo 1

Introdução

Numa sociedade imersa em novas tecnologias, onde a sucessiva competitividade e o elevado ritmo gera cada vez mais a criação e a procura de novas soluções capazes de permitir o avanço, o progresso e a ascensão de empresas, pessoas e instituições, surge com pertinência a necessidade de desenvolver produtos de custo mais acessível com a profunda intenção de superar a investida de outras soluções, promovendo assim o desenvolvimento económico.

O tema apresentado nesta dissertação será estudado para um caso particular de robótica móvel – os veículos guiados autonomamente (*AGV*). Este trabalho será desenvolvido em colaboração com a empresa DELTAMATIC – Engenharia e Automação Industrial, S.A. (www.deltamatic.pt) – uma vez que o estudo e os resultados se revelam pertinentes para a mesma.

A constante necessidade de optimizar recursos humanos para diminuir custos de produção, aumentar a produtividade de um sistema, eliminar a presença humana em ambientes potencialmente perigosos e hostis para a saúde, reduzir danos nos produtos por manuseamento, aliada à franca ascensão dos sistemas robóticos autónomos na indústria e serviços, tem possibilitado a utilização de *AGV*'s em diversas aplicações. Tal crescimento seria ainda mais significativo se estes sistemas se inserissem num mercado de custo reduzido, possibilitando o seu acesso às pequenas e médias empresas.

1.1 Enquadramento Geral

Até ao aparecimento da robótica móvel, o transporte e manuseamento de materiais era realizado com uma forte intervenção humana, como era o caso dos veículos guiados manualmente. Actualmente, estas tarefas são realizadas por veículos guiados autonomamente, vulgarmente designados por *AGV*.

O primeiro *AGV* apareceu no mercado em 1950, por Barrett Electronics de Northbrook, em Illinois, e consistia apenas num veículo de reboque que seguia um fio eléctrico embutido no pavimento, substituindo o tradicional sistema de carril. Ao longo dos anos, com a progressiva sofisticação da tecnologia os veículos tornaram-se autónomos, potencializando a tecnologia laser *LGV* para a sua localização. Sendo assim, considerando o caso típico de um robô móvel que se desloca num ambiente bidimensional, localizar o mesmo significa conhecer a sua posição cartesiana e orientação $[x \quad y \quad \theta]^T$ em coordenadas do mundo. Aliado à localização surge o conceito de navegação [1].

A navegação pode ser considerada como a plataforma intermédia que faz cumprir a ordem de chegar a um certo destino. De uma forma geral, é possível entender a navegação de robôs móveis como o problema de determinação de trajectórias, proporcionando condições para que as tarefas (objectivos) sejam executadas com desempenho aceitável. Estas técnicas utilizam diferentes tipos de sensores (infravermelho, ultra-sons, táctil, visão) como parte do equipamento do robô móvel, para realizar as suas tarefas de movimentação e controlo, desde uma posição inicial até uma posição final definida.

Em sistemas de processos automatizados, os *AGV's* são programados para cooperarem com outros robôs, com o intuito de garantir que o transporte do produto é realizado de uma forma suave. Presentemente, os veículos guiados autonomamente desempenham um papel importante na concepção de novas fábricas e armazéns, permitindo o transporte de mercadorias em segurança [2].

No final do século XX, os *AGV*'s assumiram novas funções e esta tecnologia começou a ser fortemente utilizada para movimentar contentores industriais (*ISO* Estandardizado). A título de exemplo, destaca-se o porto marítimo de Roterdão o qual utiliza mais de 100 *AGV*'s na realização desta função, figura 1.1 [2]. É de salientar que o controlo destes veículos depende das características do meio ambiente[1], bem como do tipo de tarefa e do desempenho exigido os quais influenciam bastante na escolha da técnica de localização [3]. No Capítulo 2 são apresentadas as técnicas de localização utilizadas actualmente nos *AGV's*.

As aplicações deste tipo de veículos são intermináveis podendo os mesmos possuir uma capacidade de carga que pode variar de apenas alguns quilos a centenas de toneladas.

---

[1] Pode-se caracterizar por um ambiente interno fechado com características previamente conhecidas, *indoor*, ou num ambiente externo não necessariamente conhecido, *outdoor*.

Figura 1.1 – Aplicação de *AGV's* no porto marítimo de Roterdão

1.2 Objectivos

A presente dissertação tem como objectivos gerais apresentar uma técnica de localização de custo reduzido baseada em marcadores passivos artificiais previamente colocados pelo percurso a percorrer pelo robô e um sistema de controlo e navegação para o *AGV*.

Assim, para a concretização destes objectivos esta dissertação será enformada nas seguintes etapas:

- Levantamento de estudos já realizados sobre veículos guiados autonomamente e de outras tecnologias similares;
- Revisão bibliográfica referente à problemática de localização;
- Simulação do comportamento do robô na plataforma de simulação *SimTwo*;
- Desenvolvimento e implementação do sistema de localização baseado em marcadores passivos localizados no pavimento;
- Desenvolvimento e implementação do *software* de controlo da movimentação;
- Desenvolvimento e implementação do sistema de navegação e seguimento de trajectórias.

1.3 Organização do documento

Esta secção tem como objectivo descrever a estrutura e organização deste documento, explicando sucintamente o que será desenvolvido em cada capítulo.

Após este capítulo introdutório far-se-á, no capítulo 2, uma análise do estado da arte quanto aos métodos mais utilizados na localização e na detecção de obstáculos de um *AGV*.

No capítulo 3 apresentam-se alguns conceitos teóricos pertinentes que posteriormente serão aplicados no desenvolvimento do trabalho em estudo.

No capítulo 4 descreve-se a implementação do robô desenvolvido neste trabalho, sendo também descritos os vários elementos constituintes do *hardware* e do *software*.

Segue-se o capítulo 5, plataforma de simulação, onde se apresenta a ferramenta de simulação utilizada para testar o comportamento do robô.

No 6º capítulo, localização e navegação, apresentam-se a técnica de localização relativa e absoluta aplicada ao robô, bem como o planeamento de trajectórias e os algoritmos desenvolvidos para a localização do robô.

O capítulo 7 exibe alguns dos resultados obtidos relativos à localização do robô quando percorre um percurso de um ponto inicial do mapa para um ponto final.

Finalmente, no capítulo 8 apresentam-se as conclusões gerais retiradas no desenvolvimento do presente trabalho e, tendo em consideração o que foi apreendido, remete-se ainda para alguma propostas a realizar no futuro.

Capítulo 2

Estado da Arte

Com o crescimento do poder de processamento computacional e com o sucessivo progresso no desenvolvimento de tecnologias de localização para veículos autónomos, surgiram os primeiros robôs móveis, com uma vasta aplicação em diversos serviços e indústrias. Como já foi referido no capítulo anterior, estes tinham como intuito a redução da intervenção humana, de forma a reduzir e optimizar os recursos humanos.

Segundo Rocha [4], no desenvolvimento de um robô móvel procura-se aliar a eficiência de manuseamento e manipulação de robôs convencionais (manipuladores robóticos) com a autonomia em sistemas de navegação de um veículo autónomo, num único dispositivo de automação móvel.

Na aplicação industrial, as fábricas de produção utilizam veículos móveis para incorporar trabalhos com outras máquinas, com a finalidade de se tornarem sistemas modernos de produção flexível [5]. Neste domínio, os veículos guiados autonomamente são os que apresentam maior aplicação em empresas de produção ou de distribuição, relativamente à robótica móvel, realizando variadíssimas tarefas, tais como a manipulação e armazenamento de materiais. Podem ser também utilizados em escritórios para troca de correspondência ou documentos, bem como nos aeroportos para o transporte de clientes embora ainda não seja muito habitual encontrar esta tecnologia nos aeroportos.

A eficiência e a eficácia dos sistemas de produção são influenciadas pelo nível de optimização do movimento de materiais dentro da fábrica. Sendo assim, o desempenho de um sistema automático de transporte é essencial para conseguir um bom desempenho geral do sistema de produção.

2.1 Métodos de localização de *AGV's*

Como já foi referido, os veículos guiados autonomamente, *AGV*, têm actualmente uma enorme importância no transporte interno e externo de materiais na indústria de produção, tendo-se assistido a um forte alargamento da utilização destes veículos para outras áreas como o armazenamento.

Os *AGV*'s são veículos equipados com sistemas de orientação automática capazes de seguir os caminhos pré-definidos. A precisão de localização é, consequentemente, de extrema importância para o alcance da verdadeira autonomia do veículo, contribuindo assim para o seu sucesso. A verdadeira problemática associada à localização resume-se à questão "onde estou?", referindo-se ao meio em que está inserido o veículo [6].

Desta forma, ao longo dos anos assistiu-se a uma evolução contínua no que diz respeito à autonomia e precisão de localização dos *AGV's*. Existem diversos métodos de localização que permitem a um *AGV* seguir um caminho fixo ou um caminho dinâmico. Assim, a determinação dos caminhos depende dos requisitos de flexibilidade, dos custos de instalação e da necessidade ou não da futura ampliação do sistema. Os sistemas com caminhos fixos são menos dispendiosos no que se refere à instalação, todavia inviabilizam a possibilidade de reagir a alterações no *layout* do trabalho sem interromper o seu funcionamento normal, acarretando por isso custos adicionais.

Na secção 2.1.1, 2.1.5 e 2.1.6 serão abordadas soluções de localização baseadas em caminhos fixos. Os métodos baseados em caminhos dinâmicos serão apresentados nas secções 2.1.2, 2.1.3 e 2.1.4.

2.1.1 Soluções envolvendo filo-guiado

O sistema baseado em filo-guiado, representado na figura 2.1 e 2.2, é a forma mais simples de navegação, uma vez que estabelece um percurso fisicamente pré-definido. Esta solução, segundo Rocha [4], é usada em grande escala devido à sua robustez e simplicidade. É baseada num fio électrico embutido no pavimento e num sistema de detecção do campo magnético do veículo, ou seja, um sensor para detectar o sinal de *RF* proveniente do fio embutido (aproximadamente 2 a 3 centímetros abaixo da superfície do piso). O sensor detecta o sinal e ajusta a posição do veículo para mantê-lo no caminho. Uma vez que este sistema implica a abertura de um corte no pavimento, estes sistemas de filo-guiado são comummente utilizados quando os percursos não são susceptíveis de alteração [7]. Apesar dessa limitação, os sistemas de filo-guiado são frequentemente usados em aplicações que requerem um alto grau de precisão no caminho a percorrer pelo *AGV*. É exemplo desta utilização a viagem de ida e volta entre duas estações de trabalho numa área congestionada.

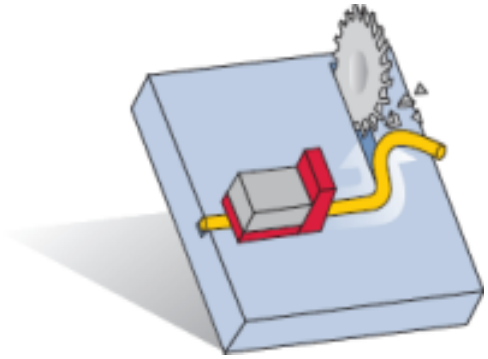


Figura 2.1 – Sistema de localização por filo-guiado [8]

Figura 2.2 – *AGV* com tecnologia filo-guiado [2]

2.1.2 Soluções envolvendo trilateração/triangulação laser

A técnica envolvendo trilateração/triangulação laser utiliza painéis reflectores instalados nas colunas, paredes, máquinas ou postos, como se ilustra na figura 2.3, sendo estabelecido um único ponto de coordenada X, Y, para cada painel [7]. Na figura 2.4 encontra-se representada esta técnica, que consiste na incorporação de um *laser scanner* no veículo. Este efectua varrimentos rotativos detectando as posições dos painéis reflectores colocados ao longo do *layout* [9]. Segundo Rocha [4], a posição do *AGV* é obtida aplicando o método da trilateração/triangulação. Este método define a localização do veículo através da medição da distância e orientação dos feixes de luz reflectidos pelos reflectores e, consequentemente, a triangulação destas medidas permite determinar a posição do *AGV*. O mapa do *layout* dos reflectores distribuídos pelo percurso é armazenado numa base de dados e acessível pelo *AGV*. Esta solução apresenta elevada flexibilidade, pois é possível criar um novo *layout* sem que seja necessário efectuar interrupções ao funcionamento normal do veículo [9] e [10]. Este método apresenta-se adequado para um ambiente industrial, desde que não exista excessiva poeira no ar.

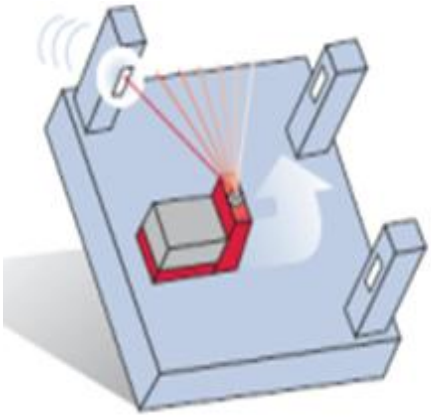


Figura 2.3 – Sistema de localização por laser [8]

Figura 2.4 – *AGV* com tecnologia laser [11]

2.1.3 Soluções envolvendo visão artificial

Na técnica envolvendo visão artificial, a navegação do veículo é realizada através da captura de informação visual obtida a partir de marcadores (*landmarks),* previamente distribuídos pelo percurso a percorrer pelo mesmo. Estes marcadores são normalmente figuras geométricas ou sinais colocados num *layout* pré-definido e pré-conhecido pelo *AGV*. Segundo Beccari [12], esta tecnologia é fortemente utilizada para a navegação de veículos inseridos em ambientes estruturados[2] e quando se pretende um custo de implementação relativamente baixo.

[2] Caracterizam-se por ambientes estáveis, onde não há variação da posição dos componentes ou equipamentos, nem um elevado fluxo na circulação de pessoas.

2.1.4 Soluções envolvendo *GPS*

A solução de localização por *GPS*, representada na figura 2.5, apenas se aplica a situações de navegação quando o veículo circula em ambientes exteriores e é utilizada normalmente para veículos de grande porte onde não é necessária uma elevada precisão na posição do *AGV*. Esta tecnologia é usualmente utilizada no transporte de pessoas nos aeroportos ou, como foi referido no capítulo anterior, para transporte de mercadorias de grandes dimensões em percursos longos, como contentores industriais. Este método, para Leitão et al [13], não permite a recolha ou armazenamento automático de produtos nem a cooperação com outros robôs num sistema de produção/distribuição flexível, devido à falta de precisão na determinação da posição do veículo.

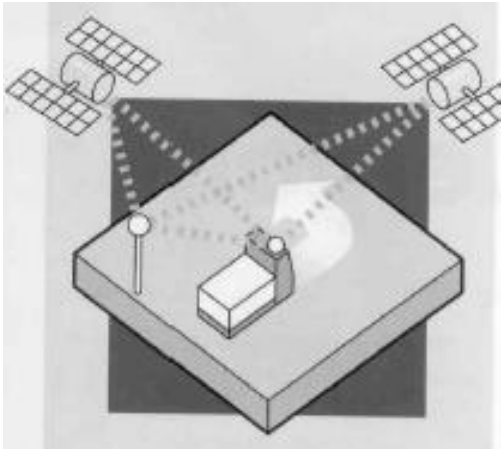


Figura 2.5 – Sistema de localização por *GPS* [13]

2.1.5 Soluções envolvendo sensores ópticos

A solução assente em sensores ópticos utiliza uma fita magnética ou colorida, como se representa na figura 2.6, para estabelecer o percurso do *AGV*. A utilização destas fitas assume algumas vantagens perante os sistemas de filo-guiado, uma vez que estas podem ser facilmente removidas e recolocadas se for necessário alterar o percurso a percorrer pelo veículo, para além do reduzido custo de instalação [10]. Este facto deve-se à não necessidade de embutir a fita no pavimento. O sistema representado na figura 2.7 é considerado um sistema passivo, uma vez que não exige que o meio de acompanhamento do veículo seja magnetizado [2]. Esta tecnologia poder-se-á tornar instável quando incorporada em áreas de elevado tráfego, onde a fita pode ficar danificada ou suja, inviabilizando a localização do *AGV*.

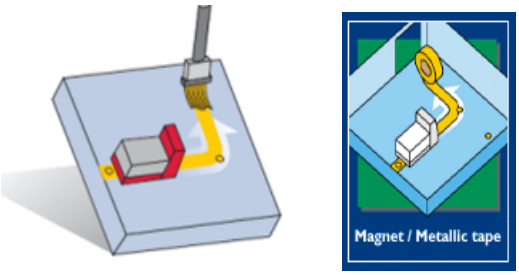


Figura 2.6 – Sistema de localização baseado em sensores ópticos [8] e [12]

Figura 2.7 – *AGV* com sistema de localização baseado em sensores ópticos [2]

2.1.6 Soluções envolvendo giroscópios

A tecnologia envolvendo giroscópios surgiu na década de 90 como alternativa à solução de filo-guiado. No pavimento são embutidos pequenos marcadores (*transponders*), ilustrados na figura 2.8, e cada *AGV* é equipado com um giroscópio. Estes dispositivos possuem uma elevada capacidade de detectar pequenos desvios na direcção do veículo e são utilizados em conjunto com os marcadores para determinar a posição do veículo guiado autonomamente [7] e [10]. Estes marcadores são utilizados como pontos de referência para corrigir qualquer pequeno erro acumulado ao longo da distância entre os marcadores. Normalmente encontra-se um marcador a cada 25 metros. Do mesmo modo que na navegação baseada na triangulação a laser, o caminho a percorrer é um jogo virtual de coordenadas armazenadas na memória de cada *AGV* ou numa base de dados central. A margem de erro para o método de inércia é de mais ou menos 1 centímetro [7].

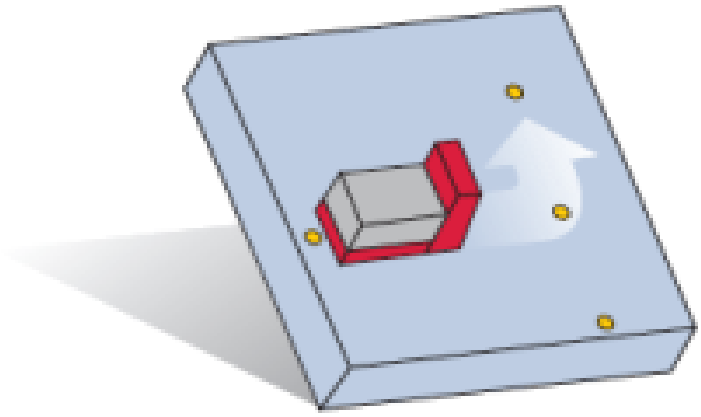


Figura 2.8 – Sistema de localização por giroscópio [8]

2.2 Soluções para detecção de obstáculos

2.2.1 Soluções envolvendo pára-choques mecânicos

Os pára-choques mecânicos, representados na figura 2.9, são a forma tradicional de detecção de obstáculos para sistemas de localização baseados em filo-guiado. A zona de detecção de um pára-choques mecânico é fixa e não varia durante o funcionamento do veículo [9]. Quando há contacto, é necessário um tempo adicional para parar o veículo, dependendo este mesmo tempo da velocidade do veículo [14]. O veículo deverá circular a velocidades baixas para evitar o contacto com um objecto, o que a acontecer poderá resultar numa perda de produtividade.

Figura 2.9 – Detecção de obstáculos através de pára-choques mecânicos [14]

2.2.2 Soluções envolvendo laser

A solução de laser, ilustrada na figura 2.10, pode ser aplicada separadamente ou em conjunto com o pára-choques mecânico, sendo esta a forma mais habitual de detecção de obstáculos embora relativamente dispendiosa [14]. Esta configuração de detecção oferece uma solução de manutenção flexível e livre quando o ambiente de operação é partilhado por pessoas [9].

Para Leitão et al [15], esta tecnologia permite que os utilizadores definam diferentes zonas de segurança para a detecção de obstáculos com diferentes actuações em cada zona por parte do *AGV*, podendo ser alteradas durante a operação do veículo. Com a ajuda do laser pode-se calcular a distância a que o obstáculo se encontra, criando-se diferentes zonas:

- Zona de Segurança Máxima - Não efectua manobras perigosas. Qualquer movimentação faz parar o *AGV*;
- Zona de Perigo – Inicia a manobra de desvio do obstáculo, ou emite aviso;
- Zona Examinada – Raio máximo observado pelo *PLS*.

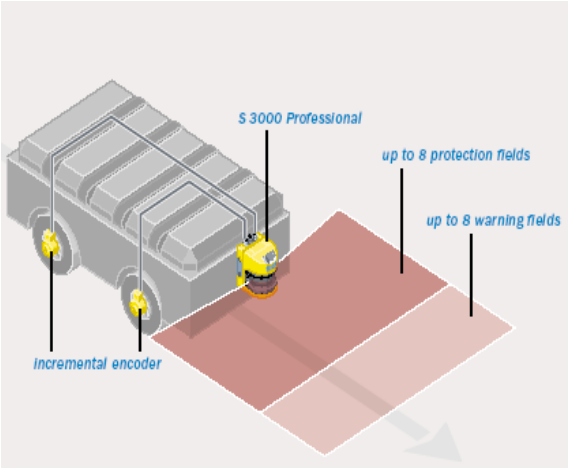


Figura 2.10 – Detecção de obstáculos por laser [15]

2.2.3 Soluções envolvendo sensores ultra-sons

A solução envolvendo sensores de ultra-sons utiliza o tempo de propagação de uma onda sonora para medir a distância a um objecto. Estes dispositivos são constituídos por um emissor e um receptor de ultra-sons. Um sistema é responsável por gerar uma onda com determinada frequência que é transmitida por um emissor de ultra-sons. Quando o sensor receptor de ultra-sons detecta o eco do sinal enviado, o sistema, que mantém o registo do tempo decorrido, desde que o sinal foi enviado até que o eco foi detectado, sabendo a velocidade do som, calcula a distância a que o objecto se encontra [16].

Esta solução ainda não se apresenta muito comum em *AGV's* embora seja uma solução que se encontra em franca ascensão na indústria automóvel. Para a indústria automóvel, a versão mais recente desta solução permite, para além de detectar obstáculos, detectar espaços livres que permitem estacionar o veículo através da utilização dos sensores ultra-sónicos colocados lateralmente [17].

Figura 2.11 – Detecção de obstáculos através de ultra-sons para a indústria automóvel [17]

Quanto aos *AGV's* esta solução é adoptada por [18] que estabelece vários sensores na zona frontal e na retaguarda, garantindo que os sensores consigam detectar um obstáculo durante o seu movimento.

Figura 2.12 – Detecção de obstáculos em *AGV's* através de sensores de ultra-sons [18]

Capítulo 3

Enquadramento Teórico

Neste capítulo introduzir-se-ão os conceitos teóricos relevantes para o desenvolvimento deste trabalho. Serão descritos de uma forma resumida com o intuito de permitir enquadrar as temáticas que serão desenvolvidas posteriormente durante a implementação do robô. Assim, na secção 3.1 serão abordados os métodos de localização relativa e absoluta, focando os métodos utilizados no desenvolvimento deste trabalho. Na secção 3.2 serão introduzidos os conceitos teóricos que permitirão a fusão da informação baseando-se no Filtro de *Kalman* (*KF),* subsecção 3.2.1, e no Filtro de *Kalman* Estendido (*EKF),* subsecção 3.2.2. A secção 3.3 apresenta algumas noções sobre análise/processamento de imagem exibindo alguns algoritmos que permitem retirar as características pretendidas de uma imagem monocromática.

3.1 Métodos de medição do estado de um robô móvel

O estado de um robô móvel, $X = [\mathrm{x} \quad \mathrm{y} \quad \theta]^{\mathrm{T}}$, designa a posição cartesiana e a orientação de um veículo móvel em relação a um referencial global.

Como foi referido anteriormente, segundo Borenstein [6], o problema da navegação de um robô móvel resume-se a três questões colocadas do ponto de vista do robô: "Onde estou?", "Para onde vou?", "Como fazer para lá chegar?". Deste modo, nesta secção pretende-se responder à primeira questão introduzida que corresponde à problemática da localização, tendo em consideração o robô em estudo.

A localização de um veículo autónomo permite definir a posição e orientação segundo um referencial mundo. Para tal, torna-se então necessário introduzir métodos que permitam medir o seu estado. Segundo Borenstein [6], estes métodos podem ser classificados em dois grupos: métodos relativos e absolutos.

3.1.1 Medição de posição e orientação relativa

A medição relativa do estado de um robô apresenta-se como um método que, normalmente, não requer a preparação do ambiente e tem como finalidade, garantir que o veículo se encontra localizado em todos os instantes. Assim, o cálculo da posição e da orientação actual do veículo é baseado na informação fornecida pelos sensores e determinado mediante a actualização de uma posição e orientação medidas anteriormente [19].

As estimativas do estado do robô fornecidas pelos métodos de localização relativa são muito úteis na avaliação de todas as outras informações disponíveis sobre a navegação.

Os métodos mais comuns utilizados para localização relativa são baseados em sensores inerciais. Nesta secção apenas será apresentado o método baseado em odometria, visto ser o que será objecto de estudo neste trabalho.

3.1.1.1 Odometria

A odometria é o método de localização relativa mais amplamente utilizado para a estimação da posição e orientação de um robô [6]. Esta técnica consiste na integração da informação incremental do movimento linear das rodas motrizes ao longo do tempo, ou seja, na medição da rotação das rodas motrizes, de forma a determinar a distância percorrida por um veículo, a sua velocidade e a direcção do movimento. Para este efeito, os sensores mais utilizados são potenciómetros, *resolvers* e *encoders* acoplados ao veio das rodas motrizes do robô.

O deslocamento linear traduzido através da rotação das rodas, por vezes não transmite uma elevada precisão para a estimação do estado do robô. A título de exemplo, destaca-se o caso de derrapagem das rodas. Este caso é traduzido em impulsos gerados pelos sensores que serão interpretados como um deslocamento linear da roda motriz e que, na maior parte dos casos, não corresponde ao real deslocamento. Este é um dos variadíssimos casos de erros aplicados à odometria do robô e que será acumulado em cada iteração no cálculo de posição e orientação do robô prejudicando igualmente, a determinação do seu estado [6] e [19].

Os erros resultantes da aplicação deste método podem ser categorizados em erros sistemáticos e erros não sistemáticos [6], [19] e [20].

Os erros sistemáticos são específicos do robô e podem ter as seguintes origens:

- Rodas com diferentes diâmetros;
- Diâmetro médio das rodas diferentes do seu valor nominal;
- Distância entre rodas diferente da nominal;
- Desalinhamento das rodas;
- Deformação dos pneus das rodas com a carga;
- Variação do raio das rodas com a carga ou distribuição da carga no veículo.

Os erros não sistemáticos são produzidos pela interacção do piso com as rodas do robô e não podem possuir as seguintes proveniências:

- Resolução dos *encoders* limitada;
- Taxa de amostragem dos *encoders* limitada;
- Incerteza na base das rodas (ponto de contacto com o piso);
- Pavimentos irregulares ou com objectos inesperados durante a trajectória do robô;

- Deslizamento das rodas devido a pisos escorregadios, acelerações excessivas, curvas efectuadas demasiado rápido, forças exteriores (interacção com corpos exteriores), forças interiores (rodas "livres"), falta de contacto pontual entre as rodas e o piso.

Apesar deste método ser de custo reduzido no que diz respeito à determinação da posição e orientação de um robô, todos os sistemas odométricos caracterizam-se por um erro cumulativo que é função da distância percorrida. Por esta razão a odometria deve ser calibrada de forma a minimizar os erros sistemáticos e é normalmente utilizada em conjunto com outros métodos, por exemplo através da identificação de referências absolutas. Informações sobre a calibração da odometria podem ser encontradas na referência [21].

3.1.2 Medição de posição e orientação absoluta

Os métodos utilizados para a localização absoluta consistem na determinação da posição e da orientação de um veículo de um modo independente de suposições sobre movimentos anteriores [20].

A estimação da posição baseada em métodos absolutos é normalmente confinada à trilateração ou triangulação com balizas, bússola magnética, reconhecimento de marcadores artificiais ou naturais, correspondência de mapas. Nesta secção apenas será apresentado o método de localização absoluta baseado em marcadores artificiais.

3.1.2.1 Marcadores Artificiais

A utilização da localização baseada em marcadores artificiais necessita que o ambiente do robô seja previamente configurado mediante a colocação de marcadores no ambiente em que o veículo se insere. Para este método, os marcadores artificiais recorrentemente utilizados são marcadores ópticos, marcadores magnéticos e *transponders* embora, usualmente, a estimação da localização de um robô seja baseada em marcadores ópticos detectados através de um sistema de visão.

A posição de cada marcador terá que ser conhecida antecipadamente pelo veículo e terá que constituir o percurso desejado para a trajectória do robô, com o fim exclusivo de possibilitar a sua localização [6] e [20]. A utilização de marcadores artificiais permite que o robô conheça desde logo as suas características, tais como a forma, o tamanho. De uma forma geral, os marcadores (*landmarks*) são cuidadosamente escolhidos para serem facilmente identificáveis/detectáveis e possuem uma posição fixa e conhecida a partir da qual o robô se pode localizar [6]. Quando se utiliza um sistema de visão artificial para detectar os marcadores, estes devem possuir como característica principal, um elevado contraste com o seu fundo (*background)*.

Este método apresenta-se bastante vantajoso quando se pretende implementar um sistema de localização de custo reduzido, pois os marcadores artificiais apresentam um custo efectivo desprezável.

A localização baseada em marcadores artificiais é inerentemente intermitente, uma vez que estes se encontram espaçados no ambiente e, geralmente, um veículo só consegue

reconhecer um marcador de cada vez e quando se encontra bastante próximo deste. Assim, a navegação baseada em marcadores artificiais é bastante inexacta quando o veículo se encontra distante do marcador que nesse momento estiver a ser utilizado para a localização. O grau de exactidão da localização apenas se torna mais elevado quando o robô detecta um marcador. Para a navegação entre dois marcadores é então necessário recorrer a outro método de localização (por exemplo, à odometria). O não reconhecimento de apenas um dos marcadores pode fazer com que o veículo se perca.

A exactidão com que o robô estima a sua posição diminui com o aumento da distância entre o robô e o marcador. A orientação de um veículo relativamente a um marcador é outro factor que permite estimar esta exactidão. Outro factor preponderante para a localização de um veículo através deste método passa pela iluminação.

A navegação por marcadores artificiais requer que o estado do robô seja aproximadamente conhecido à partida, para que o robô só precise de procurá-los numa área limitada.

3.2 Métodos Estocásticos para fusão sensorial

A localização em sistemas robóticos autónomos é um assunto complexo, e que requer especial atenção quando é necessária uma elevada precisão na obtenção da posição de um robô. A constante necessidade de reduzir o custo efectivo de um sistema através da substituição de um sensor de elevada precisão (custo elevado), por vários sensores de baixo custo (menor precisão), permitiu introduzir uma nova área denominada fusão multi-sensorial de dados [22].

Define-se fusão multi-sensorial, neste contexto, como a fusão de informação proveniente de diversos sensores, que combinados, permitem obter um melhor desempenho, ou seja, obter mais precisão do que com um único sensor, fornecendo assim uma descrição mais robusta e completa do ambiente [23].

O termo fusão sensorial, apesar de parecer bastante complexo, envolve poucas ideias essenciais: múltiplas entradas, uma ou mais técnicas de fusão, o resultado pretendido e o mecanismo de controlo do sistema, figura 3.1. A fusão de sensores é um termo que abrange uma série de métodos e algoritmos, incluindo o filtro de *Kalman*. O filtro de *Kalman* através de informação proveniente dos diferentes sensores, permite realizar uma estimação do estado do robô.

Nesta secção, o método estudado e apresentado foi o filtro de *Kalman* e a sua vertente *EKF* (*Extended Kalman Filter*). Apresentam-se também, de uma forma resumida, as equações que regem estes conceitos.

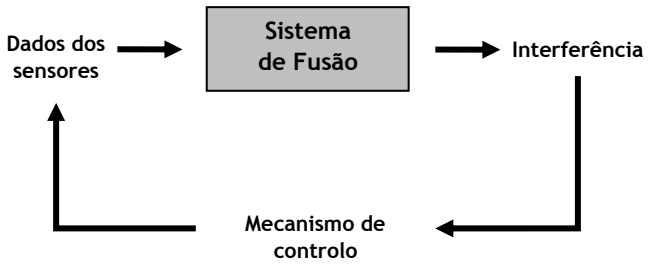


Figura 3.1 – Elementos constituintes de um sistema de fusão sensorial

3.2.1 Filtro de *Kalman*

O filtro de *Kalman* ou filtro de *Kalman-Bucy,* introduzido pelo matemático R. E. Kalman, é o método mais amplamente utilizado por fusão sensorial em aplicações de robótica móvel. Trata-se de um algoritmo recursivo de fusão de informação, utilizado para prever os estados de um sistema dinâmico, baseado na medição da fiabilidade dos dados observados a partir de uma matriz de co-variâncias calculada a cada novo processo de estimação que se utiliza o filtro, garantindo a convergência para a solução óptima [24]. Com base num conjunto de equações matemáticas é estimado o estado do processo, de modo a minimizar a co-variância do erro estimado. Se um sistema pode ser descrito como um modelo linear e as incertezas dos sensores e do sistema modeladas como ruído branco[3], então o *KF* fornece uma estimativa estatisticamente aceitável para a informação recebida. Isto significa que, sob certas condições, o filtro de *Kalman* é capaz de encontrar a melhor estimativa baseada na correcção de cada medida [24].

As premissas em que este filtro se baseia são [25]:

- O sistema deve ser linear;
- O ruído que afecta o sistema é *gaussiano* com média nula e variância constante;
- O ruído do sistema e as medidas não estão correlacionados.

3.2.1.1 Equações em tempo discreto

O filtro de *Kalman* pretende estimar o estado $x \in \mathrm{R}^n$ de um processo controlado em tempo discreto com base na equação às diferenças lineares estocásticas

$$x_{k+1} = A_k x_k + B_k u_k + w_k, \tag{3.1}$$

com medições de $y \in \mathrm{R}^m$ obtidas por

$$y_{k+1} = C_{k+1} x_{k+1} + v_k. \tag{3.2}$$

As variáveis aleatórias v_k e w_k representam o processo de mensuração dos ruídos e são assumidas como sendo independentes uma da outra, com ruído branco e distribuição normal

$$P(w) \sim N(0, Q), \tag{3.3}$$

$$P(v) \sim N(0, R), \tag{3.4}$$

[3] Caracteriza-se por ser ruído associado a determinada medida ou processo cuja característica principal é a sua aleatoriedade, logo não determinístico, o que resulta num erro de medida ou estimação do processo.

em que Q representa o ruído do processo da co-variância e R o ruído da mensuração da co-variância.

Este filtro permite obter um estimador óptimo se as variáveis, v_k e w_k, forem processos *gaussianos* de média nula.

O estado $x = [x_1\ x_2 \ldots x_n]^T$ de um processo, representa a partir de *n* a dimensão do sistema. As entradas deste são caracterizadas por *p* através de $u = [u_1\ u_2 \ldots u_p]^T$. A mensuração $y = [y_1\ y_2 \ldots y_q]^T$ do processo estipula *q* como o número de saídas do sistema.

Na equação (3.1), a matriz A com $\dim(A) = n \times n$ e a matriz B de $\dim(B) = p \times n$, relacionam, o estado anterior, k, com o estado actual, k+1, e com as entradas $u \in R^p$, respectivamente. Na equação (3.2) representa-se a matriz C de $\dim(C) = q \times n$ que relaciona o estado actual com a medição y_{k+1} [24].

O filtro de *Kalman* pressupõe duas fases distintas: estimação e actualização. A estimação usa o modelo do sistema e o estado anterior para produzir uma estimativa do estado actual. Esta fase corresponde à utilização dos sensores de localização relativa (*encoders*) para proceder à estimação do estado do sistema, o que conduz a um crescimento sem limites do erro de estimação, devido à sucessiva integração do ruído dos sensores. Para diminuir o erro de estimação é necessário então recorrer à segunda fase do *KF*, a actualização. A fase de actualização usa informações de medidas existentes para refinar essa estimação, a fim de alcançar uma nova estimativa mais precisa do estado, a iteração corrente. Para o caso de estudo em concreto, esta fase realiza-se sempre que um marcador artificial é detectado.

Considerando então as condições iniciais do processo como

$$\hat{x}_{0/0} = E[x_0], \tag{3.5}$$

$$P_{0/0} = P_0 = Var[x_0], \tag{3.6}$$

é possível auferir as equações da estimação e actualização em tempo discreto. Assim, a propagação de estado é obtida por

$$\hat{x}_{k+1/k} = A_k \hat{x}_{k/k} + B_k u_k \tag{3.7}$$

com, a co-variância associada obtida a partir de

$$P_{k+1/k} = A_k P_{k/k} A_k^T + Q_k. \tag{3.8}$$

Para a actualização, que pode não ocorrer em todas as iterações, é necessário calcular o ganho de *Kalman*. Este é dado por

$$K_{k+1} = P_{k+1/k} C_{k+1}^T (C_{k+1} P_{k+1/k} C_{k+1}^T + R_{k+1})^{-1}, \tag{3.9}$$

e a actualização da co-variância por

$$P_{k+1/k+1} = (I - K_{k+1}C_{k+1})P_{k+1/k}. \tag{3.10}$$

Por fim, a actualização de estado é obtida através de

$$\hat{x}_{k+1/k+1} = \hat{x}_{k+1/k} + K_{k+1}(y_{k+1} - C_{k+1}\hat{x}_{k+1/k}). \tag{3.11}$$

A figura 3.2 apresenta de forma sumária o processo de estimação em paralelo com o de actualização. As expressões descritas acima podem ser confirmadas em Welch et al [24].

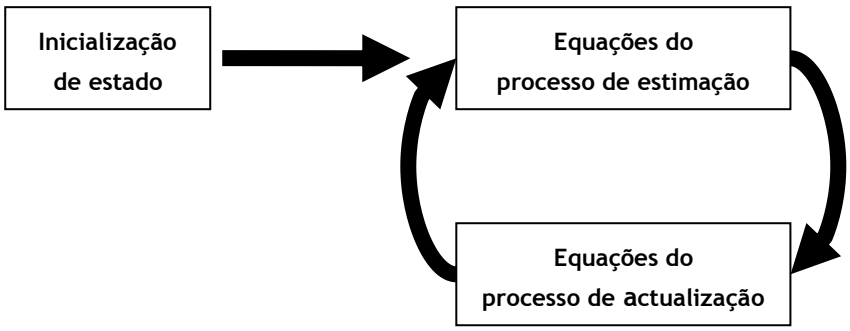


Figura 3.2 – Resumo do processo de estimação e actualização do *KF* em tempo discreto

3.2.2 Filtro de *Kalman* Estendido

Conforme descrito anteriormente, o filtro de *Kalman* aborda o problema geral de tentar estimar o estado de um sistema em tempo discreto controlado através de uma equação linear estocástica. No entanto, a maioria dos sistemas são não-lineares. Um filtro de *Kalman* que lineariza em torno da média e co-variância actual é referido como um filtro de *Kalman* estendido, *EKF*. São apresentadas de seguida as equações que descrevem o filtro de *Kalman* estendido.

A propagação de estado é dada por

$$\hat{x}_{k+1/k} = f(\hat{x}_{k/k}, k) \tag{3.12}$$

sendo, a previsão da variância determina através de

$$P_{k+1/k} = A_k P_{k/k} A_k^T + Q_k \tag{3.13}$$

com$A_k = \left.\frac{df}{dx_k}\right|_{x_k = \hat{x}_{k/k}}$, e$Q_k \approx Var\,[w_k]$.

A fase de actualização pressupõe o cálculo do ganho de *Kalman* que pode ser obtido a partir de

$$K_{k+1} = P_{k+1/k} C_{k+1}^T (C_{k+1} P_{k+1/k} C_{k+1}^T + R_{k+1})^{-1} \tag{3.14}$$

com, $C_{k+1} = \left.\frac{dh}{dx_{k+1}}\right|_{x_{k+1} = \hat{x}_{k+1/k}}$ e $R_{k+1} \approx Var\,[v_{k+1}]$.

Mediante as equações anteriores, obtém-se então a actualização da estimação. Esta é dado por

$$\hat{x}_{k+1/k+1} = \widehat{x}_{k+1/k} + K_{k+1}(y_{k+1} - h(\hat{x}_{k+1/k}, k + 1)) \tag{3.15}$$

e a actualização da variância obtida através de

$$P_{k+1/k+1} = (I - K_{k+1} C_{k+1}) P_{k+1/k}. \tag{3.16}$$

3.3 Análise de imagem

Uma imagem pode ser definida como uma matriz de pixéis em que cada pixel é função da sua coordenada (x, y) no plano da imagem. Assim, uma imagem monocromática é uma função $f(x, y)$, em que $f(x, y)$ está compreendida entre valores de preto e branco, ou seja, valores numa escala de níveis cinzentos.

Desta forma, numa imagem monocromática a função $f(x, y)$ está compreendida entre os valores

$$0\ (preto) \leq f(x, y) \leq 255\ (branco). \tag{3.17}$$

Nesta secção serão abordadas algumas técnicas que permitem retirar as características mais importantes de uma imagem, tendo em conta o fim desejado.

3.3.1 Pixéis e níveis de uma imagem

Uma imagem possuirá maior qualidade quanto maior for o número de pixéis e os níveis de intensidade que compõem essa mesma imagem.

Um conceito geralmente associado a uma imagem é a sua resolução. A resolução pode ser descrita como a resolução espacial ou resolução de intensidade. A resolução espacial é um termo que se refere ao número de pixéis utilizados na construção de uma imagem digital, expressa em *dpi* ou em *ppi,* e a distância entre cada pixel e como tal a profundidade da qualidade da imagem. Duas imagens com o mesmo nível de intensidade e diferente número de pixéis, resultam em imagens com qualidade diferente, ou seja, uma imagem com maior

resolução espacial é composta por um maior número de pixéis do que as de menor resolução espacial. Intensidade de resolução representa os níveis de intensidade numa imagem.

3.3.2 Processamento de imagem binária

Uma imagem binária caracteriza-se por ser uma imagem constituída por apenas dois níveis de intensidade, normalmente branco e preto. Na figura 3.3 mostra-se a diferença entre uma imagem de níveis cinzentos e sua imagem binária. Numa imagem binária, os pixéis do objecto possuem o valor 1 (preto) e os pixéis do fundo possuem valor 0 (branco), ou vice-versa, dependendo da convenção.

Com a mesma resolução espacial, uma imagem binária requer menos poder computacional que a sua imagem em níveis cinzentos. Isso ocorre porque o valor de um pixel binário pode ser representado por uma variável lógica, enquanto a representação de um valor de na escala de cinzentos necessita da representação da imagem como uma função $f(x, y)$.

Em geral, os sistemas de visão binária são úteis nos casos em que a forma contém informação suficiente para permitir o reconhecimento de um objecto e onde o ambiente pode ser adequadamente controlado. Desta forma, os objectos devem ser facilmente separados do fundo para garantir que se obtém a correcta forma do objecto. A qualidade da informação obtida por binarização pode ser melhorada se a imagem possuir poucos objectos e utilizando técnicas de iluminação apropriadas. Estas serão descritas na secção 3.3.3.

3.3.2.1 Aquisição de imagem binária

Quando se utiliza a visão artificial para adquirir informação de uma imagem em tempo real, não se pretende obter a imagem original mas sim uma determinada característica da imagem.

O *threshold* é uma operação pontual sobre a imagem original. Esta operação tem como finalidade separar na imagem os objectos do fundo. Assim, através desta operação converte-se uma imagem de níveis cinzentos numa imagem binária. Desta forma, define-se exactamente os pixéis brancos, cujo valor está acima de um determinado limiar e fixam-se os restantes pixéis com a cor preta. Nas imagens monocromáticas, o fundo possuí níveis de intensidade mais elevados comparativamente com os objectos.

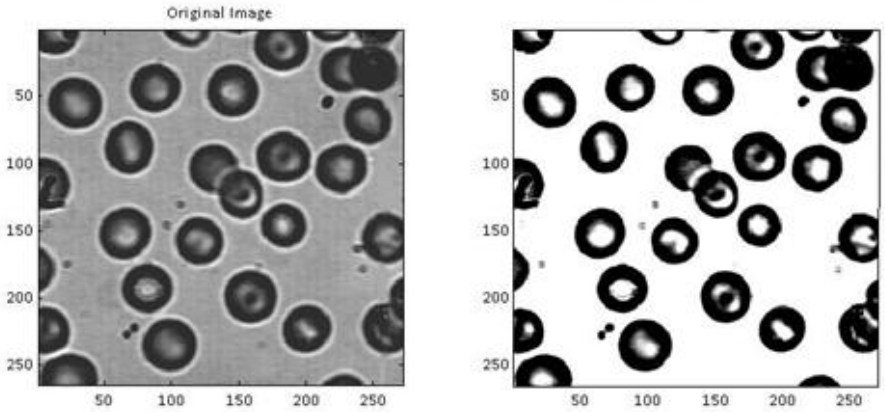


Figura 3.3 – Imagem em escala de cinzentos e imagem binária

A imagem binária $F_T = [f_T(i,j)]_{m\times n}$ obtida a partir da imagem original $F = [f(i,j)]_{m\times n}$ pode ser descrita por

$$f_T(i,j) = \begin{cases} 1, & f(i,j) \leq T \\ 0, & caso\ contrário\ . \end{cases} \tag{3.18}$$

No caso dos valores dos níveis de intensidade dos objectos serem conhecidos e pertencerem ao intervalo $[T_1, T_2]$, então a imagem binária correspondente pode ser obtida de acordo com

$$f_T(i,j) = \begin{cases} 1, & T_1 \leq f(i,j) \leq T_2 \\ 0, & caso\ contrário\ . \end{cases} \tag{3.19}$$

3.3.3 Técnicas de iluminação

Em sistemas sensíveis à luminosidade, a iluminação de uma imagem captada por um sensor passivo (por exemplo, uma câmara) é um factor importante e que, muitas vezes, afecta a complexidade do processamento da imagem. A utilização de iluminação natural, ou arbitrária, pode resultar em baixos contrastes de imagem, sombras, reflexos e detalhes irrelevantes. Deste modo, por vezes, para a obtenção de resultados óptimos é necessária uma iluminação cuidada. Um sistema de iluminação bem projectado permite minimizar a complexidade de uma imagem com a finalidade de obter uma melhor informação para a detecção de um determinado objecto. Existem diversas técnicas de iluminação de imagem sendo que os mais usuais são [26]:

- Difusa – aplicada quando as características da superfície dos objectos são importantes;
- Direccional – realça o contraste dos contornos e dos baixos-relevos de um objecto;
- Iluminação estruturada – a estrutura da iluminação fornece informação tridimensional;
- Iluminação na retaguarda do objecto (*backlight*) – permite adquirir a forma de um objecto.

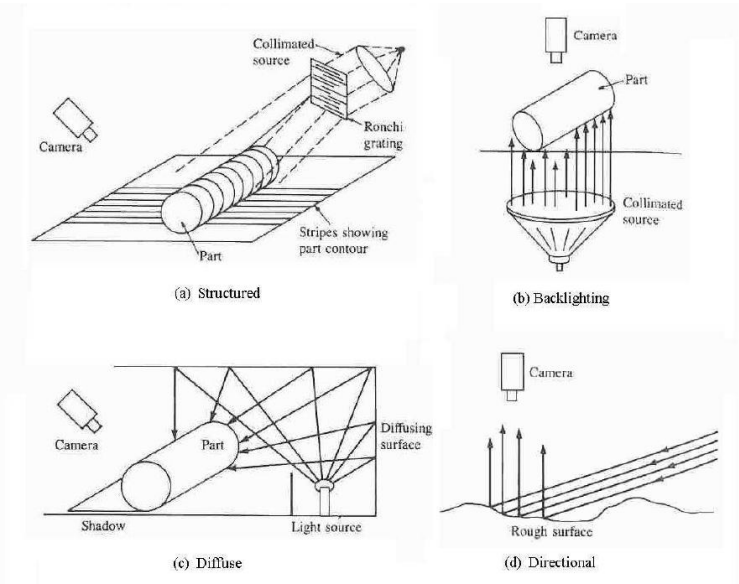


Figura 3.4 – Esquemas básicos de iluminação

Capítulo 4

Mini-AGV – Concepção Estrutural

Neste capítulo apresenta-se de uma forma geral, a infra-estrutura experimental utilizada para o desenvolvimento deste trabalho, bem como algum do material herdado de projectos anteriores. Inicialmente serão estabelecidos alguns traços gerais sobre a concepção deste sistema no que diz respeito ao tipo de locomoção e versatilidade do robô e também relativamente à arquitectura de processamento. Na secção 4.2 será apresentada a arquitectura funcional do *hardware* que compõe o robô e nas subsecções seguintes será descrito em detalhe cada elemento da plataforma de *hardware* que compõe o mesmo. A descrição em pormenor da estruturação do *software* desenvolvido para a realização deste estudo, bem como todos os protocolos de comunicações empregues e os controladores de velocidade dos motores, serão apresentados na secção 4.3.

4.1 Descrição Geral

Na figura 4.1 pode-se visualizar o robô utilizado no estudo deste trabalho. A principal característica da plataforma robótica Mini-*AGV* é o facto de esta possuir um sistema de tracção diferencial, sendo este o mais popular e usado em veículos guiados autonomamente. O sistema desenvolvido consiste na disposição diametral de duas rodas acopladas a dois motores de corrente contínua, dispostos no mesmo eixo central e simetricamente opostos. Para proporcionar uma estabilidade e apoio à estrutura são também colocadas duas rodas "livres", como se pode verificar na figura 4.2. Esta configuração traduz-se num simples mecanismo de locomoção, permitindo à plataforma uma enorme liberdade de movimento, como rodar sobre o seu centro. No capítulo 6 será descrito em detalhe a cinemática referente ao modelo de odometria para um robô com accionamento diferencial.

Uma característica não menos importante do robô concebido é a versatilidade. O robô é constituído por uma plataforma deslizante movida vertical e horizontalmente em relação à base do sistema, capaz de recolher e armazenar volumes em várias posições previamente estabelecidas pelo utilizador.

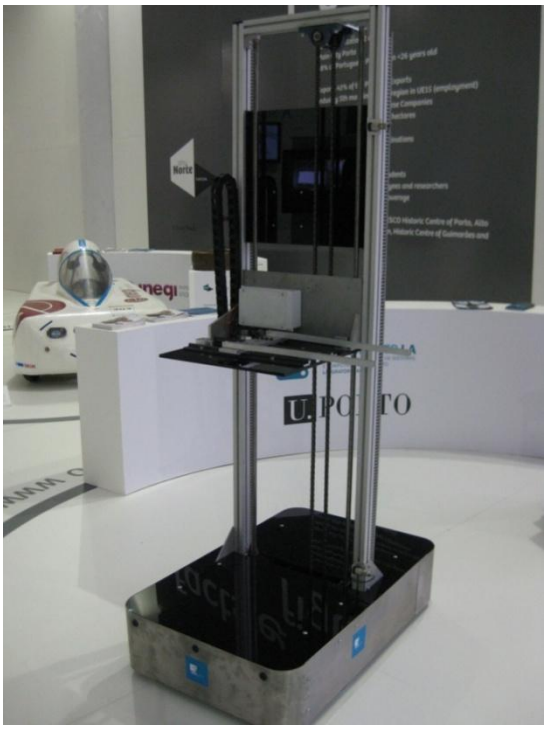


Figura 4.1 – Robô desenvolvido para a realização do trabalho em estudo

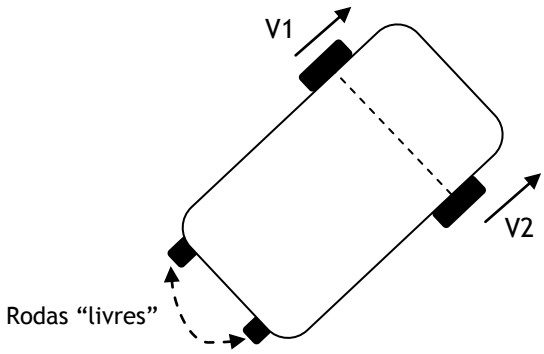


Figura 4.2 – Sistema de tracção diferencial com duas rodas "livres"

4.2 Plataforma de *Hardware*

A arquitectura funcional da plataforma de *hardware* do robô em estudo é apresentada na figura 4.3. O *AGV* apresenta uma arquitectura complexa, integrando diversos dispositivos externos com diferentes necessidades de comunicação. É de salientar que o processamento de dados é dividido em dois níveis distintos. O nível inferior, designado por camada de processamento inferior, é responsável por processar a informação dos sensores actuando nos motores. O nível superior, designado por camada de processamento superior, encarrega-se de efectuar o controlo e a decisão do movimento do robô, tendo como uma das principais funções a determinação da localização do mesmo. Estas duas camadas serão apresentadas em pormenor na secção 4.3.

Seguidamente, apresenta-se em detalhe os elementos constituintes do *AGV* e que figuram na arquitectura funcional do *hardware*.

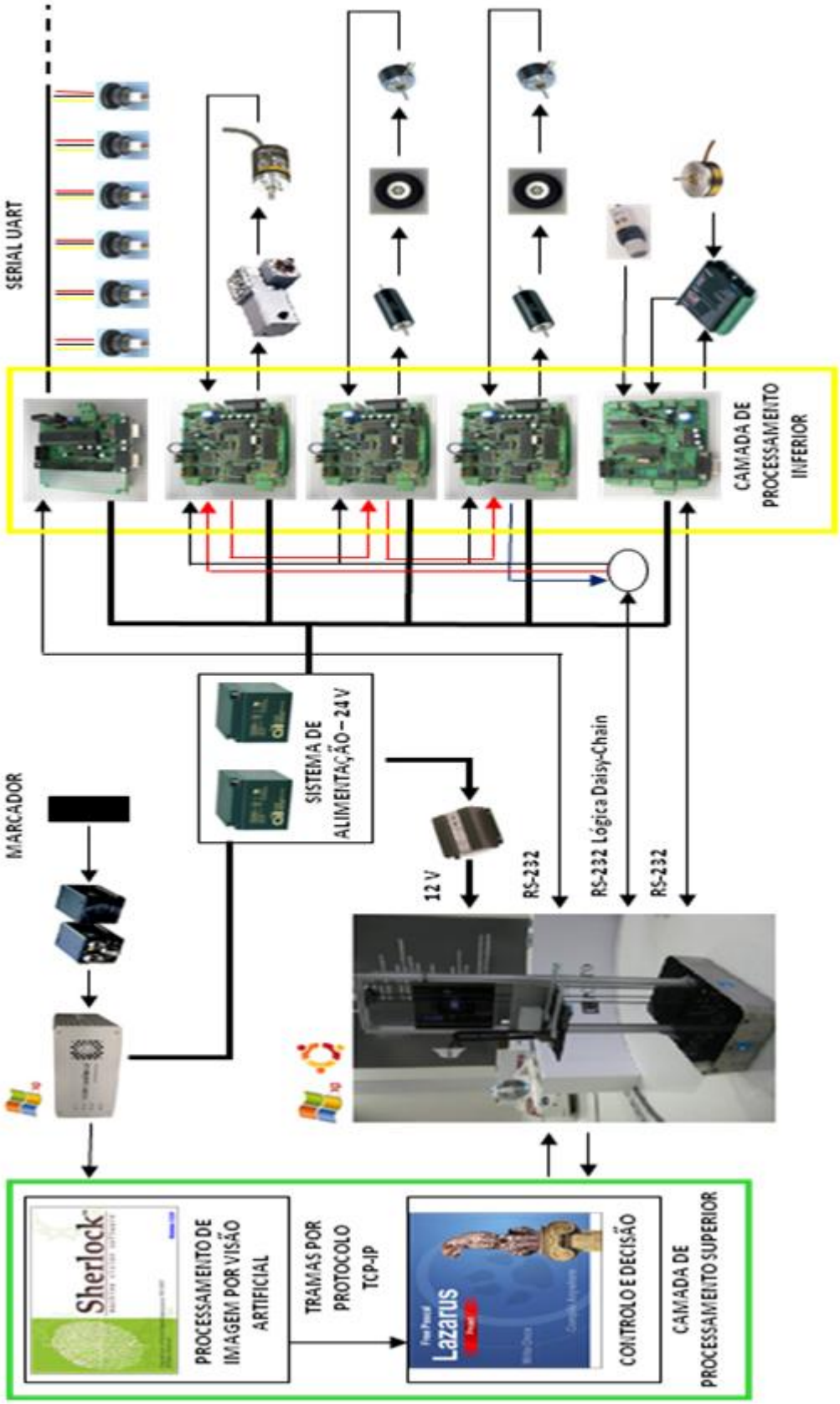


Figura 4.3 – Arquitectura funcional da plataforma de *hardware*

4.2.1 Motores

O robô desenvolvido para o estudo deste trabalho contempla quatro motores. Dois para o movimento do mesmo e dois destinados ao movimento da plataforma deslizante.

Para a locomoção do *AGV* recorreu-se aos motores *DC* apresentados na figura 4.5. Estes são dotados de 150 *Watts* de potência e alimentados por uma tensão nominal de 24 *Volts*. Realça-se o facto de estes motores de tracção possuírem uma caixa redutora incorporada sendo a relação da caixa de 43:1. Desta forma transmite-se um maior binário em prol de uma redução na velocidade de rotação. Estes motores apresentam uma excelente razão entre volume/performance, altíssima eficiência, baixa indutância, alta fiabilidade e alta aceleração graças à baixa inércia, [27].

Para o deslocamento da plataforma responsável pela recolha/armazenamento de materiais utilizaram-se dois motores para a translação vertical e para a horizontal. Assim, para o deslocamento vertical utilizou-se o motor da Mini Motor com uma potência nominal de 37 *Watts* e para o deslocamento horizontal o motor eléctrico síncrono *Brushless (*motor *DC* sem escovas) com uma potência de 30 *Watts*, comutado electronicamente. A figura 4.4 ilustra os motores utilizados para o movimento da plataforma deslizante.

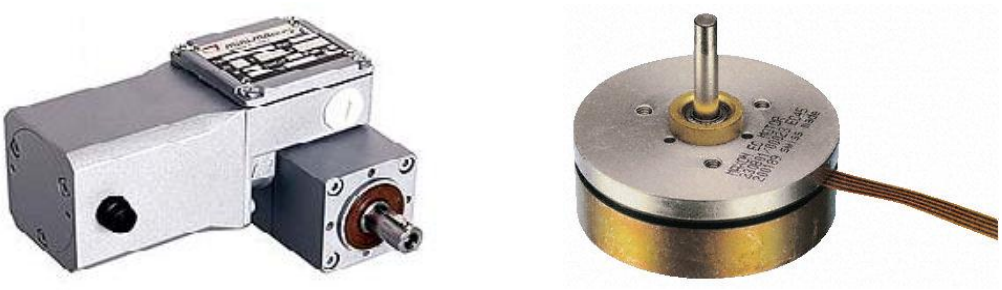


Figura 4.4 – Motores para o movimento da plataforma deslizante [28]

4.2.2 *Encoders*

Com o objectivo de quantificar a distância percorrida por cada roda, o sentido e a velocidade de rotação das mesmas, acoplou-se codificadores ópticos – *encoders*, directamente ao veio dos motores. Na figura 4.5 encontra-se representado o modelo do *encoder* utilizado para a locomoção da plataforma robótica integrado no motor e na caixa redutora. Estes *encoders* possuem uma resolução de 256 impulsos por revolução (360º) e três sinais de saída (A, B e I). Os canais A e B permitem determinar o sentido de rotação do veio e encontram-se desfasados de 90º. O canal I é um ponto de referência para a determinação precisa do ângulo de rotação e gera um impulso sempre que o eixo do codificador óptico realiza uma volta completa [29]. Uma vez que, são contabilizadas todas as transições de das fases obtém-se não os 256 impulsos por rotação mas antes 1024.

A posição de carga ou descarga de um determinado material é também determinada através da leitura de um codificador óptico que, posteriormente, converterá os impulsos eléctricos gerados pelo *encoder* em distância percorrida pela plataforma, permitindo desta forma a recolha ou armazenamento do material. Esta posição será previamente estabelecida

pelo utilizador, uma vez que o robô não possui qualquer outro dispositivo óptico capaz de lhe fornecer a informação da posição precisa do objecto. Para esta finalidade utilizou-se o *encoder* representado na figura 4.6 que por cada rotação completa gera 100 impulsos eléctricos.

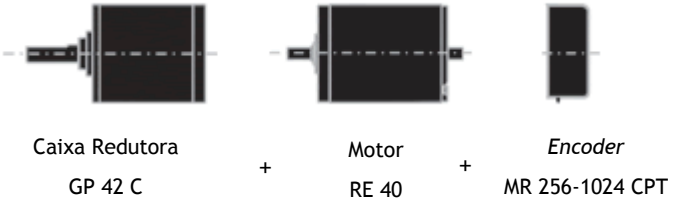


Figura 4.5 – Esquema de motor, caixa redutora e *encoder* utilizado para a locomoção

Figura 4.6 – *Encoder* Omron E6A2-CW5C [28]

4.2.3 Sistema de alimentação

A alimentação da plataforma robótica foi estabelecida tendo em consideração a tensão requerida pelos motores. Assim, o sistema de alimentação do *AGV* é composto por duas baterias de chumbo cíclicas cada uma contendo uma tensão nominal de 12 *Volts*, ligadas em série, como se pode observar na figura 4.7. As baterias utilizadas são próprias para a aplicação em sistemas industriais, permitindo ciclos contínuos de carga e descarga. Estas baterias assumem a responsabilidade de alimentar todos os dispositivos que incorporam o robô, bem como o microcomputador e o ecrã táctil.

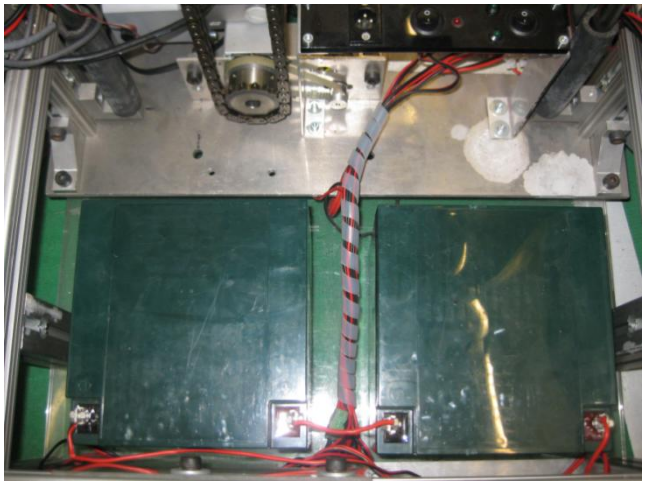


Figura 4.7 – Sistema de alimentação

4.2.4 Microcomputador e *LCD* Táctil

Para facilitar a configuração do Mini-*AGV*, instalou-se um microcomputador na parte superior do robô e um ecrã táctil de 10,2 polegadas apresentados na figura 4.8 e 4.9. Este microcomputador encontra-se equipado com uma fonte de alimentação de 12 *Volts* sem partes móveis (ventoinhas), de um disco de 160 *Gb* e de uma placa que garante o acesso a redes *wireless*. A ausência de partes móveis confere-lhe uma maior robustez e durabilidade.

A utilização de um ecrã táctil promove a interacção entre Homem-Máquina uma vez que, a acessibilidade ao sistema de controlo se torna mais prática e simples, evitando a utilização de outro tipo de periféricos de entrada de dados, como o teclado.

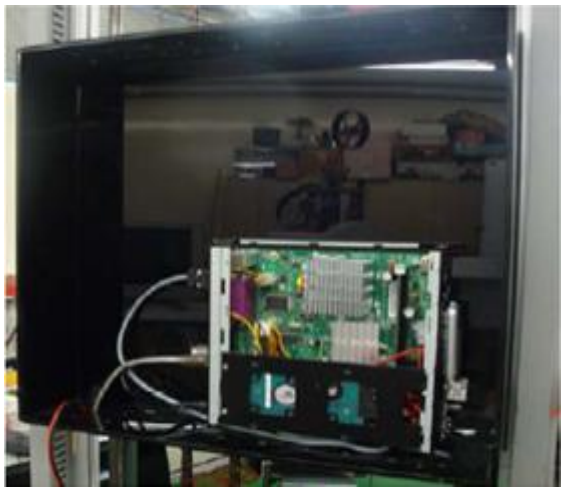


Figura 4.8 – Microcomputador instalado

Figura 4.9 – Ecrã Táctil

4.2.5 Sensores de ultra-sons

Os sensores de ultra-sons têm como finalidade a detecção de obstáculos, evitando assim a colisão com os mesmos. Deste modo, utilizaram-se três sensores de ultra-sons localizados na parte frontal do pára-choques e outros três na parte traseira. A escolha recaiu sobre o sonar representado na figura 4.10, pois apresenta uma capacidade de detecção de obstáculos numa gama de medições entre 0 centímetros e 6 metros, dependendo do modo de operação estipulado. A comunicação com os sensores pode ser feita através de uma interface *serial UART* e suporta até 16 sonares no mesmo barramento. Para tal, apenas é necessário garantir que todos possuam endereços diferentes.

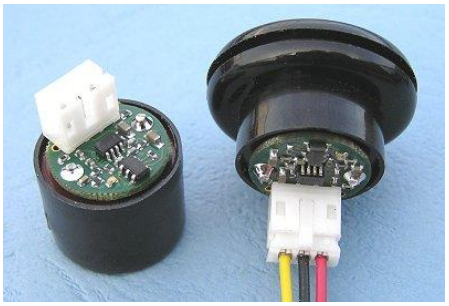


Figura 4.10 – Sonar SRF01

4.2.6 Sistema autónomo de visão artificial

Com vista a detectar os marcadores artificiais distribuídos pelo percurso a percorrer pelo robô, utilizou-se um sistema compacto de visão artificial. Este sistema assemelha-se a um *CPU* mas com características bastante específicas para visão artificial com capacidade de conjugar até quatro câmaras. Este sistema integrado permite obter medidas com excelente precisão, pois é provido da tecnologia *subpíxel*. A escolha recaiu sobre o modelo DALSA IPD-VA40 da *Infaimon* representado na figura 4.11, que fornece duas interfaces distintas designadas por iNspect e Sherlock. A aplicação iNspect proporciona versatilidade e facilidade na utilização, permitindo a instalação e implementação de uma solução de visão num curto período de tempo. Para tarefas mais complexas, o *software* Sherlock oferece ainda maior flexibilidade e controlo, fornecendo as ferramentas e os recursos capazes de se aplicar à maioria das situações.

Figura 4.11 – Sistema de aplicação para visão artificial [30]

Com vista a identificar marcadores e determinar a sua localização, obtendo assim informação do ambiente no qual o robô se insere, utilizou-se uma câmara fornecida em conjunto com o sistema integrado de visão artificial. Esta câmara reunida com outros sensores (*encoders* e sonares) permite obter o posicionamento do *AGV*. Na figura 4.12 encontra-se ilustrada a câmara jAi CV-A11 utilizada. Esta possui as seguintes características:

- Câmara monocromática progressiva compacta com um sensor de 1/3 polegadas;
- Robusta e de custo reduzido concebida para aplicações industriais;
- Resolução: 659x494;
- Extrema sensibilidade à luz, possibilitando o trabalho em ambientes com baixa iluminação.

Figura 4.12 – Câmara jAi CV-A11 [30]

4.2.7 Sensores de fim-de-curso

A utilização de sensores de fim-de-curso tem como finalidade a detecção de objectos. Neste caso concreto são utilizados os sensores fotoeléctricos representados na figura 4.13, para auxiliar no controlo da plataforma deslizante, tanto no deslocamento vertical como no deslocamento horizontal. Assim, foram colocados dois sensores na posição superior e inferior servindo de *reset* ao posicionamento da plataforma. Logo, quando os sensores ficam activos, ou seja, quando há uma reflexão do feixe luminoso emitido, provocam a situação de paragem no movimento vertical. Relativamente ao movimento horizontal foram aplicados dois sensores utilizando o mesmo conceito descrito para o movimento vertical, embora referente ao movimento de recolha e armazenamento de um objecto.

Figura 4.13 – Sensor fotoeléctrico Omron E3F2 [28]

4.2.8 Controladores dos motores

Para o controlo dos motores do *robô* seguiram-se duas abordagens distintas, considerando o tipo de motor a controlar. Desta forma, com o objectivo de controlar os motores encarregues de movimentar o *AGV* e de deslocar verticalmente a plataforma deslizante, utilizou-se um módulo constituído por uma placa de circuito impresso desenvolvido pela equipa 5DPO da FEUP/INESCPorto, vulgarmente designado de *driver*. Este tem-se demonstrado muito útil revelando grande fiabilidade na aplicação da robótica móvel, daí a sua larga utilização em projectos desta equipa, nomeadamente no futebol robótico (*Small and Middle Size League*).

Estes módulos, ilustrados na figura 4.14, têm como finalidade, para além de controlar os motores através de transístores que formam uma ponte em H, a leitura dos impulsos gerados pelos *encoders*, a gestão das comunicações entre a aplicação da camada superior e a da camada inferior através do protocolo *RS-232* numa interface *Daisy Chain* especial (será abordado mais detalhadamente na secção 4.3.1), bem como a interface com dispositivos através das entradas e saídas digitais e analógicas. Os sensores fotoeléctricos utilizados na plataforma deslizante são um exemplo da utilização destas mesmas entradas e saídas, realizando o pré-processamento dos dados adquiridos.

Tendo como objectivo a realização do controlo do motor encarregue pelo deslocamento horizontal da plataforma deslizante, optou-se por utilizar um controlador específico, representado na figura 4.15, uma vez que se trata de um motor *brushless DC* comutado electronicamente. Foi também utilizado um módulo que incorpora um microprocessador atmega8 e uma interface de comunicação *RS-232,* apresentado na figura 4.16, que serve de "ponte" entre as instruções enviadas pela camada superior e o controlo do motor.

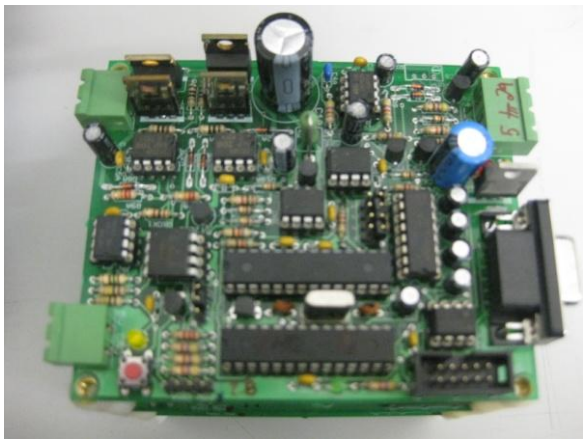


Figura 4.14 – *Driver* dos motores de locomoção e deslocamento vertical da plataforma deslizante

Figura 4.15 – Controlador de velocidade do motor Maxon EC-45

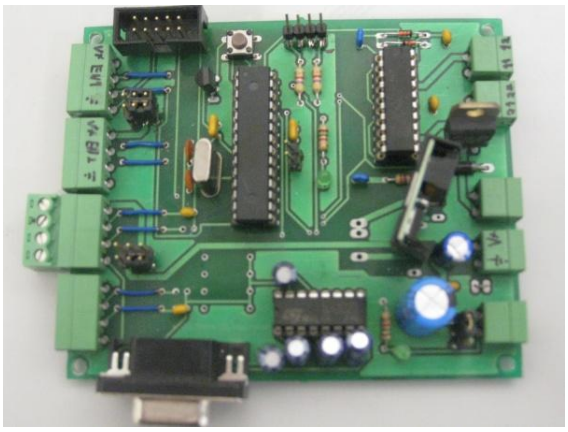


Figura 4.16 – Módulo generic atmega8

4.2.9 Conversores de tensão *DC-DC*

Atendendo à existência de diferentes níveis de tensão nos diversos sistemas que integram esta plataforma robótica, foi necessário proceder à transformação da tensão (*DC*) fornecida pelo sistema de alimentação do *AGV* noutra tensão (*DC*), para permitir o correcto funcionamento dos mesmos. Desta forma, especificam-se de seguida os diversos níveis de tensão existentes no robô:

- 24 V – 12 V: Microcomputador e ecrã táctil;
- 24 V – 5 V: Circuitos de comando dos *drivers;*
- 24 V – 3,6 V: Sistema de iluminação da câmara de visão.

4.3 Plataforma de *Software*

A organização da arquitectura de *software* de um sistema afecta totalmente o desempenho de um robô móvel. A integração de múltiplos sistemas de sensores, sem uma aquisição e pré-processamento de dados adequada e eficiente, reflecte-se numa enorme ineficácia de desempenho por parte do robô. Por outro lado, múltiplas formas de desenvolvimento e tarefas de controlo podem ser mais eficientes e manipuláveis por uma arquitectura organizada que promove a programação orientada a eventos, processos de interacção e de controlo [12].

Nesta secção apresentam-se os vários componentes que constituem a plataforma de *software*, assim como as suas principais características.

Através da figura 3.3 pode-se visualizar os diferentes tipos de comunicação utilizados para a integração total dos sistemas que constituem o *AGV*. Estes são também abordados nesta secção e descritos de uma forma detalhada. Os protocolos de comunicação utilizados permitem a simulação, navegação, localização e o controlo do mesmo.

O controlo do *software* em tempo real será organizado por camadas de processamento de diferentes níveis, de acordo com as prioridades exigidas pelo sistema. A arquitectura desenvolvida para o sistema será abordada nesta secção, descrevendo de uma forma organizada as várias camadas (subsistemas) que compõem esta plataforma robótica.

Será ainda objecto de estudo nesta secção, o desenvolvimento e a calibração dos controladores de velocidade dos motores responsáveis pela locomoção do *AGV*.

O *software* de decisão e controlo do robô foi realizado numa aplicação desenvolvida na camada de processamento superior, usando a plataforma de desenvolvimento *Lazarus* que incorpora a linguagem *Free Pascal* sobre o sistema operativo *Windows*. Salienta-se também que a utilização desta plataforma permite transpor facilmente o *software* de decisão e controlo do sistema operativo *Windows* para o sistema operativo *Linux*, necessitando apenas de ser novamente recompilado. Pelo facto da linguagem *Free Pascal* ser orientada a objectos, permite o fácil acesso aos dispositivos e sistemas periféricos, assim como a fácil construção de interfaces para o utilizador.

4.3.1 Comunicação assente no protocolo série *RS-232*

No trabalho em estudo seguiram-se duas abordagens distintas relativamente à utilização do protocolo de comunicação série *RS-232*.

Para a estratégia de comunicação entre a camada de processamento superior - aplicação de controlo e decisão - e os *drivers* dos motores aplicou-se uma lógica *serial* Daisy Chain que permite interligar mais do que um dispositivo através da mesma conexão física. Esta conexão utiliza um protocolo próprio, protocolo Daisy Bin, que adaptada a ligação série *RS-232* a uma tipologia em anel. Esta lógica é fortemente utilizada pela equipa 5DPO da FEUP/INESCPorto no desenvolvimento de projectos na área da robótica móvel, nomeadamente, nas ligas de futebol robótico (*Small and Middle Size League*).

A outra abordagem recorre a uma ligação série *RS-232*, do tipo *unicast*, para a comunicação entre o módulo de controlo dos sensores de ultra-sons ou o módulo de controlo

do motor responsável pelo movimento horizontal da placa deslizante e a aplicação de decisão e controlo.

Na figura 4.17 exibe-se o esquema de ligações aplicado para a utilização da lógica Daisy Chain, baseada no protocolo de comunicação série *RS-232*.

O protocolo Daisy Bin utiliza uma arquitectura do tipo *Master-Slave*, sendo o *Master* o responsável pela inicialização da comunicação e o único capaz de gerar pedidos. Este protocolo apenas permite a utilização de um dispositivo para desempenhar as funções de *Master* podendo-se agregar n *Slaves* à rede *serial* Daisy Chain. Para o caso em estudo, utilizam-se 3 dispositivos (*Slaves*) como se pode visualizar através da figura 4.17.

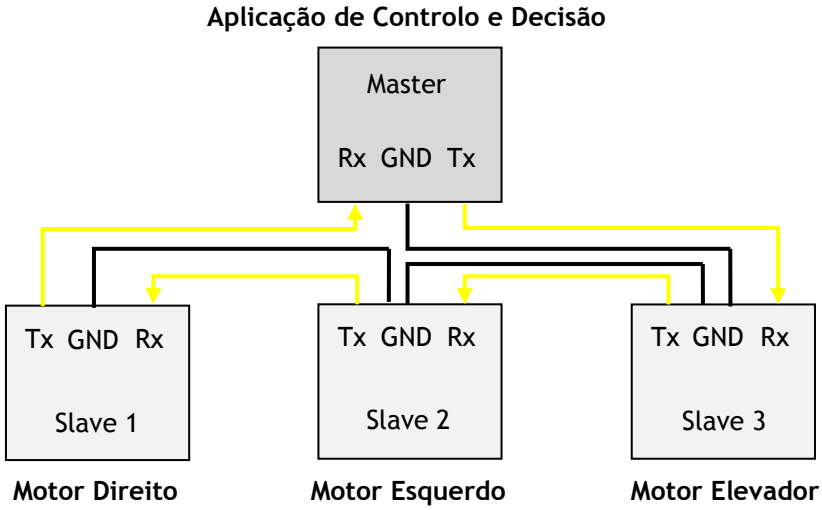


Figura 4.17 – Esquema de ligações *serial* Daisy Chain

A comunicação é realizada apenas num sentido - comunicação unidireccional - pelo que cada dispositivo da rede *serial* Daisy Chain terá que responder e reencaminhar a mensagem (trama) para permitir que esta circule e que retorne ao *Master*. A estrutura da mensagem possui o formato representado na tabela 4.1 que contém um cabeçalho, com as definições necessárias ao transporte pelo anel, e uma trama de dados. Esta trama inclui os dados do pedido do *Master*, *DataIn*, e os dados fornecidos pelo *Slave*, *DataOut*.

Tabela 4.1 – Formato da trama do protocolo Daisy Bin

	7	6	5	4	3	2	1	0	
Byte 1	'D'								Cabeçalho
Byte 2	'C'								
Byte 3	I45		O45		B	n			
Byte 4	I				O				
Byte 5	ma								
Byte 6...n	DataIn								Dados
Byte n...m	DataOut								

A trama é inicializada com os caracteres 'D' e 'C' nos bytes 1 e 2, respectivamente. No byte 3 é definido o comprimento dos dados de entrada e saída, *I45* e *O45*, bem como o modo de comunicação que se pretende para o *Broadcast*, *B*, e o número do nó ao qual se destina a mensagem. Quanto ao *Broadcast* pode ser estabelecido o valor de 1, modo *Broadcast*, caso se pretenda que os dados de entrada sejam iguais para todos os dispositivos a quem a trama se destina; ou o valor de 0, modo *Compact,* onde os dados de entrada e saída são especificados para cada nó de destino da trama. O número do nó, *n*, é inicializado pelo *Master* com o valor 0 e incrementado sempre que passa por um dispositivo.

No byte 4 define-se o número de bytes de entrada e saída de dados e, no byte 5 definem-se os dispositivos a quem se destinam as mensagens de dados. Para indicar os nós a quem se destina a mensagem coloca-se o respectivo bit a 1 como, 00000111 mensagens para os dispositivos 0, 1 e 2.

O conjunto dos 5 bytes anteriores forma o cabeçalho da mensagem sendo que os restantes bytes constituem a informação dos dados de entrada e saída. Cada byte de entrada ou saída de dados permite informação até a um máximo de 63 bytes.

Na tabela 4.2 apresenta-se, de forma resumida, o significado de cada campo que constitui a mensagem.

Tabela 4.2 – Resumo dos campos que constituem a trama do protocolo Daisy Bin

Campo	Descrição
'D' e 'C'	Caracteres para inicialização da trama
'I45' e 'O45'	Bits reservados para definir o comprimento dos dados de entrada e saída, respectivamente
B	*Flag* indicativa do modo de *Broadcast*
n	Número do dispositivo ao qual se destina a mensagem
I	Número de bytes de dados de entrada
O	Número de bytes de dados de saída
ma	Máscara do nó a quem se destina a mensagem
DataIn	Dados de entrada, destinados ao *Slave* e fornecidos pelo *Master*
DataOut	Dados de saída fornecidos pelo dispositivo ao *Master*

4.3.2 Comunicação assente no protocolo *UDP*

Para a troca de pacotes de mensagens entre a aplicação de controlo e decisão e a plataforma de simulação *SimTwo,* utiliza-se a comunicação baseada no protocolo *UDP*.

Este protocolo não é orientado à conexão e, como tal não garante que os pacotes cheguem ao destino. A entrega dos pacotes de dados poderá ser realizada de forma desordenada e a integridade dos dados pode ser conferida pelo sistema de detecção de erros, *Checksum* [31]. A ausência de estruturas de controlo complexas no protocolo *UDP* garante uma elevada eficiência, já que cada pacote é composto praticamente apenas por dados [32]. O *software* desenvolvido para o processamento dos pacotes *UDP* encontra-se estruturado de modo a reduzir os impactos na falha de um pacote *UDP*.

Na tabela 4.3 apresenta-se o cabeçalho das mensagens do protocolo *UDP*.

A utilização deste protocolo para a comunicação entre as duas aplicações referidas anteriormente, permite a entrega de mensagens entre aplicações em sistemas *host*[4], sendo apenas necessário estabelecer os portos[5] de origem e destino para permitir a comunicação entre elas. No capítulo 5, secção 5.4, será descrito em pormenor o formato dos pacotes das mensagens trocadas entre as aplicações, bem como estas reagem a mensagens danificadas.

Tabela 4.3 – Cabeçalho das mensagens do protocolo *UDP*

Porto de origem	Porto de destino
Comprimento da mensagem	Detecção de erros

4.3.3 Comunicação assente no protocolo *TCP*

O envio da trama (mensagem) com a informação do marcador detectado por parte do sistema de visão artificial é realizado sob a comunicação baseada no protocolo *TCP*.

O protocolo *TCP* foi desenvolvido para garantir uma comunicação fiável entre pares de processos através de redes fiáveis e não fiáveis. Ao contrário do protocolo descrito na secção anterior, este protocolo é orientado à conexão. A aplicação envia um pedido de conexão para o destino e usa essa conexão para transferir os dados [33]. No caso em estudo, o *software Sherlock* efectua o pedido de conexão à aplicação de controlo e decisão usando esta conexão para transferir a trama. Esta contém a informação da orientação e distância do robô ao marcador.

O recurso a este protocolo deve-se ao facto de o sistema de visão artificial apenas permitir a comunicação através de *TCP*.

Este protocolo assegura a fiabilidade e robustez das mensagens enviadas garantido a entrega ordenada dos dados [31]. Considerando que a mensagem enviada por *TCP* contém informação relevante para a determinação da localização do robô, a utilização deste protocolo revela-se de uma enorme importância.

Para permitir o estabelecimento da comunicação ponto a ponto sem a intervenção de um modem ou router, torna-se apenas necessário definir o *IP*[6] e Máscara de sub-rede[7], tabela 4.4.

Tabela 4.4 – Definição de *IPs* e Máscara de sub-rede para a comunicação baseada no protocolo *TCP*

	DALSA IPD-VA40	Minicomputador
Endereço *IP*	192.168.253.100	192.168.253.110
Máscara de sub-rede	255.255.255.0	255.255.255.0

No capítulo 6 será descrito em pormenor o formato do pacote da mensagem enviada do sistema de visão artificial para a aplicação de controlo e decisão.

---

[4] Caracterizam-se por sistemas para comunicações locais definidas pelo *IP* 127.0.0.1.

[5] Porto ou porta, é uma estrutura de *software* representada por um número de 16 bits que se encontra ligado a uma aplicação.

[6] Endereço que indica o local de um determinado equipamento numa rede privada ou pública.

[7] Caracteriza-se por ser um número de 32 bits usada para separar num *IP* a parte correspondente à rede pública, à sub-rede e aos *hosts*.

Mais detalhes sobre o funcionamento do protocolo *TCP* podem ser encontrados em [31].

4.3.4 Descrição da camada superior

A camada de processamento superior é responsável pelo controlo e decisão do comportamento do robô. As acções são tomadas tendo em consideração o estado do robô e o objectivo a alcançar. Deste modo, a estrutura de controlo é representada como uma sequência de operações muito semelhante à execução de uma máquina de estados, figura 4.18.

Como se refere anteriormente, a plataforma de desenvolvimento *Lazarus* utilizada para o desenvolvimento do processamento da camada superior é um ambiente de desenvolvimento integrado desenvolvido para o compilador *Free Pascal* sendo este um compilador de *Object Pascal*. Desta forma, o *software* de controlo e decisão foi desenvolvido baseado na programação orientada a eventos. Estes eventos são definidos como internos ou externos, despoletados por acções externas, como recepção de pacotes através do protocolo *UDP* ou *TCP*, ou por acções internas, evento externo e interno, respectivamente.

O evento interno, *timer*, controla o tempo entre ciclos de controlo, sendo invocado de 40 em 40 ms. As decisões e processamentos mais complexos são realizados neste ciclo. Os principais blocos de acções presentes no ciclo de controlo encontram-se apresentados na figura 4.18.

Cada evento externo é processado de forma assíncrona e independente. Desta forma, e uma vez que a plataforma de desenvolvimento *Lazarus* não se rege por níveis de prioridade, este apenas é processado quando a acção que se encontra a ser realizada terminar.

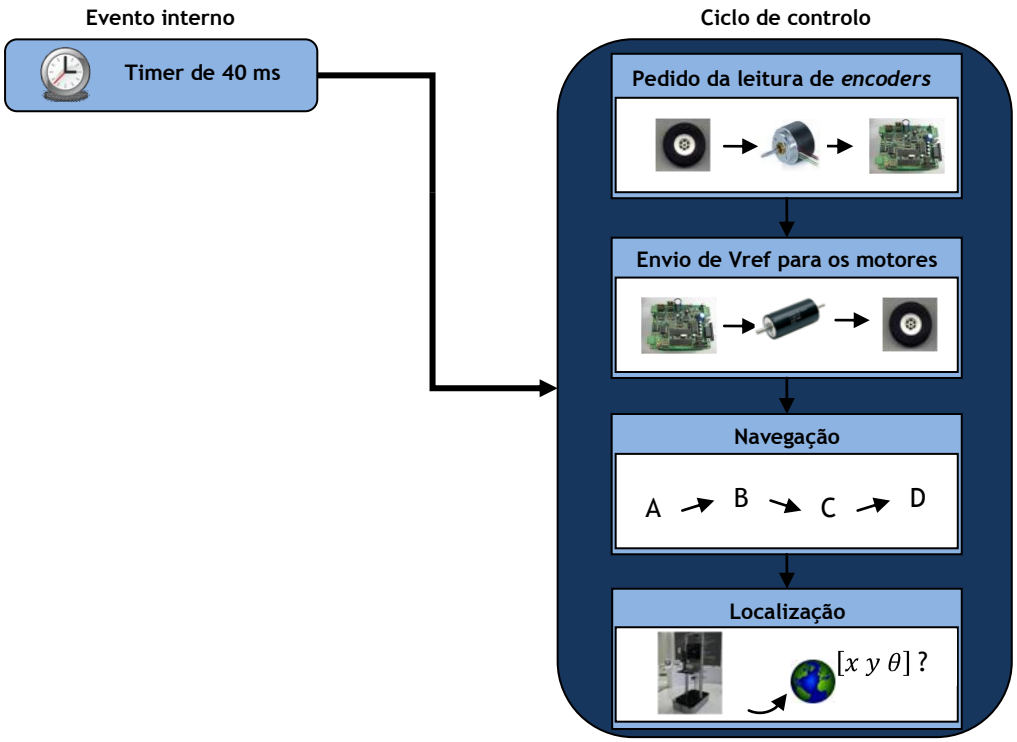


Figura 4.18 – Esquema sequencial das acções a tomar no ciclo de controlo

A máquina de estado que rege o controlo do robô é apresentada na figura 4.19. Esta é representada por três estados: inicialização, evento interno e ciclo de controlo. No estado *inicialização* é estabelecida a comunicação entre a camada de processamento superior e a camada de processamento inferior e são também inicializadas algumas variáveis internas necessárias para o correcto funcionamento do sistema. É neste estado que o robô irá determinar pela primeira vez sua posição absoluta. O estado inicialização apenas é executado quando se inicia a aplicação de controlo e decisão do robô. O estado *evento interno* apenas é responsável pela contagem do tempo de ciclo de controlo. Desta forma, sempre que o *timer* atinge o valor estipulado (40 ms) são despoletadas uma série de acções no ciclo de controlo. O estado *ciclo de controlo* corresponde ao controlo e decisão do comportamento do robô. Neste estado executadas as acções mais importantes que permitem a navegação e localização do robô no referencial mundo. É neste estado que é possível a escolha da tarefa a realizar e o seu modo de funcionamento.

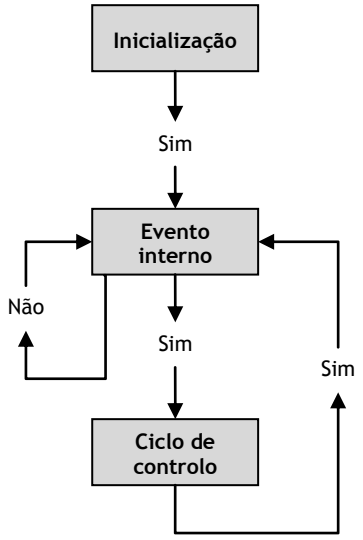


Figura 4.19 – Máquina de estados do robô

Na figura 4.20 apresenta-se um esquema da acção a tomar na recepção de dados a partir do sistema de visão artificial ou da plataforma de simulação. Estes eventos são realizados após a recepção completa dos dados e no caso de não estar a ser executada nenhuma tarefa. Deste modo, quando estes eventos são desencadeados, a aplicação de controlo e decisão executa uma função que permite a descodificação dos dados recebidos. Os dados recebidos a partir do sistema de visão artificial serão utilizados, posteriormente, no algoritmo de localização do robô. Já os dados proveniente do SimTwo são utilizados tanto na localização como na navegação do robô. Deste modo, na navegação são utilizados para proceder à actualização da posição do robô com base nos impulsos gerados por um ciclo de controlo e na localização são utilizados para determinar a distância e a orientação do robô em relação ao marcador (parede), possibilitando a correcção de posição e, consequentemente, a sua localização.

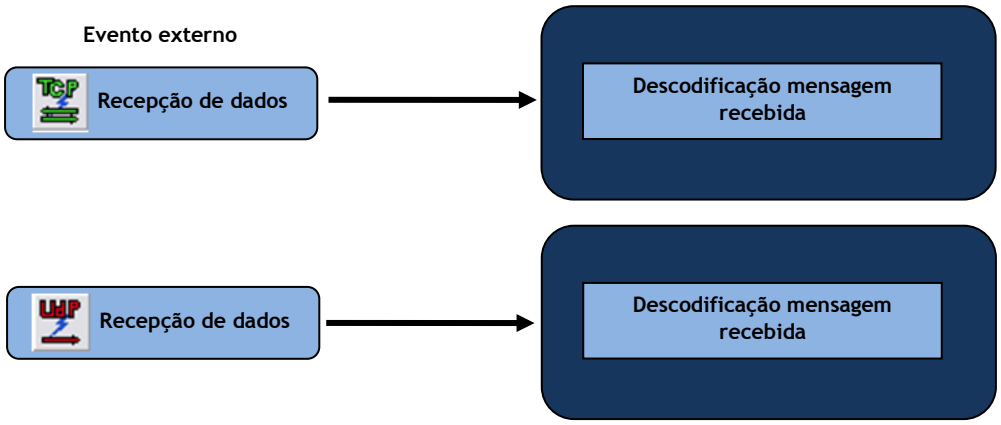


Figura 4.20 – Esquema da acção a tomar para eventos externos

Nas figuras 4.21 e 4.22 são apresentadas as interfaces gráficas disponíveis na aplicação de controlo e decisão do robô.

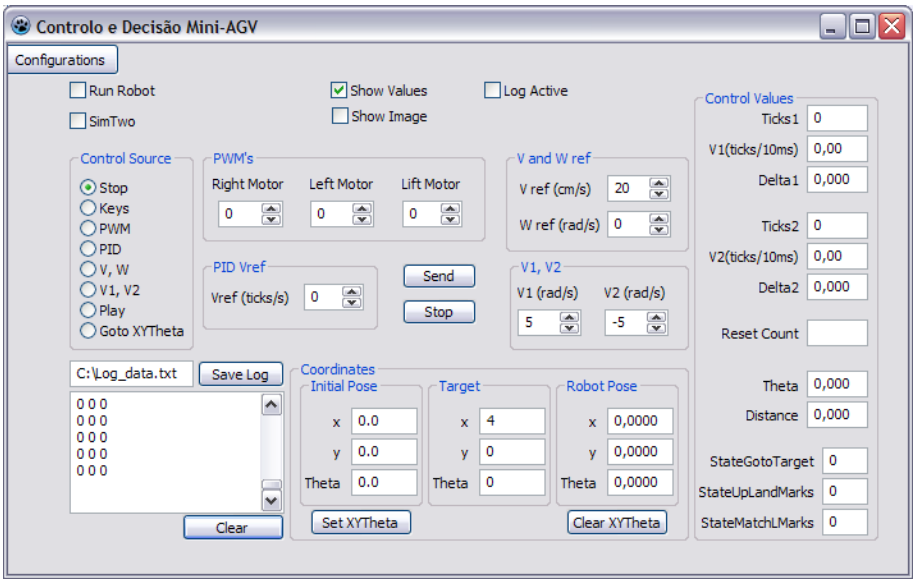


Figura 4.21 – Interface gráfica da aplicação de controlo do robô

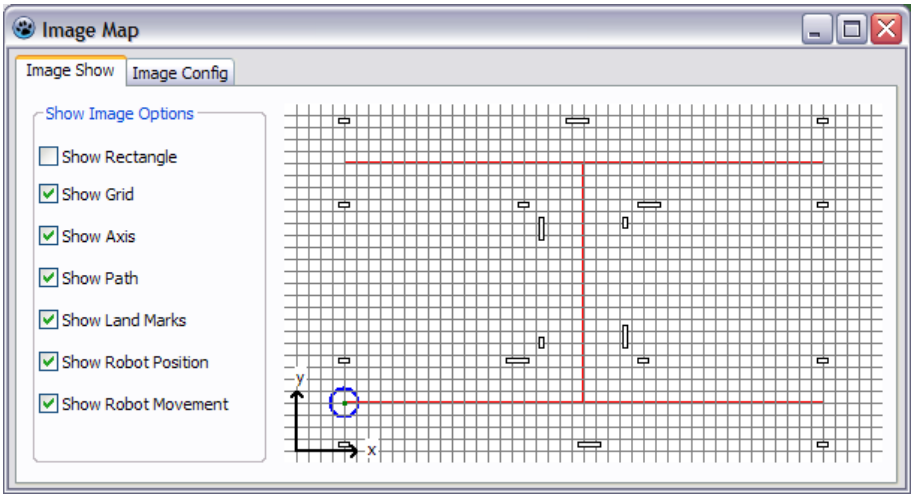


Figura 4.22 – Interface gráfica para visualização do movimento do robô

4.3.5 Descrição da camada inferior

Como já referido, o robô possui alguns controladores periféricos, responsáveis pelo processamento local de certas tarefas. Estes controladores permitem libertar o computador de tarefas simples e repetitivas, mas que requerem níveis de latência muito baixos, e por isso normalmente só disponíveis em sistemas operativos de tempo real.

Esta camada é responsável pelo processamento das leituras dos *encoders*, pela actuação nos motores de tracção, pelo controlo da velocidade dos motores e pelo encaminhamento dos dados para a camada de processamento superior. A leitura dos *encoders* é realizada num microcontrolador sendo os dados desse processamento posteriormente utilizados pelo microcontrolador, responsável pelo controlo e actuação dos motores. O controlo do sentido de rotação é feito por um sinal digital de um porto I/O do microcontrolador. A recolha dos dados odométricos permite efectuar um controlo em malha fechada da velocidade dos motores, tendo sido implementado um algoritmo de compensação tipo *PI* que efectua o controlo de 10 ms em 10 ms.

4.3.6 Controladores de velocidade dos motores

Os controladores de velocidade dos motores são responsáveis pela actuação nos motores acoplados às rodas, garantindo que estes se encontram a rodar à velocidade de referência estabelecida. Deste modo, um controlador de velocidade dos motores exige o conhecimento prévio do modelo dinâmico aproximado do robô.

Os dois motores *DC* que estão acoplados às rodas motrizes são controlados de forma independente. O controlo das velocidades é implementado no microcontrolador dos *drivers* de cada motor. Logo, quando se pretende que o robô se desloque com uma determinada velocidade, é dada uma referência a cada motor, em *ticks* (números de impulsos por intervalo de amostragem).

Na figura 4.23 exibe-se o diagrama de blocos que permite efectuar o controlo da velocidade de referência nos motores em malha fechada.

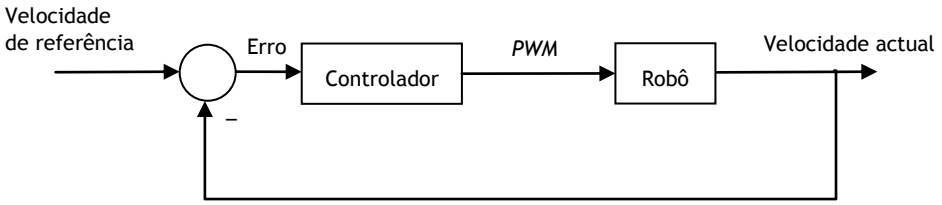


Figura 4.23 – Diagrama de blocos do sistema

Nas secções seguintes são descritos os procedimentos para obtenção do modelo dinâmico do robô, bem como o dimensionamento e implementação dos controladores de velocidade dos motores.

4.3.6.1 Modelo dinâmico do robô

Os movimentos de um robô surgem como resultado da aplicação de tensões *DC* em cada motor, criando um determinado binário em cada roda motriz. Estes binários são convertidos pelas próprias rodas em força de tracção paralelas ao chão que obrigam o robô a descrever uma certa trajectória.

Como foi referido anteriormente, a calibração de um controlador de velocidade para os motores requer um conhecimento *a priori* do comportamento do robô. Este comportamento pode ser descrito através de um modelo dinâmico aproximado que permitirá controlar uma dada trajectória através do controlo da velocidade a aplicar nos motores de tracção diferencial.

Deste modo, para se obter este modelo dinâmico é necessário actuar nos motores, aplicando um degrau de tensão em ambos. Alguns dados sobre a movimentação do robô são guardados num ficheiro de texto onde, posteriormente, são utilizados na construção gráfica da evolução temporal das velocidades dos motores, figura 4.24.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Motor da roda direita						Motor da roda esquerda					
2	Vact/40ms	Delta	Vact/10ms	Delta(s)	exp	erro^2	Vact/40ms	Delta	Vact/10ms	Delta(s)	exp	erro^2
3	0	0	0	0	0	0	0	0	0	0	0	0
4	200	4	50	0,04	20,64042	861,9849	-318	4	-79,5	0,04	-29,9233	2457,853
5	200	4	50	0,08	39,57256	108,7315	-318	4	-79,5	0,08	-56,265	539,8634
6	300	4	75	0,12	56,9378	326,243	-318	4	-79,5	0,12	-79,454	0,002116
7	300	4	75	0,16	72,86583	4,554676	-399	4	-99,75	0,16	-99,8675	0,013806
8	300	4	75	0,2	87,4756	155,6405	-399	4	-99,75	0,2	-117,838	327,166
9	404	4	101	0,24	100,8762	0,015327	-399	4	-99,75	0,24	-133,657	1149,693
10	404	4	101	0,28	113,1677	148,0533	-570	4	-142,5	0,28	-147,583	25,83803
11	404	4	101	0,32	124,4419	549,5246	-570	4	-142,5	0,32	-159,842	300,7559
12	456	4	114	0,36	134,7831	431,9359	-570	4	-142,5	0,36	-170,634	791,5348
13	456	4	114	0,4	144,2683	916,1714	-716	4	-179	0,4	-180,134	1,287019
14	456	4	114	0,44	152,9685	1518,547	-716	4	-179	0,44	-188,498	90,20509
15	570	4	142,5	0,48	160,9487	340,3545	-716	4	-179	0,48	-195,86	284,2537
16	570	4	142,5	0,52	168,2684	664,0097	-900	4	-225	0,52	-202,341	513,4374
17	570	4	142,5	0,56	174,9823	1055,098	-900	4	-225	0,56	-208,046	287,4329
18	948	5	189,6	0,61	182,5987	49,01835	-900	4	-225	0,6	-213,069	142,3582
19	948	5	189,6	0,66	189,4355	0,027066	-1062	4	-265,5	0,64	-217,49	2304,968
20	948	5	189,6	0,71	195,5725	35,67016	-1062	4	-265,5	0,68	-221,382	1946,393

...

Figura 4.24 – Folha de cálculo para construção gráfica da evolução temporal das velocidades em ambos os motores

Na figura 4.25 apresenta-se a evolução temporal das velocidades dos motores, após colocar o robô a efectuar um movimento de translação o mais linear possível, isto é, após aplicar um degrau no motor da roda motriz direita e um degrau de sinal oposto no motor da roda motriz esquerda.

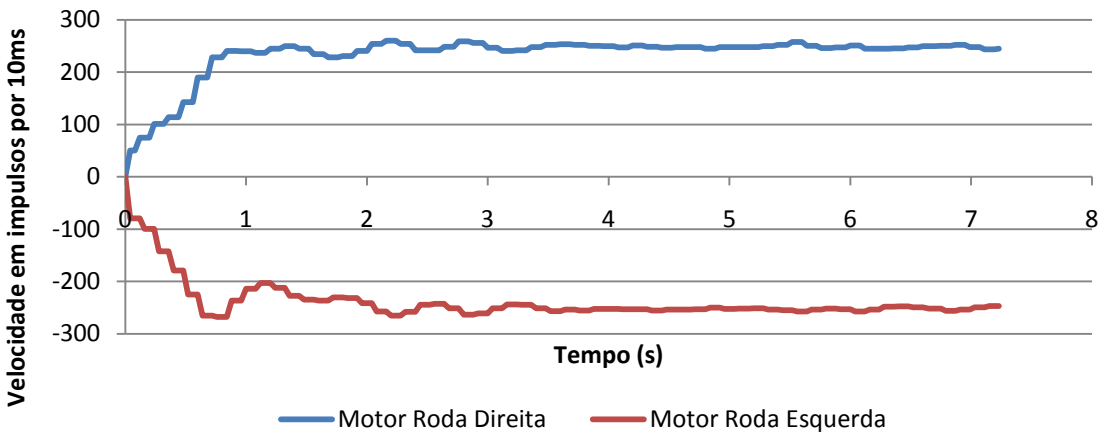


Figura 4.25 – Evolução temporal das velocidades dos motores

Com base na figura 4.25 verifica-se que a resposta do sistema é semelhante a um sistema de 1ª ordem com resposta exponencial. É também possível observar que a resposta do sistema se apresenta estável embora com algumas oscilações. Tais oscilações devem ser introduzidas por perturbações entre a estrutura mecânica e o piso. O tempo de estabelecimento do sistema é de, aproximadamente, 1 s.

Um sistema de 1ª ordem possui a seguinte função de transferência:

$$G(s)=\frac{Y(s)}{X(s)}=\frac{G_{DC}}{\tau s+1}, \tag{4.1}$$

correspondendo à equação diferencial

$$\tau y'(t)+y(t)=G_{DC}x(t). \tag{4.2}$$

Assim, o sistema modelado e apresentado na figura 4.24 pode ser representado por

$$v(t)=G_{DC}\left(1-e^{-\frac{t}{\tau}}\right)u(t), \tag{4.3}$$

sendo v(t) o sinal de saída, u(t) a entrada em degrau, G_{DC} o ganho *DC* e τ a constante de tempo para regime permanente.

Uma vez definida a equação que modela o robô é agora necessário determinar os parâmetros desta equação, nomeadamente G_{DC} e τ. Desta forma, recorreu-se ao método de Aproximação do Mínimos Quadrados, usando como pares ordenados os valores dos impulsos por período de amostragem, '*Vact/10ms*', e os valores do período de ciclo de controlo, *Delta(s),* (figura 4.24), estimando assim, os parâmetros G_{DC} e τ para cada roda motriz. Seguidamente, é realizada a média entre a estimação dos parâmetros obtidos para a roda motriz direita e a roda motriz esquerda. Na tabela 4.5 apresentam-se os valores obtidos para

os parâmetros do modelo dinâmico do robô que serão utilizados, posteriormente, para o dimensionamento dos controladores de velocidade dos motores.

Tabela 4.5 – Parâmetros do modelo dinâmico do robô

Ganho *DC*	4,17
Constante de tempo (s)	0,39

Na figura 4.26 apresenta-se as funções de aproximação do modelo do robô para a roda direita - *Motor Roda Direita (curva de tendência)* - e para a roda esquerda - *Motor Roda Esquerda (curva de tendência)*.

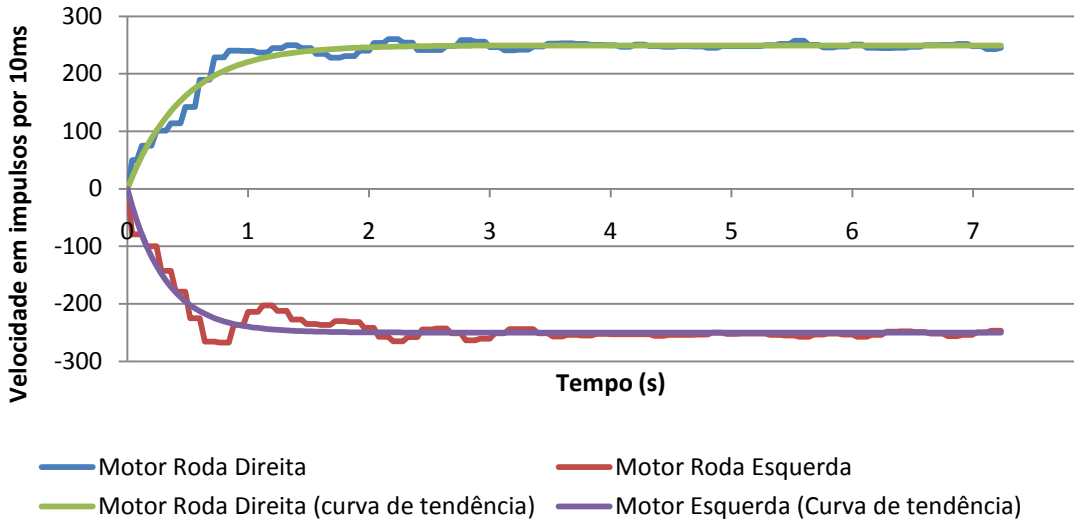


Figura 4.26 – Resultados da aproximação do modelo dinâmico do robô a um sistema de 1ª ordem

4.3.6.2 Dimensionamento do controlador

Como referido anteriormente, o controlo da velocidade dos motores é realizado em malha fechada. Este controlo é efectuado recorrendo a um algoritmo de compensação do tipo *PI* com filtro de entrada e *Feed-Forward*. A figura 4.27 apresenta o diagrama de blocos do sistema controlado.

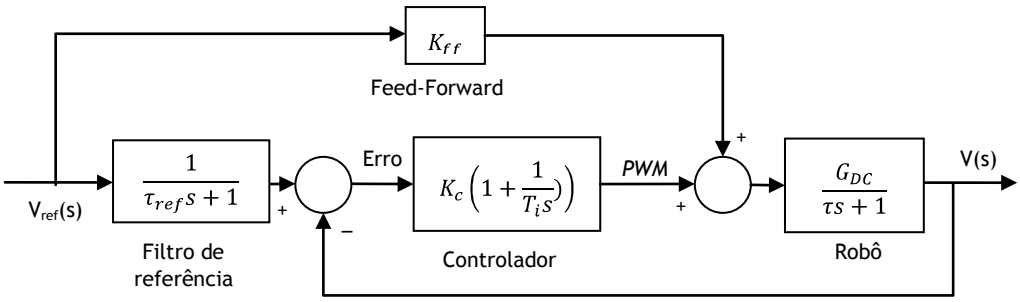


Figura 4.27 – Diagrama de blocos do sistema controlado

Na figura 4.27 pode-se observar, para além do controlador e da planta (Robô), um controlo *Feed-Forward* e um filtro do sinal de referência. O filtro de referência aplicado na entrada do sistema permite anular o zero introduzido pelo controlador, ou seja, anular a sobreelongação do sistema. Deste modo, o pólo do filtro terá que ser igual ao zero do controlador. O controlo *Feed-Forward* permite uma resposta mais rápida e melhorada sem interferir na estabilidade.

A função de transferência em malha fechada do diagrama de blocos representado na figura 4.27 é dada por

$$\frac{v(s)}{v_{ref}(s)} = \frac{\frac{1}{\tau_{ref}s+1} * K_c\left(1+\frac{1}{T_i s}\right) * \frac{G_{DC}}{\tau s+1}}{1+\frac{K_c\left(1+\frac{1}{T_i s}\right) * G_{DC}}{\tau s+1}} = \frac{\left(\frac{K_c G_{DC}}{T_i \tau}\right)}{s^2+\left(\frac{1+K_c G_{DC}}{\tau}\right)s+\frac{K_c G_{DC}}{T_i \tau}} . \quad \textbf{(4.4)}$$

Para dimensionar os parâmetros do controlador, nomeadamente, K_c e T_i recorreu-se ao protótipo de *Bessel*. Assim, definiu-se um tempo de estabelecimento[8] de 1 s e calculou-se os pólos da função de transferência através dos pólos de *Bessel* [34]. Os parâmetros determinados para o controlador *PI* encontram-se representados na tabela 4.6.

Tabela 4.6 – Parâmetros do *PI* utilizados nos motores

Ganho do Controlador	0,518
Tempo integral	0,253
Constante de tempo filtro de referência (s)	0,253

Após a determinação dos parâmetros do controlador realizou-se uma simulação com o intuito de verificar se a resposta do sistema era satisfatória, figura 4.28.

A partir da figura 4.29 é possível verificar que a resposta do sistema está dentro do que era esperado, garantindo o tempo de estabelecimento máximo de 1 s.

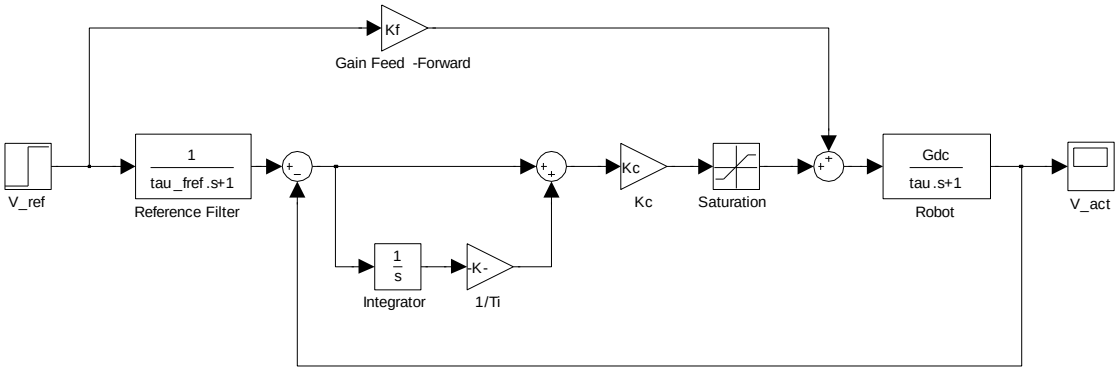


Figura 4.28 – Sistema simulado com filtro de entrada

[8] Corresponde ao tempo decorrente desde o instante em que o sistema começa a responder à solicitação de movimento e o instante em que o valor de regime permanente é obtido.

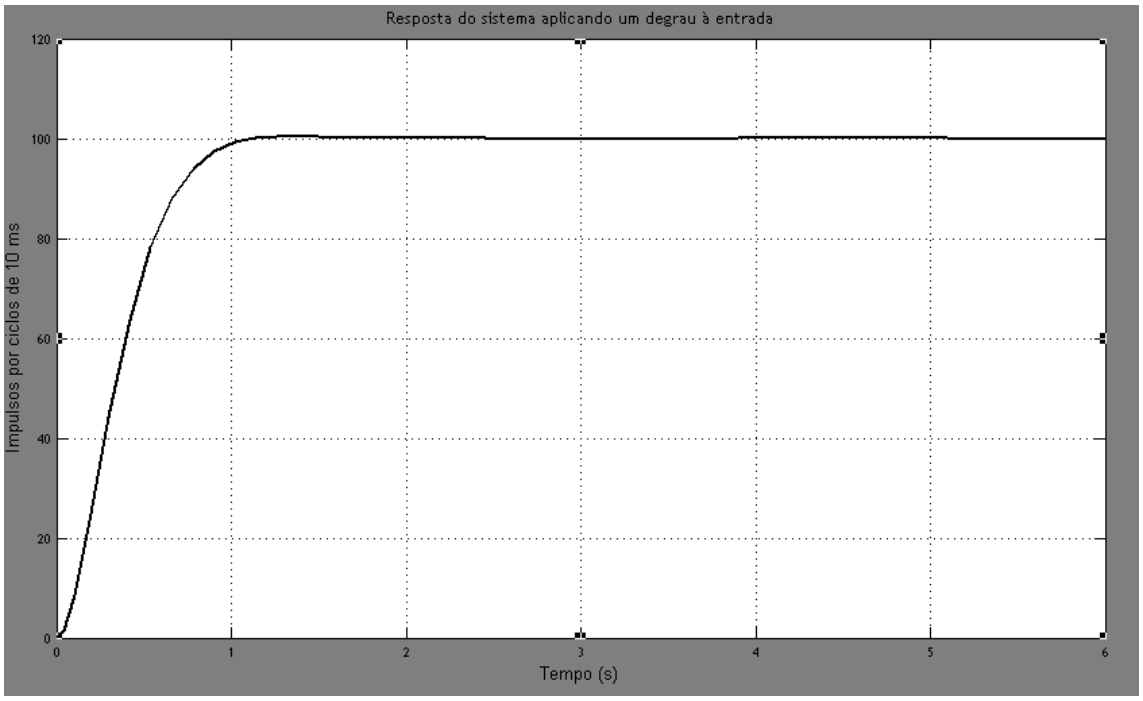


Figura 4.29 – Resposta ao sistema simulado com filtro de entrada

Com os parâmetros obtidos através do protótipo de *Bessel* e após a simulação do sistema a controlar, calibrou-se o *PI* de cada motor do robô e aplicou-se a mesma velocidade de referência aos motores. Na figura 4.30 exibe-se a resposta do sistema à aplicação de um degrau a cada motor. Pode-se observar também que a resposta é idêntica à resposta do sistema simulado, figura 4.29, comprovando a correcta calibração dos controladores e modelização do sistema.

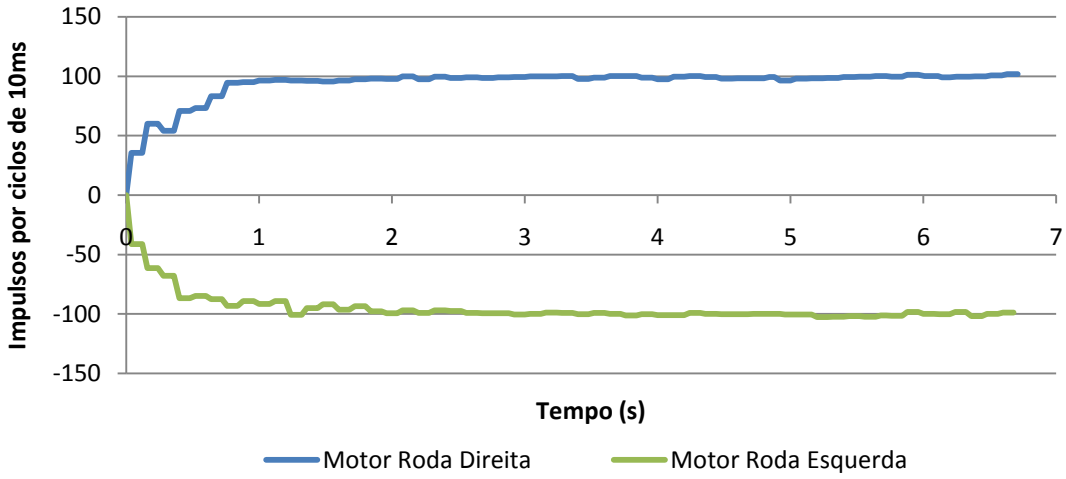


Figura 4.30 – Resposta ao degrau da velocidade aplicado às duas rodas motrizes do robô

Capítulo 5

Plataforma de Simulação

Neste capítulo realça-se a necessidade da utilização de uma ferramenta que possibilite a realização de simulações do sistema robótico apresentado, com o intuito de testar diferentes estratégias de decisão e controlo. A aplicação desta ferramenta permite economizar tempo, evitar acidentes e evitar danos no robô e sensores.

A plataforma de simulação utilizada para testar os algoritmos de controlo e decisão foi desenvolvida pelo Prof. Dr. Paulo Costa e apelidada de *SimTwo* [35].

O simulador permite verificar os resultados em tempo real através de uma interface gráfica *3D*. A partir desta, torna-se possível observar a posição, o movimento e a trajectória actual do robô. Assim, para além das vantagens da utilização de um simulador descritas anteriormente, este permite também supervisionar o comportamento do robô em tempo real, auxiliando na detecção de possíveis erros no algoritmo de decisão e controlo de forma a aperfeiçoar o software desenvolvido.

Neste capítulo, na secção 5.1, é apresentada uma visão geral do simulador e dos seus recursos. Na secção 5.2 é descrita a implementação da configuração do modelo do robô em estudo, bem como a estruturação da interligação do simulador com a aplicação de controlo e decisão. Por fim, apresenta-se na secção 5.3 uma estratégia de localização do robô aplicando um sistema de detecção de paredes através de sensores que mensuram distâncias.

5.1 Descrição Geral

A plataforma de simulação *SimTwo* é um sistema de simulação realista onde é possível implementar vários tipos de robôs[9]. Este realismo relativo à dinâmica do robô permite decompor o mesmo em duas partes: uma referente ao realismo introduzido nos sistemas de corpos rígidos utilizando a biblioteca *ODE* (*Open Dynamics Engine*) [36]; e outra relativa ao realismo aplicado nos modelos dos motores eléctricos e controladores [35]. A biblioteca *ODE* é direccionada para a simulação de estruturas articuladas, tais como veículos terrestres ou

[9] Robôs móveis, manipuladores industriais, quadrúpedes, humanóides, entre outros.

criaturas com pernas. Esta supõe dois níveis de simulação, o nível da dinâmica que se encarrega das propriedades dinâmicas dos corpos, tais como massa e velocidade, e o nível das colisões, que se ocupa da forma dos corpos.

As estruturas articuladas ou rodas, nos robôs encontram-se associadas a eixos que normalmente supõem um sistema de accionamento. Este sistema pode ser constituído por um motor de corrente contínua com caixa redutora, um controlador e um *encoder* óptico. O controlador permite ser definido aplicando um *PID* ao sinal de posição ou velocidade [35]. Este simulador realça um especial cuidado na modelização dos modelos dos motores e tem como objectivo capturar as não linearidades inerentes a estes actuadores, tais como limitações de corrente, saturação da tensão aplicada e atrito de *Coulomb* [35]. Estas não linearidades são factores imutáveis nos motores de todos os robôs.

Este simulador oferece dois níveis de controlo do robô. Um sistema de controlo de baixo nível responsável pelo controlo dos motores que, por defeito, efectua 100 ciclos por segundo (invocado periodicamente de 10 ms em 10 ms) e um controlador de mais alto nível implementado pelo utilizador, que permite desenvolver tarefas de controlo mais complexas e que, por defeito, corre com um período de 40 ms (25 ciclos por segundo). Este controlador pode ser implementado recorrendo a um programa remoto que comunica com o *SimTwo* através de protocolo *UDP* ou porta série, ou recorrendo a uma linguagem de *script* que se baseia na biblioteca *Pascal Script* [37] editável e compilável no próprio simulador [35]. É de salientar que o controlador de baixo nível encontra-se desenvolvido de origem, mas permite as alterações de muitos parâmetros.

A plataforma de simulação permite observar o comportamento do sistema em tempo real, pois é dotado de uma interface de visualização *3D*.

Para se exibir de uma forma perceptível o aspecto gráfico desta ferramenta são expostas de seguida algumas capturas das várias janelas que contemplam o *SimTwo*.

Na figura 5.1 apresenta-se a janela da interface gráfica *3D* deste simulador. O *SimTwo* fornece também uma janela de configuração da plataforma de simulação, figura 5.2. Através desta última é possível configurar parâmetros gráficos da interface gráfica *3D*, figura 5.2 a), parâmetros da configuração do robô, figura 5.2 b), bem como parâmetros da configuração física dos objectos, figura 5.2 c). O *SimTwo* é ainda dotado de um editor de código que permite a execução de código escrito numa variante da linguagem *Pascal* que é um subconjunto do *ObjectPascal*[10], figura 5.3. Na figura 5.4, ilustra-se a janela de gráficos. Esta janela permite registar a evolução de algumas variáveis em função do tempo e também visualizar esta progressão em tempo real.

Como foi referido anteriormente, o *SimTwo* permite simular diferentes configurações de robôs. Para tal, este simulador recorre a ficheiros com uma sintaxe *XML* [38] editáveis a partir de uma janela fornecida pela plataforma de simulação, figura 5.5. Ao iniciar, a aplicação reproduz o ficheiro *scene.xml* onde se encontram definidos os ficheiros *XML* que contemplam a configuração dos robôs, obstáculos e objectos. A diferença entre obstáculos e objectos é que os primeiros são inamovíveis e objectos são elementos livres com dinâmica [35].

---

[10] Tal como existe para o *Delphi* e *Free Pascal*.

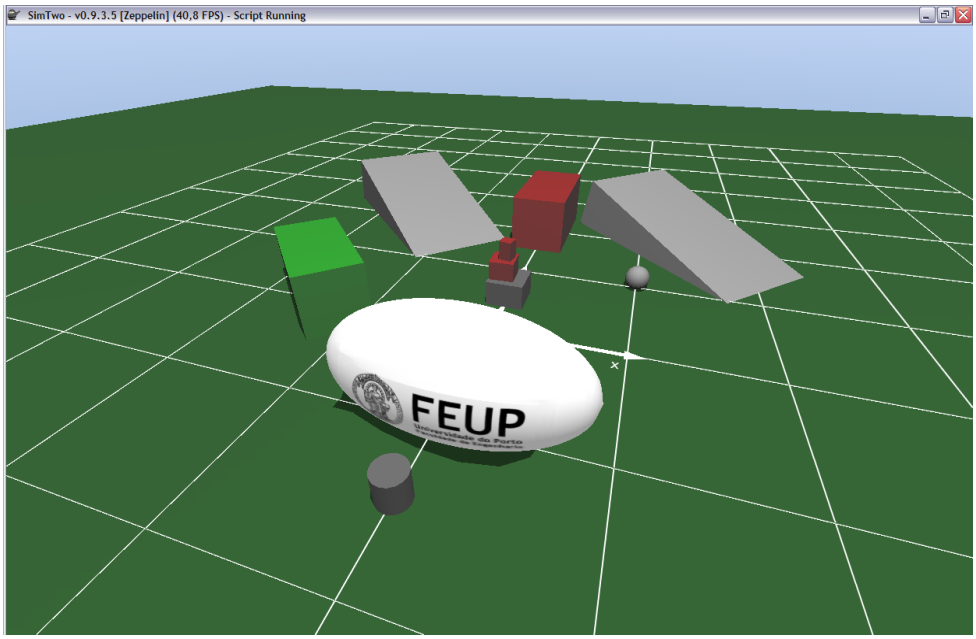
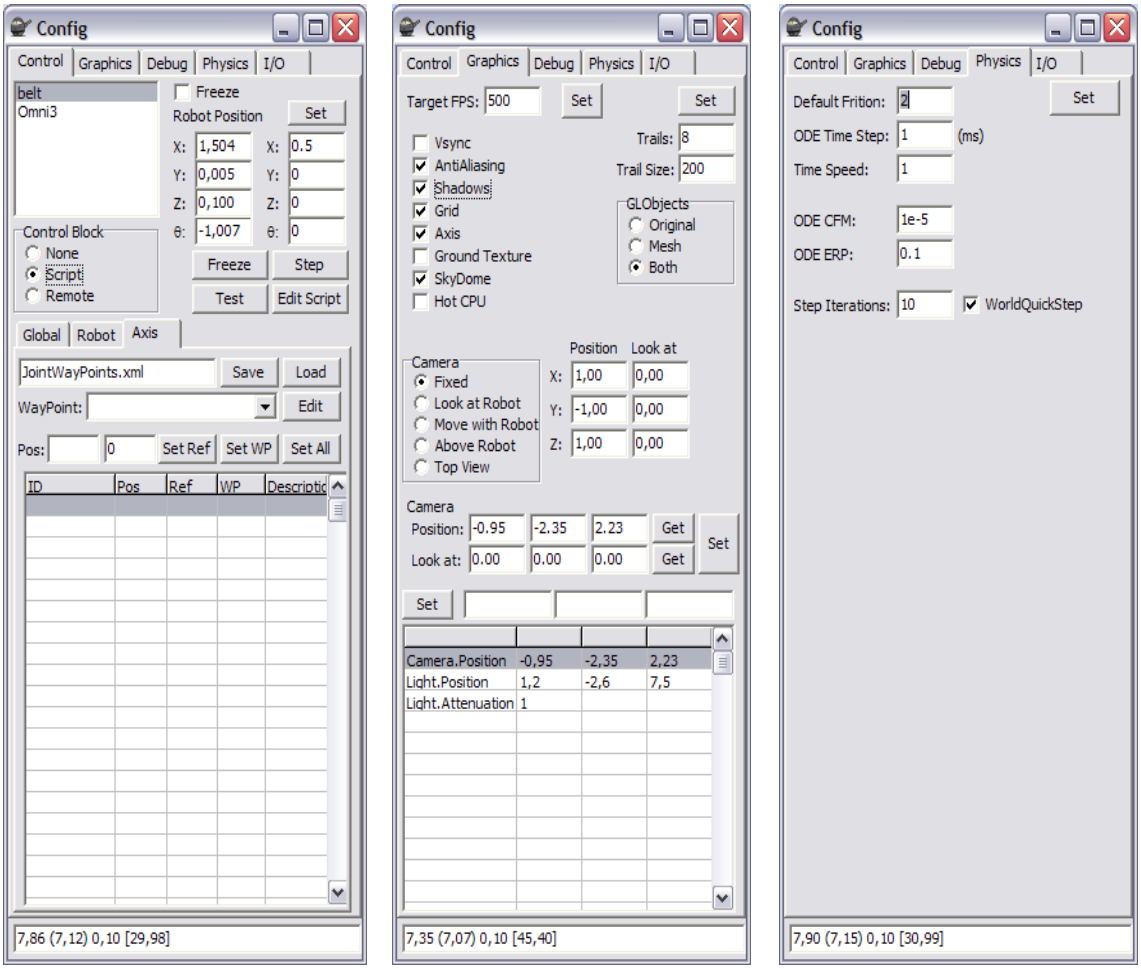


Figura 5.1 – Janela de visualização da interface gráfica *3D* do *SimTwo*



a) **b)** **c)**

Figura 5.2 – Janela de configuração do *SimTwo*

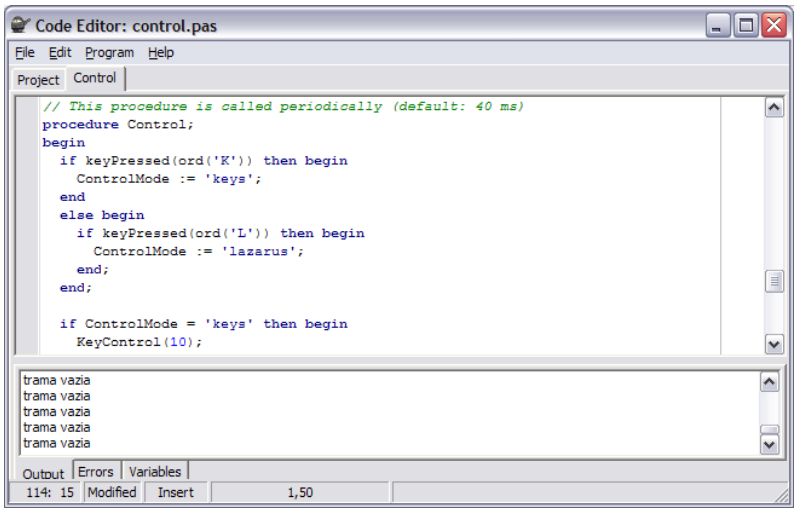


Figura 5.3 – Janela do editor de código do *SimTwo*

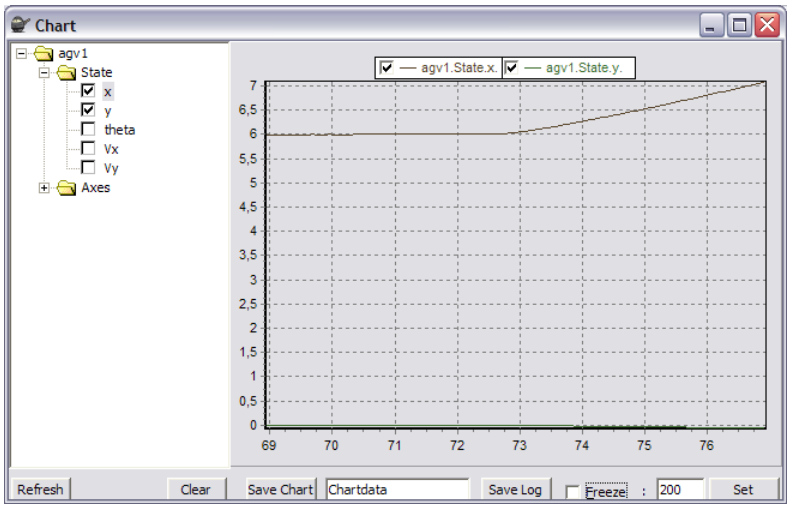


Figura 5.4 – Janela de gráficos do *SimTwo*

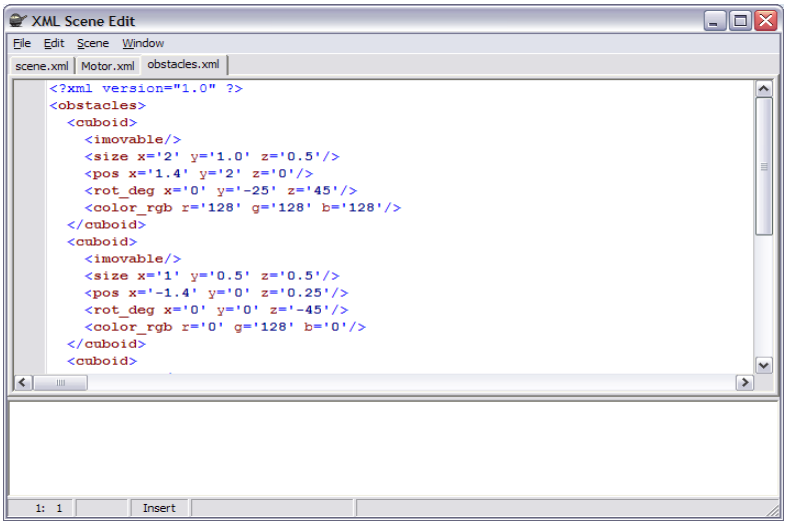


Figura 5.5 – Janela de edição em *XML* dos objectos do *SimTwo*

5.2 Configuração do simulador

Como havia sido exposto na secção anterior, o *SimTwo* especifica o *mundo virtual*[11] como uma organização hierárquica de informação que contempla no nível mais elevado desta hierarquia o ficheiro *XML, scene.xml*. A partir deste nível são introduzidas as classes ou secções, que definem, no caso concreto em estudo, a configuração do robô, os obstáculos ou paredes e a pista, sendo que esta última expressa o ambiente das trajectórias definidas para o robô. Na tabela 5.1 apresenta-se um resumo das classes abrangidas pelo ficheiro *scene.xml*.

Tabela 5.1 – Classes abrangidas pela estrutura hierarquia mais elevada do *SimTwo*

Classes	Descrição
Robot	Configuração da estrutura do robô
Things	Configuração de corpos dinâmicos
Obstacles	Configuração de obstáculos (inamovíveis)
Track	Configuração de ambientes com trajectórias desenhadas no pavimento

Das várias classes descritas anteriormente, a classe *robot* é a única instância controlável na plataforma de simulação. A configuração desta classe é estabelecida recorrendo a estruturas virtuais que são definidas por objectos. Estes podem ser divididos em dois grupos, um que incorpora os objectos básicos, *solids*, permitindo definir a estrutura física do robô, e outro que incluí os objectos mais complexos, tais como sensores, articulações, que permitem a interacção com o ambiente. Os objectos básicos possuem algumas características fundamentais, tais como forma, massa, dimensão, permitindo simular de uma forma realista a dinâmica dos corpos rígidos. Na tabela 5.2 exibe-se os diferentes tipos de objectos básicos suportados pelo *SimTwo* e as suas propriedades. Os restantes objectos são apresentados na tabela 5.3. Um exemplo de código desta implementação é mostrado de seguida:

```
<solids>
   <cuboid>
     <ID value='AgvDown'/>
```

Tabela 5.2 – Objectos básicos do *SimTwo* e suas propriedades

Objecto	Descrição	Propriedade	Componente	Descrição da Propriedade
Cuboid	Paralelepípedo	Id	value	Identificação do objecto
		Mass	value	Massa do objecto, em Kg
Cylinder	Cilindro	Size	x, y, z	Tamanho, em metros
Belt	Tapete rolante	Pos	x, y ,z	Posição no referencial mundo
		Rot_deg	x, y, z	Rotação, em graus
Sphere	Esfera	Color_rgb	r, g, b	Cor na base RGB (8bits)

[11] Subentende-se todos os objectos constituintes do meio ambiente desenvolvido para a simulação, tais como robôs, objectos, obstáculos e pista.

Tabela 5.3 – Objectos mais complexos suportados pelo *SimTwo*

Objecto	Descrição
Sensors	Sensores
Articulations	Articulações
Wheels	Rodas
Shells	Superfícies de contacto

A utilização dos objectos descritos na tabela 5.3 confere um nível de complexidade mais elevado à configuração dos robôs. As *shells* possuem a finalidade de tornar a forma do robô mais complexa e de resolver a limitação imposta pela biblioteca *ODE,* pois esta não implementa colisões entre cilindros. As *wheels* e as *articulations* correspondem ao sistema de actuação dos robôs, permitindo definir eixos articulados entre corpos. Por fim, os *sensors* permitem introduzir sensores no robô, embora presentemente só se encontram disponíveis os sensores que medem distâncias, similares aos sensores de infravermelhos.

Na tabela 5.4 apresentam-se as principais propriedades existentes nos objectos mais complexos.

Tabela 5.4 – Propriedades dos objectos mais complexos abrangidos pelo *SimTwo*

Propriedade	Componente	Objecto	Descrição
Motor	r_i, k_i, v_{max}, i_{max}, active	*Articulations/Wheels*	Parâmetros do motor
Gear	ratio	*Articulations/Wheels*	Relação da caixa redutora
Friction	B_v, f_c, coulomblimit	*Articulations/Wheels*	Parâmetros da fricção nos actuadores
Encoder	ppr, mean, stdev	*Articulations/Wheels*	Parâmetros do codificador óptico
Controller	mode, k_p, k_i, k_d, k_f, active	*Articulations/Wheels*	Parâmetros do controlador
Axis	x, y, z	*Articulations/Wheels*	Parâmetros do eixo da roda
type	value	*Articulations*	Tipo de articulação
Connect	B1, B2	*Articulations*	Ligação entre dois objectos
Pos	x, y, z	*Articulations/Sensors /Sheels*	Posição no referencial mundo
Limits	Min, Max	*Articulations*	Limites da articulação
Beam	length, initial_width, final_width	*Sensors*	Parâmetros dos sensores
Rot_deg	x, y, z	*Shells/Sensors*	Rotação, em graus
Color_rgb	r, g, b	*Shells/Sensors*	Cor na base RGB (8bits)
Omni	—	*Wheels*	Define a roda como omnidireccional
tyre	Mass, radius, width, centerdist	*Wheels*	Parâmetros da roda

5.3 Configuração do corpo do robô, paredes e pista

5.3.1 Modelação do robô simulado

Os objectos descritos na secção anterior apresentam-se como subclasses da classe *robot*. A descrição da configuração do robot na plataforma de simulação é realizada por intermédio de um ficheiro de texto com formato *XML*, que supõe quatro subclasses ou subsecções: a subclasse *solids*, que caracteriza o corpo do robô; a subclasse *shells* que representa as superfícies de contacto para o modelo de colisões; a subclasse *articulations* que é responsável por todo o sistema de movimentação, desde os motores até às rodas; e a subclasse *sensors* encarregue de introduzir sensores no robô.

A modelação do robô simulado realizou-se em duas fases. Na primeira representou-se o corpo do robô como um corpo simples, recorrendo apenas a objectos básicos do tipo *solids*. Este possuía a forma de um paralelepípedo representando apenas a base do robô, cuja massa é distribuída uniformemente pelo objecto, figura 5.6.

Numa segunda fase, desenvolveu-se o corpo do robô, com o intuito de se obter um aspecto próximo da estrutura real, e que possuísse uma distribuição de massa idêntica à do robô real, figura 5.7. Desta forma, decompôs-se este em diversos segmentos, tal como é representado na realidade. Esta concepção apresenta um nível de complexidade mais elevado e como tal, para simplificar a construção do modelo recorreu-se ao uso da subclasse *shells* que permite definir superfícies de colisão mais complexas. Deste modo, a estrutura final do corpo do robô utilizada na simulação é um conjunto de diversos segmentos referenciados sempre a partir de um elemento da base do mesmo, figura 5.8. Todas as partes foram previamente medidas e posteriormente recriadas para o robô simulado.

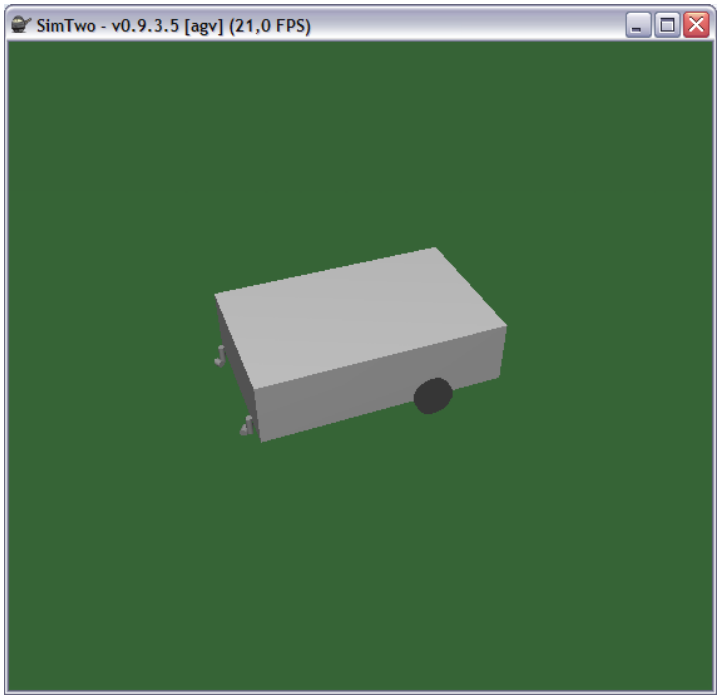


Figura 5.6 – Robô construído numa primeira fase

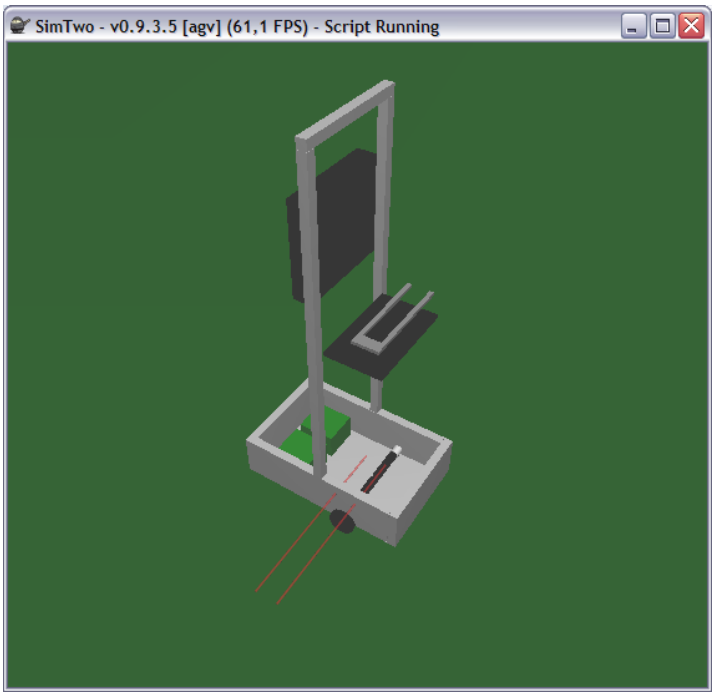


Figura 5.7 – Estrutura final do robô com visualização da distribuição de massa pelo corpo do robô

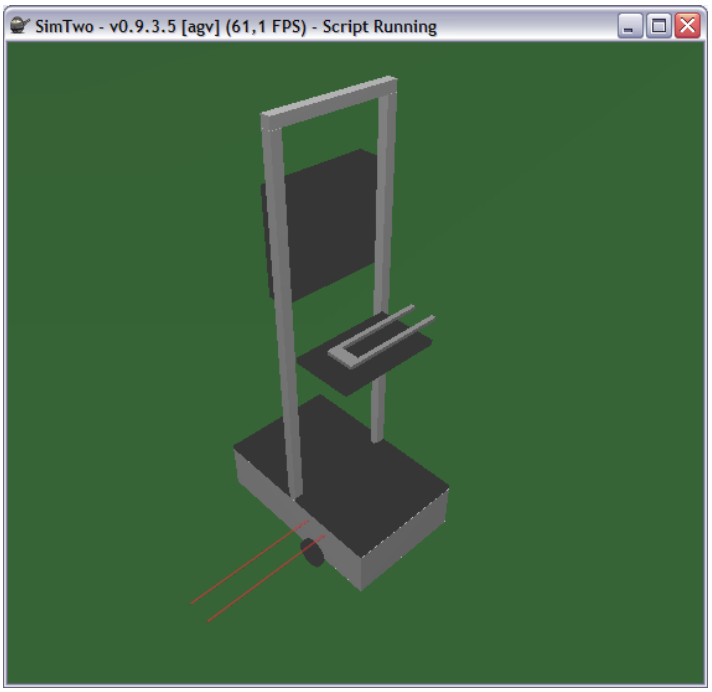


Figura 5.8 – Estrutura final do corpo do robô desenvolvido para a simulação

Para dotar o robô de comportamento próprios, isto é, para permitir que este se conseguisse movimentar, foi necessário introduzir a classe *articulations* no ficheiro de descrição do robô. Esta define através da subclasse *default*, as propriedades comuns das rodas, criando posteriormente uma subclasse para cada roda, contendo os seus parâmetros individuais.

Relativamente às propriedades comuns das rodas, estas são modeladas com base em pressupostos previamente adquiridos. As propriedades comuns contemplam o sistema de

actuação que engloba os motores, com as respectivas caixas redutoras, *encoders* e controladores. Assim, os parâmetros do motor da etiqueta *motor*, o parâmetro da relação de transmissão da caixa redutora, *ratio*, da etiqueta *gear*, bem como o parâmetro da resolução dos *encoders, ppr,* da etiqueta *encoder*, são estabelecidos recorrendo às *datasheets* dos fabricantes. No que diz respeito aos parâmetros do controlador da etiqueta *controller,* estes são baseados nos valores obtidos na secção 4.3.6.

Os parâmetros individuais atribuídos a cada roda são introduzidos através da subclasse *joint*. Esta permite definir através da etiqueta *connect* quais os objectos que serão interligados, através da etiqueta *pos* a posição do eixo de actuação e através da etiqueta *axis* o sentido de rotação do eixo.

De seguida é mostrado um exemplo de código para esta implementação:

```
<articulations>
  <default>
    <motor ri='0.316' ki='0.0302' vmax='24' imax='12' active='1'/>
    <gear ratio='43'/>
    <friction bv='0.001' fc='0.0615' coulomblimit='0.0001'/>
    <encoder ppr='44032' mean='0' stdev='0'/>
    <controller mode='pidspeed' kp='0.52' ki='0.0208' kd='0.0' kf='0.0' active='1' period='10'/>
```

A detecção de marcadores, nomeadamente a detecção da parede lateral é realizada através da utilização de sensores. Estes são introduzidos no ficheiro da descrição do robô na subclasse *sensors* e permitem especificar através da etiqueta *beam* os parâmetros dos sensores. A posição destes em relação ao robô é introduzida na etiqueta *pos*.

5.3.2 Modelação dos obstáculos

Uma vez que o simulador apenas permite a utilização de sensores que medem distâncias, tornou-se necessário criar objectos idênticos a paredes. As paredes pretendem simular a detecção de marcadores previamente definidos no ambiente e que fariam parte da trajectória do robô. O marcador utilizado na simulação é diferente do marcador detectado pelo robô real embora se deva salientar que o algoritmo de localização implementado na simulação é o mesmo para se detectar as linhas no chão (ou marcadores) com um sistema de visão.

Os obstáculos são definidos numa classe diferente da classe descrita na secção anterior, ou seja, na classe *obstacles*. Uma vez que estes são objectos inamovíveis é necessário inserir em cada subclasse a etiqueta *imovable*.

5.3.3 Modelação da pista

A modelação da pista através da classe *track*, para o trabalho em estudo, apenas possui um carácter ilustrativo uma vez que o robô não incorpora um sensor para detectar linhas no pavimento, mas sim para medir distâncias a um objecto, neste caso a uma parede. Assim,

recriou-se um ambiente que permite visualizar as trajectórias que se pretendem que o robô seja capaz de realizar. Estas podem ser vistas como um percurso a realizar em "H", representado através da cor azul, delimitando a área ou ambiente de actuação do robô através do quadrado a preto.

Na figura 5.9 apresenta-se o resultado final das modelações descritas anteriormente.

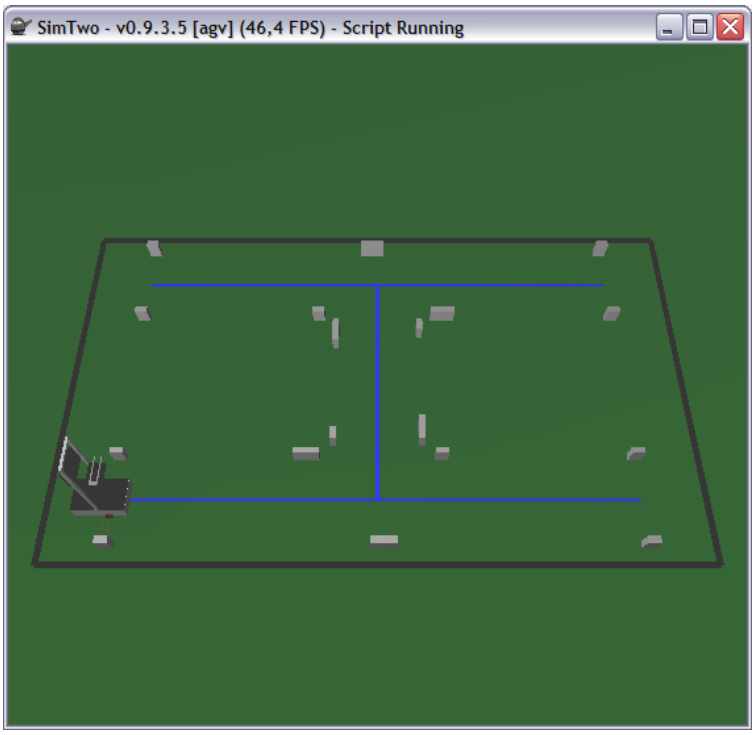


Figura 5.9 – Ambiente final do robô simulado no *SimTwo*

5.4 Controlo do robô simulado

O controlo do robô simulado é realizado através de uma aplicação externa, baseado no protocolo de comunicação *UDP*. Esta divisão em duas aplicações tem como finalidade, emular o comportamento do robô real de forma transparente para o *software* de decisão e controlo.

O princípio de funcionamento consiste na troca de mensagens entre o *SimTwo* e a aplicação de controlo e decisão.

Quando a aplicação externa pretende enviar uma ordem de movimento ao robô simulado, é transmitida uma mensagem ou trama. Esta trama contém a informação destinada aos controladores dos motores sendo processada pelo programa de controlo da simulação. A aplicação de controlo do *SimTwo* corre com um período de 40 ms e, cada vez que o ciclo de controlo se inicia, esta verifica se tem alguma trama de dados para processar. Se a resposta for positiva, é verificada a integridade da mensagem e de seguida descodificada. São então obtidas as referências de velocidade e aplicadas aos motores através da função *SetAxisSpeedRef(R: LongInt; i: LongInt; aSpeed: Double)*. Se a resposta for negativa ou se durante a descodificação da mesma for verificado a não integridade dos dados, esta é descartada permanecendo a referência de velocidade aplicada aos controladores introduzida no ciclo anterior.

No ciclo de controlo, o *SimTwo* também se encontra encarregue de enviar uma mensagem para a aplicação externa. Esta contém a odometria do robô, o número total de impulsos de cada roda, e a distância aos obstáculos medida pelos sensores de infravermelhos. Esta trama, uma vez preenchida e enviada por rede, é interpretada pela aplicação de controlo e decisão como se fosse proveniente da camada de processamento inferior (*drivers* dos motores). Esta interpretação passa por despoletar um evento, sempre que a aplicação externa recebe uma trama de dados. Tal como no simulador, na aplicação externa também se verifica a integridade da mensagem recebida. O processo é idêntico e no caso de não se verificar a integridade dos dados também é descartada, permanecendo o número de impulsos total para cada roda e a distância do robô à parede do ciclo anterior.

Na tabela 5.5 e 5.6 apresenta-se o formato das mensagens enviadas e recebidas pelo *SimTwo*.

Tabela 5.5 – Formato da mensagem enviada pelo *SimTwo*

'#'	CR[12]	Odo_0	CR	Odo_1	CR	Sensor_0	CR	Sensor_1	CR	'*'

Tabela 5.6 – Formato da mensagem recebida pelo *SimTwo*

'#'	CR	Vel_Motor_0	CR	Vel_Motor_1	CR	'*'

[12] CR ou *carriage return*

Capítulo 6

Localização e Navegação

6.1 Descrição Geral

Para o trabalho em estudo, recorreu-se ao método baseado em odometria para a medição relativa do estado do robô e à detecção de marcadores artificiais para a medição absoluta do estado do robô. Desta forma, através da odometria, que proporciona uma boa precisão a curto prazo, determina-se o estado actual do robô e através da detecção de marcadores por visão artificial corrige-se, periodicamente, a posição absoluta do mesmo.

Neste capítulo apresenta-se para além da localização relativa e absoluta, o controlo e planeamento de trajectórias para o robô, responsável pelo correcto movimento do robô durante os percursos a realizar, secção 6.4. Na secção 6.5 exibe-se o algoritmo de localização, descrevendo a estrutura deste e o método matemático utilizado para a auto-localização de linhas. A secção 6.6 contempla os métodos de fusão sensorial que permitem uma melhor estimativa do estado actual do robô.

6.2 Odometria – Localização relativa

Como referido anteriormente, a odometria é uma técnica que permite quantificar as distâncias percorridas por um veículo. A quantificação do movimento dos motores, e consequentemente da rotação das rodas, é possibilitada pelos *encoders* incrementais presentes na plataforma. Sabendo o estado anterior do veículo e a distância percorrida por cada roda motriz é possível conhecer o estado actual do mesmo, ou seja, a sua posição cartesiana e a sua orientação em relação ao referencial mundo. Para que um robô se localize no mundo e conheça com precisão as suas velocidades, é necessária uma correcta calibração da odometria.

6.2.1 Modelo de odometria

O sistema modelado é constituído por um robô móvel de duas rodas com accionamento diferencial. A figura 6.1 apresenta uma representação esquemática do robô modelado.

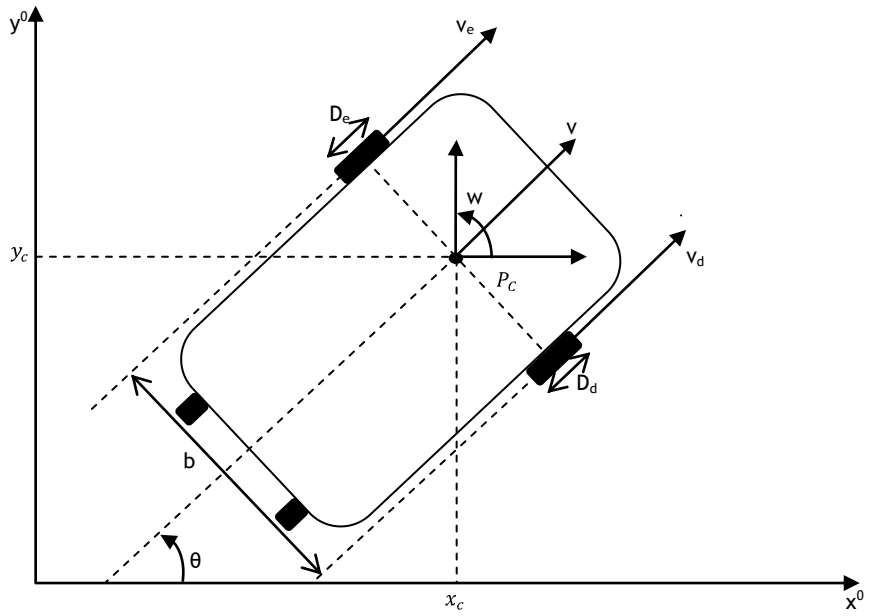


Figura 6.1 – Robô diferencial

O modelo cinemático do robô pode ser descrito com base na figura 6.1, onde x^0 e y^0 são as coordenadas de base do sistema de referência; P_C representa a intersecção entre o eixo de simetria e o segmento que passa pelas rodas; $v_{d,e}$ são as velocidades lineares das rodas direita e esquerda, respectivamente; v é a velocidade linear do robô; b é a distância entre as rodas do robô; $D_{d,e}$ são os diâmetros das rodas direita e esquerda, respectivamente; w é a velocidade angular do robô.

Assim, o modelo cinemático do robô diferencial [20] que permite simular o seu comportamento é dado por

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} v(t) * \cos(\theta(t)) \\ v(t) * \sin(\theta(t)) \\ w(t) \end{bmatrix} \tag{6.1}$$

com,

$$v(t) = \frac{v_d(t) + v_e(t)}{2} \tag{6.2}$$

e

$$w(t) = \frac{v_d(t) - v_e(t)}{b}. \tag{6.3}$$

A odometria do robô é obtida através da integração da informação provida pelos *encoders* incrementais de cada roda, em cada período de amostragem, i, e permite estimar com base no estado actual o deslocamento linear e angular durante o período de amostragem.

Para se obter os deslocamentos lineares de cada roda, $d_d(i)$ e $d_e(i)$, no período de amostragem i, torna-se então necessário relacionar o deslocamento das rodas, $d_{d/e}(i)$, com os impulsos dos *encoders*, $N_{d/e}(i)$. Assim, o deslocamento $d_{d/e}(i)$ é dado por

$$d_{d/e}(i) = c_m * N_{d/e}(i) \tag{6.4}$$

com,

$$c_m = \frac{\pi D_n}{n C_e}. \tag{6.5}$$

A equação 6.5 permite determinar c_m que representa o factor de conversão entre impulsos dos *encoders*, $N_{d/e}(i)$, e o deslocamento linear da respectiva roda onde, D_n é o diâmetro nominal das rodas, C_e a resolução dos *encoders* e n a relação da caixa redutora.

Assim, discretizando o modelo da cinemática do robô a partir das diferenças centradas, actualiza-se a o estado do robô através de

$$\begin{bmatrix} x(i+1) \\ y(i+1) \\ \theta(i+1) \end{bmatrix} = \begin{bmatrix} x(i) + d(i) * \cos\left(\theta(i) + {}^{\Delta\theta(i)}/_{2}\right) \\ y(i) + d(i) * \sin\left(\theta(i) + {}^{\Delta\theta(i)}/_{2}\right) \\ \theta(i) + \Delta\theta(i) \end{bmatrix} \tag{6.6}$$

com,

$$d(i) = \frac{d_d(i) + d_e(i)}{2} \tag{6.7}$$

e

$$\Delta\theta(i) = \frac{d_d(i) - d_e(i)}{b}. \tag{6.8}$$

A utilização de diferenças centradas para discretizar o modelo cinemático tem como finalidade diminuir o erro introduzido pela discretização na equação 6.1.

Após a exposição das equações que determinam o modelo cinemático do robô em estudo, pode-se tirar algumas elações em relação aos parâmetros a obter para o modelo de odometria. O modelo de odometria ficará completamente exposto mediante a calibração dos do raio das rodas e da distância entre rodas.

6.2.2 Calibração da odometria

A calibração da odometria é um processo extremamente importante quando se pretende uma estimativa fiável das distâncias percorridas pelo robô. O processo de calibração caracteriza-se por ser um procedimento moroso e complexo. A sua incorrecta calibração

causa erros grosseiros na precisão das medidas de distâncias devido ao erro ser acumulativo. Esta causa é estabelecida pois o erro é integrado em cada período de amostragem i o que inviabiliza a utilização desta técnica por longos períodos de tempo. Mesmo que o erro acumulado seja pequeno, haverá sempre perturbações externas que inviabilizam a utilização prolongada.

Como referido anteriormente, para uma correcta utilização do método de odometria é então necessário calibrar dois parâmetros. O diâmetro das rodas, D_n, e a distância entre rodas do robô, b. Uma vez que este último não influencia nos movimentos de translação, o primeiro parâmetro a ajustar deverá ser o diâmetro das rodas. Considerou-se que ambas compadecem do mesmo diâmetro.

Para medir o diâmetro das rodas foi necessário determinar, inicialmente, a relação de impulsos através do deslocamento do robô. Assim, colocou-se o robô a realizar um movimento de translação e armazenou-se a quantidade total de impulsos medidos pelos *encoders* de cada roda, figura 6.2. Para uma distância de 8,073 m o número total de impulsos medidos pelo *encoder* da roda direita e esquerda foi 930609 e 930876, respectivamente.

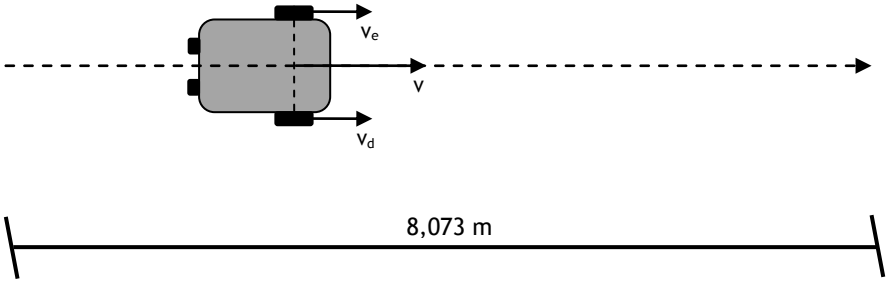


Figura 6.2 – Movimento linear para determinar a relação de impulsos através do deslocamento

Desta forma, determinou-se que os *encoders* do robô em estudo geram 115291 impulsos/metro.

Através das equações 6.4, 6.5 e 6.7 e do valor determinado para os impulsos/metro obtém-se um diâmetro médio, D_n, para as duas rodas de 12,16 cm.

Após a medição do diâmetro das rodas, procedeu-se à calibração da distância entre rodas. Para tal colocou-se o robô a realizar um movimento de rotação aplicando a mesma velocidade a cada roda mas com sinais contrários, para que o robô rodasse sobre si próprio. Durante este procedimento a velocidade deve ser reduzida para evitar situações de deslizamento das rodas e para minimizar os efeitos inerciais.

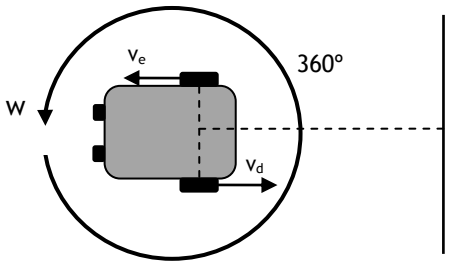


Figura 6.3 – Movimento circular para determinar a distância entre rodas

Após 10 voltas completas registou-se o valor total dos impulsos medidos pelos *encoders* obtendo-se 2150535 e 2150551 para a roda direita e para a roda esquerda, respectivamente.

Através da equação 6.8 obtém-se uma distância entre rodas, b, de 59,38 cm.

Na tabela 6.1 representam-se os parâmetros obtidos através da calibração da odometria que permitirão obter uma estimativa do estado do robô.

Tabela 6.1 – Parâmetros da odometria obtidos para a plataforma em estudo

Diâmetro das rodas (D_n)	12,16 cm
Distância entre rodas (b)	59,38 cm
Resolução dos *encoders*	44032 impulsos/volta

As calibrações descritas acima não garantem uma elevada precisão na determinação dos parâmetros do sistema de odometria, pois os métodos utilizados envolvem a aplicação de sensores (*encoders*) que, normalmente, fornecem medidas com erros relativos à resolução dos *encoders*.

Para além dos erros introduzidos por sensores, os erros sistemáticos específicos do robô também prejudicam na obtenção de uma precisão elevada na calibração. Estes podem ser de diferentes origens, tais como tempos de amostragem variáveis, variação do ponto de contacto com o piso, variação da distância entre rodas.

Assim, para uma melhor calibração e, consequentemente, para a obtenção de uma melhor precisão na estimativa da posição e orientação do robô, deve-se proceder à utilização de métodos de calibração mais eficientes de forma a identificar e compensar os erros sistemáticos.

K. S. Chong e L. Kleeman [21] apresentam um método sistemático de calibração do sistema de odometria de forma a compensar o efeito dos erros sistemáticos designado por *UMBMark* e identificam três fontes dominantes de erros sistemáticos: o erro devido ao diâmetro das rodas diferentes, E_d; o erro devido à incerteza da distância entre os pontos de contacto das rodas com o piso, E_b; e o erro de escala, E_s.

O método de calibração *UMBMark* permite identificar separadamente o erro E_d e E_b e, assim, compensar correctamente o efeito que cada um provoca na odometria.

Para informações mais detalhadas sobre o método de calibração *UMBMark* consultar a referência [21].

6.3 Sistema de detecção de marcadores - Localização absoluta

A utilização de sistemas de detecção de marcadores artificiais baseados em visão artificial permite a estimação da posição absoluta do robô, garantindo desta forma uma melhor precisão na localização do *AGV*.

Dotar um robô de visão é uma tarefa ambiciosa que implica a capacidade de gerir múltiplas tarefas simultaneamente, ou seja, a capacidade de processar dados de sensores exteriores e fundi-los de forma a se mover autonomamente. É de uma forma semelhante o que o cérebro de um ser humano faz quando recebe a informação sensorial dos olhos e depois a transmite aos membros inferiores encarregues pela locomoção.

A visão num robô aumenta significativamente as suas capacidades sensoriais e consequentemente operacionais, bem como a sua versatilidade e segurança de funcionamento. Pretende-se com a visão robótica poder determinar as posições dos objectos (*landmarks*), assim como as relações entre os mesmos e o ambiente que os cerca, e assim poder realizar tarefas ou "missões", como navegar entre os objectos e detectá-los num determinado percurso predefinido.

6.3.1 Visão Artificial

A visão artificial consiste na aquisição e processamento da imagem obtida através da câmara. A aquisição é realizada através do sistema autónomo de visão artificial e o processamento executado através do *software Sherlock*. Fisicamente, a aquisição da imagem é obtida através da utilização de um espelho posicionado a 45º em relação à câmara. Isto deve-se ao facto da câmara se encontrar instalada no interior do *AGV* e paralela ao plano do objecto (marcador). O esquema da aquisição da imagem pode ser observado na figura 6.4.

Deste modo, durante o movimento do robô sempre que é detectado um objecto idêntico ao objecto utilizado como amostra, é despoletado um conjunto de operações que visam o processamento da imagem obtida em tempo real. Após o término do algoritmo inerente a este processo e quando um marcador é identificado, é estabelecida a comunicação entre o subsistema de processamento de imagem e o subsistema da aplicação de controlo e decisão. É de salientar que a comunicação entre estas duas aplicações é unidireccional, pois apenas o subsistema responsável pela aquisição e tratamento da imagem envia informação (dados) para o subsistema da aplicação de controlo e decisão. A comunicação é realizada através do envio de uma mensagem por *TCP* e após algum processamento é possível actualizar a localização do *AGV*. A mensagem enviada contém os dados das medições da posição, do ângulo do marcador em relação ao seu referencial de base e da semelhança entre o marcador detectado e o marcador utilizado como amostra. Após a identificação do marcador visual, a posição global do *AGV* torna-se de alguma forma conhecida. Este facto só acontece quando o *AGV* se movimenta relativamente perto do marcador.

A imagem capturada pela câmara é apresentada na figura 6.5. Pode-se observar que o marcador apresenta um elevado contraste em relação ao fundo (*background*) o que facilitará a sua identificação e detecção.

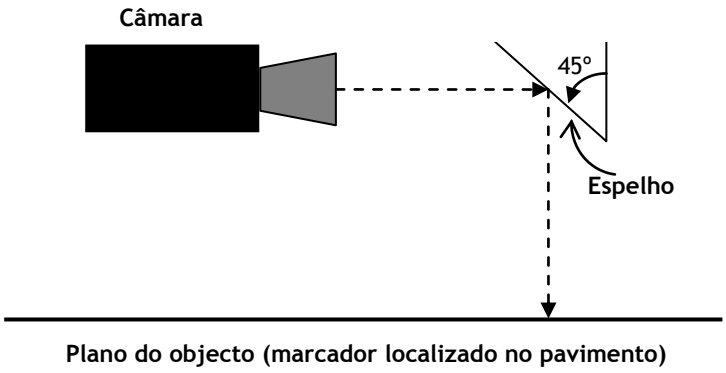


Figura 6.4 – Esquema representativo da aquisição da imagem pela câmara

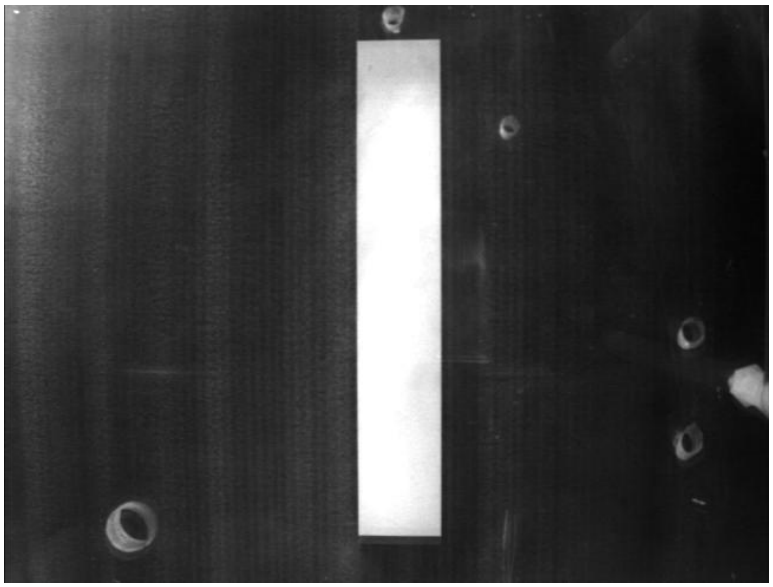


Figura 6.5 – Imagem original capturada pela câmara instalada no robô

6.3.1.1 Calibração da câmara

A calibração de uma câmara refere-se ao estabelecimento de uma transformação de coordenadas entre as coordenadas de pixel da câmara e um sistema de coordenadas real. Isto permite medições precisas a serem realizadas em unidades de real a partir de qualquer janela de imagem.

O software utilizado para o processamento da imagem apresenta três métodos de calibração: Pontos, Escala e Grelha *(Grid)* [39]. Para o presente trabalho, recorreu-se ao método *Grid* que utiliza uma imagem em grelha ou uma imagem idêntica a um tabuleiro de xadrez, para estabelecer um mapeamento de quatro pontos. Este método baseia-se na suposição de que o tabuleiro ou grelha é constituído por (apenas) linhas perpendiculares e paralelas que definem células quadradas. Definindo a dimensão das células quadradas em coordenadas mundo é possível, após a calibração da imagem, associar a dimensão de um pixel a uma unidade das coordenadas reais. Uma vez localizados os quatro pontos utilizados para o mapeamento, a transformação de coordenadas encontra-se estabelecida. Este método é composto por um método de perspectiva radial que assume as células como uniformes em tamanho e executa correcções de distância não-lineares a partir de um ponto central.

Todos os pontos e linhas são representados no sistema de coordenadas da imagem para que foram definidos. Desta forma, após a calibração da imagem, as dimensões passam a ser lidas em unidades reais, os ângulos são estabelecidos em relação ao eixo X do mundo real, e até mesmo no caso da existência de círculos estes seriam computados como ovais, devido à distorção de perspectiva. Neste caso concreto, a imagem captada apresenta uma distorção em barril provocada pela lente utilizada.

Na figura 6.6 apresenta-se o método de calibração utilizado para calibrar as imagens captadas pela câmara.

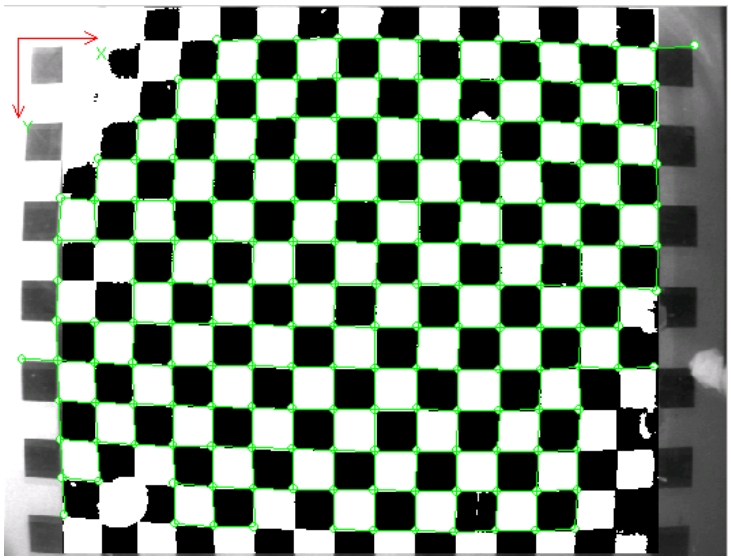


Figura 6.6 – Imagem utilizada para método de calibração

6.3.1.2 Análise/Processamento da imagem

O *software* utilizado para a análise da imagem captada pela câmara é uma aplicação *user-friendly* onde os algoritmos utilizados para o pré-processamento e posteriormente para a segmentação são escolhidos a partir de uma lista, conforme a finalidade pretendida para o tratamento da imagem.

Inicialmente, estabelece-se uma região de interesse na imagem (*ROI*). Estas são zonas onde o objecto escolhido tem uma probabilidade relevante de ser encontrado. Na figura 6.7 pode-se visualizar esta região delimitada por um rectângulo verde.

A criação de uma região de interesse permite processar apenas os pixéis que se encontram no seu interior e nas margens da área delimitada. Desta forma, uma vez que o algoritmo não necessita de percorrer todos os pixéis da imagem captada, o processo de detectar um objecto na *ROI* realiza-se de uma forma mais célere.

No pré-processamento é realizada a atenuação dos ruídos e outros factores, como a luminosidade do ambiente, com o objectivo de se obter a informação necessária da imagem utilizando os algoritmos específicos de visão. Desta forma, recorreu-se à transformação da imagem monocromática em imagem binária, aplicando um nível de *threshold* de 100 numa escala de 0 a 255. Logo, todos os pixéis que constituem a *ROI* e que se encontram acima do nível estabelecido adoptam o valor de 1 (branco), e os que se encontram sob o nível estabelecido assumem o valor de 0 (preto), figura 6.8.

Posteriormente, para a detecção do marcador utilizou-se um algoritmo de pesquisa de figuras geométricas, *Search*, que consiste na procura de uma determinada estrutura na imagem. Inicialmente, o algoritmo passa por uma fase de treino onde se introduz o marcador de forma a este algoritmo treinar a forma e tamanho do objecto a identificar, tornando-o numa amostra que servirá de referência para a identificação e detecção dos marcador visuais que serão obtidos quando o robô se encontrar a funcionar em tempo real. Após a realização do treino do objecto, o tratamento da imagem encontra-se pronto a detectar os marcadores.

A utilização do algoritmo de detecção de formas geométricas possibilita estimar a orientação do marcador e a distância do mesmo em relação à amostra.

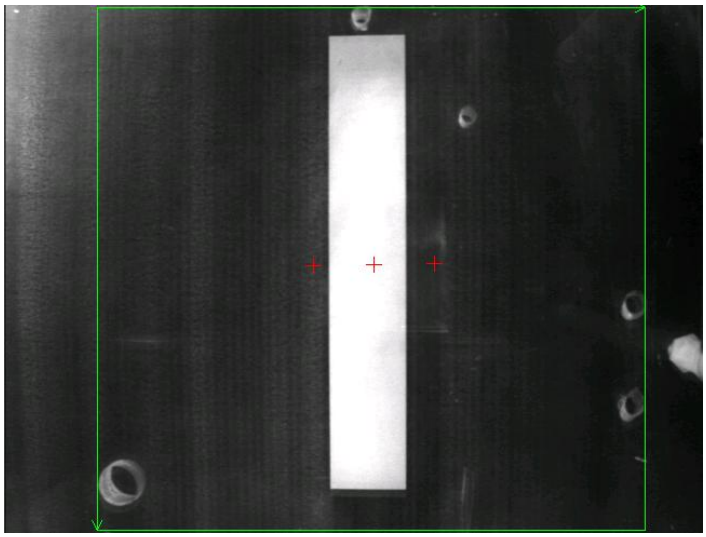


Figura 6.7 – Imagem original delimitada a verde pela *ROI*

O resultado pós-processamento é mostrado na figura 6.8.

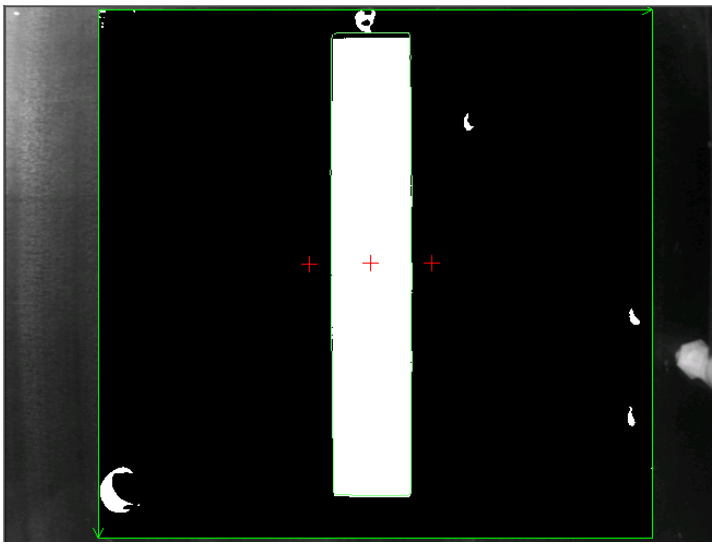


Figura 6.8 – Imagem após pré-processamento e segmentação

6.3.2 Formato das mensagens enviadas por *TCP*

Na tabela 6.2 apresenta-se o formato das mensagens enviadas do sistema autónomo de visão artificial para a aplicação de controlo e decisão.

O primeiro parâmetro corresponde à semelhança entre o marcador detectado e a amostra responsável pelo teste de similaridade. Os restantes parâmetros correspondem ao ponto do centro de massa do objecto e do ângulo relativamente ao objecto idêntico utilizado como amostra.

Tabela 6.2 – Formato da mensagem enviada por *TCP*

Pattern_Similarity	Position_X	Position_Y	Position_Theta

6.4 Controlo e Planeamento de Trajectórias

O controlo e o planeamento de trajectórias para um robô consiste na determinação de um percurso que permita movimentá-lo desde uma origem, localização inicial, até a um destino, alvo ou meta. Esta movimentação deve garantir que o robô não colida com nenhum obstáculo do ambiente e que o movimento planeado seja consistente com as restrições da cinemática do veículo.

No planeamento de trajectórias são tidas em consideração as habilidades do robô e a estrutura do ambiente. Desta forma, são assumidos os seguintes pressupostos:

- Robô rígido modelado como um ponto;
- Ambiente estático e conhecido;
- Movimentos do robô segundo segmentos rectilíneos ou curvilíneos.

Com o objectivo de permitir ao robô movimentar-se num ambiente semi-estruturado, definiram-se alguns procedimentos de baixo nível, que reunidos possibilitam o desempenho de uma tarefa pelo robô, como percorrer uma trajectória de um ponto inicial para um ponto final. Por sua vez, a definição das trajectórias do robô é implementada por um procedimento de mais alto nível responsável por garantir que o robô desempenha correctamente o movimento para o qual foi proposto.

Deste modo, na secção 6.4.1 são descritas em pormenor as rotinas básicas que permitem o controlo do robô. A secção 6.4.2 descreve o procedimento de mais alto nível que se baseia nas rotinas mais simples, com o objectivo de percorrer uma determinada trajectória.

6.4.1 Procedimentos de baixo nível

Para o procedimento de baixo nível foram desenvolvidas as seguintes rotinas básicas com o intuito de permitir o controlo do movimento do robô:

- Goto_Target (Target_X,Target_Y,V);
- Follow_Line (X_1, Y_1, X_2, Y_2, V);
- Follow_Circle (X_c, Y_c, Raio, V, sentido);
- Stop.

A implementação de diferentes procedimentos de baixo nível possibilita alterar os percursos do robô em qualquer momento, sendo apenas necessário definir o tipo de percurso que se pretende aplicar ao movimento do robô.

6.4.1.1 Goto_Target

Esta função tem como objectivo gerar os valores de velocidade linear e velocidade angular (V, W) a enviar para os motores, que possibilitam ao robô atingir um determinado ponto (target) definidos como parâmetros de entrada do procedimento.

Na figura 6.9 representa-se esquematicamente o procedimento abordado nesta secção.

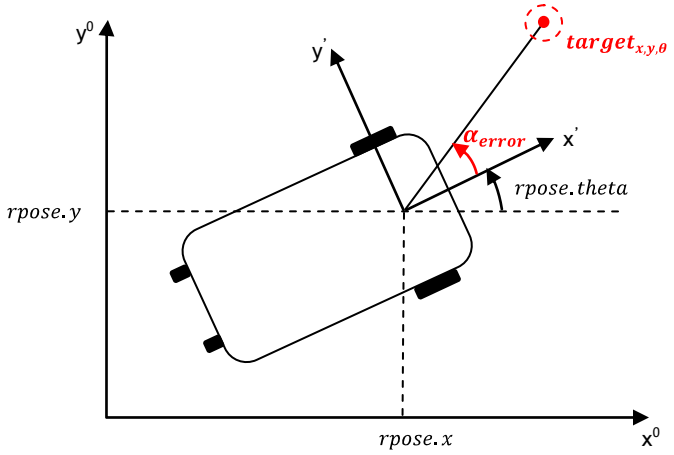


Figura 6.9 – Esquema representativo do procedimento *Goto_Target*

Este procedimento, representado na figura 6.10, pode ser visto como uma máquina de estados que percorre os vários estados com base na posição do robô, verificando as condições e actuando nos motores do robô com o intuito de atingir o objectivo final, o ponto definido por $target_{x,\ y,\ \theta}$.

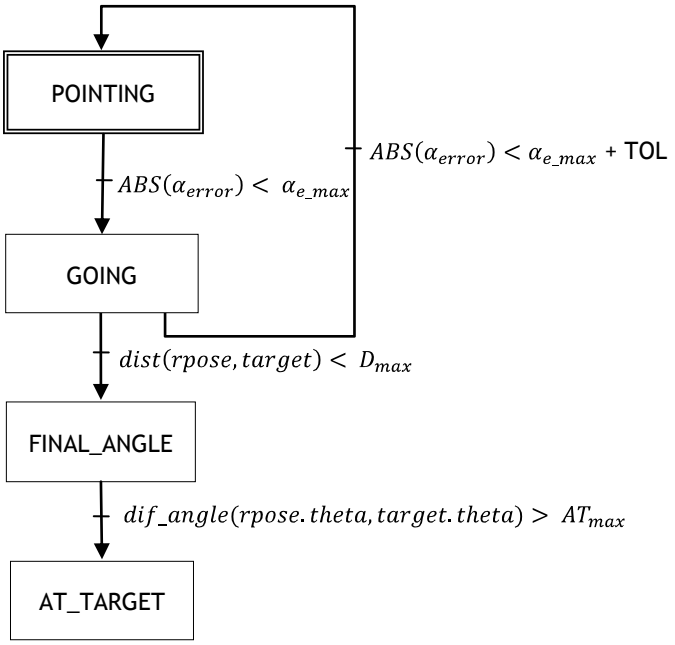


Figura 6.10 – Máquina de estados do procedimento *Goto_Target*

A variável α_{error}, apresentada na figura 6.10, é definida por

$$\alpha_{error} = dif_angle(arctan2((target.y - rpose.y), (target.x - rpose.x)), rpose.theta). \quad \textbf{(6.9)}$$

6.4.1.2 Follow_Line

Este procedimento tem como objectivo gerar os valores de velocidade linear e velocidade angular (V, W) a enviar para os motores que conduzam o robô a seguir um vector definido pelo par de coordenadas (x_1^0, y_1^0), (x_2^0, y_2^0) no sentido do ponto 1 para o ponto 2, e que são parâmetros de entrada deste procedimento. Na figura 6.11 apresenta-se esquematicamente o significado deste procedimento.

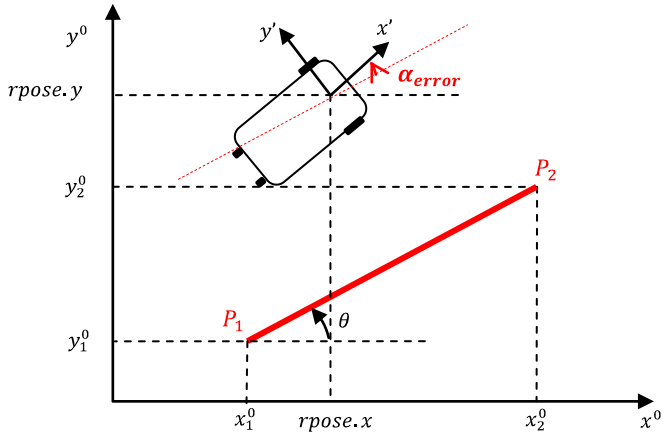


Figura 6.11 – Representação esquemática do procedimento *Follow_Line*

Para o procedimento *Follow_Line*, implementou-se uma máquina de estados, representada na figura 6.12, que permite aproximar o robô à linha formada pelos pontos (x_1^0, y_1^0) e (x_2^0, y_2^0) e, posteriormente, segui-la. Desta forma, a máquina de estados é composta por dois estados: *POINTING* e *GOING*. No estado *POINTING* ajusta-se o controlador de velocidade, actuando na velocidade de rotação (W). Inicialmente determina-se a distância do robô à linha, aplicando posteriormente uma velocidade (W) proporcional a essa distância. Se o robô se encontrar a uma distância muito elevada executa-se primeiro uma trajectória em direcção ao ponto mais próximo da linha. O estado *GOING* é responsável por aplicar aos motores as velocidades V e W que permitam seguir o vector definido por P_1 e P_2 (Anexo A).

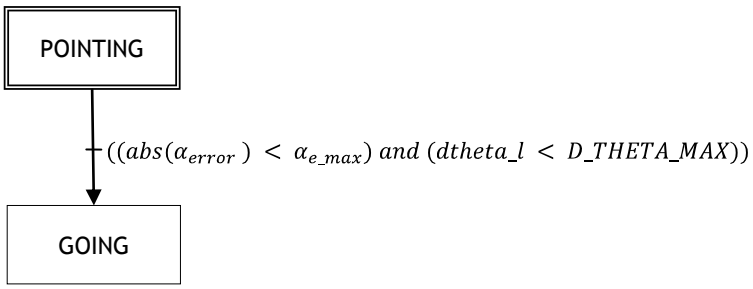


Figura 6.12 – Máquina de estados do procedimento *Follow_Line*

Na figura 6.12, o parâmetro α_{error} é dado por

$$\alpha_{error} = dif_angle(line_pointing_angle(x1, y1, x2, y2), rpose.theta). \quad \textbf{(6.10)}$$

6.4.1.3 Follow_Circle

O procedimento *Follow_Circle* (X_c, Y_c, *Raio, V, sentido)* tem como objectivo gerar os valores de velocidade tangencial e velocidade angular (V, W) a enviar para os motores que permitem ao robô seguir a trajectória de um circulo com um determinado raio, centrada no ponto (X_c,Y_c) definidos como parâmetros de entrada deste processo. Esta trajectória também permite estipular o sentido de movimentação. Assim, assume-se o sentido horário para "sentido = 1" e anti-horário no outro caso.

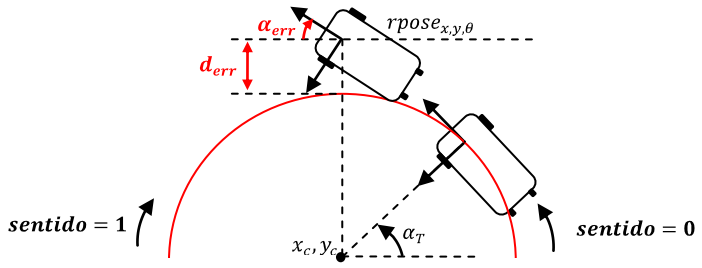


Figura 6.13 – Esquema representativo do procedimento *Follow_Circle*

O processo básico de seguir um círculo também foi implementado baseado numa máquina de estados, figura 6.14.

Desta forma, a implementação do procedimento *Follow_Circle* seguiu a mesma metodologia indicada para a acção *Follow_Line*. Este processo consiste no ajuste da velocidade de rotação do robô em função da velocidade linear deste e do raio de curvatura.

A máquina de estados contempla dois estados: *GOING* e *AT_TARGET*. O estado *GOING* verifica se o movimento é efectuado no sentido horário ou anti-horário e, se o robô se encontrar distanciado da trajectória circular, procede inicialmente à aproximação à trajectória. Quando o robô se encontra dentro da trajectória é despoletado o estado AT_TARGET que é responsável por garantir que o robô realiza a trajectória requerida, actuando nas velocidades V e W.

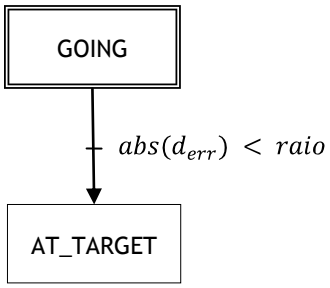


Figura 6.14 – Máquina de estados para o procedimento *Follow_Circle*

A variável d_err representa a distância entre o robô e a trajectória em círculo. Esta pode ser obtida através de

$$d_{err} = abs(dist(rpose, centro)) - raio. \quad \textbf{(6.11)}$$

6.4.1.4 Stop

Esta função tem como objectivo gerar os valores de velocidade tangencial e velocidade angular a enviar para os motores (V, W) que conduzam o robô a permanecer imóvel.

6.4.2 Procedimento de mais alto nível

Para atingir o objectivo global, ou seja, percorrer percursos através de uma trajectória previamente definida, assumiu-se que o robô móvel se encontrava inserido numa linha de produção com um armazém industrial e dois postos de trabalho. Estabeleceu-se para dois cantos do espaço utilizável cada um dos postos de trabalho e o armazém automático num outro canto. O quarto canto livre admitiu-se como o posto de carga do robô. Os corredores em que o robô se pode deslocar formam um "H" que une os 4 locais mencionados. Na figura 6.15 apresenta-se o ambiente, bem como a definição dos sentidos ao longo dos diferentes percursos pelo qual o robô se poderá movimentar.

Para permitir que o robô se desloque pelos corredores em "H", implementou-se um procedimento de mais alto nível (*Follow_Path*) que utiliza sequencialmente (máquina de estados), a função mais simples *Goto_Target*.

A máquina de estados que permite ao robô movimentar-se entre pontos numa trajectória em "H" é apresentada na figura 6.16. Os pontos estabelecidos representam percursos a efectuar pelo robô que em conjunto formam a trajectória a realizar, com a finalidade de atingir o objectivo global, ou seja, a posição pretendida.

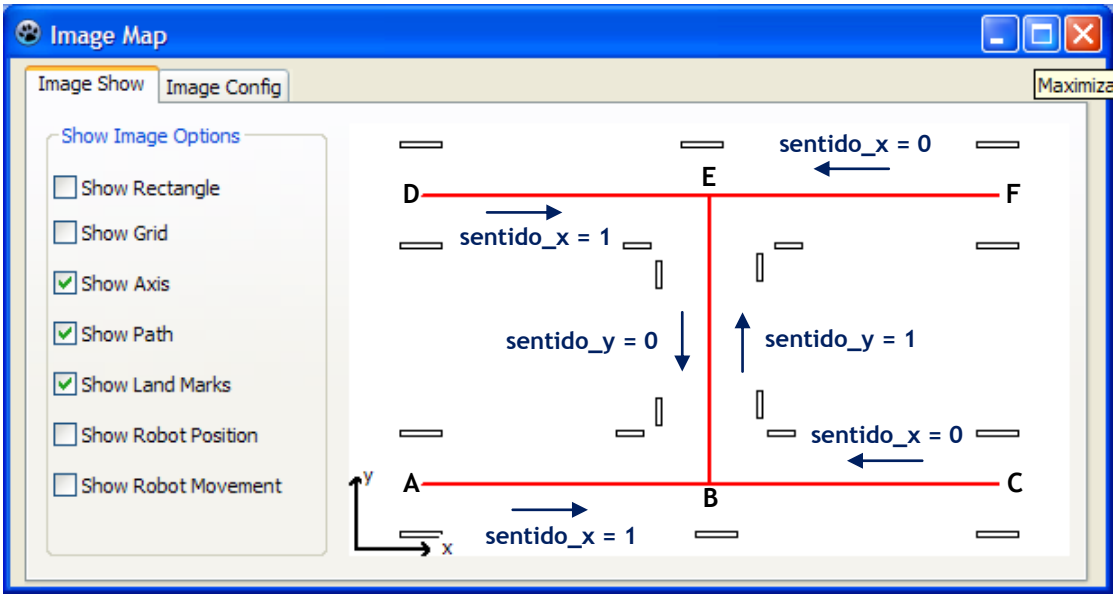


Figura 6.15 – Imagem representativa dos percursos e sentidos a realizar pelo robô ao longo da trajectória em "H"

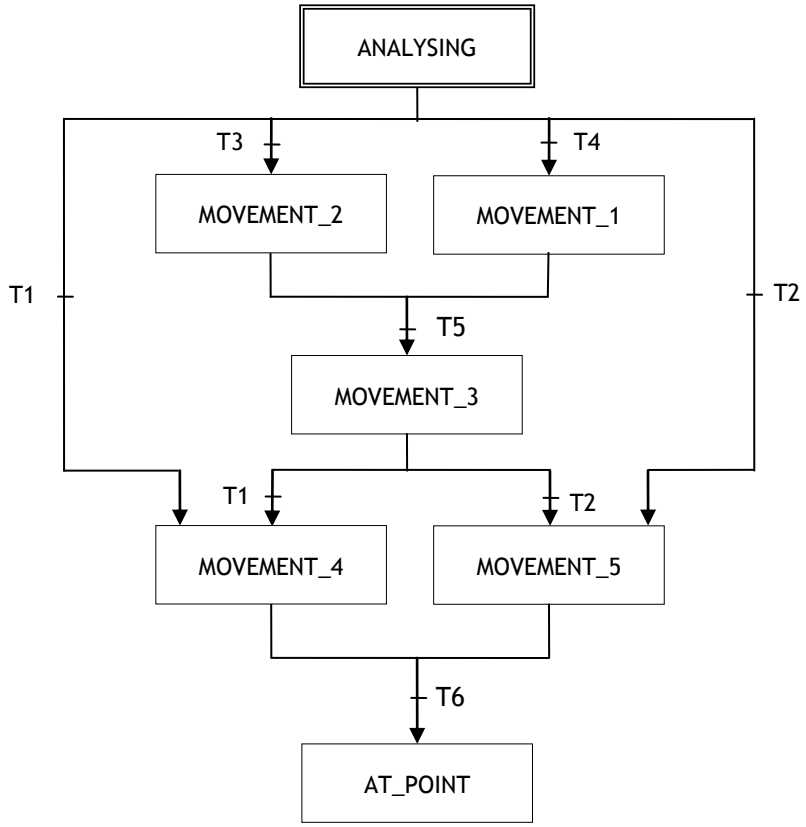


Figura 6.16 – Máquina de estados para o procedimento *Follow_Path*

Tabela 6.3 – Transições da máquina de estados do procedimento *Follow_Path*

Transição	Condição
T1	$rpose.y = target.y\ and\ sentido_x = 0$
T2	$rpose.y = target.y\ and\ sentido_x = 1$
T3	$rpose.y \neq target.y\ and\ sentido_x = 0$
T4	$rpose.y \neq target.y\ and\ sentido_x = 1$
T5	$((rpose.x \geq length/2\ and\ sentido_x = 1)\ or (rpose.x \leq length/2\ and\ sentido_x = 0))$
T6	$((rpose.x \geq target.x\ and\ sentido_x = 1)\ or\ (rpose.x \leq target.x\ and\ sentido_x = 0))$

Os estados definidos na figura 6.16 descrevem processos na atribuição de diferentes posições para cumprir com um determinado requisito no seguimento de uma trajectória. Na tabela 6.4 apresentam-se os diferentes estados que formam o procedimento *Follow_Path*, assim como os percursos realizados pelo robô em cada estado.

Tabela 6.4 – Percursos efectuados em cada estado do procedimento *Follow_Path*

Estado	Percurso
MOVEMENT_1	A -> B ou D -> E
MOVEMENT_2	F -> E ou C -> B
MOVEMENT_3	B -> E ou E -> B
MOVEMENT_4	C -> B -> A ou F -> E -> D ou B -> A ou E -> D
MOVEMENT_5	A -> B -> C ou D -> E -> F ou B -> C ou E -> F
AT_POINT	A ou B ou C ou D ou E ou F

Segue-se uma breve explicação dos estados que compõem este procedimento. Assim, no estado *ANALYSING* verifica-se o sentido do deslocamento do robô em relação ao referencial global. Desta forma, analisa-se se o robô se deslocará no sentido positivo ou negativo do eixo dos *xx*, *sentido_x*, ou no sentido positivo ou negativo do eixo dos *yy*, *sentido_y*, a partir da diferença entre a posição actual do robô, *RPose*, e a posição de destino, *target*. Posteriormente são encadeados sequencialmente os estados necessários para que o robô atinja a meta pretendida.

Desta forma, e a título de exemplo, descreve-se de seguida as acções realizadas no estado *MOVEMENT_1* da figura 6.16, sendo este despoletado quando se dá a confirmação da transição *T4* pelo robô. As acções que compõem este estado são direccionadas para a realização de um movimento que em conjunto com outros terá como finalidade atingir um objectivo global. Para este estado concreto, a condição de transição *T4* estabelece que a posição da ordenada do robô, *RPose.y*, difere da ordenada da posição final, *target.y*. Logo, o robô terá que estabelecer para a rotina básica *Goto_Target* um destino que, neste caso, será o ponto intermédio da trajectória em "H", sendo este o único ponto estipulado que permite ao robô transitar de posição segundo o eixo *y*, ou seja, movimentar-se segundo o percurso de B para E. O estado *MOVEMENT_1* apresenta-se como um estado intermédio de transição na realização de um dado percurso.

6.4.2.1 Exemplo de um possível percurso

A título de exemplo apresenta-se na figura 6.17 os estados a percorrer no procedimento *Follow_Path* quando se pretende movimentar o robô de A para F. Neste caso, assume-se que o ponto A é a base do robô (posto de carga) e o ponto F um posto de trabalho.

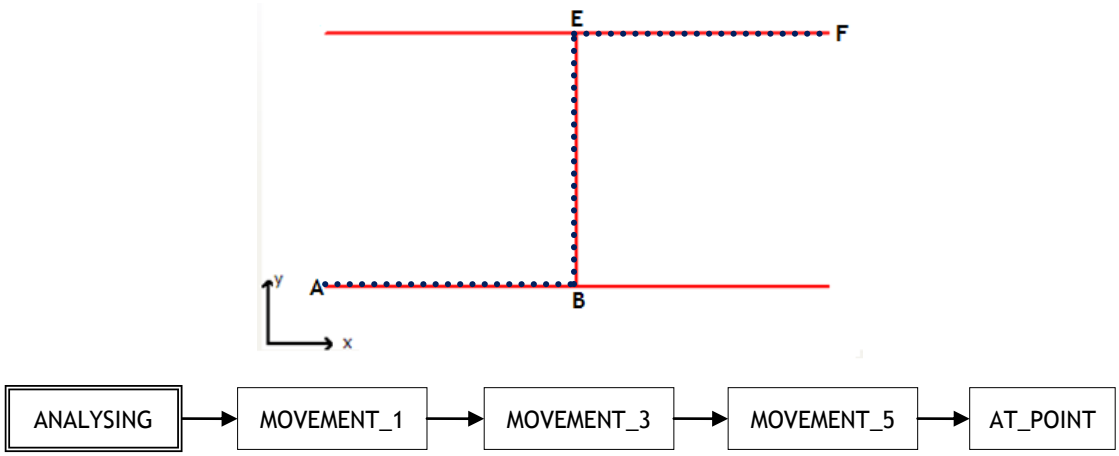


Figura 6.17 – Exemplo de um percurso a realizar pelo robô

6.5 Algoritmo de localização absoluta

O algoritmo de localização desenvolvido para este estudo tem como finalidade permitir corrigir o estado do robô. Desta forma, implementou-se um procedimento através de uma máquina de estados que permite localizar marcadores e, quando necessário, corrigir o estado do robô. Como referido anteriormente, é de salientar que o algoritmo de localização utilizado

no simulador *SimTwo* para a detecção de paredes laterais é similar ao utilizado pelo robô real para a detecção de marcadores por visão artificial, permitindo assim utilizar o algoritmo de uma forma transparente em ambas as situações.

Na simulação a localização absoluta encarrega-se de avaliar a distância entre o robô e uma parede lateral mensurada por sensores de infravermelhos. Na situação real, a localização absoluta verifica a distância e a orientação do robô em relação a um referencial de base estabelecido para a posição dos marcadores artificiais.

Na secção 6.5.1 é descrito o modelo matemático para detecção de linhas e na secção 6.5.2 apresenta-se a estrutura e a descrição do algoritmo de localização.

6.5.1 Modelo matemático do método de auto-localização por linhas

A utilização do método de auto-localização por linhas tem como finalidade localizar o robô no espaço onde se encontra. Para este método, utilizam-se duas abordagens em relação à detecção de marcadores. Desta forma, para a simulação, a detecção de marcadores baseia-se na medição da distância nas laterais do robô através de dois sensores, sensores de infravermelhos. A partir da informação obtida por cada sensor é possível determinar a orientação do robô e a sua distância, em relação à parede. Para o robô real, a detecção de marcadores baseia-se em visão artificial, utilizando o sistema autónomo de visão artificial para estimar o estado do robô através dos marcadores artificiais previamente colocados no ambiente.

É de salientar que este método apenas permite definir a coordenada na direcção perpendicular à recta [33]. Na figura 6.18 e 6.19 apresenta-se a descrição do método de auto-localização por linhas.

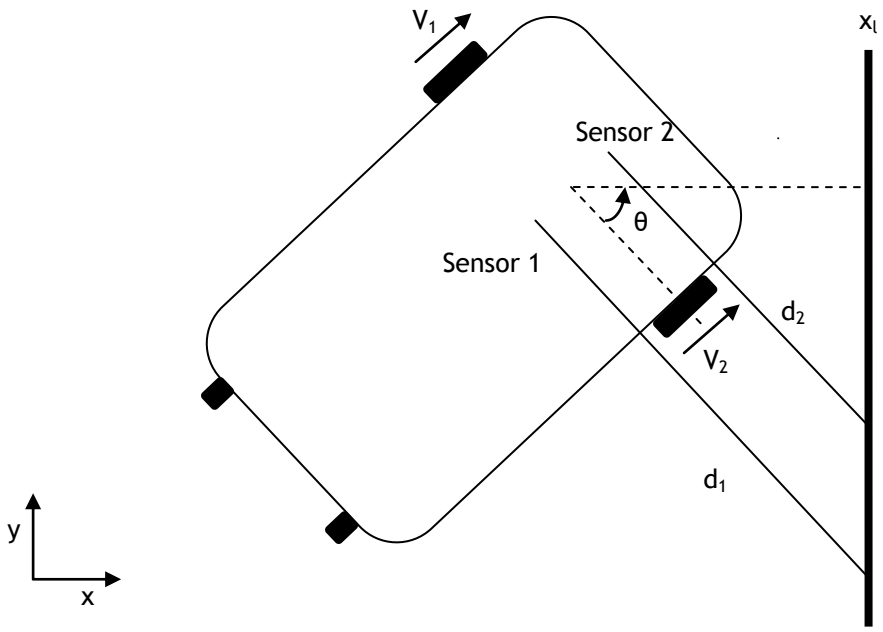


Figura 6.18 – Esquema representativo da medição da distância do robô a uma parede lateral

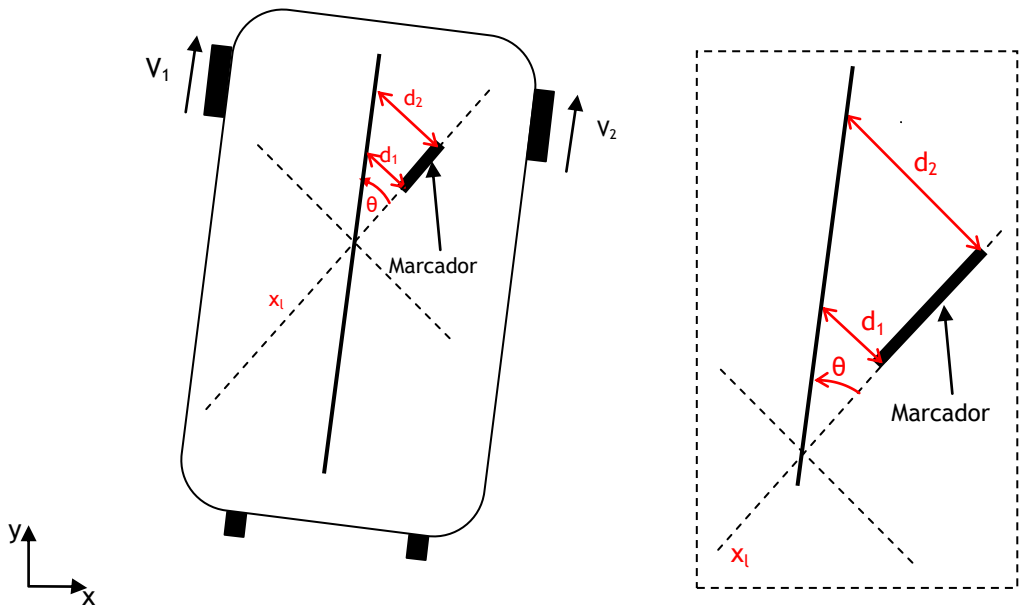


Figura 6.19 – Esquema representativo da medição da distância do robô a um marcador artificial

Assim, a orientação (θ) do robô à parede é dada por

$$\theta = \arctan\left(\frac{d_1 - d_2}{2L_S}\right), \tag{6.12}$$

em que $2L_S$ é a distância entre os dois sensores ou entre os dois pontos das extremidades do marcador.

A distância (x) do robô à parede obtém-se a partir de

$$x = x_l - \left(\frac{d_1 + d_2}{2}\right)\cos(\theta). \tag{6.13}$$

Na detecção do marcador artificial o sistema de visão utilizado para este trabalho já fornece o ângulo de orientação do robô e a sua distância ao marcador não sendo então necessário calcular θ e x a partir de d_1 e d_2.

6.5.2 Estrutura e descrição do algoritmo de localização

Nesta secção apresenta-se o diagrama de estados do processo de localização absoluta (figura 6.20). Para cada estado será fornecida uma descrição das tarefas efectuadas.

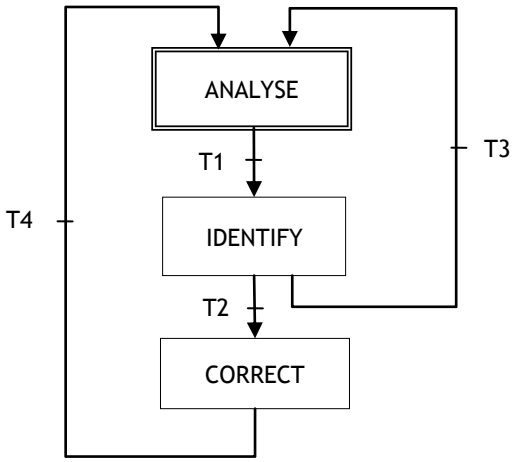


Figura 6.20 – Diagrama de estados do processo de localização

Tabela 6.5 – Transições da máquina de estados do processo de localização

Transição	Condição *SimTwo*	Condição Visão Artificial
T1	$\text{Sensor1} \leq \text{dist1 and Sensor2} \leq \text{dist2}$	$\text{pattern_similarity} \geq \text{value}$
T2	$\text{distRobot2LandMark} \leq \text{dist_max}$	
T3	$\text{distRobot2LandMark} > \text{dist_max}$	
T4	$\text{correctPose} = \text{TRUE}$	

O estado *ANALYSE* representa o primeiro estado do procedimento *Udpate_LandMarks*. Neste estado o robô processa a trama recebida pelo *SimTwo,* caso se esteja a simular o robô, ou pelo sistema autónomo de visão artificial, caso se esteja na situação de funcionamento real. A condição para transição de estado (*T1*) verifica se o robô está perante algum marcador, avaliando um parâmetro facultado pelo sistema autónomo de visão artificial ou pelo *SimTwo, pattern_similarity* ou *sensor1* e *sensor2*, respectivamente. Este parâmetro no caso de ser facultado pelo *SimTwo* representa a distância medida por cada sensor de infravermelhos e no caso do funcionamento real do robô representa a semelhança do marcador detectado com o marcador estabelecido como referência de marcadores. Assim, se este parâmetro se encontrar dentro de uma gama previamente estabelecida, a condição de transição (*T1*) para o estado seguinte é assegurada encontrando-se o robô na presença de um marcador ou obstáculo.

No estado *IDENTIFY* efectua-se uma pesquisa à base dados criada com os marcadores existentes no ambiente do robô (Anexo B), avaliando através da posição e orientação do robô e da posição e orientação dos marcadores artificiais se o objecto detectado é realmente um marcador com o objectivo de se proceder à correcção do estado do robô. Desta forma, durante o ciclo de pesquisa efectua-se o cálculo da distância entre a posição do robô e a posição dos diferentes marcadores existentes na base de dados, estimando através desta distância qual dos marcadores se encontra mais próximo do robô. Esta distância mínima final é obtida recorrendo a um ciclo de *n* iterações, sendo *n* o número de marcadores existentes na base de dados. Assim, na primeira iteração do ciclo efectua-se o cálculo da distância do robô ao primeiro marcador armazenado na base de dados e guarda-se o valor obtido para a distância entre os dois. Na iteração seguinte efectua-se novamente o cálculo da distância do

robô em relação ao marcador seguinte, *n+1*, e compara-se o valor obtido com o guardado anteriormente para a distância. Se a nova distância obtida se apresentar inferior à anterior significa então que o marcador *n+1* se encontra mais próximo do robô do que o marcador *n*. Desta forma, o marcador *n* é descartado e o marcador *n+1* passa a ser uma referência para a possível identificação do objecto. Este processo é realizado sucessivamente até ao término da pesquisa de marcadores na base de dados, procedendo sempre à actualização da distância mínima entre o robô e o marcador pela forma descrita anteriormente. Após esta pesquisa avalia-se a condição de transição de estado (*T2*), verificando-se se a distância obtida reside dentro de uma gama estabelecida. Se a resposta for positiva então o marcador detectado é o marcador correcto e a posição e orientação do robô será corrigida em função deste marcador, estado *CORRECT*.

O estado *CORRECT* tem como objectivo corrigir a posição e orientação do robô em relação a um dado marcador. Desta forma, através das equações 5.1 e 5.2 procede-se à correcção da posição e orientação do robô.

6.6 Fusão Sensorial

6.6.1 Média Pesada

Um dos mais simples e intuitivos métodos de fusão ao nível do sinal é a média pesada da informação redundante fornecida por um grupo de sensores.

A média pesada da medida de n sensores x_i com pesos $0 \leq w_i \leq 1$ é dada por

$$\bar{x} = \sum_{i=1}^{n} w_i * x_i \tag{6.14}$$

onde, $\sum_{i=1}^{n} w_i = 1$ e $w_i = 0$ se x_i não se encontrar dento de um limite pré-estabelecido.

Os pesos podem ser utilizados para distinguir diferenças de precisão entre sensores, sendo ainda possível utilizar-se uma média "deslizante" para combinar uma sequência de medidas adquiridas por um único sensor, atribuindo-se às mais recentes um maior peso.

6.6.2 Filtro de *Kalman*

Um robô possui a capacidade de se locomover em ambientes estruturados, e reconhecer tanto a sua posição inicial, quando a dos marcadores do sistema. Com base na informação de vários sensores, o filtro de *Kalman* é utilizado para fundir esta informação, permitindo obter uma estimativa global para o estado do sistema, ou seja, uma estimativa da posição real do robô em ambientes discretizados, compensando os erros acumulados obtidos pelas informações sensoriais. Esta estimativa terá elevada qualidade pois a fusão de informação é realizada tomando em consideração a modelização dos erros associados a cada tipo de medidas.

Capítulo 7

Resultados

Neste capítulo são resumidos e apresentados os resultados obtidos no trabalho desenvolvido no âmbito desta dissertação. Devido à escassez de tempo, apenas serão apresentados os resultados obtidos na simulação do comportamento do robô através do *SimTwo*.

7.1 Robô em simulação

Os resultados presentes nesta secção dizem respeito ao movimento efectuado pelo robô de um ponto inicial do mapa, A(0,0) para um ponto final do mesmo, F(8,4). Uma vez que a trajectória do robô foi planeada para corresponder a um percurso em "H", o robô terá que efectuar duas rotações, 90º e -90º, para permitir que este atinja o destino pretendido.

Nas subsecções seguintes apresentam-se os resultados obtidos para este percurso utilizando apenas a localização relativa, subsecção 7.1.1 e 7.1.2, e a localização relativa e absoluta, subsecção 7.1.3, 7.1.4 e 7.1.5. É de salientar que se apresentam resultados tanto para a localização baseada apenas em odometria, como para a localização baseada em odometria e marcadores artificiais, aplicando ruído aos *encoders* com o intuito de verificar o comportamento do robô a estas não linearidades.

7.1.1 Trajectória baseada em localização relativa sem ruído

Primeiramente, pretendeu-se verificar o comportamento do robô, recorrendo apenas à informação provida pelos *encoders*, ou seja, a quantificação da distância percorrida por cada roda é apenas determinada através da mensuração dos impulsos por volta gerados pelos *encoders*. O objectivo passa por observar a trajectória descrita pelo robô comparando-a com a trajectória real observada pelo simulador. Na figura 7.1 apresenta-se os resultados obtidos para esta trajectória, recorrendo somente à localização relativa.

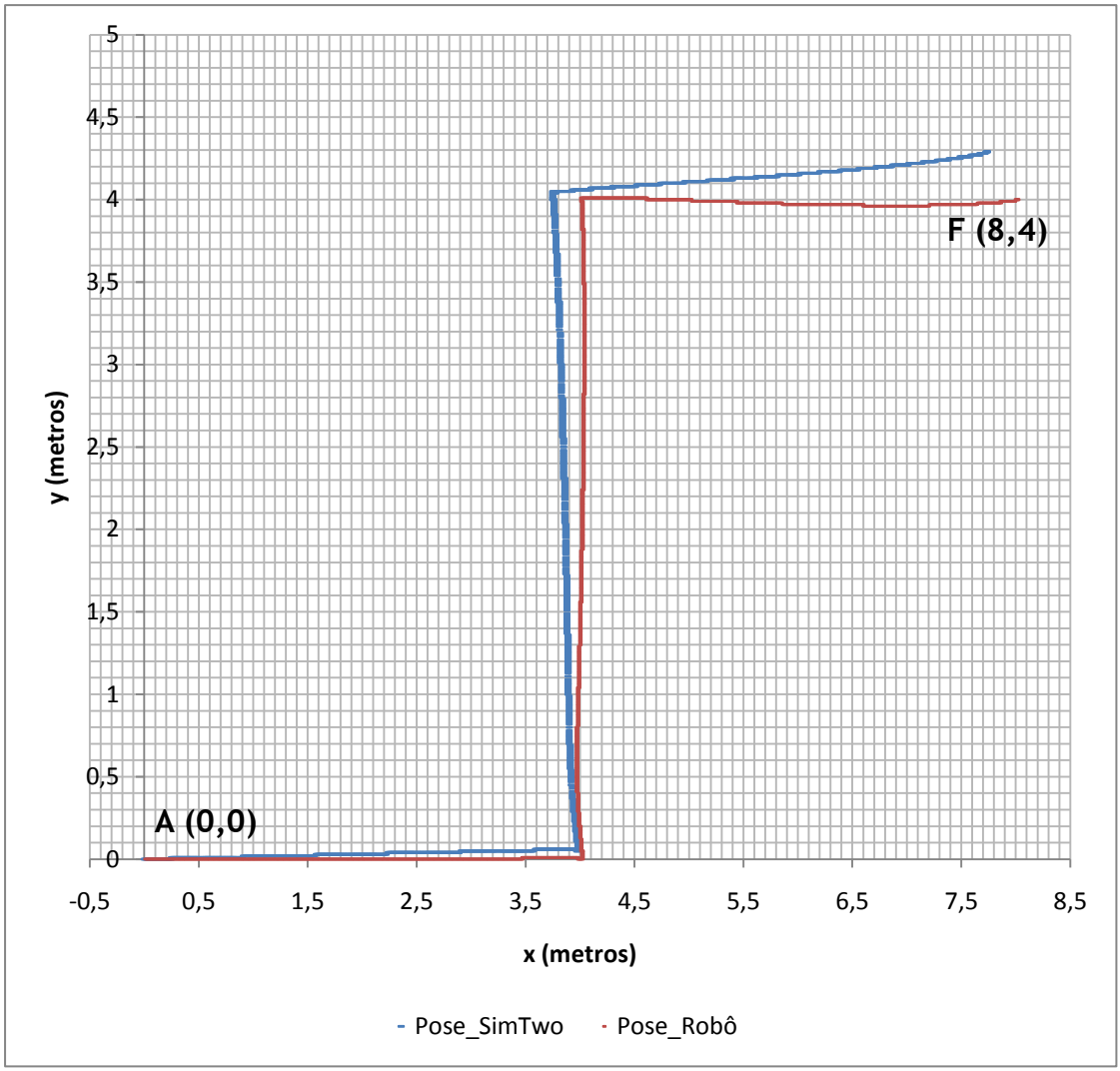


Figura 7.1 – Trajectória baseada em odometria

Através da figura 7.1 é possível constatar que o robô não atinge o destino pretendido, embora a sua localização final não se situe muito distante da observada através do simulador. É de salientar que quando o robô efectua uma rotação de 90º ou -90º este acumula erro na orientação que afectará directamente a sua posição final, contribuindo desta forma para o desvio em relação ao destino pretendido. Este erro na orientação do robô pode dever-se ao facto do simulador não permitir determinar o real ponto de contacto entre as rodas e o piso, afectando a distância entre rodas, *b*, sendo esta utilizada para determinar a orientação do robô. Apesar da posição final obtida através do método de localização baseado em odometria não corresponder à posição final pretendida, é possível observar que a utilização destes sensores em conjunto com outros sensores poderá tornar a posição e orientação do robô bastante fidedigna.

7.1.2 Trajectória baseada em localização relativa com ruído

Uma vez que o simulador utilizado permite introduzir ruído na informação dos *encoders* e, tendo em consideração que este ruído afectará o comportamento do robô, pretendeu-se com a simulação da figura 7.2 observar de que forma o ruído provido pelos *encoders* perturbava a posição e a orientação do robô durante o percurso.

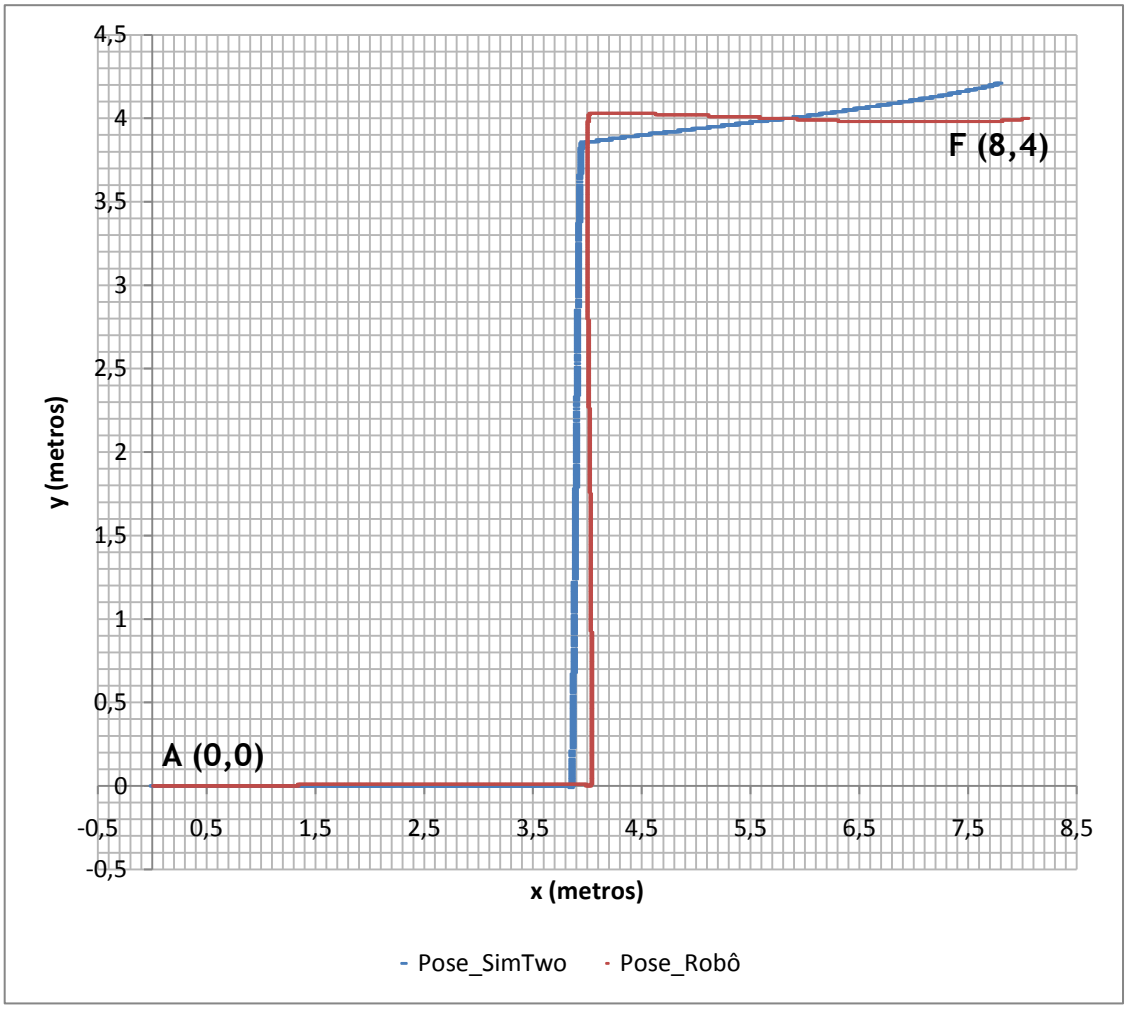


Figura 7.2 – Trajectória baseada em odometria com ruído aplicado aos *encoders*

Através da análise da figura 7.2 é possível verificar que o ruído afecta a quantificação das distâncias medidas pelos *encoders* de cada roda, perturbando o movimento do robô durante o percurso efectuado. Desta forma, a posição final do robô difere da posição observada pelo simulador.

Numa trajectória baseada em odometria sem ruído o erro apenas provém da distância entre rodas que varia devido ao ponto de contacto com o pavimento também variar. Este facto apenas afecta as rotações do robô. Na figura 7.2 constata-se que a presença de ruído também inclui erro na distância percorrida pelo robô.

7.1.3 Trajectória baseada em odometria com correcção da orientação do robô através de marcadores

De forma a aperfeiçoar o estado do robô durante o percurso estipulado, utilizou-se um método de localização absoluta em conjunto com o método baseado em odometria, possibilitando assim corrigir a posição e orientação do robô com o intuito deste atingir a posição final pretendida. Deste modo, introduziram-se marcadores previamente estabelecidos no ambiente do robô, figura 5.9, e facultou-se a posição destes no mapa do robô. Estes marcadores foram estrategicamente colocados nas zonas de rotação do robô para possibilitar a correcção da sua orientação. O método de localização absoluta utilizado é inerentemente intermitente, na medida em que os marcadores se encontram espaçados no ambiente e, geralmente, o robô só consegue reconhecer o marcador quando se encontra bastante próximo deste. Na figura 7.3 apresentam-se os resultados obtidos com a aplicação deste método. É de salientar que primeiramente, apenas se pretendeu corrigir a orientação do robô. Na secção 7.1.4 será apresentado o resultado da utilização deste método para a correcção da posição e orientação do robô.

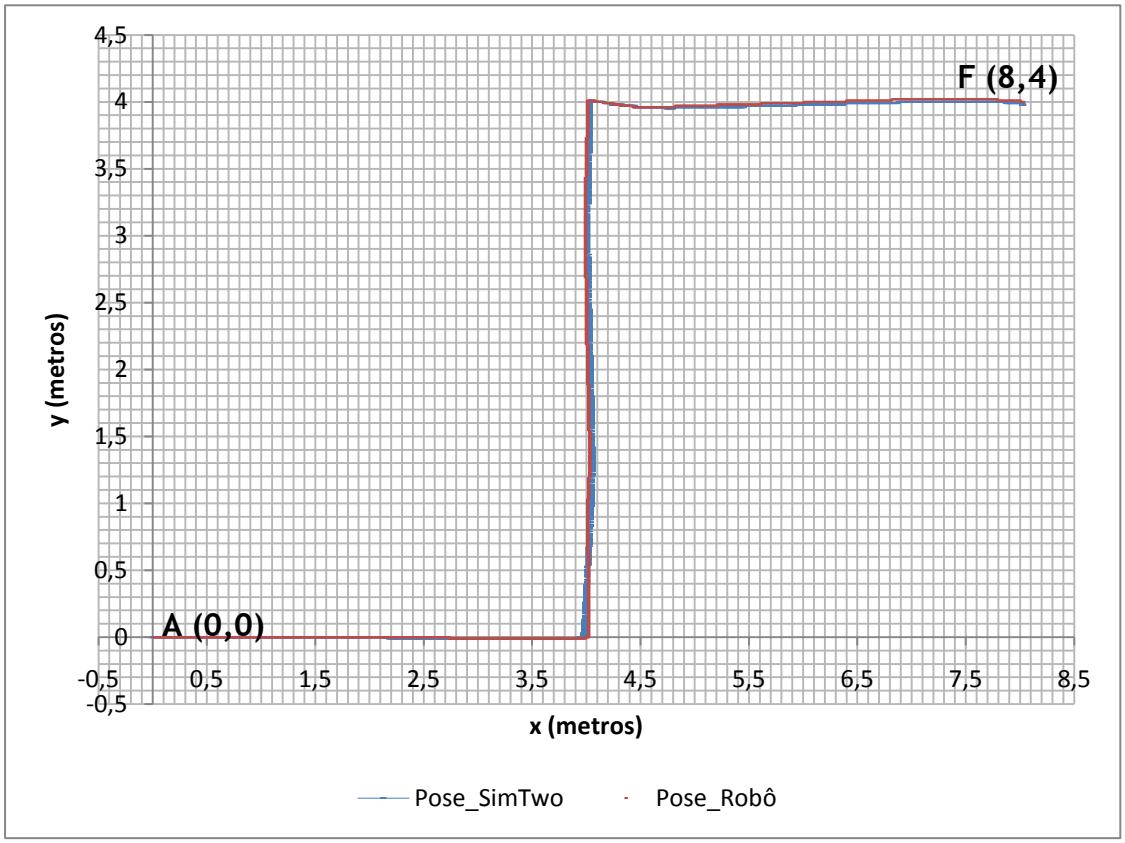


Figura 7.3 – Trajectória baseada em odometria com correcção da orientação do robô através de marcadores

Com base na figura 7.3 é possível verificar que a trajectória descrita pelo robô se aproxima da trajectória observada pelo simulador. Quando o robô detecta um marcador actualiza a sua orientação em relação a este. Assim, observa-se que a posição final obtida para o robô se aproxima da posição pretendida, com um erro na posição do robô inferior a 0,1%.

7.1.4 Trajectória baseada em odometria com correcção do estado do robô através de marcadores

Nesta secção pretende-se apresentar os resultados do comportamento do robô durante a trajectória pretendida, baseando-se na localização a partir de marcadores e da odometria.

Na figura 7.4 exibe-se o resultado obtido para o comportamento do robô a partir da correcção da posição e orientação do mesmo em relação aos marcadores detectados no percurso.

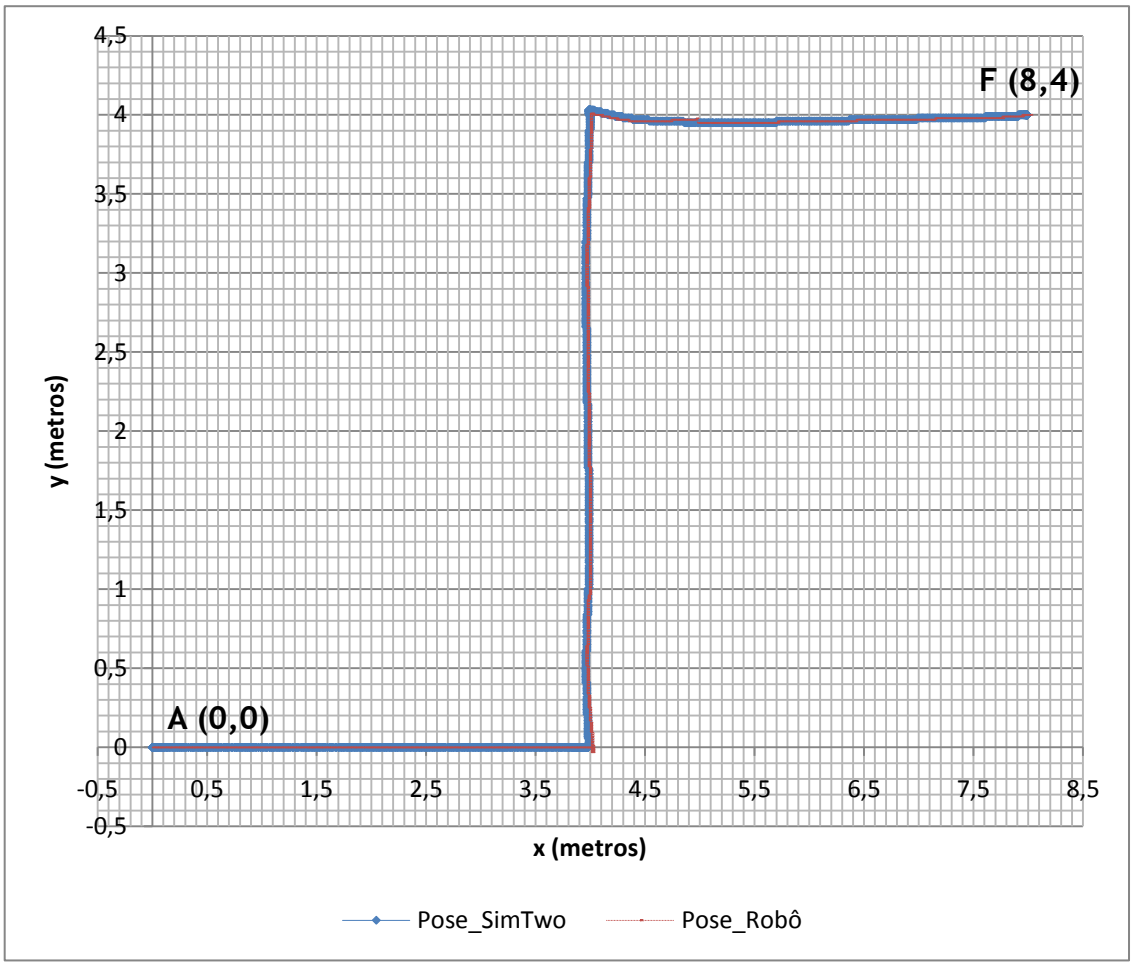


Figura 7.4 – Trajectória baseada em odometria com correcção da orientação e posição do robô através de marcadores

Através da figura 7.4 pode-se constatar que a diferença entre o comportamento do robô e o comportamento observado pelo simulador é mínima, pois a principal fonte dos erros no estado do robô advém de uma incorrecta orientação do mesmo. Desta forma, pode-se verificar que a posição e orientação do robô em relação a um marcador garantem uma melhor localização do mesmo, contribuindo para que a posição final pretendida seja atingida.

7.1.5 Trajectória baseada em odometria com correcção do estado do robô através de marcadores e ruído aplicado aos *encoders*

Como referido anteriormente, a informação dos *encoders* é afectada por ruído. Desta forma, nesta secção pretendeu-se analisar a contribuição deste ruído no comportamento do robô durante a trajectória pretendida e verificar se a posição e orientação do mesmo era corrigida ao longo do percurso.

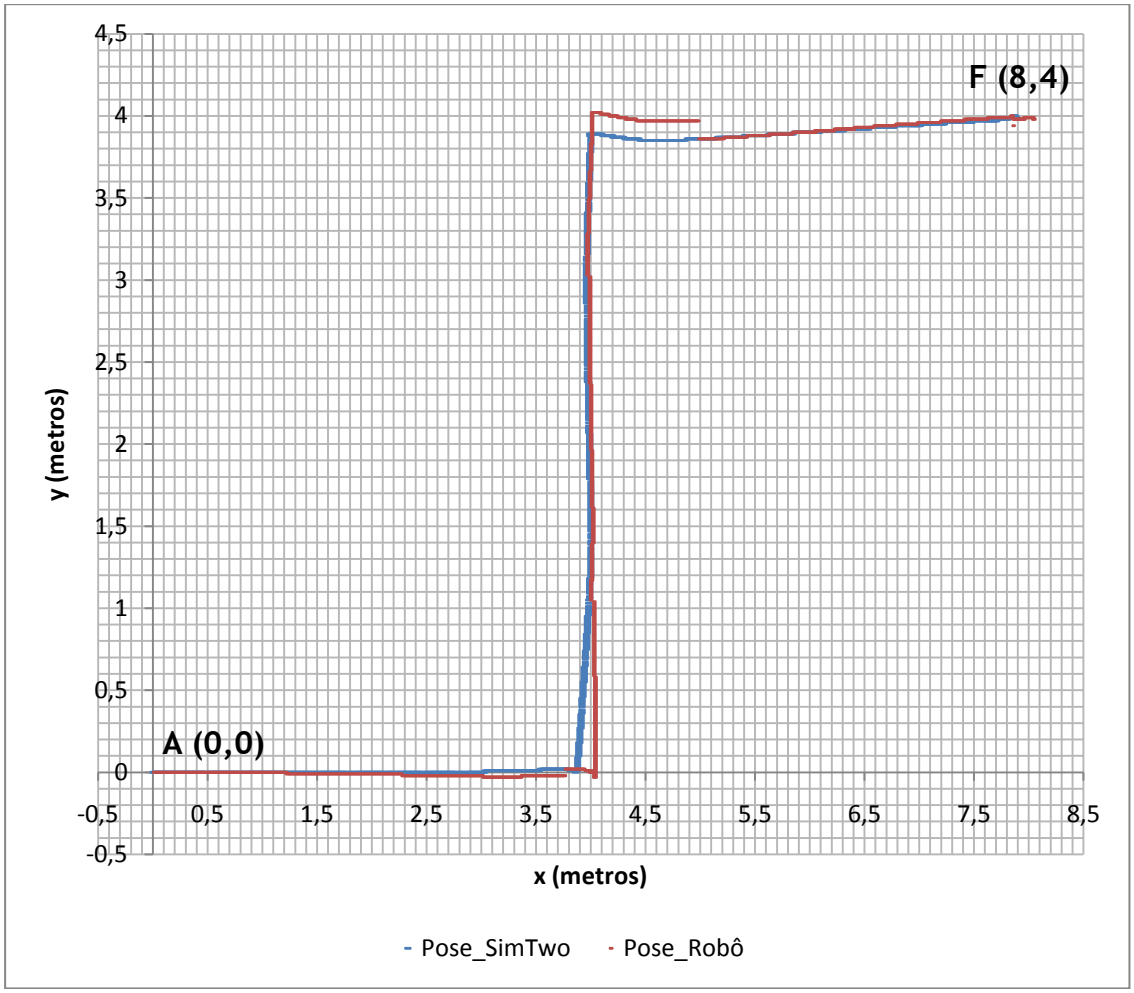


Figura 7.5 – Trajectória baseada em odometria com correcção do estado do robô através de marcadores e ruído aplicado aos *encoders*

Com base na figura 7.5 é possível verificar que a odometria é afectada pelo ruído provido pelos *encoders*. É também possível observar que, apesar deste ruído contribuir para uma quantificação errónea da distância percorrida por cada roda, a utilização de uma localização absoluta através de marcadores artificiais permite corrigir o estado do robô, garantindo desta forma que a posição final pretendida é atingida.

Pode-se verificar a partir da figura 7.5 a presença de uns saltos na trajectória do robô. Isto deve-se ao facto de quando detecta o marcador corrige a sua posição em relação a este. Com a utilização de um filtro de *Kalman* este efeito poderia ser atenuado.

Capítulo 8

Conclusões e Trabalhos Futuros

8.1 Conclusões

Nesta dissertação estudou-se e implementou-se um algoritmo de localização de custo reduzido baseado na detecção de marcadores artificiais. Desta forma, o sistema de localização absoluta assumiu duas abordagens: para o funcionamento real do robô utilizou-se um sistema autónomo de visão para detectar os marcadores artificias; para o funcionamento em simulação utilizou-se um sistema de detecção de paredes laterais.

Pode-se concluir após a finalização deste trabalho que o controlo e o planeamento de trajectórias baseadas no conhecimento do cálculo odómetrico não deve ser o único método a utilizar para estimar a localização absoluta ou relativa de um robô pois os erros são cumulativos, tornando-se cada vez maiores durante o movimento do robô. Desta forma, a utilização da localização absoluta para estimar e corrigir a posição absoluta do robô torna-se uma abordagem imperativa em qualquer sistema autónomo robótico em que a localização exacta dos robôs seja imprescindível para a correcta execução das tarefas.

A utilização de uma plataforma de simulação assumiu um papel preponderante na execução deste trabalho uma vez que, permitiu simular o comportamento do robô e aperfeiçoar os algoritmos de decisão e controlo que regem o mecanismo deste sistema, permitindo também detectar de uma forma mais transparente possíveis erros introduzidos durante a implementação do *software*.

Os resultados experimentais obtidos através da simulação comprovam que o sistema de localização baseado em marcadores artificiais avalia correctamente a qualidade das medidas do estado do robô, impedindo que este possua uma "falsa certeza" da sua localização. Desta forma, valida-se a implementação do algoritmo de localização.

Quanto à plataforma de software, a utilização da comunicação série *RS-232* numa lógica Daisy Chain, entre a camada de nível superior e a camada de nível inferior, revelou-se

bastante proveitosa e de grande eficácia, consolidando e comprovando a sua enorme aplicabilidade pela equipa 5DPO da FEUP/INESCPorto.

8.2 Trabalhos Futuros

Como trabalhos futuros, em primeiro lugar dever-se-á testar o comportamento do robô em funcionamento real e implementar o sistema de detecção de obstáculos para permitir que o robô circule de forma segura pelo ambiente, garantindo que este não colide com obstáculos. Desta forma assegura-se a integridade do sistema.

Propõe-se a estimação do estado do robô através da fusão da odometria e de marcadores artificiais com recurso a um filtro de *Kalman* Estendido (*EKF*).

Embora se tenha procedido às calibrações da odometria, recomenda-se a utilização do método *UMBMark* que permitirá corrigir eficientemente os erros sistemáticos inerentes ao robô e consequentemente melhorar o sistema odométrico do robô.

A aplicação desenvolvida cumpre com os requisitos, todavia, na medida em que a plataforma de desenvolvimento *Lazarus* contempla o funcionamento no sistema operativo *Linux* é de todo o interesse integrar o *software* de controlo e decisão do *AGV* num sistema operativo *Open Source*. Desta forma, e uma vez que não foi possível devido à falta dos *drivers* do Ecrã Táctil, recomenda-se para trabalho futuro a integração no sistema operativo *Linux*. Em relação à aplicação desenvolvida, esta apenas necessita de ser recompilada para possibilitar o correcto funcionamento deste *software*.

Propõe-se o estudo de diferentes tipos de marcadores artificiais com o intuito de estabelecer a melhor forma de detecção do mesmo.

Para finalizar, aconselha-se o desenvolvimento de um software de visão artificial integrado na aplicação de controlo e decisão do *AGV* para detecção de marcadores, permitindo desta forma reduzir os consumos de energia e aumentar a autonomia do robô.

Referências

[1] Sousa, A. (2003). *Arquitecturas de Sistemas Robóticos e Localização em Tempo Real através de Visão - aplicações no domínio do futebol robótico*. Tese de Doutoramento.

[2] *Veículos guiados autonomamente*. Acedido em: http://en.wikipedia.org/wiki/Automated_Guided_Vehicle.

[3] Figueiredo, M. (1999). *Navegação Autónoma de Robôs*.

[4] Rocha, R. P. (2000). *Estado da arte da robótica móvel em Portugal*. Acedido em: http://mail.isr.uc.pt/~mrl/admin/upload/37.pdf.

[5] Groover, M. P. (2000). *Automation, Production Systems, and Computer Integrated Manufacturing*. Prentice-Hall.

[6] Borenstein, J., Everett, H. R., Feng, L. (1996). *Where am I? Sensors and Methods for Mobile Robot Positioning*. University of Michigan.

[7] *The Basics of Automatic Guided Vehicle Systems*. Acedido em: http://www.agvsystems.com/basics/.

[8] *Segurança e Navegação dos AGV´s*. Acedido em: http://www.ek-automation.com/products/automated-guided-vehicles/navigation.html

[9] *Automatic Guided Vehicle Navigation*. Acedido em: http://www.jbtc-agv.com/Knowledge/Navigation.aspx.

[10] *Guidance Options for Automatic Guided Vehicles*. Acedido em: http://www.jervisbwebb.com/Products/guidance_options.aspx?pid=308&qs=1_6_.

[11] *Laser Navigation Controls*. Acedido em: http://www.fmcsgvs.com/content/products/nav.htm.

[12] Beccari, G. (1997). *Vision-based Line Tracking and Navigation in Structured Environments*. University of Parma (Italy).

[13] Leitão, P., Gonçalves, J. (2005). *Notas de Apoio a Robótica Móvel da Disciplina Automação e Robótica*. Acedido em http://www.ipb.pt/~pleitao.

[14] *AGVs Make The Workplace safer*. Acedido em: http://www.hksystems.com/agv/safety.cfm?m=2&s=5.

[15] Costa, A., Simão, P. (2006). *Segurança em AGV´s*. Acedido em: www.ieeta.pt/lse/str/str-0607/docs/agvs.ppt.

[16] Sobreira, H. (2009). *Clever Robot*. Tese de mestrado.

[17] *Mercedes-Benz - Assistência ao estacionamento*. Acedido em: http://www.mercedes-benz.pt/content/portugal/mpc/mpc_portugal_website/ptng/home_mpc/passengercars/home/new_cars/models/b-class/b245/overview/comfort.0006.html

[18] Obstacles detection. Acedido em: http://www.kivasystems.com/

[19] Esteves, J. S. (2005). *Metodologia de auto-localização absoluta em ambientes quase-estruturados*. Tese de Doutoramento.

[20] Moreira, A. P. (2009). *Sistemas Robóticos Autónomos*. 1 diaporama (107 diapositivos). Acessível na Faculdade de Engenharia da Universidade do Porto.

[21] Borenstein, J. (1996). *Measurement and Correction of Systematic Odometry Errors in Mobile Robots*. IEEE Transactions on Robotics and Automation.

[22] Fonseca, J. F. (1999). *Aplicação de técnicas de fusão sensorial de dados no levantamento do relevo de objectos*. Tese de doutoramento.

[23] Mitchell, H. B. (2007). *Multi-Sensor Data Fusion*. Springer.

[24] Welch, G., Bishop, G. (2006). *An introduction to the Kalman filter*. Acedido em: http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf.

[25] Scorba, M. (1997). *Simultaneous Localisation and Map Building*. Robotics Research Group Department of Engineering Science University of Oxford. Tese de doutoramento.

[26] Muratore, J.. *Illumination for Machine Vision*. Dolan-Jenner Industries. Acedido em: http://www.pinnaclevision.co.uk/illum02.htm.

[27] *Maxon DC Motor*. Acedido em: 05 de Janeiro de 2009, em: http://www.maxonmotor.com/product_overview_details_maxon_dc_motor.html.

[28] *Omron Componentes Electrónicos*. Acedido em: http://omron.pt/.

[29] *Maxon sensor*. Acedido em: http://shop.maxonmotor.com/.

[30] *Sistemas de Visão Artificial*. Acedido em: http://www.infaimon.com/.

[31] Stalling, W. (1997). *Data and computer communications*. Prentice-Hall.

[32] *User Datagram Protocol*. Acedido em: http://pt.wikipedia.org/wiki/User_Datagram_Protocol.

[33] *Transmission Control Protocol*. Acedido em 04 de Janeiro de 2010. URL: http://pt.wikipedia.org/wiki/Transmission_Control_Protocol.

[34] Vaccaro, R. (1995) *Digital Control: A state-space approach*. Mcgraw-Hill College.

[35] Costa, P. (2010). *Simtwo*. Acedido em: http://paginas.fe.up.pt/~paco/wiki/index.php?n=Main.SimTwo.

[36] Smith, R. (2009). *Open dynamics engine*. Acedido em: http://www.ode.org/.

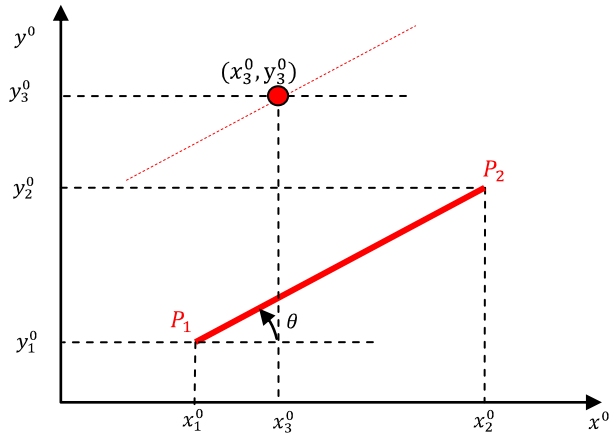
[37] Kok, C. (2009). *Pascal script*. Acedido em: http://wiki.freepascal.org/Pascal_Script.

[38] W3C-WorldWideWeb Consortium (2009). *Extensible markup language (xml)*. Acedido em: http://www.w3.org/XML/.

[39] DALSA Corporation (2007). *Sherlock Machine Vision Software – User's Reference Manual*. Infaimon.

Anexo A

Distância de um ponto a uma recta



$$\theta = arctg2\left(\frac{y_2^0 - y_1^0}{x_2^0 - x_1^0}\right)$$

$$R_1^0 = R_{z,\theta} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ponto P_3:

$$P_3^0 = R_{z,\theta} * P_3^1 \Leftrightarrow P_3^1 = R_{z,\theta}^{-1} * P_3^0$$

$$R_{Z,\theta}^{-1} = R_{z,\theta}^T$$

$$P_3^1 = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = [-x_3^0 sin\theta + y_3^0 cos\theta] \equiv y_3^1$$

Ponto P_1:

$$P_1^1 = R_{z,\theta}^{-1} * P_1^0 = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = [-x_1^0 sin\theta + y_1^0 cos\theta] \equiv y_1^1$$

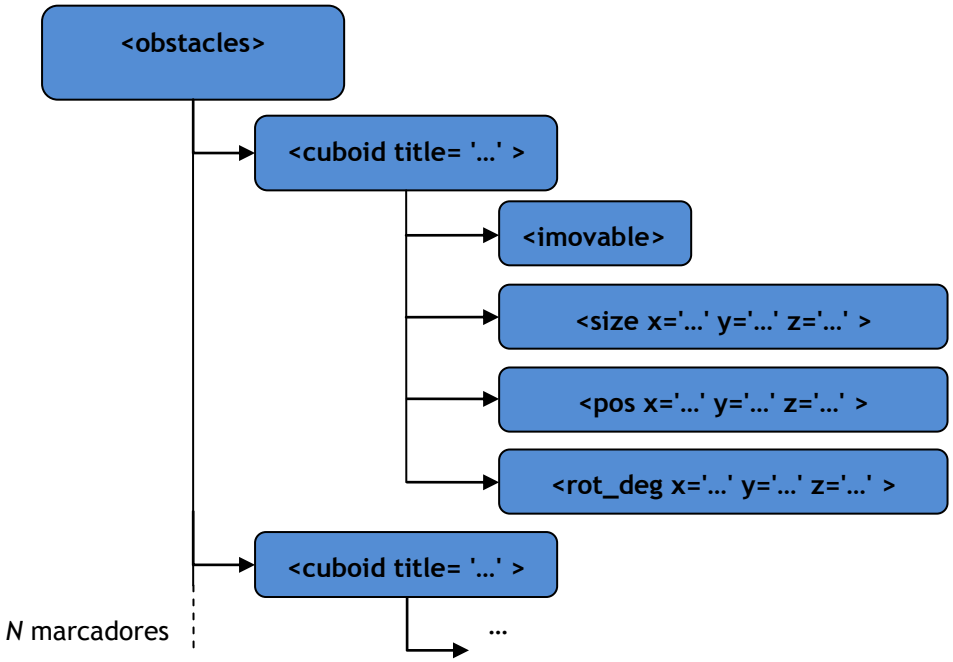
Donde:

$$Distância \ = -x_1^0 sin\theta + y_1^0 cos\theta - (-x_3^0 sin\theta + y_3^0 cos\theta)$$

Anexo B

Mapa de marcadores

A configuração da distribuição de marcadores pelo ambiente para permitir a navegação e localização do robô está representada num mapa de marcadores. Este mapa de marcadores, descrito num ficheiro *XML*, permitiu a sua utilização tanto na simulação como na aplicação de controlo e decisão do *AGV*.

Esta abordagem também teve em consideração a questão de flexibilidade do sistema. Assim, a alteração da posição de marcadores poderá ser realizada por um utilizador sem recorrer à alteração do código fonte desenvolvido para o robô. A aplicação de controlo e decisão, sempre que é iniciada, processa o ficheiro, ficando assim com a base de dados dos marcadores.

Segue-se a representação da estrutura do mapa de marcadores em *XML:*