INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia Eletrotécnica e Automação

Robótica Móvel – Sistema de Condução Autónoma

CÉSAR VALENTINO RIBEIRO COUTINHO
(Licenciado em Engenharia Eletrotécnica)

Dissertação para a obtenção do grau de Mestre
em Engenharia Eletrotécnica – Ramo Automação e Eletrónica Industrial

Orientador:
Professor Coordenador Fernando Manuel Fernandes Melício

Júri:

Presidente: Professor Elmano da Fonseca Margato (ISEL)
1º Vogal: Professor Fernando Manuel Fernandes Melício (ISEL)
2: Vogal: Professor Carlos Baptista Cardeira (IST)

Abril de 2014

Resumo

Na Robótica, o recurso à Visão Computacional sobressai como a forma mais abrangente de conhecer o ambiente onde determinado robot opera.

Tanto na indústria como no sector automóvel, são diversos os exemplos onde esta tecnologia é aplicada com resultados excelentes, impulsionadores de novos desenvolvimentos nesta área.

Nesse sentido, este trabalho pretende apresentar uma nova abordagem aos sistemas de controlo dos robots móveis diferenciais. Centrando-se numa fusão sensorial entre Visão e Ultrasons, implementa um sistema de controlo que privilegia a simplicidade e baixo custo, sem contudo descurar a eficácia necessária para que este tipo de sistemas possa ser considerado funcional, confiável e aplicável na prática.

Recorrendo apenas a vulgares *webcams*[1] e sensores de ultrasons, desenvolveu-se um sistema de condução autónoma capaz de orientar um robot móvel diferencial através de um percurso semelhante a uma faixa de rodagem real, respeitando a sinalização existente e evitando os obstáculos que lhe vão surgindo.

Este sistema, sustentado num algoritmo baseado em métodos de tratamento e realce de imagens como o Método de Otsu [80] e de extração de características como o método de seguimento de contornos proposto por Satoshi Suzuki e Keiichi Abe [99], identifica em cada instante o desvio face ao centro da trajetória a seguir e determina, através de um controlador *Fuzzy* [84], a forma de controlar a velocidade das rodas do robot para que este regresse à orientação pretendida. Em simultâneo com esta tarefa fundamental, uma análise particular de pontos-chave da imagem, coadjuvada com a informação proveniente dos sensores de ultrasons, analisa continuamente o espaço envolvente e avalia a existência de sinalização ou obstáculos que tenham que ser levados em conta para a movimentação do robot.

[1] Designação usualmente atribuída a dispositivos de captação de imagem, com interface USB, destinados à realização de chamadas de vídeo pela internet.

Abstract

In Robotics, the appeal to Computational Vision emerges as the most comprehensive way of knowing the environment where a certain robot operates.

Both in industry and in the automotive sector, there are several examples where this technology is applied with excellent results, mentors for new developments in this area.

Accordingly to that, this study aims to present a new approach to control a mobile differential robot. By focusing on a sensory fusion between Vision and Ultrasound, implements a control system which emphasizes simplicity and low cost, without neglecting the effectiveness required for this type of systems be considered functional, reliable and applicable in practice.

Using only common webcams[2] and ultrasonic sensors, was developed an autonomous driving system capable of guiding a mobile differential robot through a path similar to a roadway, while respecting the existing signaling and avoiding the obstacles that may emerge.

This system, sustained on an algorithm based on image treatment and enhancement methods such as Otsu [80] and extraction of characteristics such as the contours follow-up algorithm proposed by Satoshi Suzuki and Keiichi Abe [99], identifies in each instant the deviation toward the center of the trajectory to follow and determines, through a *Fuzzy* controller [84], the way to control the robot wheels speed to return to the desired orientation. In parallel with this fundamental task, a particular analysis of image key points, associated with the information coming from the ultrasonic sensors, continuously analyzes the surrounding environment and assesses the existence of signaling or obstacles that have to be taken into account for the movement of the robot.

[2] Designation usually attributed to image capture devices, with USB interface, intended for the realization of video calls over the internet.

Palavras-Chave

- Robótica
- Robots Móveis
- Condução Autónoma
- Visão Computacional
- Processamento de Imagem
- Fusão Sensorial
- Controlo *Difuso*

Keywords

- Robotics
- Mobile Robots
- Autonomous Driving
- Computer Vision
- Image Processing
- Sensory Fusion
- *Fuzzy* Control

Agradecimentos

A realização deste trabalho não seria possível sem o contributo direto ou indireto de um grupo de pessoas que, para além de demostrarem uma amizade sincera, sempre se disponibilizaram incondicionalmente para me ajudar de alguma forma no cumprimento deste objetivo de vida.

- Ao meu orientador, Professor Coordenador Fernando Manuel Fernandes Melício, por toda a ajuda, incentivo, orientação e valiosas contribuições ao longo da construção desta dissertação.

- Ao Professor José Carlos de Ponte Ribeiro, pela disponibilização da estrutura robótica utilizada e por todo o apoio prestado na fase inicial deste trabalho.

- À minha família, por toda a dedicação, apoio e confiança que depositaram em mim desde o primeiro momento.

- A todos aqueles que me perguntavam pelo trabalho e se mostravam interessados por este, depositando em mim toda a confiança necessária para que esta meta fosse alcançada com êxito.

A todos, os meus mais sinceros agradecimentos,

César Valentino Ribeiro Coutinho

Lista de Tabelas

Lista de Figuras

Índice

1. INTRODUÇÃO

Desde sempre o homem ousou pensar e sonhar no dia em que máquinas teriam características e habilidades que permitiriam que estas o substituíssem em determinadas tarefas com maior velocidade, precisão e segurança.

Desta intensão, surgiu a Robótica, área cientifica interdisciplinar, em constante evolução, que reúne ferramentas, metodologias e tecnologias, desenvolvidas através de conceitos teóricos de grandes áreas como a matemática, física, química, biologia e de áreas aliadas ao entendimento do cérebro e do corpo humano como neurologia, fisiologia e psicologia.

Atualmente, a robótica móvel e a visão computacional são os sectores da robótica em maior destaque. Assim, este trabalho centrou-se particularmente nestas duas temáticas.

1.1 ***Motivação***

A Robótica e a Visão Computacional são duas áreas tecnológicas que pela sua abrangência, potencialidade e perspetiva de evolução despertam bastante interesse a quem procura um tema para o desenvolvimento de uma dissertação de mestrado.

Aliando a isto, o gosto pelo sector automóvel, porventura uma das áreas onde a aplicação de sistemas interligando estas áreas tem sido mais evidente, torna-se óbvia a opção de desenvolver um trabalho sobre estas temáticas.

Sobretudo a possibilidade de aprofundar o conhecimento e poder contribuir para novos pontos de vista, que possam servir de base a futuros desenvolvimentos a serem implementadas nas diversas áreas e sectores da vida quotidiana, serviram de estímulo ao trabalho e pesquisa efetuada.

O desejo de que este trabalho não se cinja apenas a uma dissertação de mestrado esteve também patente. Pensando em evoluções futuras, foi utilizada uma abordagem o mais concisa possível, tendo-se optado pela utilização de equipamentos de baixo custo, permitindo que investimentos mais significativos possam ser efetuados apenas na expansão das funcionalidades agora apresentadas.

1.2 ***Enquadramento***

Este trabalho enquadra-se na área da condução autónoma de um robot/veículo móvel terrestre.

Nesta área são privilegiados desenvolvimentos que possam ser testados e aplicados em veículos, como forma destes serem dotados de capacidades autónomas de planeamento e seguimento de trajetórias, visando não só o cumprimento de determinado objetivo mas também o acréscimo de segurança inerente a estas funcionalidades.

1.3 ***Objetivos***

Esta dissertação tem como principais objetivos desenvolver e implementar num robot móvel diferencial, um sistema de condução autónomo sustentado numa fusão sensorial entre Visão Computacional e Ultrasons.

Pretende-se que este sistema consiga orientar um robot móvel num percurso definido por linhas semelhantes a uma faixa de rodagem real, evitando embater em obstáculos e identificando e respeitando a sinalização existente.

No decorrer deste trabalho, foi sempre tido em consideração o objetivo de manter o desenvolvimento deste sistema o mais conciso possível, recorrendo para isso a equipamentos e metodologias que permitam que o seu desenvolvimento futuro seja simples e viável do ponto de vista económico.

1.4 ***Metodologia de Trabalho***

Este trabalho foi desenvolvido por etapas, podendo-se destacar as seguintes como as mais representativas das várias tarefas necessárias ao cumprimento dos objetivos propostos:

- Pesquisa Bibliográfica
- Estudo sobre o funcionamento dos robots móveis, mais especificamente robots móveis diferenciais.
- Estudo sobre as pesquisas e desenvolvimentos efetuados na área da condução autónoma.
- Avaliação das formas de controlo mais eficazes para desempenharem esta função.
- Estudo sobre as potencialidades da Visão Computacional e dos Ultrasons como formas de controlo de um robot móvel.
- Avaliação dos benefícios de uma fusão sensorial.
- Desenvolvimento do sistema de seguimento de trajetória e deteção de obstáculos.
- Expansão do sistema com a capacidade de deteção de sinalização.
- Implementação do sistema desenvolvido no robot.
- Testes, conclusões e análise das possibilidades de desenvolvimento futuro.

1.5 ***Estrutura***

Este documento encontra-se dividido em seis capítulos.

No presente capítulo, **Capítulo 1 – Introdução**, para além desta secção, é feita uma introdução ao trabalho e são apresentadas as motivações, objetivos e a metodologia de trabalho seguida.

Capítulo 2 – Estado da Arte – Neste capítulo é apresentada a evolução e o estado da arte das principais áreas científicas envolvidas neste trabalho.

Capítulo 3 – Metodologia - Descreve os métodos propostos e os respetivos conceitos teóricos necessários ao desenvolvimento e implementação de um sistema de condução autónoma.

Capítulo 4 – Implementação - Apresenta a estrutura mecânica e a implementação do sistema de controlo proposto.

Capítulo 5 – Resultados – Neste capítulo são apresentados e discutidos os resultados obtidos nos testes efetuados às várias etapas do algoritmo implementado e à movimentação do robot.

Capítulo 6 – Conclusões e Trabalho Futuro - Finalmente, neste capítulo são apresentas as principais conclusões sobre os resultados obtidos, é feita um balanço das limitações da metodologia adotada e são avaliadas possíveis perspetivas de desenvolvimento futuro do trabalho realizado.

2. Estado da Arte

Neste capítulo é apresentado o estado da arte no que diz respeito às principais áreas científicas envolvidas neste trabalho.

Inicialmente desenvolve-se a temática da robótica, sendo abordado o seu surgimento, os tipos de robots mais vulgares e as suas principais características. É dado particular enfase aos robots móveis, tendo estes sido mais detalhadamente analisados.

Posteriormente, faz-se um breve resumo das estratégias de controlo mais utilizadas e sublinham-se alguns dos mais significativos trabalhos de pesquisa desenvolvidos nesta área.
Aborda-se também nesta fase a realidade da robótica em Portugal e a modalidade da Condução Autónoma.

Por fim, apresentam-se os marcos mais significativos da evolução das linguagens de programação, as suas principais características e bibliotecas ou funcionalidades que lhes podem ser adicionadas.

2.1 ***Robótica***

Já há algum tempo que o termo robot nos é familiar, sendo por norma associado a máquinas ou sistemas dotados de algum tipo de inteligência, que as permite executar tarefas em auxílio ou substituição do ser humano ou para seu entretimento.

Tendo origem na palavra checa *robota* [87], que significa trabalho forçado ou trabalho escravo, este termo, usado pela primeira vez pelo escritor checo Karel Capek na sua peça teatral *Rossum's Universal Robots* em 1921, não tem uma definição universal, contudo, de acordo com a *Robotics Industries Association* pode-se considerar que um robot é um dispositivo mecânico articulado reprogramável, que consegue, de forma autónoma e recorrendo à sua capacidade de processamento, obter informação do meio envolvente utilizando sensores, tomar decisões sobre o que fazer com base nessa informação e em informação à priori e manipular objetos do meio envolvente utilizando atuadores [31].
Embora o uso de máquinas ou dispositivos automáticos remeta a tempos antes de Cristo, o primeiro robot industrial apenas foi utilizado em 1961 [52]. Desenvolvido pela Unimate, a sua estrutura era semelhante à de um braço humano e foi utilizado numa linha de montagem da fábrica da General Motores nos Estados unidos.

2.1.1 Robots Manipuladores

Os manipuladores industriais, tais como o desenvolvido pela Unimate, basearam as suas estruturas e funcionalidades na constituição de um "braço humano".

As juntas constituintes de um robot manipulador tentam replicar as capacidades de um "braço humano". Por norma, as primeiras juntas, denominadas de braço (correspondentes no homem ao ombro e cotovelo), posicionam a estrutura formada pelas juntas seguintes, designadas por punho, que são utilizadas para orientar o elemento-terminal.

Geralmente, o punho tem duas configurações: pitch-yaw-roll (yxz) como o punho humano ou roll-pitch-roll (zyz) denominado punho esférico. Devido à maior simplicidade, o ultimo é o mais usado em robótica de manipulação.

São cinco os tipos de braços manipuladores mais utilizados em robótica: cartesiano, cilíndrico, polar, revolução e SCARA.

Figura 2.1 - Robot SCARA ou articulado horizontal da Toshiba

Os primeiros trabalhos com manipuladores robóticos ocorreram ao fim da segunda guerra mundial, onde foram introduzidas e desenvolvidas máquinas do tipo Master-Slave [83] para manipular materiais perigosos (radioativos). Estas máquinas eram formadas por um manipulador "master", acionado diretamente por um operador humano responsável pelas sequências de movimentos desejados, e um manipulador "slave", capaz de reproduzir os movimentos realizados remotamente pelo “master”.
Os vínculos entre os manipuladores "master" e "slave" eram realizados através de sistemas de transmissão mecânicos [23].

Exemplos destes manipuladores são o “Gantry-robot” desenvolvido pela General Mills Corporation - USA:

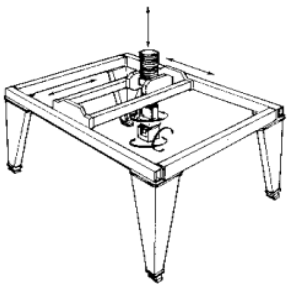


Figura 2.2 - "Gantry-robot"

O “Planetbot” (1957), que foi o primeiro robot manipulador comercial com coordenadas polares:

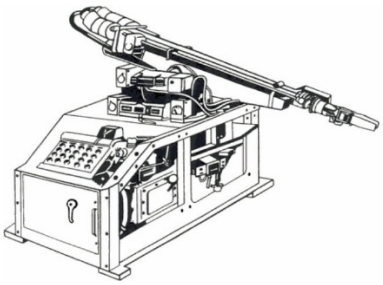


Figura 2.3 - "Planetbot"

E o robot desenvolvido por Norman Diedrich no Instituto Case da Western Reserve University (Cleveland - USA) que foi o primeiro manipulador elétrico com juntas de revolução:

Figura 2.4 - Robot do Instituto Case da Western Reserve University

Como abordado anteriormente, o primeiro robot industrial, o "Unimate" [52] foi desenvolvido por George Devol e Joseph Engelberger na companhia americana Unimation Inc. entre 1959 e 1962. Este robot era programado através de um computador e podia ser usado em diversas aplicações, bastando para isso reprogramá-lo corretamente e equipá-lo com as ferramentas adequadas à função.

Figura 2.5 - "Unimate"

Embora muito poderosa para a época, tornou-se óbvio que a flexibilidade e adaptabilidade desta nova ferramenta poderia ser consideravelmente melhorada utilizando a retroação sensorial.

Nos anos 60 e 70, vários trabalhos de investigação e desenvolvimento conduziram aos primeiros robots controlados por computador com retroação sensorial [83], o T^3 (Tool of The future) produzido pela Cincinatti Millacron (1974) e comercializado a partir de 1978, o braço de Stanford (fim dos anos 60 e inicio dos 70) que deu origem mais tarde ao PUMA da Unimation Inc (1978), o manipulador da IBM (1975) e o SCARA (Selective Compliance Assembly Robot Arm) desenvolvido entre 1978-79 e produzido por vários fabricantes, sendo a primeira das quais a Sankio Seikl.

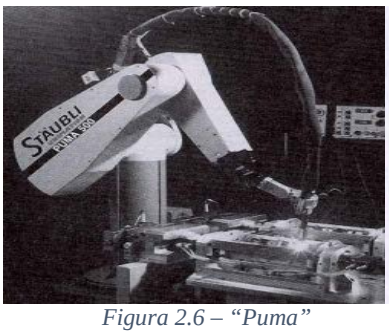


Figura 2.6 – "Puma"

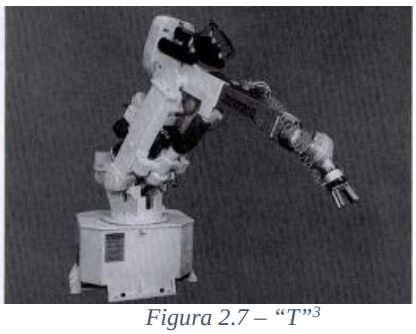


Figura 2.7 – "T"[3]

Os manipuladores mecânicos progrediram bastante, tendo surgido várias técnicas de controlo. Porém, esta área da robótica continua a ter grandes perspetivas de evolução, nomeadamente em termos de formas de programação, controlo de força, retroação visual, integração sensorial e

até sobre nas novas estruturas e materiais que permitam a construção de novos robots, mais leves e flexíveis.

Atualmente um robot manipulador industrial é constituído por vários elos rígidos ligados em série por juntas, tendo uma das extremidades fixa (base) e outra livre para se mover (elemento-terminal). As juntas são geralmente atuadas por motores elétricos (atualmente são geralmente motores trifásicos síncronos, embora também se usem atuadores pneumáticos e hidráulicos).
Por norma, um sistema computorizado controla e supervisiona o movimento do robot, recorrendo a informação sensorial para obter o seu estado e o do ambiente. O sistema utiliza informação sensorial como a posição das juntas (recorrendo a sensores de posição), a força de contacto (usando sensores de força/momento) e a distância a objetos, para calcular os sinais de controlo a enviar para obter o movimento desejado.

Figura 2.8 - Mitsubishi Movemaster RV-M2

Apesar da sofisticação, atualmente os robots são utilizados sobretudo em ambientes de produção para realização de tarefas repetitivas em linhas de montagem, tirando partido das suas capacidades de velocidade e repetibilidade.

Características Típicas de um Manipulador Robótico	
Repetibilidade	Até 0.03 mm (0.1 mm é comum)
Velocidade	Até 5 m/s
Aceleração	Até por volta de 25 m/s^2
Payload	De 2-3 kg até 350 kg
Peso/Payload	Por volta de 30-40
Eixos	6
Comunicações	Profibus, Can, Ethernet e canais série (RS232, 485)
Capacidades I/O	Capacidades de PLC para tratar sinais analógicos e digitais

Tabela 2.1 - Características típicas de um manipulador atual

2.1.2 Robots Móveis

Apesar do grande sucesso dos robots manipuladores, estes têm uma grande limitação, a falta de mobilidade.

Aliando a intenção de ultrapassar esta limitação, ao desenvolvimento da inteligência artificial e às extensas aplicações possíveis para um robot móvel, este ramo da robótica tornou-se o principal foco de atenção, sendo atualmente o sector onde se desenvolvem as mais promissoras pesquisas.

Um dos primeiros robots móveis conhecidos data de 1969, altura em que na Universidade de Stanford, Nils Nilsson apresentou o robot móvel Shakey [77], o primeiro a usar inteligência artificial para controlar os seus movimentos. Recorrendo à perceção visual, utilizando uma câmara, e a sensores táteis binários, navegava através de ambientes altamente estruturados, tais como edifícios de escritórios.

Figura 2.9 - Shackey

Este robot navegava entre as salas do laboratório, trocando sinais de rádio com a sua unidade de processamento externa. A bordo do robot apenas eram recolhidas as informações sensoriais, externamente era efetuado o processamento dos dados recolhidos e o cálculo dos comandos a enviar ao mesmo.

Após este, os trabalhos nesta área foram constantes. Na década de 70, foi desenvolvido no Laboratório de Propulsão a Jato da NASA, o Lunar Rover. Projetado para a exploração planetária, usava uma camara e vários sensores, sendo capaz de se movimentar num terreno desconhecido.

No final desta década, Hans Moravec desenvolveu no Laboratório de Inteligência Artificial em Stanford o robot CART [73]. Capaz de seguir uma linha branca numa estrada, possuía uma câmara na sua parte superior que tirava fotografias de vários ângulos diferentes e as enviava para um computador onde era determinada a trajetória a seguir.
O aumento da capacidade de manipulação de dados dos computadores tornaram esta área emergente da robótica ainda mais ativa, os robots apresentavam cada vez mais potencialidades, tornando-se progressivamente capazes de encontrar soluções para os seus próprios problemas num ambiente sem supervisão externa.

Destacam-se pela inovação apresentada, muitos outros trabalhos nesta área:

- Em 1986, foi desenvolvido no LAAS – Laboratório de Arquitetura e de Análise de Sistemas, um robot multissensorial, o robot Hilare [3].

- Em 1994, o Instituto de Robótica da Carnegie Mellon desenvolveu o Dante II [51], um robot de seis pernas, que chegou a explorar o Vulcão Mt. Spurr no Alasca para analisar gases vulcânicos.

- Ainda nesta década, surgiu uma das aplicações de robots móveis mais notáveis, a família Rocky. Resultado das pesquisas do Laboratório de Propulsão a jato do Instituto de Tecnologia da Califórnia, o modelo Rocky 4, com dimensões comparadas às de um brinquedo telecomandado e batizado carinhosamente como Sojourner (hóspede temporário em inglês) [33], foi capaz de realizar uma das maiores façanhas da pesquisa do espaço pelo homem: a exploração, à distância, do planeta Marte. Ficou provado com este que o controlo à distância em ambientes completamente desconhecidos é perfeitamente viável.

Figura 2.10 - Rocky 4 – "Sojourner"

- No ano de 1999, as universidades de Carnegie Mellon, Pitsburgh e Bohn desenvolveram em conjunto um veículo de nome Minerva [40], que teve como objetivo servir de guia turístico no Museu Nacional da História Americana. É completamente autónomo e opera em ambientes dinâmicos e povoados, graças a métodos probabilísticos aliados à informação dos diversos sensores que tem acoplados (laser, visão estereoscópica e uma câmara apontada para o teto).

- Em 2006, a Universidade de Stanford em colaboração com a Volkswagen, desenvolveu o robot Stanley [101]. Tendo como base um Volkswagen Touareg R5 TDI e uma plataforma de computação constituída por seis processadores e um conjunto de atuadores e sensores, o objetivo deste projeto foi construir um sistema extremamente seguro e com boa precisão, capaz de conduzir o veículo a velocidades elevadas num ambiente todo-o-terreno, variado e não pré-determinado.

Figura 2.11 - "Stanley"

2.1.2.1 Tipos de Sistemas de Locomoção

O tipo de sistema de locomoção de determinado robot é selecionado para que este desempenhe da melhor forma possível as tarefas para que é proposto no ambiente onde estará inserido. Neste processo de escolha, vários aspetos são tidos em conta:

- Capacidade de Manobra: Facilidade em mudar de direção.
- Controlabilidade: Hardware e Software utilizado para o controlo.
- Tração e Estabilidade.
- Irregularidades do piso.
- Eficiência Energética
- Manutenção.

Quanto ao tipo de locomoção, de acordo com o ambiente, os robots podem ser classificados de acordo com o seguinte fluxograma:

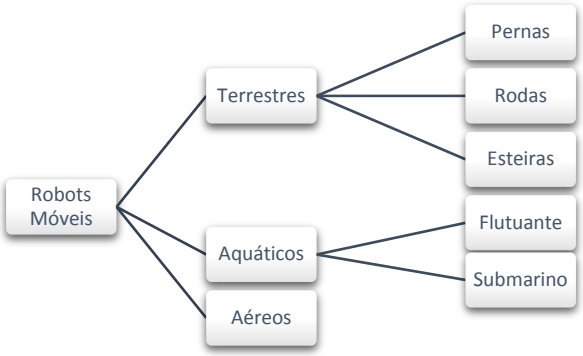


Figura 2.12 - Classificação de robots

a) b c)

Figura 2.13 - a) Terrestre – SIL06, b) Aquático – Ziphius, c) Aéreo – Pelicano

2.1.2.2 Tipos de Rodas

A mobilidade dos robots móveis com rodas é resultado de dois aspetos fundamentais, o tipo de rodas que possui e a sua disposição na estrutura mecânica.

As rodas utilizadas em robótica móvel são basicamente de dois tipos, convencional e omnidirecional (também conhecida por roda sueca), sendo considerado que durante o movimento, o plano da roda se mantém vertical e que estas giram em torno dos seus eixos horizontais. A sua orientação em relação à estrutura pode ser fixa ou variável.
Supõe-se para uma roda convencional que o contato entre a roda e o terreno satisfaz condição de rotação pura sem deslizamento. O que significa que a velocidade do ponto de contato é igual a zero (tanto para a componente paralela como para a perpendicular).

Rodas Convencionais

As rodas convencionais distinguem-se em três tipos:

- Roda fixa: O eixo da roda está fixo na estrutura do robot. Em geral, este tipo está associado ao sistema de tração do robot.

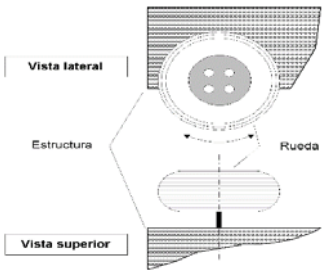


Figura 2.14 - Roda fixa

- Roda orientável centrada: Nesta roda, o movimento do plano em relação à estrutura é uma rotação ao redor de um eixo vertical que passa através do centro da roda. Em geral é usada como roda de direção ou de tração-direção.

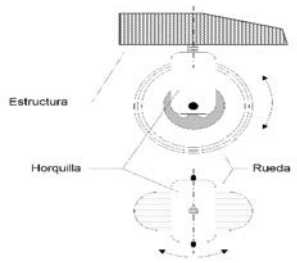


Figura 2.15 - Roda orientável centrada

- Roda orientável não centrada (roda livre): Sendo também conhecida por roda castor (castor wheel), é uma roda orientável com relação à estrutura tal que a rotação do plano da roda ocorre em redor de um eixo vertical que não passa através do centro da roda. A sua principal função é estabilizar a estrutura mecânica do robot.

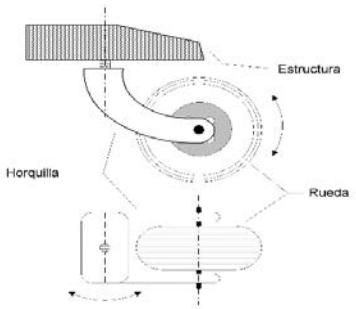


Figura 2.16 - Roda livre

<u>Rodas Omnidirecionais</u>

Nas rodas omnidirecionais, supõe-se que apenas a componente da velocidade do ponto de contato da roda com o terreno ao longo do movimento é igual a zero.
Esta roda é responsável pela tração na direção perpendicular ao eixo do motor e permite deslizar na direção do seu eixo.
Para garantir a característica associada à omnidirecionalidade é necessário que as rodas tenham pouco atrito na direção do eixo do motor, facilitando o movimento segundo esta direção.

Durante a deslocação do robot, podem ser combinados movimentos de translação e rotação, fazendo com que determinado robot possa chegar ao local de destino com o ângulo desejado.

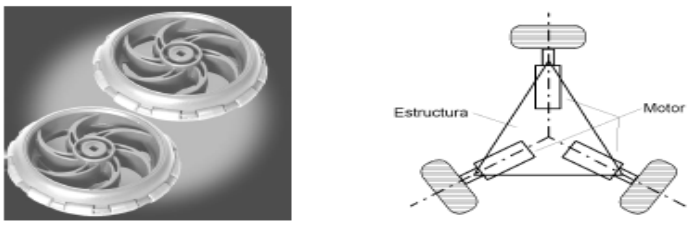


Figura 2.17 - Roda omnidirecional e exemplo de disposição sobre um robot

2.1.2.3 Disposição das rodas

Como referido, a manobrabilidade de determinado robot móvel resulta da conjugação do tipo de rodas que utiliza e da sua disposição na estrutura.
De acordo com o seu grau de manobrabilidade, os robots móveis são classificados em diferentes tipos. De seguida, apresentam-se os mais vulgares. Apesar de existirem muitos outros, são variantes destes tipos base.

Robot Omnidirecional

Os robots omnidirecionais tem uma manobrabilidade total no plano, ou seja, podem-se mover em qualquer direção sem necessidade de se reorientarem. Contudo, o controlo sincronizado deste movimento é complexo.

A imagem seguinte, apresenta o robot omnidirecional desenvolvido na Universidade do Michigan, o Uranus [6], e as suas capacidades de manobra. De acordo com a rotação de cada uma das suas rodas, o robot pode andar em linha reta, girar ou deslocar-se lateralmente sem necessidade de mudar a sua orientação.

a) b)

Figura 2.18 - Robot com rodas omnidirecionais. (a) Manobrabilidade. (b) "Uranus"

Na imagem seguinte, apresenta-se uma versão diferente de um robot omnidirecional. Este, com rodas orientáveis centralizadas, pode mudar a direção do seu movimento simplesmente alterando a orientação das rodas. Um exemplo deste tipo de estrutura é o robot Seekur [47].

a) b)

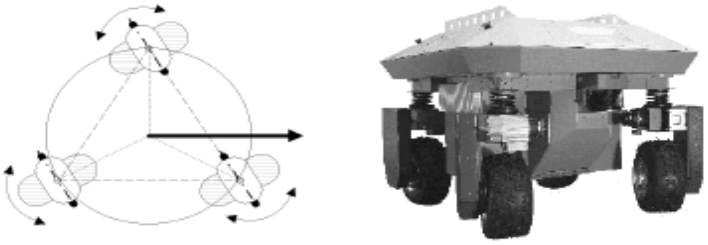


Figura 2.19 - Robot omnidirecional de rodas orientáveis. (a) Disposição. b) "Seekur"

O movimento sincronizado pode ser obtido mecanicamente, através de correias, ou eletronicamente, através de sinais de acionamento que comandam cada um dos motores das rodas.

a) b)

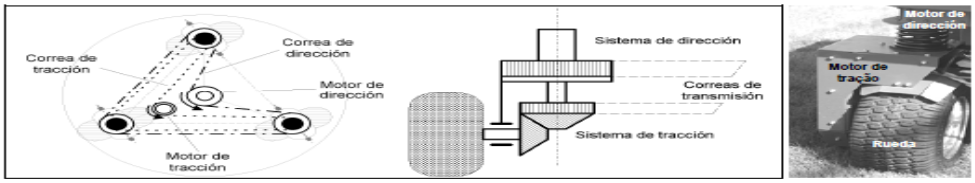


Figura 2.20 - Sincronismo tração/direção. a) Sistema mecânico. b) Sistema eletrónico

Robot Uniciclo

O robot Uniciclo, pela simplicidade da cinemática e controlo envolvido, é por norma a opção dos pesquisadores para plataforma de testes de novas estratégias de controlo.
A sua estrutura é formada por duas rodas fixas convencionais, dispostas sobre o mesmo eixo e controladas independentemente e por uma ou mais rodas livres, que lhe conferem estabilidade. O sistema de tração-direção associado ao robot necessita de controlar de forma independente as velocidades linear e angular das duas rodas.

a) b)

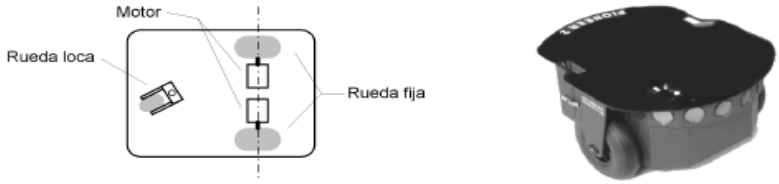


Figura 2.21 - Robot Uniciclo. a) Estrutura. b) "Pioneer"

Robot Triciclo

O sistema de locomoção deste robot consiste em duas rodas convencionais fixas sobre um mesmo eixo, e uma roda convencional, centrada e orientável que reúne as funções de tração e direção. Tal como no caso do robot Uniciclo, a cinemática deste robot é bastante simples, tornando-o adequado para pesquisas em diversas áreas. Este tipo de estrutura é apta para transporte de cargas pesadas em baixa velocidade em ambiente industrial.

A estrutura deste robot apresenta contudo uma limitação séria à sua utilização. Quando em movimento, o centro de gravidade do veículo posiciona-se em alguns casos, nos limites da superfície de equilíbrio definida pelas três rodas. Este fato, pode produzir perda momentânea de tração e é fonte de erro no memento da determinação da posição do robot.

Se a aplicação onde o robot será utilizado necessitar de uma grande precisão, este robot poderá não ser o adequado.

a) b)

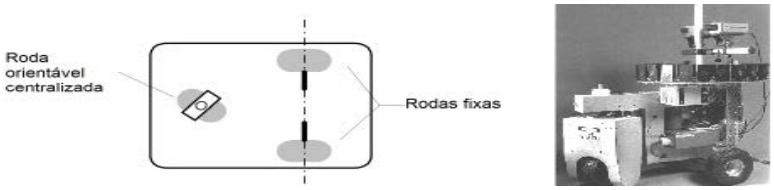


Figura 2.22 - Robot Triciclo. a) Estrutura. b) "Neptune"

Robot Quadriciclo (ou Ackerman)

O robot quadriciclo, também conhecido por Ackerman, surgiu da necessidade de ultrapassar o problema do erro na determinação do posicionamento do robot triciclo.
Tal como se verifica na imagem seguinte, neste robot, os eixos das duas rodas frontais intercetam-se num ponto (C na imagem) pertencente ao eixo comum das rodas traseiras.
No plano, o lugar geométrico dos pontos traçados por cada roda ao redor do ponto comum, forma um conjunto de arcos concêntricos, sendo todos os vetores de velocidade instantânea tangenciais a estes arcos. Esta estrutura contribui para uma maior estabilidade e evita o deslizamento das rodas, reduzindo portanto os erros de posicionamento (odometria).
Apesar da maior complexidade, esta configuração é bastante utilizada em robots que circulem em meios bastante irregulares, por exemplo em terra batida.

a) b)

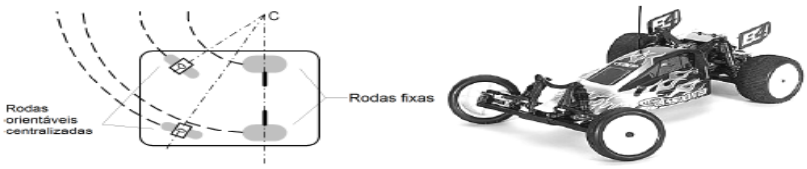


Figura 2.23 - Robot Quadriciclo. a) Estrutura. b) Robot Quadriciclo Vulgar

2.1.2.4 Tração e Direção

As capacidades motoras de determinado robot advêm do sistema de tração e direção que este utiliza. Estes sistemas, resultam da conjugação da disposição de rodas adotada, dos algoritmos de controlo local dos motores e da mecânica a eles associada.
À medida que as exigências relativamente à manobrabilidade, tração e aderência vão aumentando, o sistema de controlo torna-se também mais complexo.

Indicam-se de seguida os sistemas de tração e direção mais vulgarmente utilizados. Variações destes sistemas base não são referenciados:

Tração e Direção em eixos independentes

Nestes sistemas, a tração é efetuada por um conjunto de rodas e a direção é garantida pelo outro conjunto.
Apesar do algoritmo de controlo deste tipo de robots ser simples, a precisão obtida na direção depende da aderência das rodas responsáveis pela mesma.
Este tipo de sistema de tração e direção apresenta como limitação a impossibilidade de realizar mudanças de direção muito bruscas. O raio de curvatura é bastante elevado face a outros sistemas.

a) b)

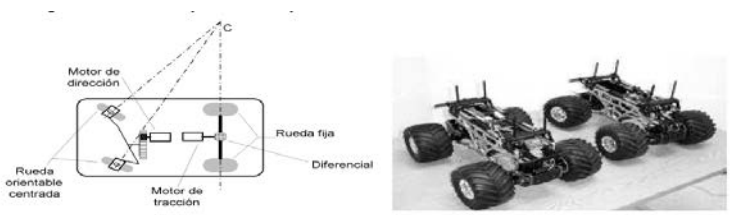


Figura 2.24 - Tração e direção independente. a) Características. b) Exemplo

Tração e Direção no mesmo eixo (Tração Diferencial)

Este tipo de sistemas funciona com motores independentes nas rodas de um eixo, auxiliados por rodas livres (uma ou mais) que fornecem a estabilidade necessária à estrutura. Sendo de construção e controlo simples, este sistema permitem raios de curvatura bastante pequenos (de acordo com a dimensão do robot) e são bastante utilizados para pesquisa e desenvolvimento de novas metodologias de controlo.
Os motores utilizados neste tipo de sistema devem ter características semelhantes sob pena do controlo se tornar mais complexo.

a) b)

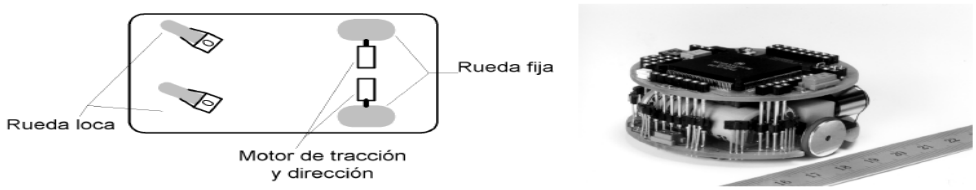


Figura 2.25 - Tração e direção no mesmo eixo. a) Características. b) Exemplo

Tração e Direção em todos os eixos

Este sistema baseia-se na capacidade individual de cada roda prover tração e direção em cada instante.
Por norma, devido à sua complexidade, este tipo de sistemas são utilizados apenas quando é necessária uma boa aderência ao terreno, por exemplo em aplicações off-road. A velocidade de translação não é valorizada neste sistema.
Este tipo de configuração necessita de um sistema odométrico complexo devido à incerteza nos raios de curvatura associados a cada roda.

a) b)

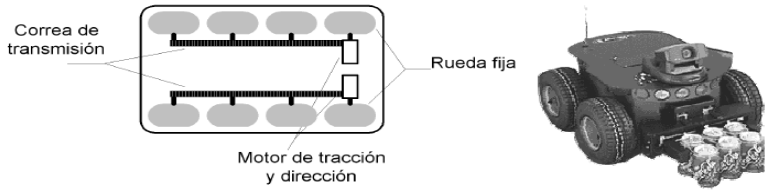


Figura 2.26 - Tração e direção em todos os eixos. a) Características. b) Exemplo.

2.1.3 Sensores

Os robots móveis caracterizam-se pela capacidade de se deslocarem de forma autónoma num ambiente parcial ou totalmente desconhecido.
Para isso, o robot tem que ser dotado com um sistema sensorial que o permita obter informação do meio envolvente, para depois extrair os seus aspetos mais significativos e reagir convenientemente a eventos imprevistos enquanto navega.
A qualidade da reação do robot é proporcional à quantidade e precisão da informação que lhe é fornecida pelos sensores, assim, raramente um robot é equipado com apenas um tipo de sensor. Idealmente, o robot teria capacidades sensoriais semelhantes ao ser humano, recebendo

várias informações do mundo exterior através de todos os seus sensores, podendo basear as suas reações com base num em específico ou na conjugação de vários.

2.1.3.1 Tipos de Sensores em Robótica

Existe uma grande variedade de sensores utilizados em robótica móvel. Alguns são usados simplesmente para obter valores como a temperatura interna ou a velocidade de rotação dos motores, enquanto outros, mais complexos, podem ser utilizados para obter informações do meio envolvente ou para determinar diretamente a posição global do mesmo.

Usualmente, os sensores são classificados segundo as suas características em dois eixos funcionais: Propriocetivos (internos) /Exterocetivos (externos) e Passivos/Ativos.

O eixo Propriocetivos/Exterocetivos refere-se à origem das informações sensoriais obtidas. Propriocetivos são sensores que medem valores internos do robot como a velocidade do motor, a voltagem da bateria ou os ângulos das juntas do robot. Pelo contrário, os sensores exterocetivos, adquirem informações do meio envolvente tais como distâncias, intensidade de luz ou amplitude de som.

Relativamente ao eixo Passivos/Ativos, este refere-se à forma como a recolha de informação é efetuada. Sensores passivos recebem apenas informações do exterior, como exemplo temos sondas de temperatura, microfones e câmaras. Os sensores ativos necessitam de emitir energia para o ambiente, sendo a resposta deste a essa energia a informação que recebem. Têm uma performance superior mas, contudo, são bastante suscetíveis a interferências. Exemplos destes sensores são encoders, sensores ultrassónicos ou lasers.

Indicam-se na tabela seguinte alguns exemplos de sensores utilizados em robótica bem como a sua classificação segundo as características que apresentam.

Classificação Geral	***Tipo de Sensor***	***Origem***	***Recolha de Informação***
Sensores Táteis	Barreiras Óticas	Exterocetivo	Ativo
Sensores de Motores/Rodas	Encoders Óticos	Propriocetivo	Ativo
Sensores de Orientação	Bússola Digital	Exterocetivo	Passivo
Sensores de Referencias	GPS	Exterocetivo	Ativo
Sensores Medição de Distancias	Sensores de Ultrasons	Exterocetivo	Ativo
Sensores Movimento/Velocidade	Radar Doppler	Exterocetivo	Ativo
Sensores Visão Artificial	CCD/CMOS Camaras	Exterocetivo	Passivo

Tabela 2.2 – Exemplos de sensores e sua classificação

Encoders

Este tipo de sensores são utilizados para determinar os estados internos e a dinâmica de um robot, permitindo controlar a velocidade angular e posição das suas rodas ou outro tipo de juntas acionadas por motores.

Um encoder é basicamente um gerador de impulsos baseado na interação de um feixe de luz com um disco perfurado, que converte um movimento rotativo ou linear em uma quantidade específica de impulsos elétricos de onda quadrada.

Existem vários tipos de encoders, sendo diferenciados pela sua forma de aplicação e resolução. Usualmente em robótica é utilizado o encoder em quadratura, neste, um segundo par fonte de luz e detetor é colocado desfasado 90º do par base.

Figura 2.27 - Encoder Ótico

Como estes sensores têm também vasta aplicação fora da robótica, as suas capacidades têm sido melhoradas e o seu preço reduzido.

Sensores de Orientação

Os sensores de orientação podem ser propriocetivos como o giroscópio e o acelerómetro ou exterocetivos como a bússola. Por norma, determinam a orientação e inclinação do robot para em conjunto com a informação da velocidade, possibilitar a determinação da posição atual.

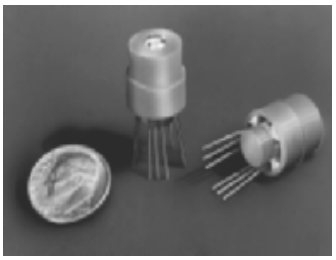


Figura 2.28 - Bússola Digital

Bússolas: Em robótica são utilizadas dois tipos de bússolas, uma baseada no efeito de Hall, que assenta em variações de potenciais e campos em semicondutores e outra baseada em fluxos, que deve o seu funcionamento, tal como o nome indica, a variações de fluxo nas bobinas que a constituem.
A opção entre os dois tipos é feita com base na sua precisão e no seu custo. As bússolas de efeito de Hall são menos onerosas, contudo são também menos precisas.

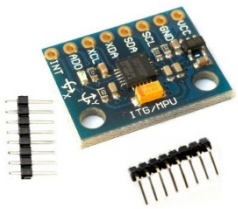


Figura 2.29 - Acelerómetro e Giroscópio Digital

Giroscópios: Os giroscópios são sensores de orientação que preservam a sua orientação em relação a um referencial fixo, o que os permite fornecer ao robot uma informação absoluta da sua orientação.
Podendo ser mecânicos ou óticos, diferem da forma como determinam a sua orientação. Enquanto os mecânicos dependem da inercia de rotação de estruturas, os óticos dependem da variação de frequências de dois feixes de raios lazer para determinar a sua velocidade angular.

Sensores de Referências – Global Positioning System (GPS)

Desde sempre o homem utilizou referências como estrelas, montanhas e outras estruturas para se localizar e orientar. Também na robótica, o recurso a referências externas é uma das formas de solucionar o problema da determinação da localização.

Após o desenvolvimento dos sistemas GPS, a utilização de referências para localização tornou-se frequente em sistemas robóticos.

A instalação de recetores de GPS nos robots, permitiu que a sua posição fosse estimada com precisão e velocidade, sendo assim uma alternativa aos sistemas que recorriam por exemplo a encoders.

De referir que estes sensores têm o seu funcionamento limitado no interior de edifícios, o que acaba por restringir o seu leque de aplicações em robótica.

Sensores de Medição de Distâncias

Este tipo de sensores continua a ser o mais utilizado em robótica. O seu baixo custo e a precisão e facilidade com que é determinada a distância a um obstáculo, tornam estes sensores a primeira escolha quando se pretende desenvolver um algoritmo de deteção e transposição de obstáculos.

Contudo, é de referir que estes sensores também apresentam algumas limitações. São bastante suscetíveis a interferências, dependem fortemente das características refletivas dos materiais presentes no meio e possuem um tempo de ciclo relativamente elevado, o que pode, consoante a velocidade a que circule o robot, ser uma limitação.

Por norma, estes sensores são classificados em sensores de medição de distâncias por tempo de voo ou sensores de medição de distâncias por propriedades geométricas consoante a sua forma de funcionamento.

Tempo de Voo

Figura 2.30 - Sensor Ultrasons

Sensores de Ultrasons: O funcionamento destes sensores baseia-se na transmissão de um pack ultrassónico de ondas de pressão e na medição do tempo que estas demoram a ser refletidas novamente em direção ao recetor.
A distância ao objeto que causa a reflexão pode ser calculada com base na velocidade de propagação do som e no tempo de voo determinado pelo sensor.

Figura 2.31 - Sensor Infravermelhos

Sensores Laser: Ao contrário dos sensores de Ultrasons, que se baseiam em ondas de pressão, o sensor laser, tal como o nome indica, utiliza um feixe laser para determinar a distância a que se encontra determinado obstáculo.
Sendo constituído por um emissor laser, e um recetor capaz de detetar a componente do feixe que é refletida ao embater no obstáculo, determina a distância a que este se encontra com base no tempo que o feixe laser demora a ser refletido para o recetor e na velocidade da luz.

Geométricos

Estes sensores diferenciam-se dos anteriores por recorrem a propriedades geométricas de padrões de luz para determinar a distância a que se encontram os obstáculos. Esta determinação,

frequentemente baseada em relações trigonométricas, é menos influenciável pela estrutura do objeto a detetar, sendo porém mais demorada.

Dividem-se usualmente, de acordo com as suas características funcionais, em dois tipos:

Sensores Triangulação Óticos: O seu princípio de funcionamento baseia-se na emissão de um feixe laser para o obstáculo, que quando refletido, é captado por uma lente que o orienta para um recetor sensível à posição. De acordo com a posição obtida no recetor, e tendo em conta a frequência do feixe, é possível determinar com precisão a distancia que os separa.

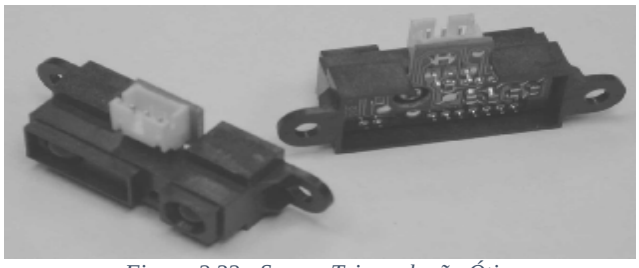


Figura 2.32 - Sensor Triangulação Ótico

Sensores de Luz Estruturada: Estes sensores diferenciam-se dos anteriores por emitirem um padrão de luz ao invés de um feixe único. Este padrão, que é conhecido, ao interagir com a estrutura de determinado objeto ou ambiente é alterado, sendo as diferenças interpretadas por um sensor de imagem.
A complexidade de implementação e o custo são duas desvantagens deste tipo de sensores, contudo, a determinação de distâncias a múltiplos pontos em simultâneo é uma clara mais-valia.

Sensores de Movimento e Velocidade – Efeito Doppler

Este tipo de sensores tem a capacidade de medir diretamente o movimento relativo entre o robot e o ambiente que o rodeia.
Recorrendo a um emissor de ondas eletromagnéticas ou sonoras de frequência definida, estes conseguem, através da análise da frequência do sinal recebido, determinar a velocidade relativa entre o emissor e o objeto onde se dá a reflexão das ondas emitidas.

Devido à sua grande precisão, mesmo a altas velocidades, estes sensores são utilizados por exemplo nos radares de controlo de velocidade ou em robots móveis que circulem a velocidades elevadas, por exemplo em vias rápidas.

Sensores de Visão Artificial

A visão, pela quantidade de informação do meio ambiente que nos fornece, é o nosso sentido mais poderoso.
Assim, não é de estranhar o esforço e dedicação presentes no desenvolvimento de sensores aplicáveis em robots capazes de replicar as capacidades humanas da visão.

Atualmente existem duas tecnologias para a criação de sensores de visão, a CCD (Charge-Coupled Device) [39] e a CMOS (Complementary Metal Oxide Silicon) [39].

É necessário ter em conta que tanto uma como a outra têm as suas limitações quando comparados com o sensor da visão humana, o olho. Assim, a opção por uma delas deve ser uma decisão bem ponderada, pesando os prós e contras de cada uma na aplicação onde serão utilizadas.

CCD – Charge Coupled Device

A tecnologia dos dispositivos de carga acoplada (CCD) [39] é a mais vulgarmente utilizada nos sistemas de visão robóticos.
Um sensor CCD é constituído por uma matriz de elementos sensíveis à luz, *pixéis*, que podem ser considerados como condensadores, que acumulam quantidades de carga variáveis consoante o número de fotões que cada um recebeu. Para cada imagem, as quantidades de carga são lidas e transferidas para circuitos auxiliares que as convertem para informação compreensível pelo restante equipamento onde será utilizada.

De referir que o processo de carga é independente da cor, assim, a obtenção de imagens coloridas resulta da existência de conjuntos de *pixéis* com filtros específicos para as cores fundamentais, sendo apenas sensíveis a estas. Estes *pixéis* podem estar agrupados no mesmo sensor ou podem ser utilizados simultaneamente três sensores CCD cada um destinado á receção de uma cor fundamental.
Este tipo de tecnologia tem como desvantagem o consumo de energia e a sua suscetibilidade a alterações das condições de captação de imagem, sobretudo a luminosidade.

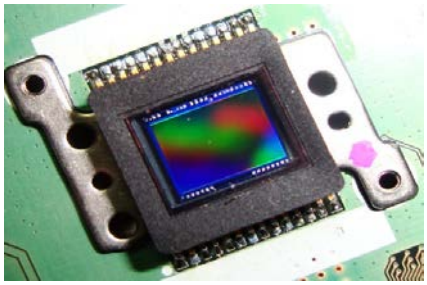


Figura 2.33 - Sensor CCD

CMOS - Complementary Metal Oxide Silicon

Os sensores que utilizam a tecnologia CMOS – Semicondutor Metal-Óxido Complementar [39], têm estrutura e funcionamento bastante semelhante aos sensores CCD.
A sua diferença mais proeminente é a presença junto a cada *pixel* de vários transístores dedicados ao processamento da informação recebida por estes.
Graças a esses transístores, a eletrónica necessária para os ciclos de descarga dos *pixéis* e obtenção de informação é bastante mais simples, o que torna o processo mais rápido, a sua produção mais acessível e o consumo de energia bastante mais reduzido.

Tal como seria de esperar, estes sensores também têm desvantagens. Devido ao espaço ocupado pelos circuitos específicos para cada *pixel* na face do sensor, estes tornam-se menos sensíveis que os CCD. O facto de ser uma tecnologia ainda relativamente recente, conduz a que a resolução dos sensores CMOS atuais também ainda seja reduzida quando comparada com a disponível nos CCD.

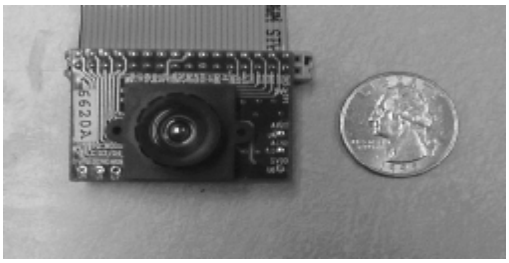


Figura 2.34 - Sensor CMOS com lente acoplada

2.1.4 Métodos de Navegação

A navegação autónoma de um robot móvel consiste na capacidade de se movimentar em determinado ambiente, detetando e evitando obstáculos ou outras situações que lhe possam surgir durante o seu percurso. Dotar um robot desta capacidade é uma tarefa bastante difícil, sobretudo devido às características do ambiente em que este estará inserido.
Segundo Russell [91], para um robot, o ambiente onde está inserido é:

- Não totalmente acessível
- Indeterminado
- Não totalmente previsível
- Dinâmico
- Continuo

Devido a esta complexidade, na robótica foram assumidas algumas simplificações em relação às características do mundo real. Desse modo, considerando um determinado ambiente estático e conhecido, foi possível desenvolver métodos para lidar com o problema da navegação autónoma. Tais métodos são oriundos da denominada Abordagem Planeada (também conhecida como Abordagem Deliberativa).

Abordagem Planeada

Neste tipo de abordagem, o robot conhece antecipadamente os obstáculos que encontrará no seu percurso. Este conhecimento, que provém do modelo do ambiente fornecido previamente ao robot, permite que este planeie o percurso de forma a evitar todos os obstáculos existentes. Como este modelo é considerado estático, sendo o robot o único objeto a movimentar-se no ambiente, não se esperam falhas ou situações imprevistas no movimento do robot até ao seu objetivo.

Os modelos fornecidos ao robot correspondem essencialmente a mapas sobre o ambiente no qual este irá interagir. São sobretudo utilizados dois tipos de modelos [100]:

- **Modelos Baseados em Células**: Nestes o ambiente é representado por um conjunto de células onde cada uma corresponde a uma região do ambiente. Cada região apresenta uma probabilidade de estar ou não ocupada (pode ser utilizada a ocupação binária, apenas com probabilidade 0 e 1).

- **Modelos Topológicos**: Consideram que o ambiente é representada por um grafo, onde os nós correspondem a diferentes situações, lugares ou marcas.

Sistemas desenvolvidas segundo esta abordagem, por norma são bastante estáveis, porém não se adaptam com facilidade a ambientes onde as mudanças ocorrem rapidamente. Seguindo esta arquitetura, algoritmos de planeamento de trajetória, auto localização e deteção de obstáculos são implementados para a realização dessas tarefas [94].

Abordagem Reativa

Em contraponto com a abordagem planeada, nesta, os movimentos ou ações de um robot são resultado de estímulos recebidos durante a sua movimentação no ambiente.
Estes estímulos, provenientes dos sensores presentes no robot, permitem que este navegue em ambientes complexos, desconhecidos e que apresentem características dinâmicas [10].

A arquitetura mais frequente neste tipo de abordagem é conhecida como arquitetura de subsunção[3] ou de Brooks [10]. Nesta, que se baseia na sinergia entre sensação e reação, as tomadas de decisão são realizadas através de um conjunto de comportamentos direcionados a tarefas a cumprir.

A reação aos estímulos é bastante rápida, sendo a autonomia do robot considerável, contudo, devido a este não ter qualquer capacidade cognitiva, não havendo nenhum conhecimento prévio nem sendo armazenada qualquer tipo de informação no decorrer da movimentação, a implementação destes sistemas em tarefas mais complexas reveste-se de grandes dificuldades [67].

Abordagens Híbridas

A abordagem hibrida surge da união dos melhores aspetos das abordagens planeada e reativa [67], o que a torna mais adequada e funcional para aplicações mais complexas, onde individualmente as 2 abordagens originárias não são bem-sucedidas.
Normalmente, a abordagem hibrida é aplicada de duas formas distintas:

- **Pré-Mapeamento**: Nesta forma, tal como na abordagem planeada, um modelo prévio do ambiente é fornecido ao robot. A partir deste, é possível realizar o primeiro planeamento dos movimentos a executar. Durante a execução desses movimentos, surgem comportamentos reativos como resposta às informações recolhidas pelos sensores. Nomeadamente no caso da presença de um obstáculo, surgirá uma reação que o permitirá evitar.

- **Construção e Atualização de Mapas em Tempo de Execução:** Esta forma de abordagem híbrida centra grande atenção nas capacidades sensoriais do robot.
  O robot planeia a sua movimentação com base num mapa do ambiente que é construído dinamicamente enquanto o robot se movimenta, através das informações sensoriais que vai recolhendo do meio. Em 2001, Tomatis et al. [102] propuseram uma arquitetura de controlo que combina reação, planeamento e capacidade de aprendizagem.

[3] Esta arquitetura, uma das mais representativas do paradigma puramente reativo, é caracterizada por os comportamentos corresponderem a módulos que mapeiam uma informação sensorial numa ação motora. Os comportamentos são interligados formando uma rede organizada em camadas de competência, sendo cada uma responsável por uma atividade do robot. Os comportamentos em cada camada funcionam de forma concorrente e independente.

Navegação Autónoma Baseada em Visão

Para nós, seres humanos, a visão desempenha um papel fundamental no quotidiano, podendo até ser considerada o nosso mais importante sentido. A quantidade e qualidade da informação sobre o ambiente que nos é fornecida por este é incomparavelmente superior à que os restantes sentidos nos proporcionam.
Tendo em conta este facto, pesquisadores na área da robótica móvel como DeSouza e Kak [19] aperceberam-se que seria uma clara mais valia apostar também nesta área sensorial. Assim, a navegação baseada em visão tornou-se um dos métodos mais relevantes e sobre o qual ainda recai grande expectativa em relação ao futuro.

A implementação de um sistema de navegação autónoma baseado em visão, consiste simplificadamente em dotar determinado robot da capacidade de adquirir representações visuais do ambiente onde se encontra e a partir destas, determinar a trajetória mais adequada para o seu movimento [62].

Para isso, é adicionado ao robot um dispositivo de captura de imagens que será responsável pela aquisição de imagens e extração das informações visuais mais pertinentes para a tarefa em causa. Estas informações, depois de tratadas, são fornecidas ao sistema de controlo que gerará os comandos apropriados para que o veículo possa ter em tempo real, uma navegação segura, baseada na informação visual obtida [82].
Por norma considera-se neste tipo de navegação a existência de dois sistemas, o sistema de aquisição de imagens, que corresponde ao dispositivo de captura de imagens e o sistema de controlo, que pode ou não estar presente na estrutura do robot e que será o responsável pelas decisões.

Face à complexidade da tarefa em causa, o desenvolvimento de um sistema de navegação autónoma deve ter em consideração vários aspetos [62]:

- Identificação da informação pertinente para a tarefa.
- Definição da forma de extração da informação das imagens.
- Definição da forma de controlo com base na informação recolhida.

Sendo a forma mais correta de o implementar dividida nas quatro tarefas seguintes segundo [8]:

1. **Adquirir informação sensorial:** Obter imagens provenientes de uma câmara.
2. **Detetar pontos de referência:** Identificar nas imagens recolhidas os contornos, regiões de interesse, distâncias ou movimentos.
3. **Estabelecer correlações entre o observado e o esperado:** Neste ponto o sistema de controlo deverá tentar comparar os pontos de referência observados com correspondentes informações previamente armazenadas no sistema.
4. **Calcular a posição:** Após a identificação dos pontos de referência, deve ser determinada a posição dos mesmos.

Referem-se algumas das perspetivas desenvolvidas nesta área:

- Pio, em 2002 [82] apresenta uma proposta de navegação para um robot aéreo onde este é equipado com uma única câmara e um transmissor rádio que são responsáveis pelo processamento das imagens e envio das mesmas ao sistema de controlo.

- Araujo e Librantz, em 2006 [1], apresentam um sistema que utiliza mecanismos de visão computacional que descrevem o ambiente para que o robot possa tomar decisões. Com base nas informações recebidas sobre o seu posicionamento, posição do alvo e dos obstáculos, este deve decidir o trajeto a fazer para atingir o objetivo.

- Kim, Bok e Kweon, em 2008 [61], apresentaram um sistema de navegação autónoma capaz de lidar com alterações no ambiente e objetos em movimento. Este, sustentado numa fase inicial de treino e num algoritmo de previsão de movimento, é capaz de seguir um percurso estabelecido, evitando os obstáculos que possam surgir na sua trajetória.

Em várias perspetivas desta área é salientada a utilidade de algum conhecimento prévio do meio envolvente ao robot e da incorporação de parte da informação recolhida durante a sua movimentação.

Com base nestas, é também possível verificar que a forma mais adequada de proceder ao controlo dos movimentos deverá ser:

- **Controlo em Malha Aberta:** Esta forma de controlo é caracterizada pelo facto da extração de informação da imagem e o controlo do robot serem tarefas que ocorrem em momentos diferentes.
  Inicialmente é efetuada a captura e o processamento de imagem, sendo após isso determinada a sequência de controlo mais adequada a enviar para o robot.

- **Controlo em Malha Fechada (Com Retroação):** Nesta, os processos de captura e processamento de imagem e a determinação dos comandos a enviar ao robot são efetuados em simultâneo.
  Com base nas imagens recebidas dos sensores, é gerada uma sequência de controlo que visa ajustar o posicionamento, diminuindo o erro relacionado à velocidade de deslocamento e posicionamento. Estes sistemas são também designados como servo visuais [82].

Sendo consoante o ambiente onde o robot se irá movimentar:

Navegação no Interior

Segundo [19] a navegação interior baseada em visão pode ser subdividida em três metodologias:

- **Navegação sem Mapa:** Nesta, nenhum conhecimento prévio do ambiente é fornecido ao robot. Os movimentos deste são determinados com base na informação relevante sobre o meio ambiente fornecida a cada instante pelos dispositivos de captura de imagens. É vulgarmente considerado [7] que esta metodologia pode ter dois aspetos

diferentes, **a Navegação utilizando o fluxo ótico,** que se baseia na análise do movimento de objetos ou características em uma sequencia de imagens [93] **e a Navegação por correspondência baseada em aparência,** onde o robot navega em determinado ambiente, comparando as imagens capturadas com outras, pré armazenadas na memória do robot, determinando a partir daí as ações a seguir.

- **Navegação baseada sem pré-mapeamento** Já referenciada na Abordagem Reativa.

- **Atualização de mapas em tempo de execução:** Já referenciada na Abordagem Hibrida.

Navegação no Exterior

Este tipo de navegação refere-se àquela que decorre no exterior, em ambientes que se podem considerar estruturados ou não.

Em ambientes não estruturados, não existem tal como o nome indica, estruturas ou propriedades regulares que possam servir de base para a navegação. Assim, podem verificar-se dois tipos de situações:

- **Exploração Aleatória:** O robot explora aleatoriamente o ambiente.

- **Exploração por Objetivo:** O robot executa uma missão com o objetivo de atingir uma determina determinada posição. Neste caso, um mapa das áreas em que o robot se move tem de ser desenvolvido e um algoritmo de localização é também necessário.

2.2 *Sistemas de Controlo*

Como abordado no capítulo anterior, o controlo da movimentação de um robot móvel autónomo depende essencialmente da informação recolhida do exterior e de um controlador adaptado e dedicado a essa tarefa.

Este controlador, recorrendo à informação fornecida pelo operador e pelos sensores presentes no robot, determina qual a ação a efetuar para que determinado objetivo seja atingido. No caso especifico da robótica móvel, este objetivo corresponde a uma posição em determinado percurso.
São utilizados para este tipo de tarefa desde simples controladores On-Off como em [78], controladores PID [79], controladores *Fuzzy* [84], controladores baseados em redes neuronais [72], controladores baseados em algoritmos genéticos [16] e controladores resultantes da fusão de alguns dos referidos, como é o caso do controlador *Neuro-Fuzzy* em [75] e Neuro-PID em [81].

Frequentemente, devido aos bons resultados que proporcionam e à facilidade com que são implementados a opção recai sobre os controladores PID e os controladores *Fuzzy*.

Neste trabalho são usados estes dois tipos de controladores, um controlador PID de baixo nível para o controlo de velocidade dos motores e um controlador *Fuzzy* que será responsável pelo movimento do robot.

Controlador PID

O controlador PID (Proporcional, Integral, Derivativo) empregue em [79] é amplamente utilizado em sistemas de controlo industriais em que existe feedback do estado do sistema. Sendo constituído por elementos com funções proporcionais, integrais e derivativas, a sua popularidade pode ser atribuída em parte ao seu desempenho robusto numa ampla gama de condições de funcionamento e em parte à sua simplicidade funcional.

Estes controladores, influenciados pelo feedback do sistema, determinam o erro entre o valor objetivo e atual e atuam nas entradas no sentido de minimizar este erro.

De referir que o projeto e ajuste do controlador, apesar de não serem muito complexos, implicam o conhecimento de várias características do sistema para que este possa ser modelizado, o que para algumas aplicações pode ser relativamente complicado.

Matematicamente um controlador PID pode ser representado pela seguinte equação:

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{d}{dt}e(t)\right) \quad (2.1)$$

Onde K_p corresponde ao ganho proporcional, T_i é a constante de tempo Integral, T_d é a constante de tempo derivativa e $e(t)$ é o erro entre o objetivo e o valor atual.

Controlador *Fuzzy*

A lógica *Fuzzy* [68] foi desenvolvida por Lofti A. Zadeh, Engenheiro de sistemas, durante a década de 1960. Esta lógica, pretende traduzir valores imprecisos, qualitativos e verbais, comuns na comunicação humana, em valores numéricos, incertos, imprecisos ou difusos.

Com o surgimento desta lógica, foi possível desenvolver controladores sustentados na mesma, que não fazem mais do que tentar incorporar nos sistemas de controlo experiencias heurísticas de um operador humano, ou seja, tentam replicar a forma de pensar humana.

Estes controladores revelam-se um grande sucesso, apresentando vantagens face às alternativas existentes como, simplicidade de implementação, fácil compreensão e ajuste devido ao uso de variáveis heurísticas, menor esforço computacional, simplificação do problema com recurso à experiencia do especialista humano e dispensa da modelização do sistema a controlar, evitando assim a determinação de equações complicadas.
Porém, sendo desenvolvidos empiricamente, necessitam de mais testes e consequentes ajustes, necessitam de conhecimento humano aprofundado sobre o comportamento do sistema e não existem definições matemáticas que o permitam representar.

Considera-se que um controlador *Fuzzy* [84] consiste nos seguintes blocos funcionais:

- ***Fuzzyficação* -** Faz a identificação dos valores das variáveis de entrada. Estes valores são "*Fuzzyficados*" em conjuntos *Fuzzy* para que se possam tornar instâncias de variáveis linguísticas.

- **Base de conhecimento -** Consiste numa base de dados e uma base de regras. A base de dados fornece as definições numéricas às funções de pertinências usadas no conjunto de regras *Fuzzy*. A base de regras caracteriza os objetivos e a estratégia de controlo utilizada por especialistas na área, por meio de um conjunto de regras normalmente linguísticas.

- **Lógica de tomada de decisões –** É a etapa onde é gerada a ação de controlo a partir da estrutura de base de regras. São decisões que simulam a atitude de um ser humano a controlar o sistema

- ***Defuzzyficação* -** Consiste na obtenção de um único valor discreto, utilizável numa ação de controlo concreta no mundo real, a partir de valores *Fuzzy* de saída obtidos. A obtenção destes valores *Fuzzy* de saída é efetuada através de dois princípios básicos de *Defuzzyficação*. O primeiro baseado no centróide e o segundo em valores máximos.

2.3 *Reconhecimento de Objetos*

O reconhecimento de objetos é um dos principais focos de interesse da visão computacional. Sobretudo em aplicações na área da robótica, algoritmos de reconhecimento são a base de grande parte dos sistemas implementados.

O desenvolvimento destes algoritmos é bastante complexo, tendo que prever e ultrapassar dificuldades na deteção devido a variações em ângulos e escalas, variações fotométricas que incluem mudança de brilho e contraste e deformações em perspetiva devido à mudança na posição do observador.

Dependendo das necessidades da aplicação, este tipo de tarefa pode ser realizado de várias formas, agrupáveis essencialmente em duas metodologias diferentes, uma baseada no aspeto global do objeto e outra em características particulares da sua forma que o permitem identificar.

Metodologia Baseada na Aparência

Os algoritmos de reconhecimento representativos desta metodologia revelam-se como os mais simples de implementar. Baseando a sua capacidade de reconhecimento numa operação genérica para determinar a semelhança entre figuras, a imagem é comparada diretamente com modelos armazenados do objeto que têm que ser totalmente representativos, ou seja, têm que ilustrar todas os tipos e variações possíveis (translação, rotação e mudanças de escala) do mesmo.

Não são métodos computacionalmente muito exigentes, porém, a pesquisa de todos os modelos presentes na base de dados em determinada imagem pode torná-los relativamente lentos. De referir ainda que pequenas variações aos modelos existentes, quer seja a nível de cor, luminosidade ou forma podem inviabilizar a deteção.

Podem-se referir os algoritmos por coincidência de gradientes, como por exemplo em [103] e comparação de histogramas em [90] como os mais utilizados e melhor sucedidos.

Metodologia Baseada em Características Particulares

Nesta abordagem, o reconhecimento de determinado objeto resulta da procura de características estruturais representativas do mesmo na imagem a analisar.

Os algoritmos implementados visam sobretudo tornar o reconhecimento imune às variações referidas anteriormente, que se podem revelar impeditivas da deteção. Para isso, baseiam-se na extração de características específicas, normalmente pertencentes à sua superfície, cantos ou arestas.
Podem-se referir como exemplo desta metodologia o reconhecimento por características geométricas [24] e por aspetos invariantes [27].

Dois dos métodos mais utilizados para a extração de características e reconhecimento de objetos são o SIFT (Scale Invariant Feature Transform) [65] e o SURF (Speeded Up Robust Features) [4]. Estes devem grande parte do seu sucesso à robustez e ao facto de se centrarem em características invariantes à escala e rotação, o quer permite que o reconhecimento seja eficaz em diversos tipos de situações.

SIFT - Scale Invariant Feature Transform

O SIFT [65] é um algoritmo de visão computacional desenvolvido por David G. Lowe que é utilizado frequentemente para o reconhecimento de objetos através da deteção e extração de descritores invariantes em escala e rotação.
Estes, são bastante utilizados em aplicações envolvendo a correspondência de diferentes imagens de um objeto ou cena devido a serem razoavelmente invariantes a mudanças de iluminação, ruído na imagem, rotação, escala e pequenas mudanças de ponto de vista.

Este algoritmo é constituído por dois elementos distintos, o detetor que é implementado com recurso a funções DoG (“*Difference of Gaussian*”)[4] e o descritor, que se obtém com base em histogramas de orientações da vizinhança local dos pontos-chave.

Resumem-se de seguida as principais fases da obtenção dos descritores de uma imagem:

- **Deteção de extremos no espaço-escala** - Neste primeiro estágio é efetuada a busca de pontos-chave em todas as escalas e locais de uma imagem. Isto é efetuado com recurso a funções DoG de modo a identificar potenciais pontos-chave[5] invariáveis à escala e orientação.

- **Localização de Pontos-Chave** - Para cada local candidato (onde foi detetado um extremo máximo e mínimo), um modelo detalhado é ajustado para determinar a localização e escala. Pontos-chave são então selecionados baseando-se na sua estabilidade.

[4] Algoritmo que salienta características particulares de imagens com recurso a subtrações de imagens processadas por filtros gaussianos a diferentes escalas.
[5] Pontos da imagem que apresentam características que os tornam distinguíveis dos restantes.

- **Determinação de orientação -** Baseadas nas direções do gradiente local da imagem, uma ou mais orientações são associadas para cada localização de pontos-chave. As operações seguintes são realizadas nos dados da imagem relativamente transformados em relação à orientação, escala e localização de cada ponto-chave, tornando-as assim invariantes a estas transformações.

- **Descritor de Pontos-Chave** - Nesta etapa, os gradientes locais da imagem são avaliados na escala selecionada em redor de cada ponto-chave. Os resultados desta avaliação são organizados numa representação das características do ponto-chave.

Após a obtenção dos descritores para cada ponto, a tarefa de encontrar a correspondências entre imagens é resumida a encontrar entre os descritores de uma imagem, os melhores candidatos a serem os seus equivalentes na outra imagem.

De referir que este algoritmo e outros que deste derivam são bastante utilizados no reconhecimento de sinais de trânsito, como é possível verificar nos trabalhos [12, 57].

SURF – Speeded Up Robust Features

O SURF [4] é um método de deteção e descrição de características invariantes em escala e rotação inspirado no algoritmo SIFT [65], tendo portanto atributos idênticos.

Desenvolvido com o intuito de tornar o processo de identificação de aspetos semelhantes entre imagens mais célere, a principal diferença face ao SIFT [65] reside na fase de deteção dos pontos-chave, que neste é efetuada com recurso a um detector Hessiano[6].

De referir que neste método, a determinação e atribuição da orientação aos pontos-chave detetados se sustenta em transformadas Haar Wavelets[7] [97].

A busca por correspondências segundo este método pode ser dividida essencialmente em três fases:

1. Encontrar pontos-chave na imagem, definindo-os como aqueles que sejam identificáveis em diferentes condições de imagem.

2. Representar por um vetor de características ou descritores a vizinhança dos pontos-chave. Devem ser distintivos e robustos ao ruído, às deformações geométricas da imagem e aos erros.

3. Comparar os vetores descritores da imagem com o objeto. A comparação é geralmente baseada na distância entre vetores.

[6] Detetor sustentado na matriz Hessiana (matriz composta por derivadas parciais de segunda ordem) capaz de identificar zonas da imagem onde se verificam variações acentuadas de intensidade.
[7] Transformada baseada nas funções de Haar, que permite descrever de forma expedita as características locais de uma imagem com recurso a um conjunto de coeficientes.

Os descritores destes pontos têm vulgarmente 64 elementos mas podem ser de dimensão superior, o que favorece a eficácia da deteção, implicando porém uma menor agilidade computacional.

2.4 ***Condução Autónoma***

A condução autónoma, tal como o nome indica, é a técnica que pretende dotar robots móveis da capacidade total ou parcial de realizar tarefas relacionadas com a condução sem intervenção de um condutor.

Os benefícios dos desenvolvimentos nesta área não se ficam pela componente tecnológica. A nível social e económico, uma vez que nesta área também se inclui a evolução de sistemas de segurança ativa e passiva e sistemas auxiliares à condução (no casos de condições de condução adversas ou de utilizadores com algum tipo de deficiência ou incapacidade para conduzir) as mais-valias também são claras.

A aplicação desta tecnologia aos automóveis será um dos maiores avanços verificados neste sector. A capacidade de autonomamente circular e evitar acidentes e lesões, contribuirá para a segurança, eficiência e conforto na utilização do veículo automóvel.

Esta tecnologia teve a sua génese em 1977 quando, no Japão, o Laboratório de Engenharia Mecânica Tsukuba desenvolveu um robot [46] capaz de seguir linhas brancas semelhantes às que delimitam faixas de rodagem.

Referem-se de seguida alguns dos principais marcos na evolução desta vertente da robótica móvel:

- Em 1980, uma carrinha Mercedes-Benz, desenhada por Ernst Dickmanns e pela sua equipa na Universidade de Bundeswehr em Munique [42], atingiu, usando visão, os 100 km/h numa rua sem trânsito.

- Ainda em 1980, o Autonomous Land Vehicle (ALV) [41] patrocinado pela DARPA conseguiu, pela primeira vez, controlar um veículo sem intervenção humana, a uma velocidade de 30 km/h, usando radares laser e visão artificial.

- Em 1994, dois veículos robóticos, VaMP e Vita-2, circularam mais de mil quilómetros numa autoestrada Parisiense de três vias de forma semiautónoma, em condições típicas de trânsito intenso, atingindo a velocidade máxima de 130 km/h [54]. Foram os primeiros a demonstrar a condução integrando uma coluna de veículos, mudanças de faixa e ultrapassagem de outros veículos.

- Em 1995, um Mercedes-Benz Classe S modificado por Dickmanns [42], realizou uma viajem de ida e volta entre Munique e Copenhaga, usando uma técnica de visão computorizada que simula o comportamento do olho humano ao procurar por pontos de interesse numa imagem.

- Entre 1996 e 2001, o governo Italiano patrocinou o projeto ARGO [38], que decorreu na Universidade de Parma e foi coordenado pelo Professor Alberto Broggi. No âmbito deste projeto, procedeu-se à modificação de um Lancia Thema para que este circulasse

autonomamente, seguindo linhas delimitadoras das faixas de rodagem de uma autoestrada normal, e em simultâneo, monitoriza-se o ambiente à sua volta por forma a localizar e evitar obstáculos no seu caminho.

- Em 2004, teve lugar a primeira edição de uma das maiores competições a nível mundial no que toca a condução autónoma, o DARPA Grand Challenge. Esta competição, patrocinada pela DARPA, a mais importante organização em termos de investigação do Ministério da Defesa dos Estados Unidos, consiste na competição entre equipas originárias de vários países.

- Em Maio de 2006, realiza-se a primeira edição do European Land-Robot Trial (ELROB) um evento que, através de dois cenários, permite às equipas participantes a demonstração dos mais recentes avanços alcançados na construção de veículos terrestres autónomos.

- Em Novembro de 2007 tem lugar a terceira edição do DARPA Grand Challenge, apelidada de Urban Challenge. Tendo lugar num recinto fechado (uma base aérea desmantelada), consistiu num desafio diferente em relação às edições anteriores. Nesta, existiram oito dias de rigorosos testes, em três cenários diferentes. Numa primeira secção, era avaliada a capacidade dos veículos lidarem com situações de mudanças de direção num circuito fechado de duas vias, com trânsito em ambos os sentidos. Numa segunda secção, eram avaliadas as capacidades de navegação, parqueamento e desvio de carros parados na via. Na terceira e última secção, os veículos tinham de lidar com cruzamentos, respeitando as regras de prioridade, e situações de estrada sem saída (para avaliar qual a capacidade dos robots em recalcular um novo percurso para completar o objetivo pretendido). O Urban Challenge constituiu, então, um enorme avanço na área da condução autónoma, na medida em que, pela primeira vez, interagiram num mesmo cenário veículos autónomos e veículos conduzidos pelo homem, sendo que todos eram obrigados a cumprir o código da estrada em vigor na Califórnia, estado onde decorreu a prova.

2.5 ***Robótica em Portugal***

Tal como no resto do mundo, em Portugal a Robótica é uma das áreas da ciência que beneficia de maior atenção, sendo considerável a atividade educativa e de investigação em torno das suas diferentes vertentes.

As universidades são as principais impulsionadoras desta área, em colaboração com os sectores empresariais e industriais, desenvolveram vários protótipos que se vieram a revelar uteis à sociedade, passando a ser produzidos em massa.

A distribuição deste tipo de produtos pela sociedade foi bastante benéfica para a expansão da sua abrangência, deixando de estar apenas confinada ao ambiente fabril e à automação, mas estendendo-se também à área de serviços (quer seja em casas, escritórios, hospitais ou no exterior).

Um dos marcos fundamentais para o desenvolvimento deste sector foi a 1ª edição do Festival Nacional de Robótica em 2001. Este evento tem como objetivo contribuir para o desenvolvimento da investigação em robótica e automação e promover a ciência e tecnologia

junto de jovens dos vários níveis de ensino, bem como do público em geral, através de competições entre robots.
O Festival decorre todos os anos em uma cidade distinta e inclui um encontro científico, onde investigadores nacionais e estrangeiros, da área da robótica, se reúnem para apresentar os mais recentes resultados da sua atividade.

Devido a tudo isto, a noção de robótica também foi sendo alterada, deixando de se restringir apenas ao manipulador ou ao veículo com rodas e alguns sensores, passando a abranger uma vasta gama de dispositivos, com diferentes graus de sofisticação e complexidade que possuem a capacidade de comunicar entre si formando uma rede de sensores autónomos e distribuídos permitindo a monitorização de vastas zonas geográficas e de ambientes específicos.
Destacam-se de seguida alguns dos projetos desenvolvidos em Portugal nas várias vertentes da robótica:

- Em 2003, foi criada na Universidade de Aveiro a CAMBADA (Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture) [36], uma equipa de futebol robótico que iria participar no Festival Nacional de Robótica. Na liga de robots médios, onde a equipa compete, equipas de no máximo seis robots maioritariamente pretos e totalmente autónomos (todos os mecanismos sensoriais estão a bordo do robot e todas as decisões são tomadas sem qualquer intervenção humana) com valores limite de 50cm para o diâmetro, 80cm para a altura e 40Kg para o peso, disputam jogos com duas partes de quinze minutos em um campo com cores bem definidas de 18x12 metros.

- O Laboratório de Sistemas Autónomos (LSA) do instituto Superior de Engenharia do Porto desenvolveu os projetos ROAZ, FALCOS e LINCE [45].
  O projeto ROAZ foca-se no desenvolvimento e construção de veículos autónomos de superfície para utilização em missões de salvamento, monitorização, segurança e de carácter científico em ambientes marítimos.
  O FALCOS é um sistema autónomo aéreo concebido para operar a baixa altitude e em missões de prevenção de fogos florestais, monitorização ambiental e captação de imagens aéreas. Possui asas com comprimento inferior a 2 metros, é alimentado a baterias e tem um módulo dedicado à deteção de incêndios.
  O LINCE é uma plataforma terrestre com potencial para ser usado em missões em terrenos irregulares, desconhecidos e com condições variáveis (catástrofe em ambiente urbano, áreas exteriores de difícil acesso e estabelecimento ou suporte a comunicações). Podem ser utilizados de forma cooperante em missões de busca e salvamento.

- Uma equipa de investigação composta por elementos do Instituto Pedro Nunes e do Instituto de Sistemas e Robótica da Universidade de Coimbra, visando a conceção de um sistema de condução autónoma para transporte de pessoas e bens, estão a estudar a criação de um Kit passível de ser instalado em qualquer AGV (Autonomous Guided Vehicle) disponível no mercado que os dotaria de mecanismos de deteção e desvio de obstáculos baseados em sensores de Ultrasons e laser.

- O projeto ATLAS [29], que surgiu em 2003, conseguiu o 4º lugar na competição nacional de robótica recorrendo a uma arquitetura Ackermann e a um sistema de visão baseado numa webcam que adquiria imagens através de um espelho, de modo a obter uma visualização de toda a pista. Desde essa altura foi continuamente evoluído,

apresentando em cada edição da competição novas capacidades, como câmaras com maiores ângulos de abertura e novas técnicas de controlo e processamento de imagem.

- O robot VERSA [34] foi desenvolvido na FEUP e participou na competição nacional de robótica em 2005. Possuindo locomoção diferencial, está equipado com um sistema de visão composto por duas câmaras, uma para detetar as linhas no solo e outra para detetar semáforos e sinais de trânsito, sensores de codificação para medir a distância percorrida por cada roda e sensores de Ultrasons distribuídos pela frente e laterais do robot para detetar obstáculos.

- No âmbito da dissertação do mestrado integrado, Héber Sobreira [95] desenvolveu um robot para fins publicitários, constituído por um robot de locomoção diferencial, munido de sensores de Ultrasons para detetar presença de degraus e obstáculos, encoders para determinar a distância percorrida por cada roda e por fim um recetor infravermelho para detetar sinais codificados por frequência provenientes de duas balizas equidistantes externas. Pretendia-se que este projeto não tivesse necessidade de recalibrações sempre que o hardware mude de lugar.

- Tendo sido desenvolvido no início de 2012, como projeto final de curso de MIEEC por André Almeida Vidal, O FEUPCar 2.0 [105] é um robot de locomoção Ackermann destinado à participação no concurso nacional de robótica na competição de condução autónoma. Este robot estava equipado com três câmaras, duas para detetar as linhas laterais da pista e outra para detetar e identificar sinais de trânsito e sensores de odometria de forma a determinar a posição do robot no mundo.

2.6 *Linguagens de Programação*

A implementação de um sistema computacional como o necessário à condução autónoma, implica o desenvolvimento de um algoritmo capaz de desempenhar as funções pretendidas e a sua codificação para que o computador esteja apto a executá-lo.
Esta codificação, que não é mais que a discrição da sua estrutura, métodos e objetivos, é conseguida por intermédio de uma "linguagem de programação".

Ao longo da história da computação, várias "linguagens de programação" foram sendo desenvolvidas, cada qual, a seu tempo, introduzindo facilidades e recursos que foram tornando a tarefa de programar progressivamente mais abrangente, acessível a utilizadores não especializados e menos suscetível a erros.

Contudo, determinada linguagem pode não ser adequada para todas as tarefas necessárias a determinada aplicação, fatores como a complexidade dos algoritmos necessários, os interfaces e as bibliotecas a utilizar ou a velocidade de execução pretendida, tornam o processo de seleção da linguagem a utilizar um passo crucial para a expedita e eficiente implementação do sistema.

2.6.1 Linguagens de Baixo Nível

As linguagens de programação de baixo nível são as mais próximas do código nativo da máquina, correspondendo quase que diretamente ao código de máquina que será enviado ao processador para execução.

Este tipo de linguagens dependem substancialmente da arquitetura utilizada, levando à existência de várias variantes das mesmas, consoante o processador onde serão interpretadas.

Podem-se referir as linguagens Código de Máquina [25], interpretado diretamente por um processador e Assembly [86], que associa instruções do processador a termos cuja semântica corresponde à operação efetuada, como as mais representativas deste tipo.

2.6.2 Linguagens de Alto Nível

As linguagens de alto nível [25] são caracterizadas por apresentarem um nível de abstração elevado, longe do código de máquina e próximo à linguagem humana. Fazendo uso de palavras reservadas extraídas do vocabulário corrente e dados nas mais diversas formas, tornam-se mais fáceis de ler e escrever.
Estas linguagens não estão diretamente relacionadas à arquitetura do computador, permitindo assim que o programador não precise conhecer características do hardware como instruções e registos para desenvolver determinada aplicação.

Os programas escritos nessas linguagens são convertidos para a linguagem de máquina através de um programa compilador ou de um interpretador.
De seguida, descrevem-se as linguagens mais representativas deste nível, agrupadas segundo o seu paradigma de programação, ou seja, a sua estruturação e execução:

2.6.2.1 Linguagens Não-Estruturadas

As linguagens não-estruturadas [25] corresponderam a um grande salto qualitativo em termos de programação quando comparadas com as linguagens de baixo nível, contudo, o desenvolvimento das linguagens estruturadas, com as suas evidentes mais-valias, depressa tornaram as linguagens não estruturadas obsoletas.

Como exemplos mais significativos deste tipo de linguagens podemos referir a Cobol [25], desenvolvida em 1959 e vocacionada para o tratamento de grandes volumes de dados e a Basic [60], que surgiu em 1963 com o intuito de tornar possível por parte dos programadores informáticos, o rápido desenvolvimento e execução de novas aplicações computacionais.

2.6.2.2 Linguagens Estruturadas

As linguagens estruturadas [25] surgiram como uma forma de programação que estabelece uma disciplina de desenvolvimento de algoritmos independente da sua complexidade e da linguagem de programação na qual são codificados. O número restrito de mecanismos de codificação utilizados incrementaram de sobremaneira a compreensão e legibilidade do algoritmo implementado.
Estas linguagens utilizam formas de raciocínio intuitivamente óbvias, sustentadas sobretudo em três mecanismos de controlo, a sequência, a condição e a repetição.
Além da definição dos mecanismos anteriores, nestas linguagens, programas mais complexos podem ser divididos em partes funcionais menores, tais como procedimentos, funções, módulos, métodos e sub-rotinas, o que simplifica e melhora a clareza do programa desenvolvido.

Devido a esta subdivisão, a "reutilização de software", impulsionada pela necessidade de economia de tempo e custo sai também beneficiada.

2.6.2.3 Linguagens Procedimentais

As linguagens Procedimentais [25] são caracterizadas pela existência de algoritmos, que determinam uma sequência de chamadas de procedimentos que constituem o programa. Essa característica é colocada em contraposição às linguagens funcionais, onde se descrevem expressões que caracterizam um certo tipo de conhecimento.

Neste tipo são de destacar a linguagem Pascal [43] e a linguagem C [25].

2.6.2.4 Linguagens Funcionais

As linguagens funcionais [25] evidenciam um estilo de programação bastante diferente das linguagens procedimentais, enfatizam a avaliação de expressões, ao invés da execução de comandos.

As expressões nestas linguagens são formadas utilizando funções para combinação de valores básicos.
Os marcos fundamentais deste tipo de linguagem foram o surgimento da linguagem LISP [86] (1958) e da linguagem Prolog [20] em 1973.

2.6.2.5 Linguagens Orientadas a Objetos

O termo Programação Orientada a Objetos [25] foi primeiramente aplicado por Alan Kay, autor da linguagem de programação Smalltalk, contudo, anteriormente já alguns conceitos da Programação Orientada a Objetos (POO) tinham sido utilizados.

A POO foi criada na tentativa de aproximar o mundo real do mundo computacional, organizando a programação como uma coleção de objetos que incorporam estrutura de dados e um conjunto de operações que manipulam estes dados, o que torna o processo de programação, manutenção ou modificação de uma aplicação mais simples.

No passado, os paradigmas da programação derivavam os módulos baseados na funcionalidade de um sistema. Esses módulos correspondiam basicamente a módulos procedimentais, que eram supridos de dados, gerando novos dados. A orientação a objeto mudou essa conceção, projetando objetos como unidades básicas comunicantes entre si por meio de mensagens e encapsulando ao mesmo tempo dados e funções, por meio de um mecanismo conhecido como tipo de dados abstratos.
A programação orientada a objetos é por si só uma técnica. Uma linguagem é dita uma linguagem orientada a objetos, se suporta o estilo de programação orientada a objetos.

Atualmente, existe uma grande diversidade de linguagens orientadas a objeto, abrangendo desde linguagens de âmbito geral, até linguagens para multimédia e programação em lógica.

- **Linguagem C++ [98] –** Linguagem de propósito geral, alia características da linguagem C previamente desenvolvida, a novas capacidades como o suporte para a criação e uso de tipos de dados genéricos, abstração, suporte ao paradigma de programação orientada a objeto, herança, polimorfismo e diversas outras pequenas melhorias nas construções.

- **Linguagem Java [76]** – Linguagem desenvolvida na década de 90, é uma linguagem parcialmente compilada e parcialmente interpretada, que surgiu da intenção de desenvolver uma uma linguagem que pudesse executar o mesmo programa em múltiplas plataformas de hardware e software.

- **Python [66]** – Linguagem desenvolvida por Guido Van Rossum em 1991, é uma linguagem de propósito geral, interpretada, bastante portável e orientada a objetos (incluindo herança múltipla). Apresenta semântica dinâmica, modernos mecanismos de tratamento de erros e exceções, formas eficientes de acesso e reutilização de código com o uso de módulos, recursos avançados de manipulação de textos, listas e outras estruturas de dados e uma sintaxe simples, quase como um pseudo-código, característica que atraí muitos utilizadores pela facilidade que um utilizador não experiente tem em desenvolver alguma aplicação.

- **Ruby [66]** – Linguagem que surgiu em 1995 da intenção do seu percursor, Yukihiro "Matz" Matsumoto, de unir e expandir em uma linguagem "script", as funcionalidades de diferentes linguagens, nomeadamente Perl, Python, Smalltalk, Eiffel, Ada e Lisp. Resultando da fusão das principais capacidades destas linguagens, é uma linguagem de programação interpretada, multiparadigma, de tipagem dinâmica e forte, com gestão de memória automática, suporte a programação funcional, orientada a objetos, imperativa e reflexiva.

2.6.3 Tendências Atuais

A evolução das linguagens de programação continua tanto ao nível da investigação como da indústria. Algumas das tendências mais recentes incluem desenvolvimentos nas seguintes áreas:

- Suporte para execução concorrente e distribuída;

- Mecanismos para aumentar a segurança e fiabilidade das linguagens;

- Mecanismos alternativos de modularidade (mixins, delegates, aspects);

- Desenvolvimento de *software* orientado por componentes

- Metaprogramação (acesso à árvore abstrata da sintaxe);

- Ênfase adicional na distribuição e mobilidade;

- Melhor integração com bases de dados, incluindo XML e bases de dados relacionais;

- Suporte para Unicode no output e no código fonte;

- Open Source como filosofia de desenvolvimento das linguagens, incluindo a coleção de compiladores GNU.

- AOP (Aspect Oriented Programming);

- Delegação e métodos mais complexos de gerir a passagem de listas de dados e funções aos métodos que depois chamam a função apropriada;

2.7 *Bibliotecas de Programação*

O crescente aumento de complexidade das aplicações informáticas, obriga a que as linguagens de programação estejam em constante evolução, para que se possam adaptar às novas exigências que lhes são solicitadas.

Uma parte dessa evolução é conseguida com o desenvolvimento de novas bibliotecas que quando adicionadas à linguagem base, acrescentam novas funcionalidades, que permitem o desenvolvimento de novas aplicações e tornam as aplicações existentes mais simples de implementar.

São desenvolvidas bibliotecas para várias linguagens, de acordo com necessidades específicas das mesmas. Tem-se verificado que os maiores desenvolvimentos têm ocorrido nas linguagens mais em voga, C++, Java e Python.

Referem-se de seguida algumas das bibliotecas mais utilizadas atualmente no âmbito das comunicações e do processamento de imagem, duas áreas fundamentais neste trabalho.

2.7.1 Bibliotecas de Comunicações

- **CSerialPort** - Classe que permite de forma simples, utilizar a linguagem de programação C++ para comunicar com periféricos através das portas série.

- **LibUSB++** [44] - Adaptação da biblioteca LibUSB para a linguagem C++. Esta biblioteca, previamente desenvolvida para a linguagem C, permite uma comunicação bidirecional, parametrizável, com qualquer tipo de dispositivos USB.

- **Javax.comm** [48] - Também designada por Java Communnications API, Atualmente, na versão 3.0, é uma biblioteca já bastante madura, que apresentavárias funcionalidades que permitem de forma muito eficaz e estável, que uma aplicação java comunique através de portas série e paralelo.
- **jUSB** [32] - Desenvolvida por Mojo Jojo and David Brownell em Junho de 2000, com o objetivo de disponibilizar varias ferramentas que permitam aceder a dispositivos USB através de uma aplicação java.

- **PySerial** [35] - Desenvolvida para a linguagem Python, permite desenvolver aplicações que comuniquem bidireccionalmente com dispositivos conectados na porta série

- **PyUSB** [37] - Biblioteca desenvolvida para a linguagem Python que permite, de forma expedita, aceder aos dispositivos USB.

2.7.2 Bibliotecas de Processamento de Imagem

- **CImg [30]** - Também conhecida por C++ Template Image Processing Toolkit, é uma biblioteca que permite a manipulação de imagens ou vídeos numa aplicação desenvolvida na linguagem C++.

- **Java Advanced Imaging** [49] - Biblioteca de processamento de imagem ou vídeo que expande as capacidades de manipulação multimédia da linguagem Java. Também designada por JAI API (Java Advanced Imaging - Application Programming Interface), esta biblioteca, que se encontra atualmente na versão 1.1.3, é bastante abrangente, permitindo por exemplo, a interação com *webcams*, o redimensionamento de imagens, a aplicação de filtros, a análise e identificação de características relevantes da imagem e a utilização de vários operadores que permitem trabalhar ao nível do elemento mais básico da imagem, o *pixel*.

- **OpenCV [63]** - A biblioteca OpenCV, Open Computer Vision, é porventura a mais conhecida e desenvolvida biblioteca de processamento de imagem e vídeo utilizada atualmente no desenvolvimento de aplicações que envolvam visão computacional. Apresentada pela Intel na sua versão inicial em 2000 para as linguagens C/C++, foi idealizada com o objetivo de tornar a visão computacional acessível a utilizadores e programadores em áreas tais como a interação humano-computador em tempo real e a robótica. A Velocidade de implementação e execução, a capacidade de comunicação com *webcams*, Kinect, câmaras FireWire e IP ou telefones móveis e as Funções de processamento de imagem, análises estrutural, análise de movimento, procura de objetos, reconhecimento de padrões, calibração de câmaras e reconstrução 3D foram os principais impulsionadores desta biblioteca.

- **Python Imaging Library** [50] - Esta biblioteca permite munir a linguagem Python de capacidade de processamento de imagem e vídeo. Devido às suas potencialidades e velocidade de execução, esta biblioteca tornou-se numa das mais utilizadas na linguagem Python, servindo até algumas das suas funções como base estrutural para o desenvolvimento de outras bibliotecas, como é o caso da SimpleCV. Podem-se destacar o interface com *webcams*, a conversão de formatos, o redimensionamento, a rotação, a apresentação múltipla de imagens, a manipulação de bits, a filtragem e a conversão de espaços cor como as mais utilizadas capacidades desta biblioteca.

- **SimpleCV** [18] - A biblioteca SimpleCV foi desenvolvida em Python como forma de disponibilizar nesta linguagem, de forma expedita, as potencialidades da biblioteca OpenCV. Encontrando-se atualmente na versão 1.3, esta biblioteca abrange a quase totalidade das ferramentas existentes na OpenCV, tendo por isso capacidades muito semelhantes.

3. Metodologia Proposta

Neste capítulo descrevem-se os métodos e os respetivos conceitos teóricos necessários ao desenvolvimento e implementação de um sistema de condução autónoma num robot móvel diferencial.

Um sistema deste tipo depende fundamentalmente da deteção e tratamento de imagem. Portanto, compreende-se que o seu bom desempenho, sobretudo no que se refere às funcionalidades motoras e de deteção de sinais, obrigue a um eficiente processamento de imagem.

Aliado ao complexo processamento referido anteriormente, um algoritmo de controlo de motores eficaz é também essencial para um bom desempenho global do sistema.

Assim, de forma a descrever detalhadamente o sistema de controlo implementado, que se sustenta essencialmente nos dois pontos anteriores, referem-se de seguida os vários aspetos a ter em conta relativamente à aquisição e processamento de imagem bem como à forma de controlo utilizada.

3.1 *Aquisição e Processamento de Imagem*

A aquisição e o processamento de imagem são porventura as etapas fundamentais deste trabalho, sendo estas que viabilizam a grande maioria das funcionalidades com que se pretende dotar a plataforma móvel robótica.

A necessidade de implementar um sistema de controlo baseado em informações extraídas de imagens obriga a que após a aquisição, seja efetuado um complexo processamento da imagem recolhida, para que os atributos e aspetos de que dependerá o correto funcionamento do sistema sejam salientados, em detrimento de outros que não representam nenhuma mais-valia para o mesmo.

O processamento de imagens, consoante a especificidade da aplicação onde se insere pode apresentar diversas fases, porém, a maioria das aplicações, tal como a que se propõe neste trabalho, consiste em basicamente três, aquisição, pré-processamento (que inclui filtragem e outros tipos de tratamento) e a extração de atributos.

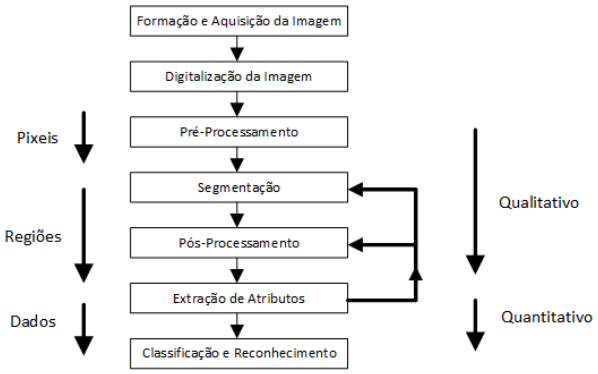


Figura 3.1 - Estrutura exemplificativa de um algoritmo de processamento de imagens

3.1.1 Aquisição e Definição de Zonas de Interesse

A aquisição de imagem é o paço primordial para o desenvolvimento deste sistema, sendo nesta etapa adquirida a maioria da informação do meio envolvente que permitirá ao robot deslocar-se convenientemente e em segurança.

Tal como descrito no capítulo anterior, as imagens são recolhidas graças a sensores que captam a informação do meio envolvente, correspondendo esta, depois de recolhida, a um conjunto de valores na forma de uma matriz de dados, onde cada elemento representa um *pixel* da imagem. Esta matriz numérica pode então ser processada pelo computador de forma a extrair as informações desejadas. No caso de imagens a cores, definidas no espaço de cor RGB que descreveremos mais à frente, correspondem a matrizes em que cada *pixel* é um tripleto que reúne as componentes vermelho, verde e azul. Imagens neste formato podem ser consideradas a sobreposição de três matrizes correspondentes às cores fundamentais.

Na figura seguinte temos uma representação da forma como o computador "vê" a imagem adquirida, um conjunto de números, organizados numa matriz de dados.

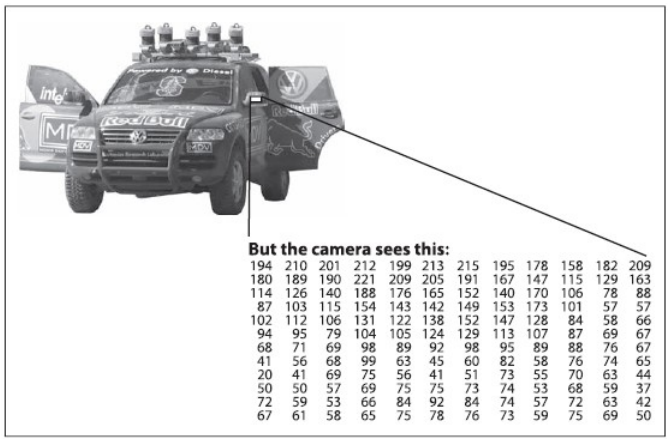


Figura 3.2 - Matriz de dados referentes ao espelho retrovisor esquerdo do automóvel

Como referido, há vários aspetos a considerar para garantir uma aquisição de imagem adequada às tarefas que dela dependem:

- **Características do Dispositivo de Captação de Imagem**

As características dos dispositivos de captação de imagem são cruciais para a forma e qualidade como a informação do exterior é recolhida.
Dependendo da aplicação e do ambiente onde irão operar, as câmaras utilizadas para esta função poderão ter que apresentar características de resolução, cadência de captação, ângulo de abertura e sensibilidade adaptadas à situação.
Neste trabalho, como as condições no local de aquisição são controladas e não é necessária informação muito pormenorizada sobre os elementos detetados, foi possível recorrer a simples *webcams* para realizar a aquisição.
Esta opção permite cumprir um dos pressupostos deste trabalho, o desenvolvimento de um sistema de condução autónoma de baixo custo.

- **Posição e Orientação dos Dispositivos de Captação de Imagem**

A posição e orientação das câmaras são fatores determinantes para o processamento de imagem efetuado, bem como para o funcionamento global do sistema.

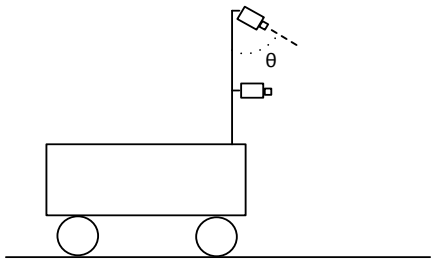


Figura 3.3 - Posicionamento das câmaras na estrutura robótica

As câmaras estão posicionadas de acordo com as funções a que se destinam.

A câmara destinada à deteção das faixas de rodagem está posicionada num ponto elevado da estrutura e orientada horizontalmente segundo o eixo central do robot e verticalmente no sentido da pista. Como se descreve no capitulo seguinte, verticalmente, o ângulo que esta descreve face ao seu suporte, representado na Figura 3.3 por θ, é determinado empiricamente de forma a maximizar a eficiência global do sistema.

A câmara destinada à deteção da sinalização, é posicionada a uma altura e orientação compatível com a sinalização existente. Esta câmara é representada na Figura 3.3 no ponto médio do suporte vertical.

- **Dimensão e Posicionamento das Zonas de Interesse**

Normalmente, quando se recorre à Visão Computacional para determinada tarefa, as imagens adquiridas contêm bastante informação desnecessária ao processo. Uma forma de tornar as tarefas subsequentes de deteção e extração mais céleres é evitar que esta informação sem significado também seja processada.

Para isso, recorre-se frequentemente à sectorização da imagem adquirida, ou seja, restringe-se o processamento a zonas específicas da imagem.

Neste trabalho, era essencial detetar independentemente cada uma das faixas de rodagem. Devido a isso, o processamento de imagem relativo à deteção de faixas é efetuado em duas zonas distintas da imagem. Estas, de dimensão pré-determinada, correspondem às áreas onde em cada momento se espera encontrar informação útil para a aplicação em causa.

De forma a maximizar a eficiência do sistema de controlo do robot, a posição das zonas de interesse consideradas depende da sua velocidade de circulação. Uma velocidade mais elevada implica que a zona onde a deteção é efetuada se localize numa área mais superior da imagem, o que na prática corresponde a uma deteção mais afastada do robot. Isto permite que apesar do acréscimo de velocidade, o robot tenha ainda capacidade de reagir adequadamente às mudanças de direção que lhe vão sendo exigidas.

Para isso, implementa-se neste trabalho, um algoritmo sustentado numa relação linear entre velocidade e posição:

$$Pos = Pos_{min} - 45(vel - 1) \quad (3.1)$$

Onde *Pos* é a posição da zona de interesse a determinar, Pos_{min} é a posição da zona de interesse (360 *pixéis* de altura) para a velocidade mínima (1 cm/s) e *vel* corresponde à velocidade de circulação do robot em dado momento. A constante utilizada na eq. 3.1 (45) foi selecionada para que se verificasse uma proporcionalidade direta entre as várias velocidades de circulação e zonas de interesse possíveis.

Representa-se na imagem seguinte a posição das zonas de interesse para a velocidade mínima de circulação do robot. As setas indicam a direção e sentido do movimento dessas zonas em resposta a um aumento da velocidade de circulação.

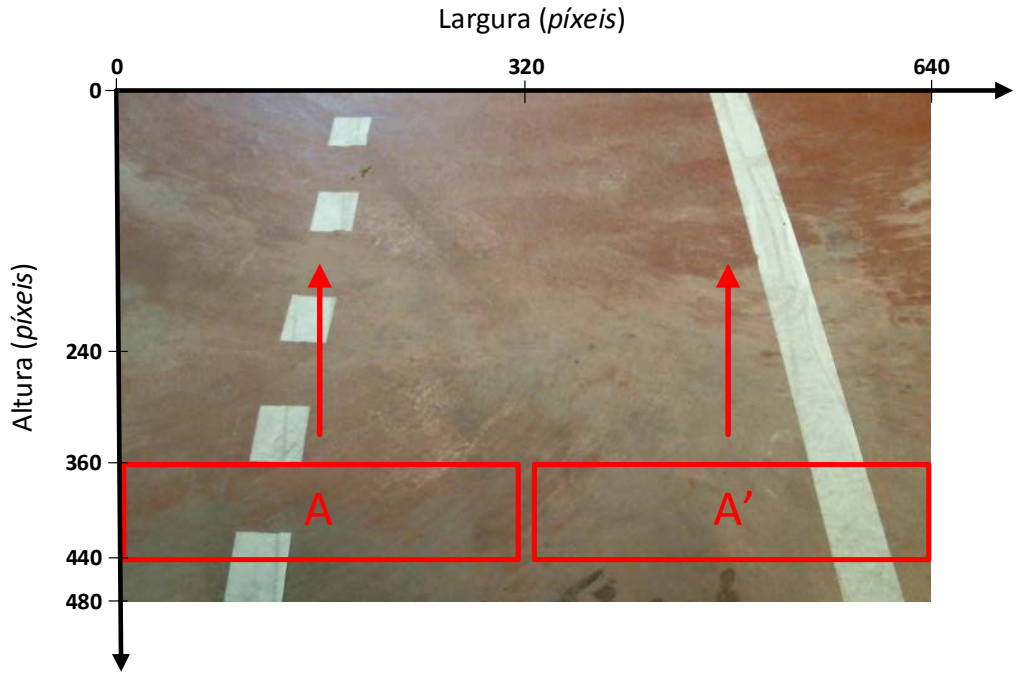


Figura 3.4 - Zonas de interesse para deteção de faixas de rodagem

Como é possível verificar posteriormente nos resultados, este algoritmo foi uma mais-valia para o sistema de controlo.

- **Calibração e Medidas Reais**

Esta é uma etapa importante para aplicações que recorrem a algoritmos de Visão Computacional com o intuito de obter valores reais de distâncias ou deslocamentos com base nas imagens adquiridas.

Contudo, para vários tipos de aplicações, tal como no presente trabalho, essa informação não é relevante. O controlo da movimentação do robot será efetuado apenas com base em diferenças absolutas de *pixéis*, não sendo necessário em nenhum momento ter conhecimento da correspondência entre determinado número de *pixéis* e a sua dimensão real.

3.1.2 Filtragem

Durante o processo de aquisição de imagem torna-se percetível a sua suscetibilidade a influências do exterior, chegando por vezes a informação que se pretende obter a ser degradada de tal forma que se torna impossível realizar qualquer tipo de reconhecimento com base nesta. Para tentar minimizar este tipo de situação, são por norma utilizadas técnicas de filtragem e suavização que tornam a imagem resultante mais adequada a determinada aplicação [21].

Os filtros mais vulgarmente utilizados no processamento de imagem classificam-se segundo as suas características essencialmente em três tipos, passa-baixo, passa-banda e passa-alto. Os filtros passa-baixo atenuam ou eliminam componentes de alta frequência que correspondem a regiões de detalhes finos. A filtragem passa-baixo provoca uma leve suavização da imagem. Os filtros passa-banda salientam aspetos específicos de uma imagem ou eliminam ruídos ou imperfeições presentes a uma frequência conhecida. Os filtros passa-alto, atenuam ou eliminam os componentes de baixa frequência e, em função disto, realçam as bordas e regiões de alto contraste da imagem [21].

O processamento de imagem a efetuar neste trabalho, centrado sobretudo na deteção de contornos, é bastante sensível a ruído, podendo na fase de deteção das faixas parte deste ser interpretado erradamente como informação útil ao sistema de controlo. A aplicação de um filtro passa-baixo, que suaviza a imagem, atenuando o ruído existente, permite minimizar este tipo de situação sem contudo eliminar informação pertinente.
Este tipo de filtragem pode ser aplicado no domínio espacial[8] ou no domínio da frequência[9] porém, a semelhança de resultados e o maior peso computacional dos processos envolvidos na aplicação no domínio da frequência tornam a abordagem espacial mais adequada à tarefa em causa. De entre os filtros mais utilizados podem-se destacar o filtro da média [21] e o filtro Smooth Gaussiano [2]. Neste trabalho foi utilizado o filtro Smooth Gaussiano devido à máscara[10] de filtragem aplicada à imagem apresentar pesos diferentes para cada posição, o que permite suavizar preservando mais eficazmente os contornos das formas. Este aspeto justifica também a sua utilização na fase inicial do método de deteção de contornos de Canny [14].

O filtro Smooth Gaussiano baseia-se na aplicação à imagem de uma função gaussiana de uma ou duas dimensões.

Uma função gaussiana de uma dimensão é representada pela seguinte equação:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \qquad (3.2)$$

Sendo a sua forma percetível na imagem seguinte:

[8] Filtragem efetuada diretamente sobre a matriz de *pixéis* representativa da imagem.
[9] Filtragem baseada na modificação da transformada de Fourier de uma imagem para atingir um objetivo específico. A imagem filtrada é obtida através da transformada inversa de Fourier.
[10] Matriz dos pesos de determinado filtro a aplicar à imagem.

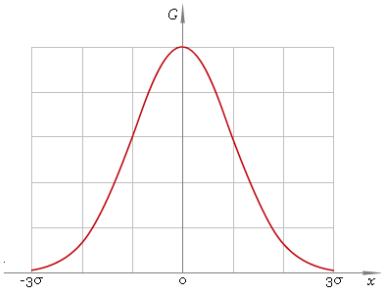


Figura 3.5 - Distribuição Gaussiana de uma dimensão limitada a 3σ

Uma função gaussiana de duas dimensões, que resulta do produto de duas equações de uma dimensão, é representada pela seguinte equação:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.3}$$

Sendo a sua forma percetível na imagem seguinte:

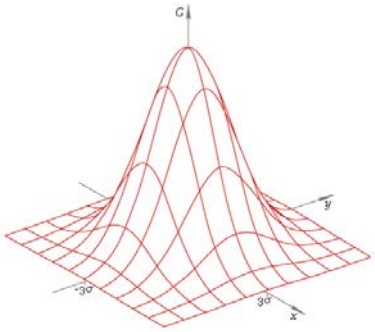


Figura 3.6 - Distribuição Gaussiana de duas dimensões limitada a 3σ

Onde x é a distância a partir da origem no eixo x, y é a distância a partir da origem no eixo y e σ é o desvio padrão da distribuição Gaussiana.

Matematicamente a aplicação deste filtro consiste na convolução da imagem com uma máscara determinada pelas funções de distribuição gaussiana indicadas anteriormente. A filtragem depende do valor de σ (desvio padrão) considerado e da dimensão da máscara (janela de convolução). Esta resulta na suavização da imagem original, sendo tanto mais visível quanto maior for o σ considerado.

Neste trabalho, aplica-se uma Filtragem Gaussiana a duas dimensões, com máscara de dimensão 3x3 e σ = 1. Estas características permitem que a aplicação da máscara culmine numa filtragem fina, preservando eficazmente os contornos das formas principais na grande maioria das condições de captação.

Resume-se na Figura 3.7 o processo de obtenção da máscara de filtragem a aplicar, determinada com base na equação 3.3.

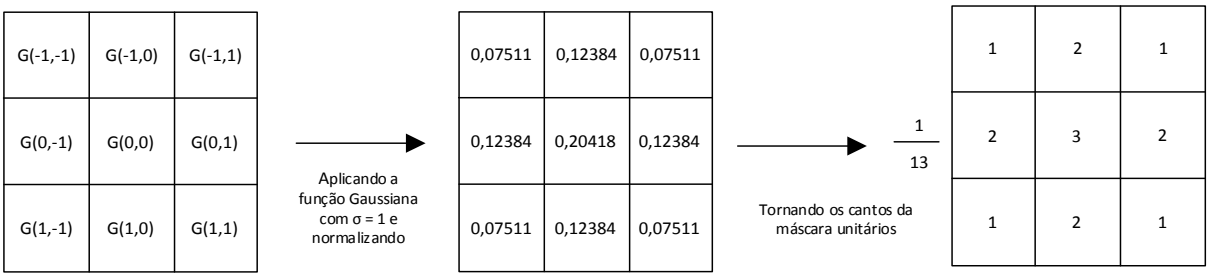


Figura 3.7 - Máscara do filtro gaussiano de dimensão 3x3 e σ=1

A imagem filtrada resulta da convolução[11] da imagem original com a máscara anterior. O resultado desta filtragem é apresentado na figura seguinte:

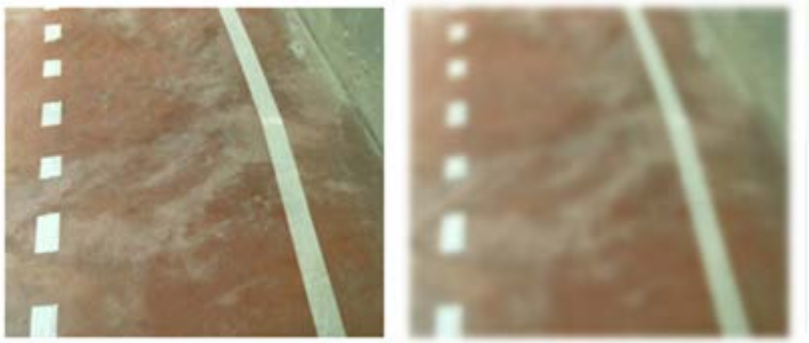


Figura 3.8 - Imagem à esquerda sem tratamento e à direita após Filtragem Gaussiana

3.1.3 Escala de Cinzentos

O reconhecimento de imagem inerente a esta aplicação implica que as imagens utilizadas sejam previamente tratadas para que as características mais significativas das mesmas sejam salientadas em detrimento de outras que sejam inúteis à tarefa em causa.
Assim, neste trabalho, as imagens serão trabalhadas em preto e branco, tendo para isso que previamente serem convertidas para tons de cinzento, ou seja, a cada *pixel* passará a estar associado um valor entre 0 e 255 que denotará o seu nível do cinzento.

Uma imagem pode ser descrita utilizando diversos modelos de cor, sendo o modelo mais utilizado, e adequado para aplicações eletrónicas, o modelo aditivo RGB - Red, Green and Blue.

[11] Processo de aplicação de uma máscara a uma imagem. Consiste na soma das multiplicações dos elementos que possuem a mesma posição nas matrizes de representação da imagem e máscara após sua inversão. A posição do elemento resultante, i.e. somatório, é a mesma do elemento que durante o processamento de toda a imagem esteve ao centro da máscara.

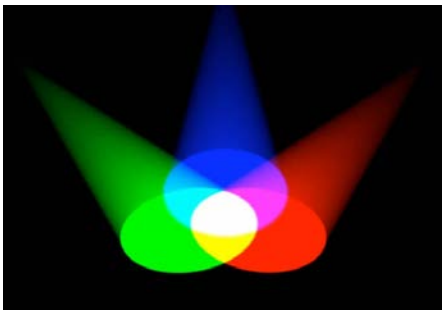


Figura 3.9 - Ilustração da aditividade do modelo RGB

Este modelo representa cada *pixel* de uma imagem através de três valores correspondentes às três cores primárias aditivas: vermelho, verde e azul. A perceção simultânea destes três valores reproduz a cor pretendida. De referir que este modelo não reproduz todas as gamas de cores existentes, limita-se à gama de cores percetíveis ao olho humano.

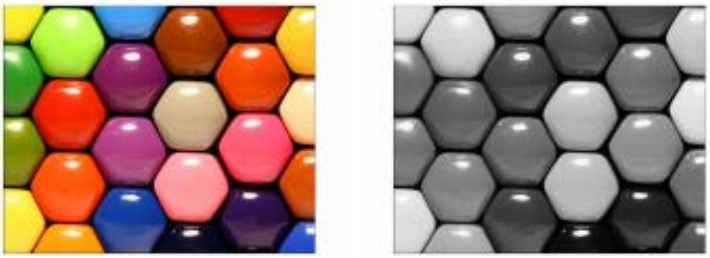


Figura 3.10 – Imagem em modelo RGB e em tons-de-cinzento.

Como indicado, uma imagem digital pode ser encarada como três matrizes, cada uma associada a uma cor fundamental: vermelho, verde e azul. Cada matriz tem a dimensão da imagem em *pixéis* e a cada *pixel* estará associado um valor entre 0 e 255 (ou entre 0 e 1) que denota a amplitude da respetiva cor nesse *pixel* (um valor elevado corresponde a uma cor mais brilhante, um valor muito baixo corresponde a uma cor mais escura).

A transformação entre o modelo RGB e o modelo Grayscale (escala de cinzentos) será feita neste trabalho de acordo com a recomendação BT.709-5 de 2002 da ITU-R (Radiocommunication Sector of International Telecomunication Union) [104], sendo contudo de referir que existem outros métodos como por exemplo a Luminância [53] e a Desnaturação [53] que apresentam resultados semelhantes.

O método utilizado consiste na ponderação da amplitude de cada cor por um fator pré-determinado que valoriza cada componente de cor de forma semelhante à que o olho humano faz [53]. O valor de cada *pixel* em escala de cinzentos é determinado segundo a equação:

$$Grayscale(x,y) = R(x,y) \times 0.299 + G(x,y) \times 0.587 + B(x,y) \times 0.114 \qquad (3.4)$$

Onde, R(x,y), G(x,y) e B(x,y) correspondem respetivamente às amplitudes do *pixel* para as cores vermelho, verde e azul.

3.1.4 Binarização

Após a conversão para escala de cinzentos, a binarização da imagem, ou seja a sua conversão para uma imagem apenas a preto e branco, é essencial para maximizar as capacidades de deteção necessárias neste algoritmo.

A binarização é o método mais simples de segmentação de uma imagem, consistindo essencialmente na separação das suas regiões consoante sejam ou não importantes para o restante processamento a efetuar. Esta separação, efetuada com base na escolha de um ponto de corte (*"threshold"*), permite evidenciar certos aspetos da imagem em detrimento de outros que não sejam relevantes para a aplicação onde a imagem será utilizada.

O método da binarização pode ser representado matematicamente pela seguinte equação:

$$g\,(x,y) = \begin{cases} 1\ se\ f(x,y) > T \\ 0\ se\ f(x,y) \leq T \end{cases} \tag{3.5}$$

Onde $f(x,y)$ é o valor do *pixel* da imagem na escala de cinzentos, $g(x,y)$ é a imagem binarizada e T é o ponto de corte ou *threshold*.

Após a aplicação deste método, todos os *pixéis* com intensidade menor do que o valor de *threshold* selecionado são convertidos para a cor preta e os restantes para a cor branca.

Os métodos mais simples para binarizar imagens utilizam um único *threshold*. Em alguns casos no entanto, devido à forma de captação da imagem, a influências externas ou à especificidade da aplicação, não se consegue apenas com um limiar obter bons resultados na segmentação de toda a imagem. Para esses casos, são utilizadas técnicas variáveis e multinível, aplicadas diferenciadamente nas várias zonas da imagem, sendo o *threshold* determinado com base nas suas características específicas.

3.1.4.1 Limiar de Decisão *(Threshold)*

A seleção de um bom valor para o ponto de corte (*threshold*) de uma binarização é uma tarefa bastante complexa. Normalmente selecionado por análise do histograma da imagem ou por tentativa e erro, a escolha de um valor não adequado às características da imagem pode inviabilizar a utilidade posterior da imagem.

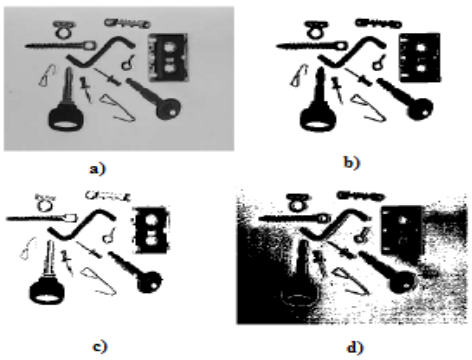


Figura 3.11 – Binarização - a) Original b) Threshold adequado c) Threshold baixo d) Threshold alto

Esta dificuldade, aliada à possibilidade do mesmo valor de *threshold* poder não ser adequado para todas as imagens captadas e para todas as áreas da mesma imagem, conduziu a que diversos métodos automáticos de seleção de *threshold* fossem desenvolvidos e testados, tendo-se revelado, graças aos bons resultados obtidos, grandes mais-valias para o processamento de imagem.

Assim, justifica-se na grande maioria das situações, tal como neste trabalho, a opção por um método automático de determinação do *threshold* a utilizar. Estes métodos, que se baseiam sobretudo no histograma da imagem a binarizar, permitem que a segmentação efetuada à imagem seja a melhor possível em cada momento, minimizando assim os efeitos de alterações (como por exemplo na iluminação) que possam ocorrer entre cada captação.

Vários autores implementaram soluções automáticas de determinação automática de *threshold*, sendo de destacar por exemplo Huang [55], Li [64] e Schanbag [98]. O método aplicado neste trabalho baseia-se na solução proposta por Otsu [80].

3.1.4.2 Histograma

O histograma de uma imagem é a forma mais comum de representar a distribuição estatística dos seus níveis de cinzento. Para uma imagem de 8 bits por exemplo, este corresponde a uma tabela com 256 entradas, indexadas de 0 a 255, onde o nível de cinzento mais escuro é representado pelo 0 e o nível mais claro é representado pelo 255 [96].
A determinação do histograma segundo Souza, Capovilla e Eleotério [96] de determinada imagem pode ser ilustrada através do seguinte exemplo:

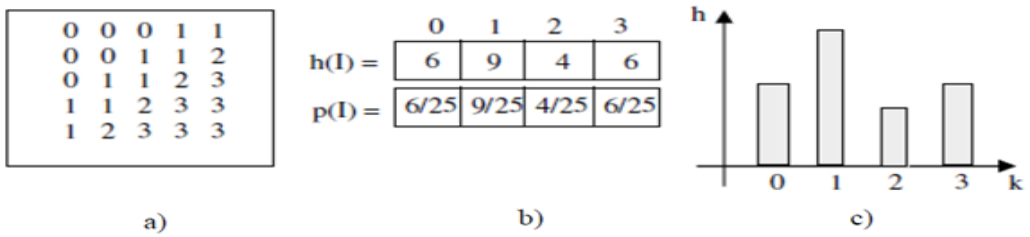


Figura 3.12 - Determinação de um Histograma

- A figura a) mostra uma matriz de dados referente a uma imagem com 25 *pixéis*. Neste caso, uma imagem com tons de cinzento de nível 0 a 3.

- Na figura b) indica-se a contagem e classificação destes *pixéis*. Como é possível verificar, em h(i), temos 6 *pixéi*s de nível de cinzento 0, 9 *pixei*s com nível de cinzento 1, 4 *pixei*s com nível de cinzento 2 e 6 *pixéi*s com nível de cinzento 3.

- Na Figura c), apresenta-se o histograma propriamente dito. Tendo sido gerado com base na contagem e classificação dos *pixéis*, neste apresentam-se quatro colunas, referentes a cada nível de cinzento encontrado na imagem. Verifica-se que o nível de cinzento mais incidente nesta imagem é o nível de cinzento 1, sendo o que aparece com maior comprimento no histograma.

Vários indicadores relativos às características das imagens podem ser retirados do histograma, sendo os mais úteis o nível global de intensidade, a gama dinâmica, o contraste e a informação estatistica (média, desvio padrão, etc.).

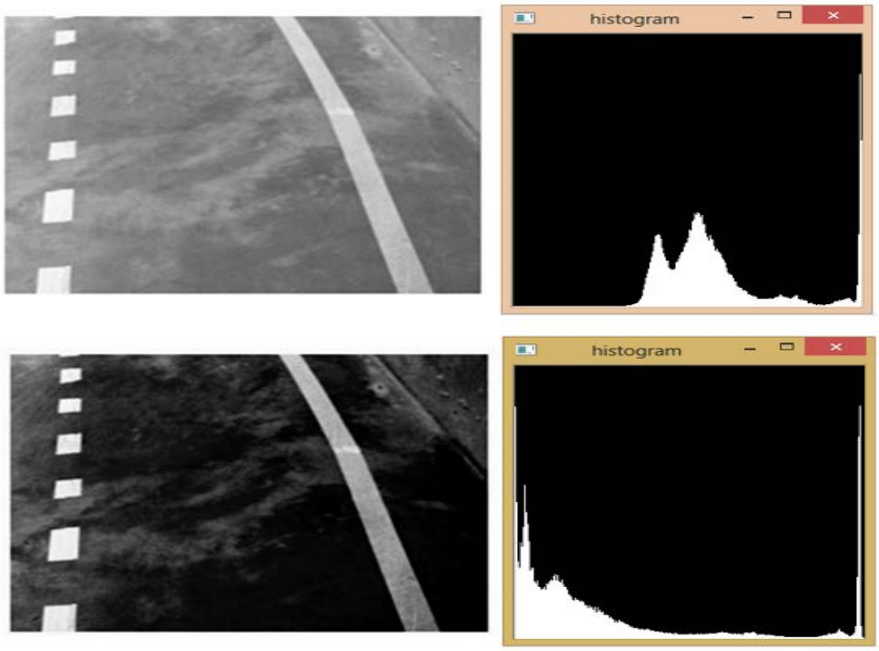


Figura 3.13 - Mais-valia do histograma na avaliação da luminosidade de uma imagem

Os histogramas são também a base de outras técnicas de processamento de imagem. Como exemplo podem-se referir técnicas utilizadas na equalização de imagens [9], na sua segmentação por cor [85] e na sua comparação [59].

3.1.4.3 Determinação Automática de Threshold Otimizado

Como abordado, para a grande maioria das situações, a opção por um método automático de determinação do limiar ótimo para a binarização/segmentação de imagens é a que conduz a melhores resultados.

De entre os vários existentes, características como a aplicabilidade a toda a imagem ou apenas a uma zona específica e, por exemplo, o tempo de execução, tornam o método adequado ou não a determinada aplicação.

Neste trabalho, a semelhança face aos resultados de outros métodos, a facilidade de implementação e a reduzida carga computacional envolvida na sua execução, tornaram uma abordagem semelhante à apresentada por Otsu [80] a opção adequada para o processamento de imagem necessário a esta aplicação.

O método de determinação de *threshold* proposto inicialmente por Nobuyuki Otsu [80] é um método automático de seleção do valor a utilizar para segmentação de determinada imagem. Considera que esta têm duas classes de *pixéis*, o *background* – conjunto de *pixéis* com intensidade de cinzento abaixo do *threshold* – e o *foreground* – *pixéis* com intensidade acima do *threshold* e determina o *threshold* ótimo através de uma busca exaustiva pelo valor que maximiza a variância entre classes.

Este algoritmo itera pelos *thresholds* possíveis e calcula uma medida de variância das intensidades de cinzento dos *pixéis* para cada classe (*background* e *foreground*), sendo o seu objetivo encontrar o *threshold* que torna a variância entre-classes máxima.

O algoritmo implementado neste trabalho, que se baseia no método originalmente desenvolvido por Otsu [80], consiste no seguinte:

Considerando uma imagem digital f, de dimensões $M \times N$ e quantificada em L níveis de cinzento, o primeiro passo para aplicação deste método é calcular o histograma p da imagem dado por:

$$p_{i=} \frac{n_i}{M\ N} \tag{3.6}$$

onde n_i é a quantidade de *pixéis* da imagem f que possuem a intensidade de cinzento i, para $i = 0, \dots, L-1$. Assim, $MN = n_0 + n_1 + \cdots + n_{L-1}$ e

$$\sum_{i=0}^{L-1} p_i = 1, \qquad p_i \geq 0 \tag{3.7}$$

Seja k o nível de cinzento que particiona o histograma da imagem em duas classes C_1 e C_2 (*background* e *foreground*) em que a primeira denota os *pixéis* com níveis de cinzento no intervalo $[0, k]$ e a segunda os *pixéis* cujos níveis de cinzento abrangem $[k+1, L-1]$, podemos definir as probabilidades das classes como:

$$P_1(k) = \sum_{i=0}^{k} p_i \tag{3.8}$$

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1\,(k) \tag{3.9}$$

E os valores médios dessas classes como:

$$\mu_1(k) = \sum_{i=0}^{k} iP(i|C_1) \tag{3.10}$$

$$\mu_2(k) = \sum_{i=k+1}^{L-1} iP(i|C_2) \tag{3.11}$$

Que utilizando o teorema de Bayes conduz a:

$$\mu_1(k) = \sum_{i=0}^{k} i \frac{P(C_1|i)P(i)}{P(C_1)} \tag{3.12}$$

(3.13)

$$\mu_2(k) = \sum_{i=k+1}^{L-1} i \frac{P(C_2|i)P(i)}{P(C_2)}$$

Onde $P(C_1) = P_1(k)$, $P(C_2) = P_2(k)$, $P(i)$ é o próprio p_i e $P(C_1|i)$ e $P(C_2|i)$ são sempre 1, uma vez que i está no intervalo de cinzentos das próprias classes. Simplificando:

$$\mu_1(k) = \frac{1}{P_1(k)} \sum_{i=0}^{k} i p_i \tag{3.14}$$

$$\mu_2(k) = \frac{1}{P_2(k)} \sum_{i=k+1}^{L-1} i p_i \tag{3.15}$$

Por sua vez, a variância de cada classe é dada por:

$$\sigma_1^2(k) = \frac{1}{P_1(k)} \sum_{i=0}^{k} p_i [i - \mu_1(k)]^2 \tag{3.16}$$

$$\sigma_2^2(k) = \frac{1}{P_2(k)} \sum_{i=k+1}^{L-1} p_i [i - \mu_2(k)]^2 \tag{3.17}$$

O que permite determinar a variância entre-classes em relação ao nível de cinzento k, que servirá como medida de separabilidade das duas classes existentes. O *threshold* que maximize esta separabilidade será o mais adequado para efetuar a binarização da imagem em causa. Assim, a variância entre-classes é dada por:

$$\sigma_e^2(k) = P_1(k)[\mu_1(k) - \mu_T(k)]^2 + P_2(k)[\mu_2(k) - \mu_T(k)]^2 \tag{3.18}$$

Que sendo:

$$\mu_T(k) = \mu_1(k)P_1(k) + \mu_2(k)P_2(k) \tag{3.19}$$

Conduz a:

$$\sigma_e^2(k) = P_1(k)P_2(k)[\mu_1(k) - \mu_2(k)]^2 \tag{3.20}$$

Com base nesta, o *threshold* ótimo para a binarização *k** resulta de:

$$k^* = \max_{0 \le k \le L-1} \sigma_e^2(k) \tag{3.21}$$

No seu trabalho [80], Otsu demonstrou que a minimização da variância intra-classes também conduziria ao *threshold* ótimo.

Sendo a variância intra-classes dada por:

$$\sigma_i^2(k) = \sigma_1^2(k)P_1(k) + \sigma_2^2(k)P_2(k) \tag{3.22}$$

O *threshold* ótimo para a binarização *k** pode também ser determinado por:

$$k^* = \min_{0 \le k \le L-1} \sigma_i^2(k) \tag{3.23}$$

Apresentam-se de seguida o resultado de dois testes efectuados com recurso a este método. A segunda imagem foi propositadamente escurecida de forma a verificar o funcionamento do algoritmo:

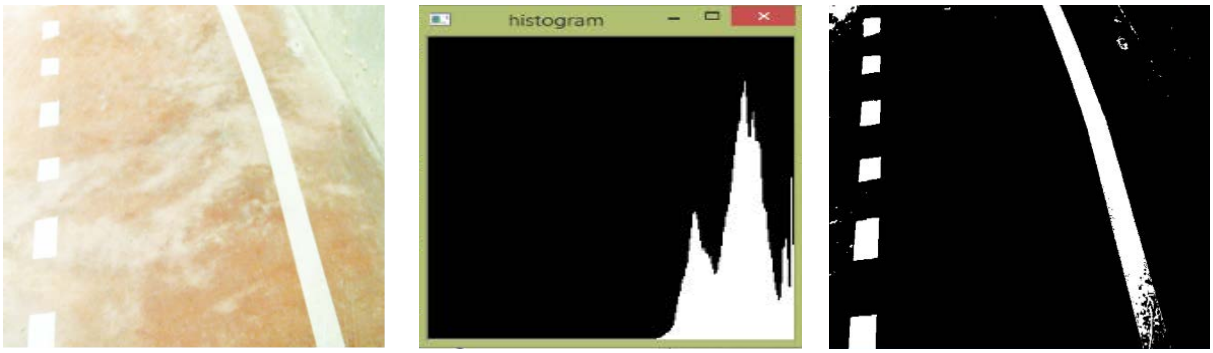


Figura 3.14 - Original à esquerda, histograma ao centro e binarizada à direita. Threshold = 210

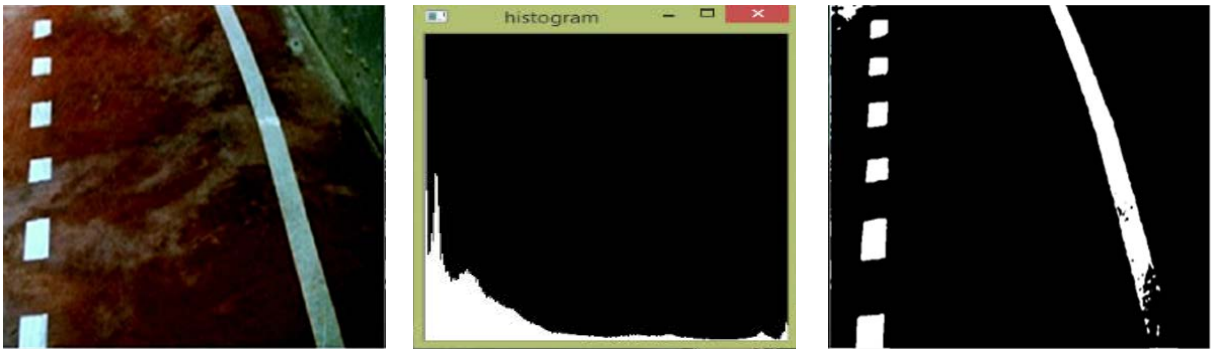


Figura 3.15 - Original à esquerda, histograma ao centro e binarizada à direita. Threshold = 111

Como se verifica nos exemplos anteriores, o benefício deste método é claro. Apesar da segunda imagem ser bastante mais escura, o método determinou *thresholds* que conduziram a resultados finais semelhantes.

O método implementado representa uma evolução da solução inicialmente proposta por Otsu [80]. Ao contrário do que era proposto por este, um *threshold* otimizado para toda a imagem, neste trabalho, a determinação automática do *threshold* é efetuada diferenciadamente para cada zona de interesse, o que beneficia a eficácia do processo de binarização de cada uma das áreas em análise.

3.2 ***Deteção das faixas de rodagem – Reconhecimento de Formas***

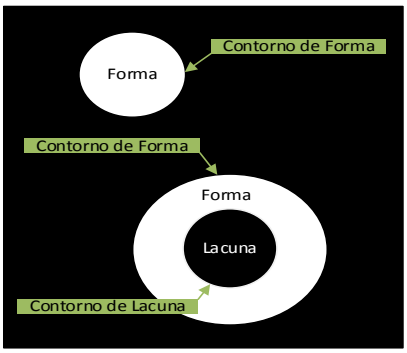
Após a imagem captada ter sido dividida em regiões de interesse e estas terem sido alvo do processamento referido nos últimos pontos, é chegado o momento de iniciar a fase fundamental da extração de características da imagem, o reconhecimento de formas.

É esta etapa que fornecerá ao robot as informações necessárias à orientação e posicionamento durante a sua movimentação. Esta etapa permite localizar formas presentes na imagem, que correspondem neste caso, devido à orientação da câmara e às fases de processamento anteriores, às faixas de rodagem. Após a sua deteção, torna-se possível determinar características como área, perímetro e centro, que as permitem identificar inequivocamente. Recorrendo a estas características, a determinação da posição relativa do eixo central do robot face ao seu objetivo (eixo central da faixa de rodagem) revela-se uma tarefa bastante simples.
O reconhecimento de formas é essencial para a maioria das aplicações que envolvem visão computacional, sendo uma área em constante desenvolvimento, em relação à qual é frequente verificar o surgimento de novas abordagens para realizar de modo mais eficiente este tipo de tarefas. Essencialmente, as metodologias apresentadas baseiam-se na deteção de contornos (linhas fechadas que delimitam as formas existentes na imagem) que, como se verifica em trabalhos iniciais nesta área [70, 88] bem como em desenvolvimentos mais recentes [15, 26], se sustentam na classificação de contornos.

Neste trabalho é implementado um algoritmo de deteção de formas baseado na solução proposta por Satoshi Suzuki e Keiichi Abe [99]. Alicerçada também no seguimento e classificação de contornos, manteve-se possivelmente como o melhor compromisso entre eficácia e rapidez apesar das várias abordagens que foram entretanto surgindo.

De referir que as funções de deteção de formas implementadas na biblioteca de imagem utilizada, a *SimpleCV*, seguem também a abordagem de Satoshi Suzuki e Keiichi Abe [99]. As capacidades do método e a possibilidade de tirar partido da sua presença na biblioteca tornam a opção por este a mais adequada.

O método proposto por Satoshi Suzuki e Keiichi Abe [99] consiste em dois algoritmos distintos. Centrando-se ambos no seguimento de contornos e na sua classificação, diferenciam-se um do outro pelo nível de informação que permitem obter em relação às formas detetadas. O primeiro algoritmo distingue contornos de formas[12] e lacunas[13], enquanto o segundo, se limita apenas à deteção das formas mais exteriores, ou seja, apenas identifica os contornos das formas.



[12] Conjunto de *pixéis* de intensidade 1.
[13] Conjuntos de *pixéis* de intensidade 0 localizados no interior de formas.

1º Algoritmo – Deteção de contornos de formas e lacunas

O seguimento de contornos que permite a deteção de formas consiste essencialmente na classificação sucessiva e hierárquica de *pixéis* numa imagem binária.
A execução deste algoritmo inicia uma análise à imagem que segue até ao *pixel* onde se verificam condições indicativas da presença de um contorno (e consequentemente uma forma). Nesse momento, esse *pixel* é considerado o ponto inicial de um contorno, e é aí iniciado o processo do seu seguimento e classificação. Após a conclusão deste processo, a análise global à imagem é retomada, repetindo-se este processo sempre que as condições indicativas da presença de contorno se verifiquem.

Descrição do Algoritmo

Considerando que o limite exterior de uma imagem definida por $F = \{f_{ij}\}$, onde f_{ij} corresponde à intensidade de um *pixel* (i, j) (que para uma imagem binária apenas assume os valores 0 ou 1), forma um contorno ao qual se atribui a classificação 1, correspondendo ao NCA (**N**úmero sequencial do **C**ontorno **A**tual) e simultaneamente ao NUCD (**N**úmero sequencial do **U**ltimo **C**ontorno **D**etetado), o processo de deteção de contornos consiste em:

Iniciar uma análise *pixel*-a-*pixel* da imagem e para cada um que verifique a condição $f_{ij} \neq 0$ seguir o seguinte procedimento:

1) Seleciona-se a opção adequada:

   a) Se $f_{ij} = 1$ e $f_{i,j-1} = 0$, Assumir que o *pixel* (i, j) é o ponto inicial para o seguimento de um contorno exterior, incrementar *NCA* e $(i_2 j_2) \leftarrow (i, j-1)$.

   b) Se $f_{ij} \geq 1$ e $f_{i,j+1} = 0$, Assumir que o *pixel* (i, j) é o ponto inicial para o seguimento do contorno de uma lacuna, incrementar NCA, $(i_2 j_2) \leftarrow (i, j+1)$ e NUCD $\leftarrow f_{ij}$ no caso de $f_{ij} > 1$.

   c) Não se verificando nenhuma das condições anteriores, seguir para o ponto 4).

2) Dependendo do tipo de contorno detetado e do contorno anteriormente identificado na presente linha (identificado pelo NUCD), classificar hierarquicamente o contorno atual através da tabela seguinte:

Tipo de Contorno Detetado (C)	Tipo de Contorno anteriormente detetado (identificado no NUCD) (C')	
	Contorno Exterior	Contorno Lacuna
Contorno Exterior	Contorno Exterior a C'	O contorno Anterior C'
Contorno Lacuna	O contorno Anterior C'	Contorno Exterior a C'

Tabela 3.1 - Tabela de decisão para classificação do contorno detetado

3) A partir do ponto inicial (i, j), seguir o contorno detetado como indicado nos subpontos seguintes (a) a d)):

a) Começando em (i_2, j_2), Analisar no sentido dos ponteiros do relógio os *pixéis* na vizinhança de (i, j) em busca de um *pixel* diferente de zero, passando este a ser (i_1, j_1). Se não for encontrado nenhum, atribuir o valor –NCA a f_{ij} e seguir para o ponto 4).

b) $(i_2, j_2) \leftarrow (i_1, j_1)$ e $(i_3, j_3) \leftarrow (i, j)$

c) Começando no elemento seguinte do *pixel* (i_2, j_2), no sentido contrário ao dos ponteiros do relógio, analisar a vizinhança do *pixel* atual (i_3, j_3) em busca de um *pixel* diferente de zero, passando este a ser (i_4, j_4).

d) Alterar o valor f_{i_3, j_3} do *pixel* (i_3, j_3) da seguinte forma:

   1) Se o *pixel* $(i_3, j_3 + 1)$ é um dos *pixéis* iguais a zero analisados na alínea c) então, $f_{i_3, j_3} \leftarrow$ –NCA.

   2) Se o *pixel* $(i_3, j_3 + 1)$ não é um dos *pixéis* iguais a zero analisados na alínea c) e $f_{i_3, j_3} = 1$, então $f_{i_3, j_3} \leftarrow$ NCA.

   3) Se não se verificar nenhuma das alíneas anteriores, f_{i_3, j_3} não deve sofrer alteração.

e) Se $(i_4, j_4) = (i, j)$ e $(i_3, j_3) = (i_1, j_1)$, regressar ao ponto inicial e depois seguir para o ponto 4). Caso contrário $(i_2, j_2) \leftarrow (i_3, j_3), (i_3, j_3) \leftarrow (i_4, j_4)$ e regressar à alínea c) deste ponto.

4) Se $f_{ij} \neq 1$, então NUCD $\leftarrow |f_{ij}|$ e a análise *pixel-a-pixel* da imagem deve ser retomada no ponto $(i, j + 1)$.

Na Figura 3.16 é exemplificada a classificação de contornos efetuada por este algoritmo numa imagem. Cada conjunto de caracteres semelhantes representa um contorno de uma determinada forma ou lacuna.

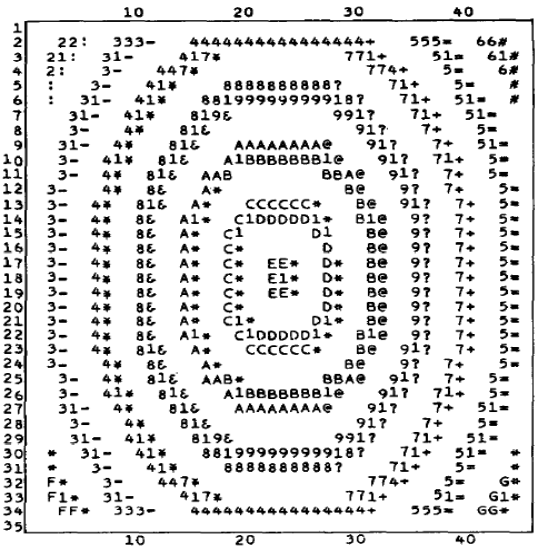


Figura 3.16 - Aplicação do algoritmo a imagem (extraído de Satoshi Suzuki e Keiichi Abe [109])

2º Algoritmo – Deteção de contornos de formas exteriores

Este algoritmo surge como uma modificação do primeiro. Restringe a deteção efetuada aos contornos exteriores (ou seja, apenas das formas, ignorando lacunas) o que o torna mais rápido, embora menos completo.

As diferenças face ao anterior são reduzidas, centrando-se basicamente em três pontos relacionados com as condições de início de seguimento de contornos e na forma de classificação dos mesmos. Indica-se de forma sucinta as mesmas:

- O seguimento dos contornos apenas é iniciado em pontos onde se verificam as condições $f_{ij} = 1$ e $f_{i,j-1} = 0$ e NUCD ≤ 0.
- A identificação do primeiro contorno passa a ser efetuada pelos valores NCA e NUCD em substituição de "2" e "-2" respetivamente.
- O valor do NUCD passa a ser reiniciado a 0 em cada mudança de linha durante a análise global da imagem.

Neste trabalho o algoritmo seguido é o primeiro, contudo, a não existência de lacunas nas formas (faixas de rodagem) conduz a que seja uma aproximação ao segundo, o que o torna consequentemente bastante mais rápido.

De referir que a deteção de formas pode ser mais específica, aplicando-se para isso um filtro que restringe a deteção a estruturas com determinadas características. Neste trabalho, poder-se-ia ter utilizado um filtro de forma a limitar a deteção a retângulos (correspondentes à faixa de rodagem), porém, toda a preparação prévia conduz a que a informação presente no momento da deteção corresponde apenas às faixas de rodagem, dispensando assim esta filtragem adicional.

3.3 ***Posicionamento relativo das faixas de rodagem – Extração de Atributos***

Após a deteção e identificação de formas realizada através do método descrito no ponto anterior, torna-se possível para cada uma extrair várias informações uteis.
Correspondendo estas a atributos que permitem posteriormente conhecer características como a sua dimensão ou posição na imagem, esta etapa é fundamental para a aquisição de informação para o sistema de controlo a implementar no robot.

A determinação de características de formas presentes em imagens digitais baseia-se na teoria dos momentos, propriedades estatísticas que as descrevem. O primeiro trabalho significativo nesta área foi apresentado por Hu [71]. Baseado nos resultados da teoria dos invariantes algébricos, derivou sete momentos invariantes à rotação de objetos 2-D que servem de base a vários trabalhos na área da visão computacional, sendo disso exemplo [22, 56].

A obtenção dos momentos de ordem$(p + q)$ de uma imagem de tamanho (m, n) é conseguida através da seguinte equação:

$$M_{pq} = \sum_{x=1}^{m}\sum_{y=1}^{n} x^p x^q f(x,y), \qquad p,q = 0,1,2,3,\ldots \tag{3.24}$$

Onde $f(x,y)$ representa a intensidade no ponto (x,y).

Sendo uma imagem binária representada por $f(x,y)$, se $f(x,y) = 1$, (x,y) define a posição de um ponto pertencente ao objeto, caso contrário o ponto considerado não pertence ao mesmo. Os momentos permitem definir algumas propriedades de elementos contidos em imagens como por exemplo, área, centróide, momentos de inércia e direção dos eixos principais de inércia. Aborda-se de seguida a forma de cálculo das duas propriedades utilizadas neste trabalho:

- **Área** - Propriedade frequentemente utilizada, corresponde ao momento m_{00}. Este momento representa o somatório do número de *pixéis* que constitui a região e é obtido pela seguinte equação:

$$Área = m_{00} = \sum_{x=1}^{m}\sum_{y=1}^{n} f(x,y) \tag{3.25}$$

- **Centróide** - Característica essencial para este trabalho, pode ser definido pela relação entre os momentos de ordem 0 e 1 como se descreve nas duas equações seguintes:

$$x_c = \frac{\sum_{x=1}^{m}\sum_{y=1}^{n} x f(x,y)}{\sum_{x=1}^{m}\sum_{y=1}^{n} f(x,y)} = \frac{m_{10}}{m_{00}} \tag{3.26}$$

$$y_c = \frac{\sum_{x=1}^{m}\sum_{y=1}^{n} y f(x,y)}{\sum_{x=1}^{m}\sum_{y=1}^{n} f(x,y)} = \frac{m_{01}}{m_{00}} \tag{3.27}$$

Com o conhecimento destas propriedades torna-se possível a localização da faixa de rodagem na imagem e a avaliação da sua posição relativamente ao eixo central do robot.

3.4 ***Deteção e Reconhecimento de Sinalização***

A capacidade de detetar, reconhecer e respeitar a sinalização existente é uma das mais-valias com que se pretende dotar o sistema de controlo a implementar no robot. Resultando de uma etapa especializada de processamento de imagem, consiste essencialmente na implementação de um método de extração e caracterização de aspetos estruturais particulares de cada sinal e na sua procura em imagens do ambiente envolvente.

Neste trabalho foi implementado um algoritmo de deteção de sinalização baseado no método SIFT – Scale Invariant Feature Transform [65]. Esta opção deveu-se à capacidade de deteção

independente da escala e rotação da imagem e aos resultados obtidos em [69], validados pelos testes previamente efetuados neste trabalho aos métodos mais vulgarmente utilizados, SIFT[65], SURF [4], FAST [89] e BRIEF [13]. Na tabela seguinte sintetizam-se os resultados dos testes realizados.

Comparação de Métodos de Extração de Características								
Sinais	***SIFT***		***SURF***		***FAST***		***BRIEF***	
	Pontos Detetados	*Falsas Deteções*	*Pontos Detetados*	*Falsas Deteções*	*Pontos Detetados*	*Falsas Deteções*	*Pontos Detetados*	*Falsas Deteções*
Sinal 50	***6***	*1*	*3*	*0*	*2*	*1*	*2*	*1*
Sinal 90	***3***	*1*	*1*	*1*	*2*	*0*	*1*	*1*
Sinal Avanço	***6***	*1*	*3*	*0*	*1*	*0*	*2*	*0*
Sinal Stop	***5***	*1*	*2*	*1*	*2*	*1*	*2*	*2*
Sinal Passadeira	***6***	*2*	*4*	*1*	*3*	*0*	*3*	*1*

Tabela 3.2 - Comparação de métodos de extração de características

Como se verifica, o método de extração de características que permitiu identificar maior quantidade de pontos correspondentes entre as duas imagens (com reduzido número de falsas deteções) foi o SIFT, o que motivou a sua escolha.

A Figura 3.17 ilustra os resultados obtidos com o método SIFT.

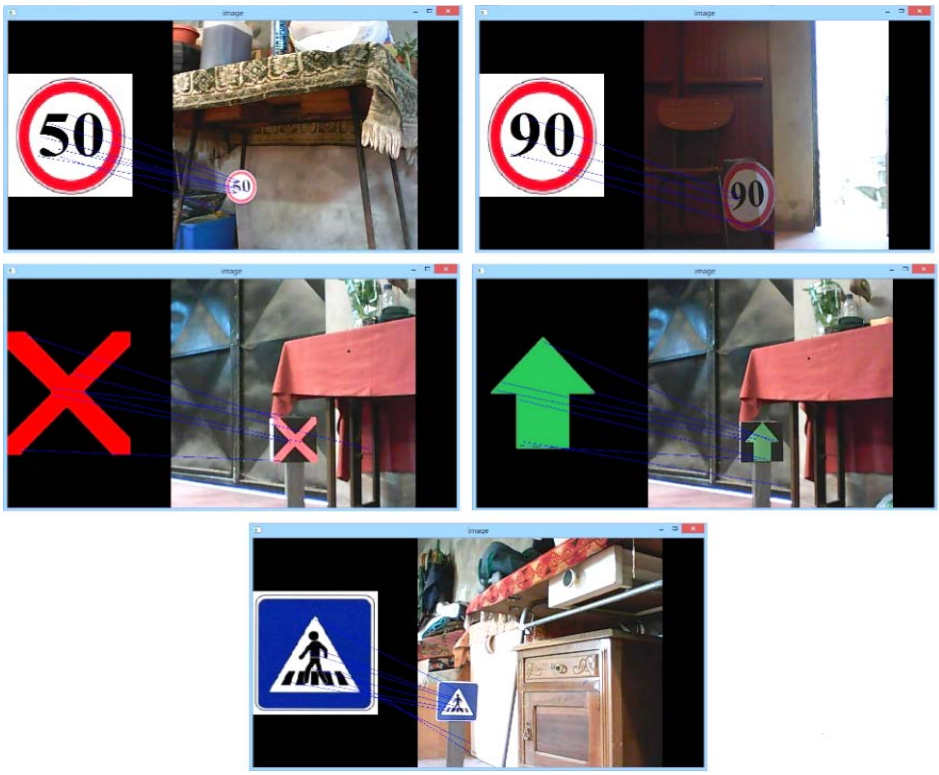


Figura 3.17 - Teste de deteção - Método SIFT (Scale Invariante Feature Transform)

O reconhecimento da sinalização é efetuado com base nas imagens obtidas por uma *webcam* dedicada à tarefa, posicionada de forma a minimizar interferências de outros objetos na deteção.

O processo de reconhecimento implementado, baseado no método SIFT [65], consiste essencialmente em duas etapas. Na primeira identificam-se pontos-chave da imagem e

estabelecem-se os seus descritores. Na segunda, estes descritores são comparados para determinar se existem semelhanças entre imagens.

1ª Fase – Identificação de Pontos-Chave e Obtenção de Descritores

Como referido anteriormente, a extração de descritores neste trabalho será efetuada com base no algoritmo SIFT [65]. Este é composto por dois elementos, o detetor e o descritor, sendo que a implementação de cada um destes envolve simplificadamente 2 etapas, que se descrevem de seguida.

Detetor

Nesta fase são detetados os pontos-chave que representam locais relevantes da imagem que se mantêm estáveis em relação a mudanças de escala e rotação.

- **Deteção de Extremos**

Esta etapa consiste em procurar pontos que sejam invariantes a mudanças de escala da imagem. Recorrendo a uma função *espaço-escala*[14], que neste caso é a função Gaussiana, são identificados os pontos que mantêm características estáveis em diferentes escalas. Estes pontos permitirão comparar imagens independentemente da distância a que se encontra o dispositivo de captação face ao objeto a detetar.

Sendo uma imagem $I\ (x, y)$ que passa a ser definida por $L\ (x, y, \sigma)$ no espaço-escala, como resultado de uma convolução de uma função gaussiana $G\ (x, y, \sigma)$ com a imagem $I\ (x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{3.28}$$

onde:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \tag{3.29}$$

A procura por pontos-chave pode ser eficientemente efetuada pela identificação de extremos numa função DoG (*Difference of Gaussian*) [11], formada pela diferença de imagens filtradas em escalas próximas, separadas por uma constante de escala *k:*

A função DoG é definida por:

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma) \tag{3.30}$$

[14] Teoria formal para lidar com estruturas de uma imagem em diferentes escalas. Estabelece as condições necessárias para a definição de transformações que possibilitem a manipulação de características presentes em diferentes níveis de maneira consistente.

Sendo o resultado da convolução da imagem com o filtro DoG definido pela função anterior:

$$D(x, y, \sigma) = \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y) \quad (3.31)$$

Que é equivalente a:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (3.32)$$

Esta função suaviza as imagens e pode ser calculada de forma simples através da subtração de imagens filtradas por um filtro Gaussiano em escalas σ e $k\sigma$. A utilização da função gaussiana visa obter representações da imagem onde detalhes indesejados e ruídos são eliminados e características fortes realçadas. Variando σ torna-se possível identificar tais características em diferentes escalas.

Ilustra-se na seguinte imagem e descreve-se de seguida, um processo expedito para a determinação da Diferença de Gaussianas (DoG) [11]:

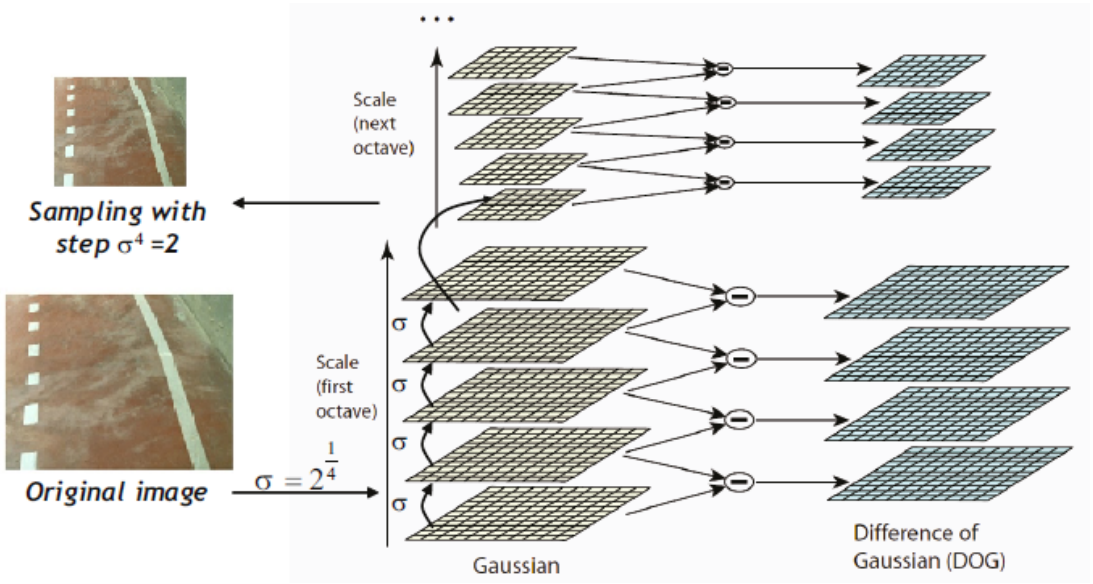


Figura 3.18 - Obtenção das Diferenças de Gaussianas DOG para diversas oitavas de uma imagem

Descrição:

- A imagem inicial sofre convoluções incrementais com filtros Gaussianos para produzir imagens separadas por um fator de escala *k* no espaço de escala.
- Considera-se que é necessário fazer a convolução da imagem até 2σ para ser possível a construção de descritores invariantes quanto à escala. Para isso, para gerar em s intervalos, o fator de escala k é definido por $k = 2^{\frac{1}{s}}$, produzindo assim s+3 imagens na oitava de forma que a deteção de extremos cubra toda a oitava.
- Imagens em escalas adjacentes são subtraídas para produzir as imagens da DoG (Diferença de Gaussianas) representadas à direita.

- Uma vez processada a oitava, é reduzida a resolução da imagem tomando-se cada segundo *pixel* da imagem no centro da oitava, gerando-se uma nova oitava (alteração da frequência de mostra por um fator de dois), e volta-se ao passo numero 1.

- Após a determinação da DoG (Diferença de Gaussianas), é efetuada uma deteção de extremos nos intervalos de cada oitava. Um extremo é definido como qualquer valor na DoG maior do que todos os seus vizinhos no espaço-escala.

Os extremos correspondem a valores de máximo ou mínimo local para cada $D(x, y, \sigma)$, que podem ser obtidos através da comparação de intensidades de determinado ponto com a de oito vizinhos na sua escala, com nove pontos vizinhos na escala superior e novo vizinhos na escala inferior.

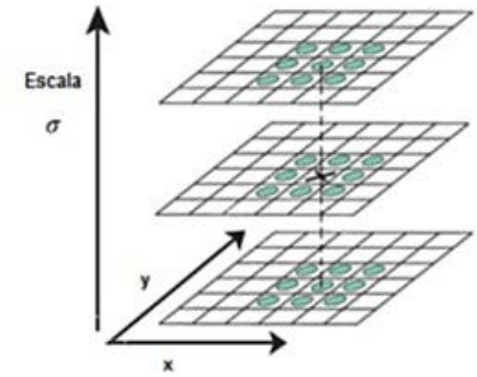


Figura 3.19 - Deteção de extremos no espaço-escala

Na imagem anterior esquematiza-se a deteção de extremos. O ponto identificado com "X" é comparado com os seus vizinhos identificados com "O". As 3 imagens representadas correspondem à diferença entre imagens adjacentes da pirâmide gaussiana.

- **<u>Localização Especifica de Pontos-chave</u>**

Nesta fase pretende-se calcular a posição exata dos candidatos a pontos-chave identificados como extremos no ponto anterior.
A localização dos mesmos é obtida através do ajuste de uma função quadrática 3D do ponto de amostragem local de modo a determinar uma localização interpolada do máximo.

Para isto, recorre-se a uma expansão de Taylor [15] da função Diferença de Gaussiano (DoG) aplicada à imagem, $D(x, y, \sigma)$, centrada no ponto de amostragem:

$$D(\bar{x}) = D + \frac{\partial D^T}{\partial \bar{x}}\bar{x} + \frac{1}{2}\bar{x}^T\frac{\partial^2 D}{\partial x^2}\bar{x} \tag{3.33}$$

$$\bar{x} = (x, y, \sigma)^T \tag{3.34}$$

[15] Método matemático que permite aproximar funções deriváveis através de funções polinomiais, cujos coeficientes dependem das derivadas da função.

Onde D, e as suas derivadas são calculadas no ponto de amostragem e $\bar{x}$ representa o deslocamento deste ponto.

A localização do ponto de interesse é dada pelo extremo da função apresentada em (3.33). Esta localização, $\hat{x}$, é determinada igualando a zero a derivada da função $D(\bar{x})$ em relação a $\bar{x}$.

$$\frac{\partial D}{\partial \bar{x}} + \frac{\partial^2 D}{\partial \bar{x}^2}\hat{x} = 0 \tag{3.35}$$

Tem-se então a posição do extremo, dada por:

$$\hat{x} = -\frac{\partial^2 D^{T-1}}{\partial \bar{x}^2}\frac{\partial^2 D}{\partial \bar{x}} \tag{3.36}$$

O valor da função no extremo, $D(\bar{x})$, é útil para a filtragem de extremos instáveis com baixo contraste, que seriam sensíveis a ruido. Isto pode ser obtido substituindo-se a equação (3.36) na equação (3.33):

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \bar{x}}\hat{x} \tag{3.37}$$

Segundo os testes efetuados por Lowe [65], devem ser rejeitados valores de $|D(\hat{x})|$ inferiores a um determinado limiar. No seu trabalho, propõe o valor 0,03 [65].

De referir que este método também apresenta "forte" resposta a pontos-chave situados ao longo de arestas, mesmo que mal determinadas, o que não é desejável. Tendo os pontos nesse tipo de localização características particulares como valores de curvatura principal elevados ao longo da aresta e reduzidos perpendicularmente a esta, torna-se possível a sua eliminação com recurso a uma matriz Hessiana 2x2, H, computada na localização e escala dos pontos-chave na função D. Esta matriz permite mensurar as magnitudes das curvaturas locais de D a partir dos seus valores próprios e é definida por:

$$H(x, y) = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{3.38}$$

Onde D_{xy} é a derivada de $D(x, y, \sigma)$ na localização e escada em relação a x e y, D_{xx} é a derivada de segunda ordem em relação a x e D_{yy} é a derivada de segunda ordem em relação a y.

Sendo os valores próprios de H proporcionais às curvaturas locais de D, e pretendendo-se apenas conhecer a sua relação, torna-se desnecessário determinar analiticamente os seus valores.

Sendo α, o valor próprio com maior magnitude, e β, o de menor, pode-se calcular a soma dos valores próprios pelo traço de H e o seu produto pelo seu determinante:

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \tag{3.39}$$

$$Tr(H) = D_{xx}D_{yy} - \left(D_{xy}\right)^2 = \alpha\beta \tag{3.40}$$

Para o caso em que o determinante seja negativo, as curvaturas possuem sinais diferentes, e o ponto é descartado, não sendo considerado um extremo.
Sendo r a razão entre o valor próprio de maior magnitude e o de menor, de modo que $\alpha = r\beta$, então:

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{3.41}$$

A equação anterior depende apenas da razão entre os valores próprios, permitindo avaliar a sua semelhança. Este valor é mínimo quando são idênticos e cresce com respeito ao valor de r. Assim, é possível eliminar pontos indesejáveis próximos a extremidades que estejam abaixo de determinado limiar (r):

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \tag{3.42}$$

Lowe propõe no seu trabalho [65] o uso de r=10 como o mais adequado para a maioria das situações.

Descritor

Nesta fase é atribuída a orientação a cada ponto-chave detetado e é estabelecido o seu descritor.

- **Atribuição da Orientação dos Descritores**

Este é a primeira etapa para o estabelecimento dos descritores dos pontos-chave detetados. Para cada um, é atribuída uma orientação que será utilizada posteriormente para a construção de descritores invariantes quanto à rotação.

Com base na escala a que o ponto-chave foi identificado, seleciona-se uma imagem suavizada $L(x,y)$, na escala mais próxima, que permite que a determinação efetuada seja invariante à escala.

Para cada amostra da imagem $L(x,y)$, na escala identificada, a magnitude do gradiente $m(x,y)$ e respetiva orientação $\theta(x,y)$ pode ser determinada usando diferenças de *pixéis* segundo as seguintes funções:

$$m(x,y) = \sqrt{\begin{pmatrix} \big(L(x+1,y) - L(x-1,y)\big)^2 \\ +\big(L(x,y+1) - L(x,y-1)\big)^2 \end{pmatrix}} \tag{3.43}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{(L(x,y+1) - L(x,y-1))}{(L(x+1,y) - L(x-1,y))}\right) \tag{3.44}$$

Com base nestas, é formado um histograma das orientações dos gradientes de *pixéis* numa região em torno do ponto-chave. O histograma possui 36 regiões, cobrindo todas as orientações possíveis $(0\ a\ 2\pi)$.

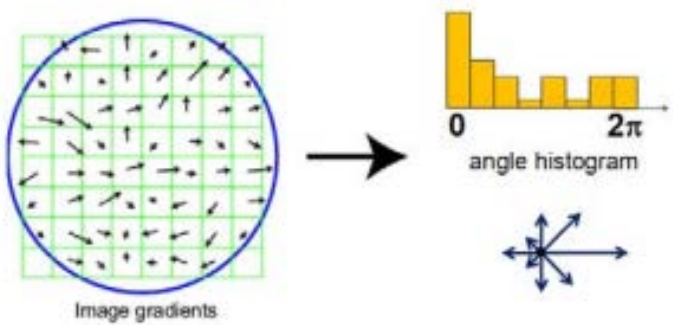


Figura 3.20 - Histograma de orientações de um ponto-chave (extraído de Lowe, 2004)

Cada ponto na vizinhança do ponto-chave é adicionado ao histograma com um peso específico. Este resulta da magnitude do seu gradiente afetada por uma janela Gaussiana circular com σ igual a 1,5 vezes maior que a escala do ponto-chave.

Os picos presentes no histograma de orientações correspondem às direções dominantes dos gradientes locais, sendo utilizados para definir a orientação dos pontos-chave.

Em situações onde existem múltiplos picos de elevada amplitude, são criados múltiplos pontos-chave na mesma localização e escala com diferentes orientações.

- **Construção do Descritor Local**

As etapas anteriores atribuem a localização, escala e orientação aos pontos-chave presentes em determinada imagem. Nesta etapa, para cada ponto-chave identificado, é estabelecido um descritor invariante à escala e rotação.

No seu trabalho, Lowe [65] propõe o seguinte para a obtenção dos descritores de pontos-chave:

A criação do descritor associado a cada ponto-chave resulta da computação da magnitude e orientação dos gradientes em redor da sua localização. Para que estes sejam invariáveis à rotação, as orientações dos gradientes destes pontos são giradas de um ângulo corresponde à orientação do ponto-chave definida na fase anterior.

No sentido de evitar mudanças súbitas do descritor em resposta a pequenas alterações na posição da janela, é utilizada uma função Gaussiana para dar peso à magnitude do gradiente em cada ponto da vizinhança do ponto-chave. A janela de suavização Gaussiana aplicada é de escala σ, igual à metade da largura da janela do descritor. Esta janela de suavização permite também reduzir o enfase nos gradientes longe do centro do descritor, mais afetados por erros.

Na Figura 3.21, ilustra-se a formação de um descritor de 2x2 histogramas de orientação (os gradientes são representados pelas pequenas setas em cada amostra da localização).

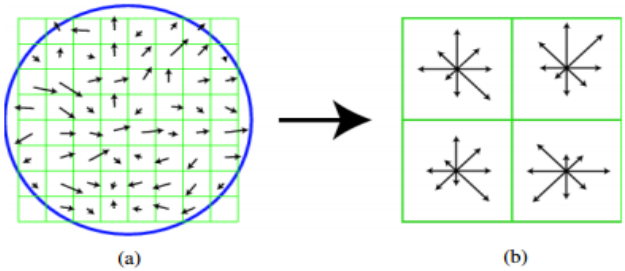


Figura 3.21 - Mapa de gradientes (a) e correspondente descritor (b) (extraído de Lowe,2004)

Na imagem anterior, são representados 8 direções para cada histograma de orientações, correspondendo o comprimento de cada vetor à magnitude da respetiva entrada do histograma.

Assim, segundo Lowe [65], o descritor é formado por um vetor que contem os valores de todas as entradas dos histogramas de orientações que, na Figura 3.21, correspondem aos comprimentos dos vetores em (b).

A utilização de descritores deste tipo, com elevado número de elementos, embora seja a mais eficiente, inviabiliza o reconhecimento em tempo real necessário ao sistema de condução autónoma.
No sentido de ultrapassar esta situação, propõe-se neste trabalho uma estrutura simplificada para os descritores, formada apenas por cinco elementos que permitem descrever a sua posição e características com precisão.

A grande diferença face ao proposto por Lowe [65] consiste na redução da informação da vizinhança do ponto-chave a dois elementos, orientação dominante e dimensão da vizinhança considerada. Os cinco elementos constituintes de um descritor são assim neste trabalho:

- Coordenadas do ponto-chave.
- Valor do extremo correspondente.
- Orientação dominante da vizinhança.
- Dimensão da área de vizinhança considerada.
- Escala a que foi identificado o ponto-chave.

Como se verifica posteriormente, embora se tenha simplificado consideravelmente a estrutura do descritor, o reconhecimento foi eficientemente efetuado.
De referir também que apesar da simplificação, o tempo de execução do algoritmo de deteção de sinalização tem bastante impacto no funcionamento do sistema de controlo.

2ª Fase – Comparação de Descritores

A segunda fase da tarefa de reconhecimento de sinais consiste na comparação dos descritores de pontos-chave de imagens.

A comparação dos descritores, que confirmará ou não a existência de pontos homólogos nas imagens, pode ser efetuada por exemplo através da distância Euclidiana, sendo neste caso os pontos candidatos à melhor correspondência os que apresentam menor distancia.

No seu trabalho, Lowe [65] utilizou uma variante do algoritmo Árvore kd (k-Dimensional Tree) designado de método de *Best-Bin-First* (BBF) [5]. Este diferencia-se do método original por lidar mais eficazmente com a avaliação de grandes quantidades de dados, sendo isso conseguido através do método de procura implementado. A metodologia utilizada penaliza a certeza do resultado obtido mas representa um considerável aumento na velocidade de análise.

Neste trabalho, o método de comparação implementado foi semelhante ao que serviu de base a Lowe, a procura da vizinhança mais próxima foi efetuada com base em um algoritmo de Árvore kd.

Algoritmo de Pesquisa de vizinhança - Árvores kd

Uma Árvore Kd [92] é uma estrutura de busca binária, de *k* dimensões, que se caracteriza por testar, em cada nó percorrido, um valor chave, que está associado a um discriminador.

O discriminador indica qual dimensão da Árvore e é utilizado para determinar o valor chave que divide os n restantes em duas subárvores. Os nós com os valores dessa altura maiores que o valor chave são armazenados do lado direito e os restantes do lado esquerdo do nó percorrido. Esta estrutura é eficiente para pesquisas que envolvam proximidade, tal como é o caso da busca da vizinhança mais próxima.

As consultas de proximidade geralmente são feitas com base num ponto central e um raio de busca. O resultado é um conjunto de pontos contidos dentro dessa região circular (área de buffer).
Há, basicamente, dois tipos de Árvores kd: a original, e a adaptativa. A diferença entre ambas está na política de escolha do discriminador, bem como no próprio armazenamento do dado espacial nos nós da árvore. Neste trabalho foi seguida a metodologia original.

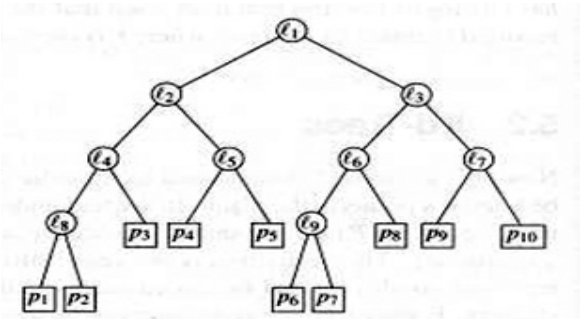


Figura 3.22 - Estrutura de uma Árvore kd

3.5 ***Sistema de Controlo***

Como referido anteriormente, pelos bons resultados obtidos em outros trabalhos e pela relativa simplicidade com que é desenvolvido, optou-se pela implementação de um controlador *Fuzzy* para desempenhar as tarefas fundamentais de controlo da movimentação do robot.

Este controlador será a base de todo o sistema, porém, o seu funcionamento dependerá também dos módulos de deteção de sinalização e obstáculos que sempre que necessário se poderão sobrepor a este de forma a controlar o robot em situações particulares.

3.5.1 Locomoção Diferencial

Conforme se descreverá mais detalhadamente no capítulo seguinte, a plataforma disponível para testar esta metodologia apresenta locomoção diferencial, sendo esta proporcionada por duas rodas diferenciais e duas rodas livres. Tal como é característico de robots móveis diferencias, não existe nenhum eixo exclusivamente responsável pela mudança de direção, esta é conseguida pela diferença de velocidades entre as duas rodas motoras que possui.

O modelo cinemático de robots com este tipo de locomoção pode ser representado por:

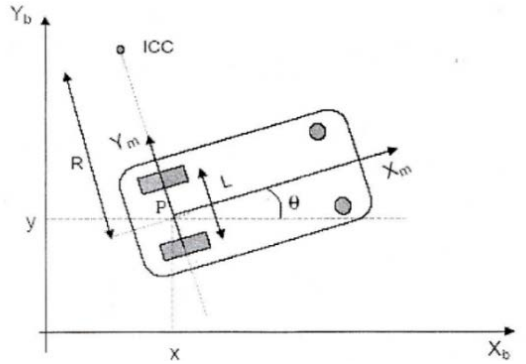


Figura 3.23 - Representação do modelo cinemático do robot

Sendo as relações entre velocidades lineares e angulares em cada roda dadas por:

$$v_d = w_d \times r_d \tag{3.45}$$

$$v_e = w_e \times r_e \tag{3.46}$$

Extrai-se da cinemática apresentada que:

$$\begin{cases} w_d(t) = \dfrac{v_d(t)}{R + \dfrac{L}{2}} \\ w_e(t) = \dfrac{v_e(t)}{R - \dfrac{L}{2}} \end{cases} \Rightarrow \begin{cases} w(t) = \dfrac{v_d(t) - v_e(t)}{L} \\ R = \dfrac{L(v_e(t) + v_d(t))}{2(v_e(t) + v_d(t))} \end{cases} \tag{3.47}$$

$$\tag{3.48}$$

De onde se pode concluir também que:

$$v(t) = w(t) \times R = \frac{1}{2}(v_d(t) - v_e(t)) \tag{3.49}$$

Notação:

$\mathbf{X_m Y_m}$ – referencial do corpo em movimento
$\mathbf{X_b Y_b}$ – referencial da base
$\boldsymbol{v_d}\mathbf{(t)}$ – velocidade linear da roda direita
$\boldsymbol{v_e}\mathbf{(t)}$ – velocidade linear da roda esquerda
$\boldsymbol{L}$ – comprimento do eixo entre as duas rodas motoras
$\boldsymbol{r}$ – raio de cada
$\boldsymbol{R}$ – raio da curvatura instantânea da trajetória do robot, relativamente ao ponto médio P, do eixo entre as duas rodas motoras.
$\boldsymbol{w_d}\mathbf{(t)}$ – velocidade angular da roda direita
$\boldsymbol{w_e}\mathbf{(t)}$ – velocidade angular da roda esquerda
$\boldsymbol{R}\mathbf{-}\boldsymbol{L}\mathbf{/2}$ – Raio da curvatura da trajetória da roda esquerda
$\boldsymbol{R}\mathbf{+}\boldsymbol{L}\mathbf{/2}$ – raio da curvatura da trajetória da roda direita

Como referido, pode-se confirmar através da equação (3.49), extraída do modelo cinemático apresentado que a direção e velocidade com que um robot móvel diferencial se move resulta da diferença entre as velocidades das suas rodas motoras.

Assim sendo, o sistema de controlo a implementar terá que controlar independentemente a velocidade das duas rodas para que o robot consiga alcançar determinado objetivo ou manter uma determinada orientação.

Um controlador *Fuzzy* [84] desempenha esta função com grande robustez.

3.5.2 Deslocamento Transversal na Faixa de Rodagem

O deslocamento transversal assume um duplo papel no que toca à condução. Se por um lado permite que o robot tenha conhecimento da sua localização na faixa de rodagem, podendo determinar se está demasiadamente próximo de qualquer uma da linhas delimitadoras da faixa de rodagem, por outro, ao controlar a distância a uma dessas linhas através do balanceamento (parâmetro descrito de seguida), torna possível desenvolver uma nova abordagem para a realização de manobras necessárias à condução, como por exemplo, a mudança de faixa de rodagem e o desvio de obstáculos.

3.5.2.1 Balanceamento

A principal função do sistema será manter o robot em movimento, centrado na faixa de rodagem que lhe é destinada. Isto consiste essencialmente em garantir um balanceamento adequado face ao objetivo a seguir.

O balanceamento pode ser encarado como o desfasamento, para a esquerda ou direita, em relação à situação de equilíbrio (eixo central, o objetivo). Este parâmetro tem implicações quer ao nível do posicionamento desejado do robot na faixa de rodagem, quer ao nível da definição de trajetórias

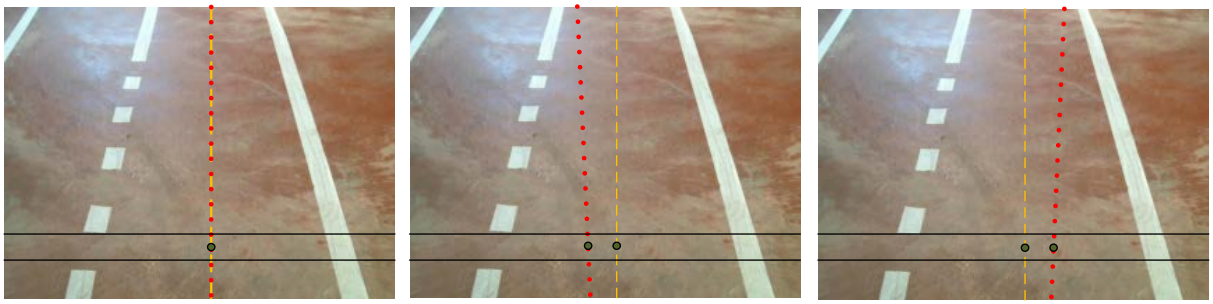


Figura 3.24 - Imagens representativas de balanceamento nulo - a), negativo - b) e positivo - c)

Os algoritmos implementados segundo o ponto (3.3), ao fornecerem as coordenadas centrais do eixo da faixa, permitem determinar numa zona específica (na imagem anterior delimitada pelas linhas a preto), o desvio ou balanceamento do eixo central do robot face ao eixo da faixa de rodagem. A zona de interesse onde é efetuada deverá ser variável consoante a velocidade, encontrando-se mais afastada do robot, ou seja, na zona mais superior da imagem, quando a velocidade for mais elevada.

De facto, com base neste parâmetro é possível fazer com que um robot móvel siga a uma distância definida do centro da faixa de rodagem onde circula, mude de faixa de rodagem ou mesmo que se desvie de um obstáculo.

Resumidamente, quando o balanceamento assume um valor negativo ou positivo, a informação que está de facto a passar para o sistema de controlo é de que o robot se encontra para a direita ou esquerda do centro da faixa de rodagem e que é necessário atuar na direcção, neste caso na velocidade dos motores, por forma a alinhar o robot com a nova posição de equilíbrio. No caso de o valor atribuído ser zero, o robot deverá efetuar o necessário para se manter alinhado.

3.5.3 Controlador *Fuzzy*

Como já referenciado anteriormente, para o controlo do movimento do robot, essencialmente para a sua orientação na faixa de rodagem, foi utilizado um controlador *Fuzzy [84]*.
A Lógica *Fuzzy* é uma técnica bastante eficaz em problemas relacionados com o controle de robots móveis [74], especialmente por se basear em conhecimento empírico, o que torna possível, dispensar o modelo dinâmico do sistema a ser controlado [17].

Pretende-se com a utilização deste tipo de controlador, tirar partido da sua capacidade de controlo independente de múltiplas variáveis (neste caso as velocidades das duas rodas) e facilidade de implementação, que dispensa uma complexa modelização do sistema.
Este controlador, receberá informação relativa ao balanceamento descrito no ponto anterior e consoante o seu valor, atuará na velocidade das rodas motoras para que este se volte a centrar no eixo da faixa de rodagem.

O funcionamento de um controlador *Fuzzy [84]* resulta da interação de várias etapas, ilustradas na imagem seguinte que representa a sua estrutura funcional:

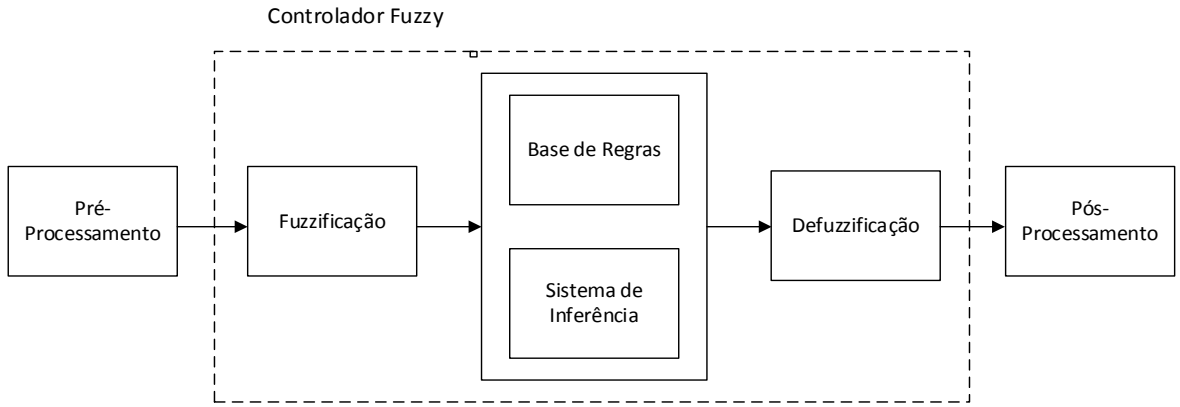


Figura 3.25 - Estrutura de um controlador Fuzzy

O controlador consiste essencialmente em três etapas, a *Fuzzificação*, o Sistema de Inferência e a *Defuzzificação*. Descrevem-se de seguida as etapas referidas e os restantes elementos constituintes do controlador:

Pré-Processamento

Frequentemente as variáveis de entrada de um controlador necessitam de ser condicionadas antes de servirem de base ao sistema de controlo.

Para isso, por norma, existe na estrutura funcional dos controladores uma etapa de pré-processamento. Nesta, são ajustados aspetos como:

- Arredondamento de valores
- Normalização ou reescalonamento
- Filtragem para eliminação de ruido
- Conversão de valor instantâneo para tendência
- Diferenciação ou integração

Fuzzificação

A *fuzzificação* é a primeira etapa do funcionamento de um controlador *Fuzzy [84]*. Esta consiste essencialmente na transformação dos valores numéricos das entradas do sistema em termos da linguagem natural com um certo grau de pertinência.

Para isso, são utilizadas funções de pertinência, definidas empiricamente com base no conhecimento do sistema, que permitem matematicamente determinar que o valor de entrada pertence a dada classificação (conjunto *Fuzzy*) com um determinado peso.

Como função de pertinência pode ser utilizada qualquer tipo de função, porém, vulgarmente, as funções de pertinência utilizadas são triangulares, trapezoidais e funções R-L[16]. Esta escolha deve-se à simplicidade da sua definição matemática.

[16] Funções do tipo trapezoidal em que um dos flancos, respetivamente o direito (**R**) ou esquerdo (**L**) se mantém constante (no valor 1) independentemente do valor da abcissa.

A definição das funções a utilizar deverá refletir um conhecimento preciso do sistema a controlar. De forma a orientar este processo, Hill, Horstkotte & Teichrow, no seu trabalho em 1990 [28], recomendam o seguinte:

- Começar com funções triangulares – As funções de pertinência para determinada entrada ou saída devem ser para valores centrais, triângulos simétricos e para os extremos funções do tipo R-L.

- Os limites das funções utilizadas devem permitir que cada valor de entrada deve pertencer a duas funções. O cumprimento desta regra pode ser dispensado para os valores extremos.

Exemplificam-se na imagem seguinte, possíveis funções de pertinência para uma entrada genérica x:

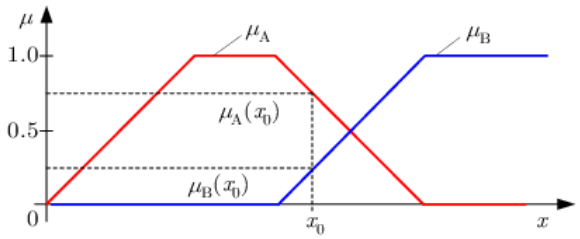


Figura 3.26 - Exemplo de funções de pertinência para variável genérica x

Como se verifica no exemplo anterior, as funções de pertinência representadas a vermelho e azul abrangem todos os valores possíveis para a entrada a que correspondem.
Estas funções, que representam determinado termo linguístico (μ_A *e* μ_B) fazem corresponder a cada valor da variável de entrada x, um certo valor de pertinência (μ) para cada termo linguístico.

Neste exemplo, para um valor x_0 é possível verificar que o valor x_0 pertence a μ_A com uma pertinencia de 0.75 e a μ_B com 0.25.

A conversão dos valores de entrada de cada variável em termos linguísticos com determinado grau de pertinência permite posteriormente que o sistema de inferência as utilize para tomar decisões.

Base de Regras

A base de regras é o elemento do controlador que descreve o comportamento que este deverá ter perante determinado valor de entrada. Como este controlador privilegia termos linguísticos, o desenvolvimento das regras de controlo é bastante simples, centrando-se o seu estabelecimento no conhecimento empírico do sistema a controlar e do seu comportamento. Para este controlador as regras serão do tipo “Se antecedente então consequente”.

Apresenta-se de seguida um pequeno exemplo de uma base de regras:

1. **Se** desvio **é** pequeno **e** velocidade **é** baixa **então** saída **é** nula
2. **Se** desvio **é** pequeno **e** velocidade **é** alta **então** saída **é** semi-moderada
3. **Se** desvio **é** grande **e** velocidade **é** baixa **então** saída **é** semi-moderada
4. **Se** desvio **é** grande **e** velocidade **é** alta **então** saída **é** moderada

Os termos a bold correspondem aos conectores que permitirão determinar uma resposta a determinada entrada. Os termos desvio, velocidade e saída são as variáveis do sistema, enquanto os termos pequeno, grande, alta, baixa, nula, semi-moderada e moderada representam os correspondentes linguísticos dos seus valores.

Sistema de Inferência

O sistema de inferência pode ser considerado o elemento mais importante do controlador. Este é o responsável por deduzir conclusões com base nos valores de entrada que lhe são fornecidos e no conhecimento disponível (presente na base de regras).

Durante a avaliação das regras, os valores de entrada provenientes da *fuzzificação*, que correspondem ao grau de pertinência de cada termo linguístico de um determinado valor de entrada, são avaliados e daí surge a dedução de novas alegações (os termos da regra de conclusão).

O passo da inferência (a avaliação da regra) é constituído por três fases, Agregação, Implicação e Acumulação:

- **Agregação**

A agregação compreende a avaliação das premissas. Este processo corresponde genericamente à aplicação dos operadores lógicos (conectores) "E" e "OU" das premissas às funções de pertinência respetivas.

Estes operadores lógicos podem ser implementados na teoria dos conjuntos difusos da seguinte forma:

Operadores de Agregação	
Mínimo ("E")	$\min(\mu_A(x), \mu_B(x))$
Máximo ("OU")	$\max(\mu_A(x), \mu_B(x))$

Tabela 3.3 - Operadores de Agregação

- **Implicação**

A implicação permite determinar o grau de certeza (ou execução) para a conclusão da regra. Tendo como base os fatores de certeza das condições (premissas) calculados anteriormente, representa a conclusão da alegação lógica "Se A Então B".

Simplificadamente, a implicação é a ligação entre os fatores de certeza das premissas e o grau de execução.

Matematicamente, o grau de certeza pode ser determinado por:

Operador de Implicação	
Mínimo	$\min(\mu_A(x), \mu_B(x))$

Tabela 3.4 - Operador de Implicação

- **Acumulação**

Frequentemente mais do que uma regra conduz à mesma conclusão, o que para os sistemas *Fuzzy* implica um tratamento diferenciado.

Tendo duas regras graus de execução diferentes, torna-se necessário que estes sejam reunidos num só. Isto é realizado pelo processo de acumulação, o qual corresponde à unificação dos resultados individuais com o operador lógico "OU".

Matematicamente, na teoria dos conjuntos difusos, pode ser utilizado o seguinte operador de acumulação:

Operador de Acumulação	
Máximo	$\max(\mu_A(x), \mu_B(x))$

Tabela 3.5 - Operador de Acumulação

Defuzzificação

A *defuzzificação* é a etapa onde se efetua a conversão dos resultados do processo de inferência descrito anteriormente para indicações concretas da ação a tomar.

Do ponto de vista matemático, o resultado do processo de inferência é a série *Fuzzy* para cada variável de saída (série *Fuzzy* de saída), isto é, a série:

Variável É termo 1 OU
Variável É termo 2 U
……OU
Variável E termo N

A série *Fuzzy* de saída da variável linguística é, por conseguinte, a união de todos os termos, tendo uma função de pertinência que é calculada com base nos seus graus de pertinência.

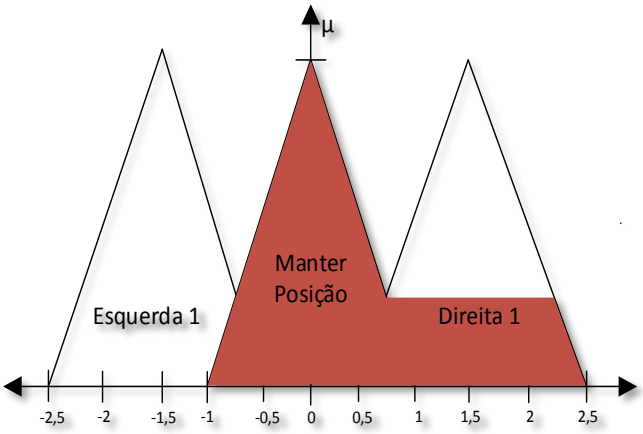


Figura 3.27 - Exemplo de função de pertinência da série Fuzzy

Na imagem acima, a zona sombreada a verde representa a função de pertinência de uma saída determinada pelo processo de inferência. O eixo horizontal representa os vários valores possíveis para a saída e o eixo vertical representa a sua pertinência.

A tarefa de defuzzificação consiste em transformar a função de pertinência obtida para a saída num resultado concreto, que possa ser utilizado para controlar diretamente o sistema em causa. Existe uma grande diversidade de métodos para concluir esta tarefa, porém, devido à questão da velocidade de computação envolvida na sua execução, as duas funções seguintes são as mais utilizadas.

Determinação do Centro de Gravidade (COG)

$$\bar{x} = \frac{\int_a^b x\mu_{saida}(x)\,dx}{\int_a^b \mu_{saida}(x)\,dx} \tag{3.50}$$

Média do Máximo

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} {x_i}^{max} \tag{3.51}$$

Onde:

$\bar{x}$ *– é o valor de saída desfuzzificado*

$\mu_{saida}(x)$ *é a função de pertinencia da série Fuzzy de saída entre o range a e b*

${x_i}^{max}$ *é o valor de x no ponto ao qual a série Fuzzy de saída atinge o máximo global*

Pós-Processamento

Após o controlador determinar as saídas adequadas para determinado estado ou situação, poderá ser necessário que estas sejam convertidas de forma a serem aplicadas diretamente ao sistema. Tal como indicado na etapa de pré-processamento, pode ser necessária uma alteração de escala para que o sistema possa determinar a saída adequada, sendo nesse momento necessário, um novo reescalonamento para tornar a saída adequada aos atuadores existentes.
Conversão dos valores matemáticos para os correspondentes valores físicos neste sistema, por exemplo, uma saída [-1,1] poderá ter que ser convertida para [-10,10] V.

Frequentemente nesta etapa é aplicado um ganho à saída, sendo também por vezes aplicado um integrador.

3.5.4 Deteção e desvio de obstáculos

A deteção e desvio de obstáculos é uma funcionalidade essencial em qualquer robot móvel autónomo. Sem esta, poderia a qualquer momento ocorrer um embate que danificasse os componentes ou ferisse alguém.

Para evitar este tipo de situação, recorre-se à deteção dos obstáculos que rodeiam a estrutura através de sensores de ultrasons. A sua grande precisão permite que a cada instante se reconheça a existência de obstáculos no ambiente que o rodeia e a distância a que estes se encontram de si. Com esta informação, torna-se possível desenvolver um algoritmo que evite o embate e consoante a situação, contorne o obstáculo ou o ultrapasse.

Este algoritmo deve funcionar em simultâneo com o controlador descrito anteriormente, verificando continuamente o meio que o rodeia e cooperando com este aquando da deteção de um obstáculo e durante o seu contorno.

3.5.5 Deteção Sinalização

A deteção e o cumprimento da sinalização existente é outra das funcionalidades com que se pretende dotar o robot.
Conseguida com recurso ao algoritmo de reconhecimento de objetos referido anteriormente, este deverá tal como a deteção de obstáculos, funcionar em simultâneo com o controlador, procurando continuamente algum sinal.
Quando é detetado algum sinal, o algoritmo responsável deverá condicionar o comportamento do robot em virtude do seu significado. Por exemplo no caso da passadeira, devem ser respeitados determinados comportamentos específicos.

4. Implementação do Sistema

Para avaliar se a metodologia proposta anteriormente é adequada ao desenvolvimento de um sistema de condução autónoma que possa ser aplicado num robot móvel, é necessária a sua implementação e teste em condições reais, numa plataforma robótica móvel.

Nesse sentido, recorreu-se a uma estrutura robótica previamente desenvolvida que possuía a maioria dos componentes necessários aos testes a efetuar. Elementos como as *webcams* utilizadas para a captação de imagem, que não existiam na estrutura base, foram facilmente adicionadas à mesma.

Os algoritmos de controlo descritos, como se refere mais detalhadamente de seguida, foram implementados com recurso à linguagem de programação Python em conjugação com bibliotecas adicionais para comunicações série, processamento de imagem e desenvolvimento de controladores *Fuzzy*.

O funcionamento do sistema de controlo fica dependente de um computador portátil que será colocado na plataforma móvel. Este estará interligado a todos os componentes do robot. Os algoritmos desenvolvidos serão executados em tempo real neste computador.

4.1 ***Estrutura e Composição***

Descrevem-se nos pontos seguintes os elementos da plataforma robótica utilizada neste trabalho:

Figura 4.1 - Robot móvel desenvolvido

4.1.1 Estrutura mecânica

A estrutura da plataforma robótica consiste em duas plataformas em perfil de alumínio com 16 mm de secção e com a base da plataforma em aço zincado de 2 mm de espessura.
A plataforma inferior destina-se à fixação dos eixos e dos diversos módulos do robot. O eixo traseiro, destinado às duas rodas motoras de 5 cm de raio e o eixo dianteiro destinado às rodas livres, são ambos em ferro de modo a oferecerem uma maior rigidez e capacidade de carga. Na plataforma superior, são fixados no perfil de alumínio os sensores sonares e o suporte para as *webcams*, sendo na sua base que assentará o computador de controlo portátil.

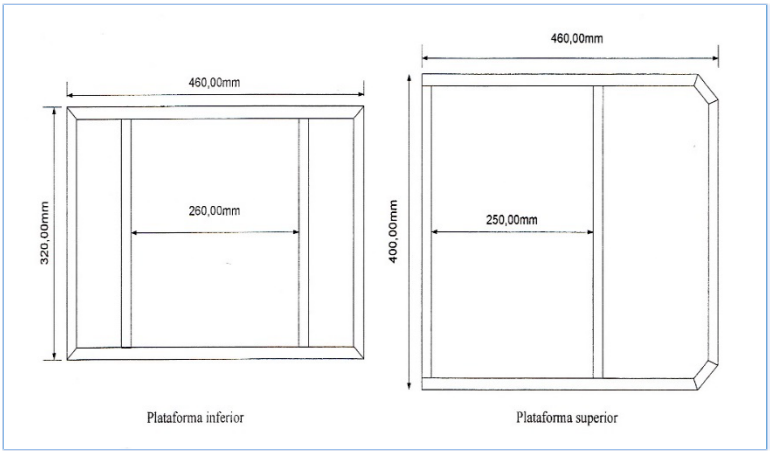


Figura 4.2 - Plataformas da estrutura do robot

4.1.2 Sistema de energia

A energia necessária á alimentação dos motores é fornecida por duas baterias BT-12M7.OAT ligadas em série, de chumbo-ácido de 12 Volts com uma capacidade de 7 Ah cada.
A energia necessária á alimentação da parte eletrónica (placas controladoras) é fornecida por uma bateria BP 12-1.3, de chumbo-ácido de 12 V com uma capacidade de 1,3 Ah. A alimentação das placas controladoras que constituem o sistema é realizada via placa de alimentação, que possui quatro reguladores de tensão de 5 V.

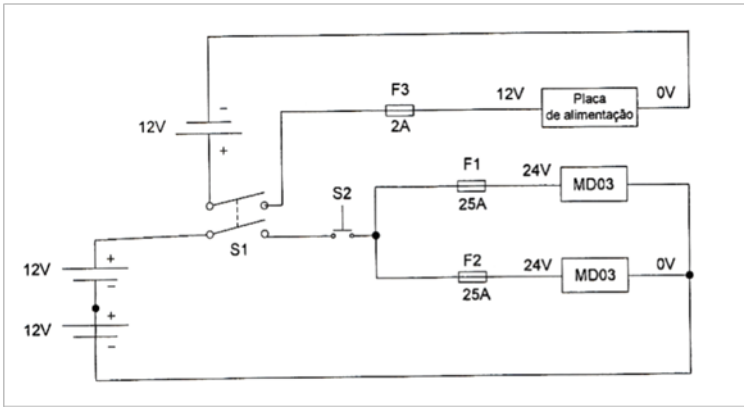


Figura 4.3 - Esquema elétrico da alimentação do sistema

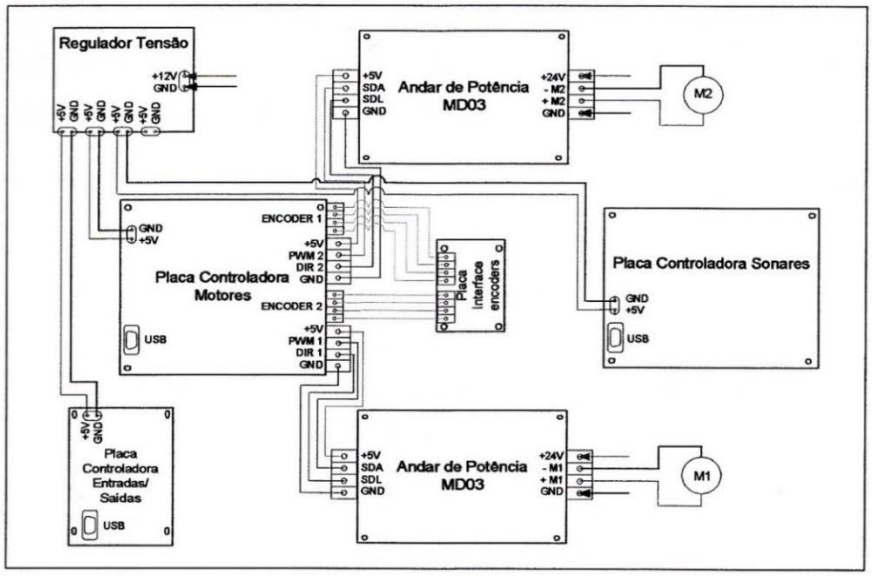


Figura 4.4 - Localização e respetivas ligações dos módulos na plataforma inferior

A alimentação às *webcams* é efetuada através da ligação USB ao computador portátil.

A cablagem utilizada nas ligações e o calibre dos fusíveis foram escolhidos de acordo com as correntes esperadas no sistema.

4.1.3 Sistema Sensorial

Para conhecer o ambiente que o rodeia e a velocidade com que se movimenta, este robot móvel possui sensores de três tipos, sensores de ultrasons, *webcams* e encoders. São estes equipamentos que fornecem as informações necessárias para que o sistema de controlo possa orientar convenientemente o robot.

4.1.3.1 Sensores de Ultrasons

Para possibilitar a implementação dos algoritmos de deteção e desvio de obstáculos, este robot móvel foi dotado de oito sensores de Ultrasons. Posicionados em torno de toda a sua estrutura, permitem cobrir toda a área em seu redor.

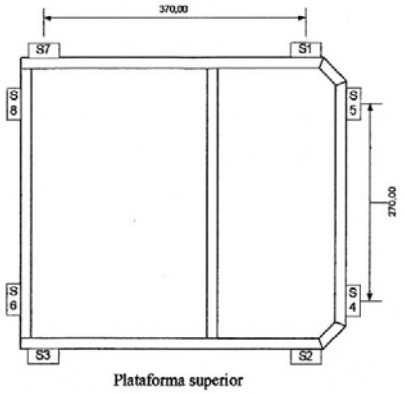


Figura 4.5 - Disposição dos sensores de ultrasons na plataforma superior

O sensor utlizado, o Devantech SRF04, funciona com uma tensão de alimentação de 5 V e um consumo de corrente tipicamente na ordem dos 30 mA, possui um transmissor e um recetor separados, permitindo obter distâncias de deteção desde os 3 cm aos 3 metros.
A frequência das ondas ultrassónicas emitidas pelo transdutor é de 40kHz, sendo pouco provável que o eco de uma onda de som desta frequência seja proveniente de outra fonte que não o próprio sensor de ultrasons.
O feixe ultrassónico tem um padrão cónico, com um ângulo de abertura de 55º que não pode ser alterado.

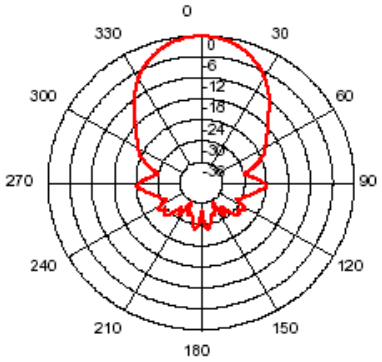


Figura 4.6 - Padrão do feixe de ultrasons

Isto justificou particular atenção ao seu posicionamento na estrutura. A altura a que foi colocado evita falsos ecos provenientes da reflexão do feixe no piso. O fabricante aconselha uma altura mínima de montagem de 30 cm.

Ligações do sensor de Ultrasons

O SRF04 requer quatro ligações para operar. A alimentação, a massa, o impulso de disparo para envio do feixe e a saída onde será fornecida um impulso proporcional ao eco de entrada. Estas devem ser ligadas diretamente a portas de registo de um microcontrolador.

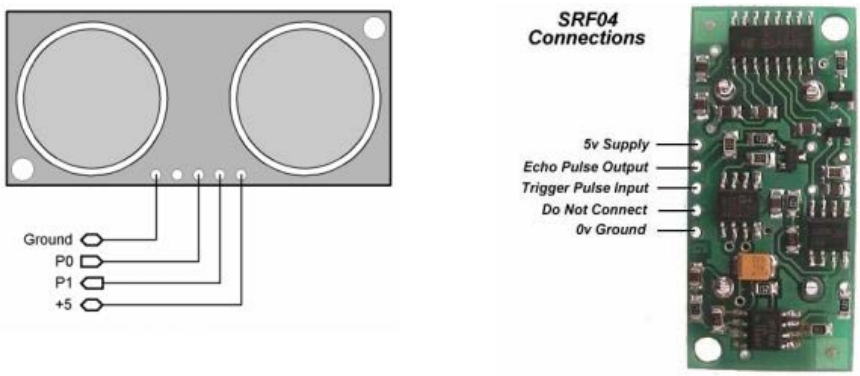


Figura 4.7 - Ligações do sensor de ultrasons

Placa controladora de Ultrasons

Esta placa com uma alimentação de 5 V permite o controlo dos oito sensores de ultrasons SRF04 através de USB. Utiliza um microcontrolador PIC 16F876 e dispõe de seletores de duas posições que permitem ligar e desligar alguns dos sensores.

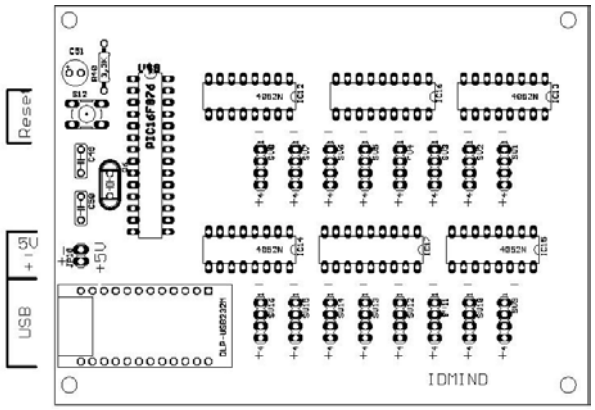


Figura 4.8 - Placa controladora de 16 sensores de ultrasons com ligação USB

A ligação USB do computador é convertida em RS-232 por um chip DLP-USB232M, possibilitando assim a ligação ao microcontrolador. A comunicação é efetuada a 115200 bps.

Procedimento de disparo dos sensores de Ultrasons

O computador começa por enviar um caráter ASCII " A" para o PIC 16F876 solicitando o início do procedimento de disparo dos sonares que é efetuado de acordo com a seguinte sequência:

1. O PIC efetua o disparo dos sonares 1 e 5, ficando a aguardar o sinal de eco de cada um dos sonares.

2. O PIC efetua o disparo dos sonares 2 e 6, ficando a aguardar o sinal de eco de cada um dos sonares.

3. O PIC efetua o disparo dos sonares 3 e 7, ficando a aguardar o sinal de eco de cada um dos sonares.

4. O PIC efetua o disparo dos sonares 4 e 8, ficando a aguardar o sinal de eco de cada um dos sonares.

Após realizar a leitura dos oito sensores de ultrasons o microcontrolador devolve uma *string*[17] de 18 bytes em que os dois primeiros são os caracteres "A" e "B".

Os oito pares de bytes que se seguem aos carateres "A" e "B" correspondem aos bytes de dados de cada um dos oito sensores de ultrasons, sendo o primeiro byte de cada par, os 8 mais significativos e o segundo os 8 bits menos significativos. O valor codificado nestes 16 bits

[17] Termo vulgarmente usado em informática para designar uma sequência de caracteres.

corresponde ao número inteiro de ciclos do oscilador que decorre entre o envio e a receção do feixe de ultrasons.

Para converter os dados enviados por cada um dos sonares, multiplica-se o ByteH (mais significativo) por 255 e soma-se o resultado ao ByteL (menos significativo).
Divide-se o resultado por 2x 1843200 para obter o valor em segundos e finalmente multiplica-se o resultado pela velocidade do Som 340,9 m/s, ficando-se assim com o valor da distância em metros. Para obter o valor da distância em centímetros basta multiplicar por 100.

$$Distância\ (cm) = \left(\frac{ByteH \times 255 + ByteL}{2 \times 1843200}\right) \times 340{,}9 \times 100 \qquad (4.1)$$

4.1.3.2 Webcams

Neste trabalho foram utilizados duas câmaras para captar informação visual do meio onde o robot se encontra. São indispensáveis para o sistema de condução autónoma que se pretende implementar, sendo a informação recolhida por estas que permitirá determinar se a trajétoria de movimento do robot é a pretendida e se existe algum tipo de sinalização no local.

Como referido anteriomente, durante o desenvolvimento deste sistema existiu sempre a preocupação de o fazer do modo mais económico possível. Nesse sentido, optou-se pelo uso de duas vulgares *webcams*. Apesar do baixo custo, as suas características permitem o desempenho da função que lhes está destinada com grande eficácia.

Figura 4.9 - Webcams no robot móvel

De referir que embora se deseje que este sistema possa ser expansível ao sector automóvel, logo, passando a ser utilizado no exterior, neste trabalho, as condições do ambiente onde o robot se movimentou são controladas, favorecendo assim a qualidade de captação das câmaras utilizadas.

Webcam destinada à Deteção das Faixas de Rodagem

A *webcam* utilizada para a deteção das faixas de rodagem é uma Logitech HD Webcam C615.

Figura 4.10 - Webcam utilizada no reconhecimento da faixa de rodagem

Características Logitech HD Webcam C615	
Resolução	1920 x 1080 *pixéis*
Cadência de Captação	30 fps[18]
Focagem	Automática
Diagonal do Campo de Visão	74º
Interface	USB 2.0

Tabela 4.1 - Principais características da Logitech HD Webcam C615

Esta câmara é responsável pela aquisição de imagens das faixas de rodagem. Com recurso a estas, será possível posteriormente fornecer ao sistema de controlo implementado a informação do seu posicionamento em relação ao eixo central do robot.

Como se verifica na Figura 4.9, esta câmara posiciona-se no topo do suporte das câmaras, está orientada horizontalmente segundo o eixo central do robot e verticalmente no sentido do piso.

Estas orientações são essenciais ao funcionamento do sistema. Um desvio na horizontal forneceria ao sistema informações erradas relativamente à orientação do robot na faixa de rodagem. Verticalmente, o ângulo que a câmara apresenta em relação ao seu suporte condiciona a eficiência do sistema de controlo.

Influência do Ângulo da Câmara no Desempenho do Sistema					
Ângulo da Câmara (Θ)	***15º***	***30º***	***45º***	***60º***	***75º***
% Conclusão do percurso (2 cm/s)	*80%*	*87 %*	*93 %*	*73 %*	*67 %*
% Conclusão do percurso (4 cm/s)	*73 %*	*80 %*	*87 %*	*67 %*	*60 %*

Tabela 4.2 - Influência do ângulo da câmara no desempenho do sistema

[18] *Frames Per Second*, ou em português, quadros por segundo, significam o número de imagens que o dispositivo regista, processa ou exibe por unidade de tempo.

Testes efetuados na pista descrita posteriormente no capítulo 4, que se encontram resumidos na tabela anterior, permitem confirmar que o ângulo da câmara face ao seu suporte condiciona o desempenho do sistema. Para as duas velocidades em teste, representativas da gama de valores que se pretende utilizar, o ângulo que conduziu a melhores resultados foi 45º. Assim, para os restantes testes efetuados ao sistema de controlo, esta câmara será posicionada de forma que o ângulo que descreve face à sua base se mantenha sempre em 45º.

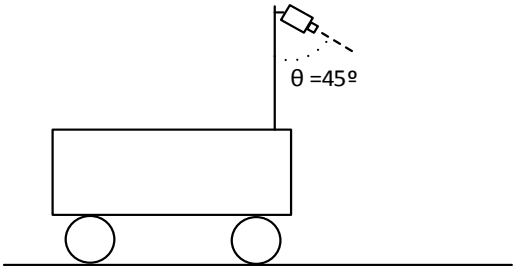


Figura 4.11 - Posicionamento da câmara de deteção de faixas

Webcam destinada ao reconhecimento de sinais

A *webcam* utilizada para o reconhecimento de sinais é uma Logitech C170.

Figura 4.12 - Webcam utilizada no reconhecimento de sinais

Características Logitech C170	
Resolução	1024 x 768 *pixéis*
Cadência de Captação	30 fps
Focagem	Fixa
Diagonal do Campo de Visão	58º
Interface	USB 2.0

Tabela 4.3 - Principais características da Logitech C170

Esta câmara será responsável por adquirir imagens do ambiente em redor do robot que permitirão ao módulo de reconhecimento de sinais detetar a existência de algum tipo de sinalização.

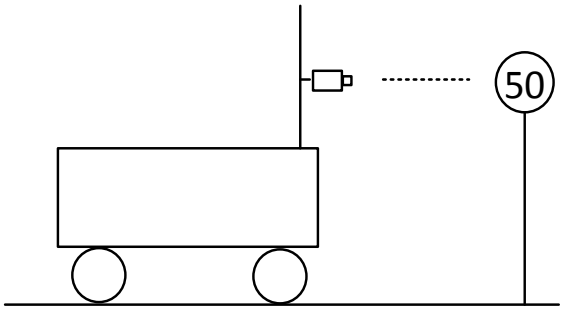


Figura 4.13 - Posicionamento da câmara de reconhecimento de sinais

Esta câmara está posicionada de forma que a sinalização existente esteja no seu campo de visão.

De referir que apesar das *webcams* permitirem resoluções superiores, neste trabalho, de forma a uniformizar e tornar mais célere o processamento de imagem, a captação das duas câmaras foi sempre efetuada à resolução de 640x480 *pixéis*.

4.1.3.3 Encoder

Neste trabalho, para a medição da velocidade das rodas motoras, foram utilizados encoders incrementais de 500 linhas por rotação, que geram dois sinais em quadratura (canal A e B), o que lhes confere uma resolução de quatro vezes o número de linhas do encoder, ou seja, 2000 posições por rotação.

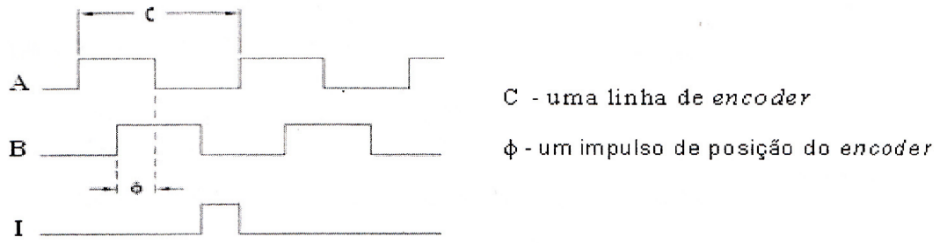


Figura 4.14 - Sinais de saída do encoder

Analisando qual dos sinais em quadratura está em avanço, é possível determinar o sentido de rotação do motor. Com o motor em rotação no sentido direto, apresenta-se em avanço o canal A. Em sentido inverso, o canal em avanço será o B.

O encoder incremental disponibiliza ainda um terceiro sinal designado por sinal de indexação. Este sinal, sincronizado com o canal B, permite contar o número de rotações do motor recorrendo a uma marca adicional no disco, geralmente designada por "origem" ou "zero".

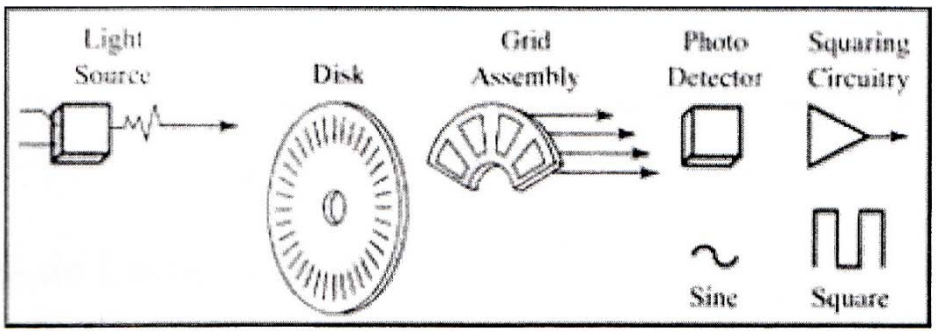


Figura 4.15 - Elementos constituintes de um encoder incremental

4.1.4 Locomoção

A estrutura robótica móvel utilizada neste trabalho, como abordado no capítulo anterior, apresenta locomoção diferencial. Isto significa que o seu movimento e orientação se deve unicamente à diferença de velocidades das suas duas rodas motoras. Descrevem-se de seguida os principais elementos constituintes do sistema de locomoção.

4.1.4.1 Motores

A força motriz do robot assenta em dois motores elétricos de magnetos permanentes.
Cada motor é alimentado a tensão contínua (DC) de 24V e disponibiliza uma potência máxima de 70W com uma corrente de arranque máxima 21,5 A.

4.1.4.2 Andar de Potência MD03

O MDO3 é um conversor estático de potência DC- DC (CHOPPER) de quatro quadrantes, que providencia amplificação de potência para alimentar o motor. A velocidade de rotação do motor é controlada por PWM, *"Pulse Width Modulation"* (modulação por largura de impulso) a uma frequência de 7,8kHz. Utiliza MOSFET's como dispositivos semicondutores e requer uma alimentação de 5V/ 50 mA para o controlo lógico e uma alimentação de 24V para os motores.

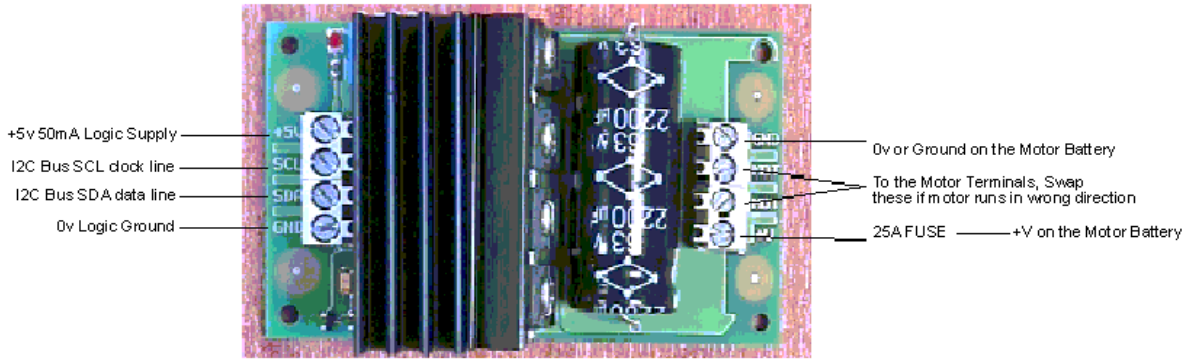


Figura 4.16 - Andar de potência MD03

Os andares de potência podem operar em diferentes modos, sendo que a configuração adotada para o funcionamento neste trabalho permite realizar o controlo de velocidade de rotação do motor a partir de um sinal PWM na entrada SDA (Figura 4.16).
O MDO3 possui um filtro resistivo-capacitivo que converte o sinal PWM do controlador de motores LM629N numa tensão analógica para realizar o controlo do motor. Uma razão cíclica do sinal PWM de 0% representa 0V (motor parado) e uma razão cíclica de 100% representa 5V (tensão de saída máxima).

O controlo da direção de rotação é feito através do nível lógico de entrada SCL (Figura 4.16), onde o nível lógico 1 indica a rotação em sentido direto e o nível lógico 0 indica a rotação em sentido inverso.

4.1.4.3 Placa Controladora de Motores

Sendo alimentada a 5V (DC), esta placa possui dois controladores LN629N que permitem o controlo dos dois motores de tensão contínua de 24V. Utiliza uma interface USB/RS-232 para possibilitar a comunicação entre o computador de controlo e o microcontrolador PIC 16F876, sendo este o responsável pelo encaminhamento da informação para o respetivo controlador de motores

Esta placa possui ainda duas saídas de quatro pinos destinadas à alimentação e controlo por modulação de largura de impulso (PWM) dos andares de potência e duas entradas de encoders para fechar a cadeia de controlo.

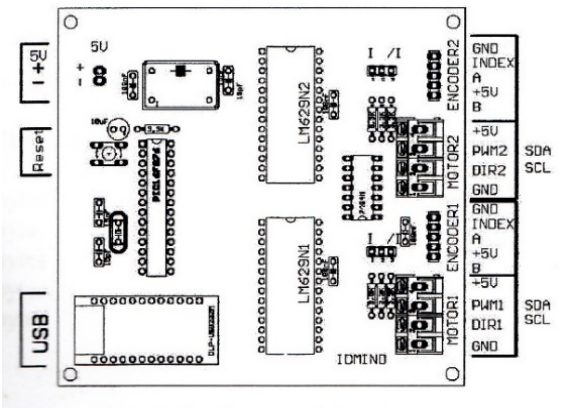


Figura 4.17 - Placa controladora de motores

Protocolo de Comunicação

A comunicação entre o computador e os controladores de motores LM629N é feita através do microcontrolador PIC 16F876.

No envio de uma mensagem, seja ela de dados ou comandos, de escrita ou de leitura, o primeiro byte de informação a enviar ao microcontrolador é sempre o byte de endereçamento. Neste byte informamos o microcontrolador das operações que vamos realizar, do número de bytes que constituem a informação a enviar e a qual dos controladores de motores essa informação se destina. O envio do byte de endereçamento é feito do bit mais significativo (bit 7) para o menos significativo (bit 0) e tem a seguinte estrutura:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2 Bit 1 Bit 0
Motor	W/R	D/C	-	-	Nº de Bytes

Tabela 4.4 - Byte de endereçamento do LM629N

Bit7- Motor
0-Motor 1
1-Motor 2

Bit6- W/R
0 – Operação de leitura
1 – Operação de escrita

Bit5 – D/C
0 – Controlo
1 – Dados

Bit4 e Bit3 – Não utilizados

Bit2 a Bit0 – Nº bytes a serem escritos ou lidos
000 – 0 bytes
001 – 1 byte
010 – 2 bytes
011 – 3 bytes
100 – 4 bytes

Controlador de motores LM629N

O controlador de motores LM629N está concebido para ser utilizado numa variedade de motores de corrente contínua que providenciem um sinal de retorno de posição incremental em quadratura. O *firmware* dos controladores disponibiliza um conjunto de comandos de alto nível, que facilita a programação do controlo digital de movimentos.

Gama de Posições	-1.073.741.824 a 1.073.741.823 (impulsos)
Gama de Velocidades	0 a 16.383 (impulsos/amostragem), com uma resolução de $\frac{1}{2}^{16}$ (impulsos/amostragem)
Gama de Acelerações	0 a 16.383 (impulsos/amostragem), com uma resolução de $\frac{1}{2}^{16}$ (impulsos/amostragem)
Sinais de controlo do motor	PWM com resolução de 7 bits e 1 bit de sinal de amplitude
Modos de Operação	Em Posição ou Velocidade
Dispositivo de Localização	Encoder Incremental, com 2 sinais em quadratura e um sinal de indexação
Algoritmo de Controlo	Filtro Proporcional Integral Derivativo (PID), com programação do limite de integração
Intervalos de Amostragem	Termo Proporcional e Integral: 2048/f_{CLK} Termo Derivativo: 2048/f_{CLK} a (2048x56) / f_{CLK} em passos de 2048/f_{CLK}

Tabela 4.5 - Características do LM629N

Este controlador funciona com uma frequência de relógio de 6MHz (oscilador externo) e providencia um sinal de saída PWM de 8 bits à frequência de 11,72KHz, para controlo direto do conversor de potência DC – DC (andar de potência MD03) a jusante.

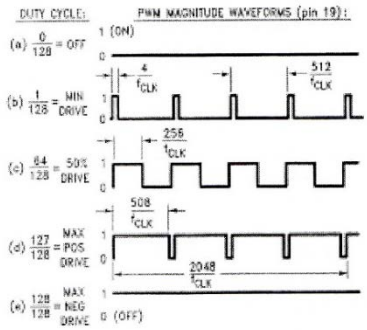


Figura 4.18 - Sinal de saída PWM

Informação de Retorno Posicional

O controlador de motores recebe a informação de retorno posicional através de um encoder incremental, que providencia dois sinais em quadratura e um sinal de impulso de indexação.
Os sinais em quadratura, sincronizados com o sinal de relógio do LM629N, são utilizados para manter a referência da posição do motor. De cada vez que uma transição lógica ocorre numa das entradas do sinal em quadratura, o registo interno de posição do LM629N é incrementado ou decrementado de acordo.

A saída do impulso de indexação fornecido pelo encoder assume o estado lógico 1 vez por rotação. Se programado, o LM629N irá guardar a posição absoluta do motor num registo dedicado (*Index register*).

Geração do Perfil de Velocidade ou Trajetória

O gerador do perfil de velocidade trapezoidal do LM629N calcula a posição desejada do motor em função do tempo. No modo de operação em posição, especifica-se a aceleração, a velocidade máxima e a posição final. O LM629N utiliza esta informação para calcular o movimento, acelerando até a velocidade máxima ser alcançada ou até ao início da desaceleração, de modo a parar na posição final especificada. Em qualquer instante do deslocamento, a velocidade máxima assim como a posição alvo podem ser alteradas, sendo refletidas pelo acelerar ou desacelerar do motor. De referir que no modo de operação em posição os valores de aceleração e de desaceleração são idênticos.

Quando em modo de velocidade, o motor acelera até uma velocidade específica com a aceleração previamente definida, mantendo a velocidade até o comando de paragem ser enviado. A velocidade é mantida constante através do avanço da posição pretendida a um ritmo constante.

Parâmetros de trajetória: Posição, Velocidade e Aceleração

Todos os parâmetros de trajetória são valores de 32- bits. (8 bytes). A posição pretendida é uma quantidade especificada de uma gama de valores inteiros entre os -[2^{30}] (C0000000 hex) e os [2^{30}]-1 (3FFFFFFF hex). A aceleração e velocidade são especificadas também em valores de 32-bits mas com uma gama de valores positiva entre os 0 (00000000 hex) e os [2^{30}]-1 (3FFFFFFF hex). Os 16 bits menos significativos (bit 0 a 15) tanto na aceleração como na velocidade correspondem á parte fracionária, o que faculta um aumento da resolução média da velocidade e da aceleração. A parte inteira (bit 31 a 16) do parâmetro velocidade específica quantas contagens por intervalo de amostragem o motor irá efetuar.

Os valores de velocidade e aceleração são multiplicados por 65536 para ajustar o formato da parte fracionária ao formato requerido para o carregamento de dados. De notar que o valor carregado para a aceleração não pode exceder o valor carregado para a velocidade, assim como depois de escalonar os valores de velocidade e aceleração não podem ser carregados valores somente fracionários. O valor da resolução do encoder é devido ao método usado para descodificar os sinais de quadratura do encoder. Os valores dos parâmetros posição, velocidade e aceleração são depois convertidos em código hexadecimal de forma a serem carregados no LM629N.

Indica-se de seguida um exemplo de cálculo dos parâmetros de trajetória em que se pretende que o motor acelere a 10 cm/s^2 até atingir a velocidade de 25 cm/s, e depois desacelere numa posição 200 cm após o início:

Nº de linhas do encoder: encLines = 500
Resolução do encoder: encRes = 4.0
Relação de transmissão: kRdutor= 91/6
Frequência do oscilador da placa dos controladores: clockTime = 6MHz

Tempo de amostragem do termo proporcional e integral: sampleTime= 2048/clokTime

- **Parâmetros Posição**

P = posição alvo (nº de impulsos do encoder)

kPosition = (encLines x enRes x kRedutor) / 31,4159

P = distância pretendida em cm x kPosition

$P = 200 \times 500 \times 4 \times \frac{91}{6} \div 31{,}4159 = \ 193108$ Impulsos (valor a carregar)

P (codificado) =0002F254 (código hexadecimal carregado para o LM629N)

- **Parâmetros Velocidade**

V = velocidade (impulsos/ amostragem)

kVelocity = encLines x encRes x kRedutor x sampleTime / 31, 4159

V= velocidade pretendida x kVelocity

$V = 20 \times 500 \times 4 \times \frac{91}{6} \div 31{,}4159 = \ 8{,}23$ Impulsos/Amostra

V=8, 23 x 65536= 539361,29

V (arredondando) = 539361 (valor a carregar)

V (codificado) = 00083AE1 (código hexadecimal carregado para o LM629N)

- **Parâmetros Aceleração**

A = aceleração (impulsos amostra/amostra)

kAcceleration = encLines x encRes x kRedutor x sampleTime x sampleTime /31, 4159

A = aceleração pretendida x kAcceleration

$A = 10 \times 2000 \times \frac{91}{6} \times 341 \times 10^{-6} \times 341 \times 10^{-6} / 31{,}4159 = 0{,}0011$

A = 0,0011 x 65536

A (Arredondado) = 72 (valor a carregar)

A (codificado) = 00000048 (código hexadecimal carregado para o LM629N)

Resumem-se na tabela seguinte os comandos do controlador de motores:

Comando	Tipo	Descrição
RESET	Inicialização	Reset ao LM629N
DFH	Inicialização	Define a posição de zero absoluto
SIP	Interrupção	Define a posição do sinal de indexação
LPEI	Interrupção	Interrupção por erro de posição
LPES	Interrupção	Paragem por erro de posição
SBPA	Interrupção	Define ponto de referência absoluto
SBPR	Interrupção	Define ponto de referência relativo
MSKI	Interrupção	Definição de potenciais interrupções
RSTI	Interrupção	Reset de interrupções
LFIL	Filtro	Carregamento dos parâmetros do filtro
UDF	Filtro	Atualização do filtro
LTRJ	Trajetória	Carregamento de trajetória (p,v,a)
STT	Trajetória	Início de movimento
RDSTAT	Leitura	Leitura do byte estados
RDSIGS	Leitura	Leitura do registo de sinais
RDIP	Leitura	Leitura da posição de indexação
RDDP	Leitura	Leitura da posição desejada
RDRP	Leitura	Leitura da posição real
RDDV	Leitura	Leitura da velocidade desejada
RDRV	Leitura	Leitura da velocidade real
RDSUM	Leitura	Leitura do erro integral acumulado

Tabela 4.6 - Conjunto de comandos do controlador de motores LM629N

Para uma consulta mais detalhada, as funcionalidades e características dos comandos apresentados na tabela anterior encontram-se referenciados no **Anexo I**.

Parâmetros do controlador PID dos motores

Independentemente do modo de operação do controlador (posição ou velocidade), o utilizador especifica os valores pretendidos para a posição, velocidade e aceleração e estes são convertidos em impulsos, impulsos/amostragem e impulsos/amostragem^2 respetivamente.

Em funcionamento, o LM629N utiliza essa informação para gerar o perfil de velocidade trapezoidal e a partir deste perfil, calcula a posição do motor a cada amostragem.

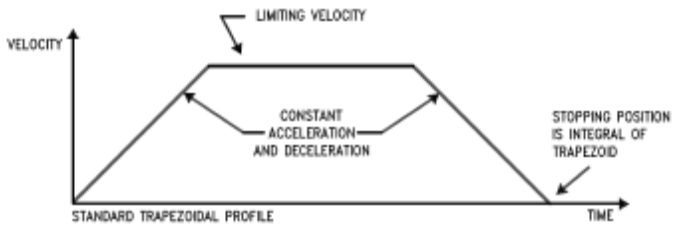


Figura 4.19 - Perfil de velocidade

A função do controlador digital do LM629N (controlador PID), consiste em corrigir o erro resultante da diferença entre o número de impulsos calculado para uma determinada amostragem (referencia) e o valor real de posição velocidade e aceleração devolvidos pelo encoder.

A sintonia dos parâmetros deste controlador poderia ser efetuada de várias formas, porém, como se pretendia de seguida tirar partido de um dos grandes benefícios do controlador *Fuzzy*, a dispensa da modelização do sistema eletromecânico, os parâmetros para o controlador PID foram determinados empiricamente, recorrendo-se para isso a ensaios práticos a partir dos quais se efetuaram os ajustamentos necessários para a obtenção de uma boa performance do sistema de controlo.

Após vários ensaios e consequentes afinações, adotaram-se os seguintes valores para os coeficientes do controlador PID:

Coeficiente de Ganho Proporcional (kp)	4
Coeficiente de Ganho Derivativo (kd)	1
Coeficiente de Ganho Integral (ki)	7
Limite de Integração (il)	32767/12=2731

Tabela 4.7 - Parâmetros controlador PID

Durante os testes do sistema de condução autónoma, recorrendo a estes valores, o controlador apresentou sempre uma boa resposta aos diversos comandos de alteração de velocidade que lhe foram sendo enviados.

4.1.5 Sistema de Comunicação

A comunicação das diferentes placas controladores com o computador é realizada por um interface USB. Cada placa dispõe de um conversor (DLP-USB232M) USB/RS-232 para a ligação a microcontroladores. As portas associadas a cada um destes dispositivos são configuradas com os parâmetros apresentados na Tabela 4.8:

Baud-Rate; Velocidade [bit/s]	11500
Stop bit	1
Data bit	8
Even (bit de paridade)	-
Flow Control	-

Tabela 4.8 - Parâmetros de configuração das portas USB-RS232

4.1.6 Computador de Controlo

O controlo dos vários elementos constituintes do robot móvel depende de um computador portátil colocado na sua plataforma superior.

Este computador, interligado às várias placas controladores e às duas *webcams* através de interfaces USB, terá em ambiente Windows o algoritmo de controlo que se descreve de seguida a ser executado em tempo real.

Neste trabalho, o computador portátil utilizado foi um Samsung 350V5C-S04PT, contudo, a preocupação em desenvolver um algoritmo eficiente, possibilitaria a utilização de um equipamento de menor desempenho sem comprometer os resultados.

4.2 ***Algoritmos de Controlo***

As metodologias apresentadas para o desenvolvimento do sistema de condução autónoma foram implementadas computacionalmente de forma que pudessem ser testadas na plataforma móvel descrita anteriormente.

Na fase inicial deste trabalho foi utilizada a linguagem de programação Java, sobretudo devido à sua portabilidade e vasta documentação de apoio existente. Posteriormente, tornou-se percetível que seria útil recorrer à biblioteca SimpleCV para agilizar o processamento de imagem necessário. Sendo essa biblioteca destinada à linguagem Python, esta tornou-se a escolha mais acertada.

Assim, toda a implementação foi efetuada em ambiente Windows, com recurso à linguagem de programação interpretada e orientada a objetos Python, apoiada em várias bibliotecas de entre as quais se destacam as já referenciadas SimpleCV, PySerial e PyFuzzy.

Descreve-se de seguida a implementação global do sistema e abordam-se com particular detalhe alguns dos seus aspetos mais significativos.

4.2.1 Estrutura Geral

Pretende-se que o algoritmo de condução autónoma desenvolvido neste trabalho possa dotar uma plataforma robótica das capacidades necessárias ao seu deslocamento autónomo, eficaz e seguro.

Para isso, implementou-se um algoritmo que engloba 8 módulos distintos, sendo da interação entre estes que resulta o funcionamento global do sistema.

Na Figura 4.20 ilustra-se a estrutura funcional do algoritmo implementado.

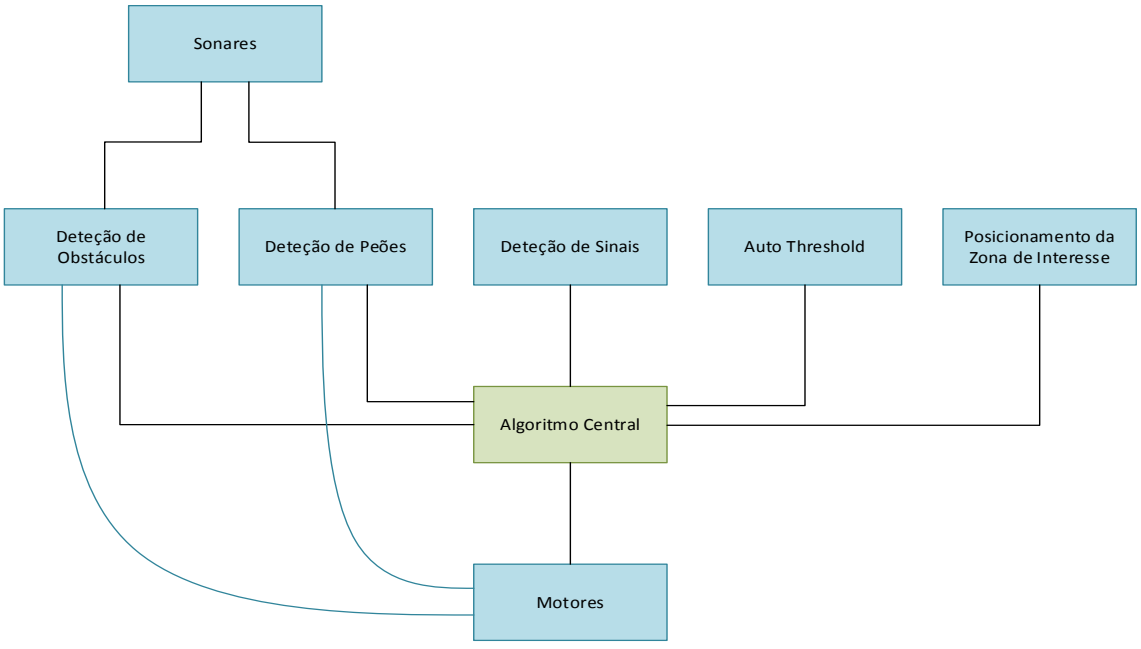


Figura 4.20 – Estrutura funcional do algoritmo implementado

Resumidamente, um ciclo de funcionamento deste algoritmo consiste nas seguintes etapas:

1º Aquisição de imagens das faixas de rodagem e ambiente envolvente.

2º Filtragem Gaussiana da imagem das faixas de rodagem.

3º Conversão para escala de cinzentos da imagem das faixas de rodagem.

4º Seleção da zona de interesse com recurso ao módulo **Posicionamento da Zona de Interesse.**

5º Binarização das zonas de interesse com recurso ao módulo **Auto *Threshold*.**

6º Deteção das faixas de rodagem nas zonas binarizada.

7º Cálculo do desvio da estrutura face ao eixo central da faixa.

8º Determinação das velocidades a aplicar às rodas com recurso ao controlador *Fuzzy*.

9º Deteção e contorno de obstáculos com recurso ao módulo **Deteção de Obstáculos.**

10º Deteção e cumprimento da sinalização com recurso aos módulos **Deteção de Sinais** e **Deteção de Peões.**

11º Envio das velocidades determinadas no ponto 8 às rodas motoras.

Detalha-se de seguida a implementação dos vários módulos utilizados neste algoritmo de controlo.

4.2.2 Motores

O módulo ***Motores*** consiste na definição de uma classe que permite instanciar individualmente os dois motores presentes no robot móvel.

É nesta classe que estão definidos os métodos que possibilitam efetuar a comunicação e comando individual das duas rodas motoras. Foi essencial para a implementação deste módulo a utilização da biblioteca adicional da linguagem Python designada por PySerial. Esta biblioteca possui funções pré-definidas essenciais à comunicação série, como é o caso em questão.

Estão definidas neste módulo diversas funções essenciais ao funcionamento dos motores. Exemplos disso são a **load_filter()** que faz o carregamento dos parâmetros do filtro PID, a **reset_motor()**, que permite parar o motor e a **start()** e **loadV()**, duas das mais utilizadas neste trabalho.

Função start():

Esta função inicia o movimento dos motores consoante os parâmetros carregados.

Abre a porta série através da função **open_serial()** e de seguida, envia o comando predefinido para start do motor, ***STT=0x01***. Após o envio do comando a porta série é fechada com recurso à função **close_serial().**

```
"""Função Start Motor"""

def start (self):

    self.open_serial()

    self.send_comand(self.STT,self.motor)

    self.close_serial()
```

Figura 4.21 - Excerto da função start()

A função responsável pelo envio do comando ao microcontrolador é a **send_comand()**. Consoante o motor a que se destina, esta função envia ao microcontrolador o byte de endereçamento adequado seguido do comando desejado, neste caso o **STT**.

Função loadV()

Esta função é a responsável pelo carregamento das velocidades desejadas para os motores.

A velocidade pretendida é convertida para o formato a enviar ao controlador e é iniciado o processo através do comando indicativo de início de carregamento de trajetória, **LTRJ=0x1F.**

```
vello=vel & 0x0000FFFF
velhi=(vel>>16) & 0x0000FFFF

#time.sleep(0.05)

self.send_comand (self.LTRJ,self.motor)
```

Figura 4.22 - Excerto da função loadV() - conversão de valores e comando de carregamento

Após o envio do comando **LTRJ,** são enviados para o controlador do motor os bytes de dados correspondentes à velocidade convertida anteriormente.

```
data_velhi[1]=velhi & 0x00FF
data_velhi[0]=(velhi >> 8) & 0x00FF

time.sleep(0.02)

self.send_data(data_velhi,self.motor)

time.sleep(0.02)

data_vello[1]=vello & 0x00FF
data_vello[0]=(vello >> 8) & 0x00FF

self.send_data(data_vello,self.motor)
```

Figura 4.23 - Excerto da função loadV() - envio de dados

4.2.3 Sonares

O módulo ***Sonares*** consiste na definição de uma classe que permite instanciar os sensores de ultrasons.

Recorrendo também à biblioteca PySerial, implementa várias funções, sendo a mais importante a **read_dist()**. Esta função é responsável por iniciar o processo de leitura dos oito sensores e posteriormente converter os valores obtidos por cada um destes em distâncias dos obstáculos relativamente a si. O processo é iniciado com o envio do caracter **"A"** à placa controladora dos sonares.

```
def read_dist (self):

    self.open_serial()

    self.serie_sonares.write("A")

    distancias=self.serie_sonares.read(18)

    self.close_serial()
```

Figura 4.24 - Excerto da função read_dist() - sequência de leitura

A conversão dos valores obtidos (tempos de voo) é efetuada com base na equação (4.1) apresentada anteriormente.

```
self.distFE=(ord(distancias[14])*255+ord(distancias[15]))*self.kdist # Distancia Sonar Frente Esquerdo
```

Figura 4.25 - Excerto da função read_dist() - conversão de valores

4.2.4 Posicionamento da Zona de Interesse

Tal como referido no subcapítulo 3.1, a definição de zonas de interesse em função da velocidade, ou seja, zonas onde se espera em determinado momento detetar as faixas de rodagem, é uma mais-valia para o desempenho do sistema implementado. O módulo responsável pelo posicionamento destas zonas, consoante a velocidade a que o robot se desloca, determina a posição da janela que as define. Isto é efetuado com recurso à função **calculo()**.

```
def calculo(velocidade):

    vel_max=9 ## Velocidade Máxima para o cálculo variável da posição
    pos_base=360 ## Valor base para a posição da zona de interesse (para baixa velocidade

    if velocidade==1: ## Verifica a velocidade e seleciona o posicionamento da zona
        pos=pos_base

    elif velocidade>vel_max: ## Valor para velocidade limite de cálculo
        pos=0

    else:
        pos=pos_base-45*(velocidade-1) ## Cálculo da Posição da Zona de Interesse com bas

    return pos
```

Figura 4.26 - Excerto da função calculo() do módulo de posicionamento da zona de interesse

4.2.5 Auto *Threshold*

Este módulo consiste essencialmente na implementação de um método automático de binarização baseado no método de Otsu [80]. Este método permite a deteção automática de um limiar ótimo para a binarização de uma imagem em escala de cinzentos.

Assim, a função **calculo()** definida neste módulo, permite que durante a execução do **Algoritmo Central**, após a conversão das imagens adquiridas para escala de cinzentos, efetuar a sua binarização de forma conveniente, minimizando o efeito das influências exteriores.

Inicialmente, a referida função prepara a imagem a ser analisada e determina o seu histograma normalizado e a sua função de distribuição acumulada.

```
def calculo (img):

    img=img.getNumpy().transpose(1, 0, 2) ## Converte o espaço de cor da imagem

    blur = cv2.GaussianBlur(img,(5,5),0) ## Aplica uma filtragem Gaussiana


    """Determina o Histograma Normalizado e a sua função de Distribuição Acumulada"""

    hist = cv2.calcHist(img,[0],None,[256],[0,256]) ## Determina o Histograma da imagem
    hist_norm = hist.ravel()/hist.max() ## Cálcula o seu histograma normalizado
    Q = hist_norm.cumsum() ## Determina a sua função de Distribuição Acumulada
```

Figura 4.27 - Excerto da função calculo() do módulo Auto Threshold - filtragem e histograma

Posteriormente, são determinados parâmetros estatísticos como as médias e variâncias para cada valor de *threshold* possível de forma a avaliar qual destes minimiza a variância intra-classes, dada pela função **fn** no seguinte código.

```
    for i in xrange(1,256):
        p1,p2 = np.hsplit(hist_norm,[i]) ## Probabilidades
        q1,q2 = Q[i],Q[255]-Q[i] ## Soma das Classes
        b1,b2 = np.hsplit(bins,[i]) ## Pesos


        """Determina Médias e Variâncias"""

        m1,m2 = np.sum(p1*b1)/q1, np.sum(p2*b2)/q2
        v1,v2 = np.sum(((b1-m1)**2)*p1)/q1,np.sum(((b2-m2)**2)*p2)/q2


        """Cálcula a função de minimização"""

        fn = v1*q1 + v2*q2
        if fn < fn_min:
            fn_min = fn
            thresh = i


    return thresh
```

Figura 4.28 - Excerto da função calculo() do módulo Auto Threshold - minimização da variância

De referir que este módulo recorre a funções da biblioteca SimpleCV para efetuar o processamento e análise dos parâmetros relativos à imagem. Exemplo disso é a determinação do seu histograma normalizado.

4.2.6 Deteção de Obstáculos

O módulo **Deteção de Obstáculos** é responsável, tal como o nome indica, por detetar e contornar os obstáculos que possam surgir durante a movimentação do robot na pista.

Este foi desenvolvido no sentido de lidar com o desvio de obstáculos imóveis, porém, alguns testes efetuados ao desvio de obstáculos em movimento, ou seja, a sua ultrapassagem, foram também relativamente bem-sucedidos.

A função **Deteccao()** deste módulo é chamada durante a execução do **Algoritmo Central** no sentido de verificar se existe algum objeto nas proximidades do robot. Caso exista, é executada uma rotina pré-definida de contorno de obstáculo definida neste módulo. Caso contrário, a execução do **Algoritmo Central** continua normalmente.

Esta função interage também com os módulos Motor e Sonares.

```
velocidade=velocidade
Sonares.read_dist()
if Sonares.distFE < 20 or Sonares.distFD < 20 :
    if tipo_faixa == "continua":
        time.sleep(5)
        Desvio_Obstaculos(velocidade)
    else:
        Desvio_Obstaculos(velocidade)
else:
    pass
return
```

Figura 4.29 - Excerto da função Deteccao() do módulo Deteção de Obstáculos - rotina de deteção

A função **Deteccao()** recebe do **Algoritmo Central** a informação se a faixa esquerda detetada é contínua ou descontinua. Se for contínua, é feita uma pausa de 5 segundos e é posteriormente iniciado o algoritmo de contorno do obstáculo através da função **Desvio_Obstaculo()**. Se for descontínua, o algoritmo é iniciado imediatamente.

A função **Desvio_Obstaculo()** consiste numa rotina de movimentações que permite transpor determinado obstáculo.
Recorre a informações provenientes dos sonares, de forma a orientar o seu movimento consoante a distância a que se encontra do obstáculo.

A parametrização desta rotina foi efetuada empiricamente por tentativa e erro.
Após a conclusão desta rotina, a execução do **Algoritmo Central** é retomada normalmente.

4.2.7 Deteção de sinais

O módulo de **Deteção de Sinais** baseia o seu funcionamento na extração de características invariantes de imagens a partir do método SIFT.

Este módulo é inicializado aquando da execução do **Algoritmo Central**, nesse momento, é chamada a função **detecao_base()** que procura segundo o referido método, as características identificativas dos sinais a serem detetados (presentes na base de dados).

```
tkp_stop = detector.detect(sinal_stop) ## Deteta as Características
tkp_stop, td_stop = descriptor.compute(sinal_stop, tkp_stop) # Determina o Discriptor

tkp_avanco = detector.detect(sinal_avanco) ## Deteta as Características
tkp_avanco, td_avanco = descriptor.compute(sinal_avanco, tkp_avanco) # Determina o Discriptor

tkp_sinal50 = detector.detect(sinal_50) # Deteta as Características
tkp_sinal50, td_sinal50 = descriptor.compute(sinal_50, tkp_sinal50) # Determina o Discriptor

tkp_sinal90 = detector.detect(sinal_90) # Deteta as Características
tkp_sinal90, td_sinal90 = descriptor.compute(sinal_90, tkp_sinal90) # Determina o Discriptor

tkp_sinalpassadeira = detector.detect(sinal_passadeira) # Deteta as Características
tkp_sinalpassadeira, td_sinalpassadeira = descriptor.compute(sinal_passadeira, tkp_sinalpassadeira)
```

Figura 4.30 - Excerto da função detecao_base() da Deteção de Sinais - extração de características

Contudo, o processo de reconhecimento de sinais apenas é iniciado quando a função **match()** é chamada durante a execução do **Algoritmo Central**.

Esta função, que recebe como entrada a imagem do ambiente em redor do robot e a sua velocidade atual, chama a função **detecao_imagem()** do módulo, que deteta e descreve as características da imagem recolhida que serão posteriormente comparadas com as detetadas nos sinais da base de dados.

```
def match(img,velocidade):

    sinal_stop=0
    sinal_avanco=0
    sinal_50=0
    sinal_90=0
    sinal_passadeira=0


    skp_imagem,sd_imagem = detecao_imagem(img)

    dist=90
    num=-1
    skp_stop, tkp = findKeyPoints(skp_imagem,sd_imagem,tkp_stop,td_stop,dist)
    skp_avanco, tkp = findKeyPoints(skp_imagem,sd_imagem,tkp_avanco,td_avanco,dist)
    skp_sinal50, tkp = findKeyPoints(skp_imagem,sd_imagem,tkp_sinal50,td_sinal50,dist)
    skp_sinal90, tkp = findKeyPoints(skp_imagem,sd_imagem,tkp_sinal90,td_sinal90,dist)
    skp_sinalpassadeira, tkp = findKeyPoints(skp_imagem,sd_imagem,tkp_sinalpassadeira,td_sinalpassadeira,dist)
```

Figura 4.31 - Excerto da função match() do módulo Deteção de Sinais - deteção de características

A função responsável pela comparação das características é a **findKeyPoints()**.
Esta função avalia a similaridade dos descritores das características identificadas através de uma estrutura de busca binária, uma **Árvore kd**.

```
def findKeyPoints(skp,sd,tkp,td, distance=200):

    flann_params = dict(algorithm=1, trees=5)
    flann = cv2.flann_Index(sd, flann_params)
    idx, dist = flann.knnSearch(td, 1, params={})
    del flann

    dist = dist[:,0]/60.0
    dist = dist.reshape(-1,).tolist()
    idx = idx.reshape(-1).tolist()
    indices = range(len(dist))
    indices.sort(key=lambda i: dist[i])
    dist = [dist[i] for i in indices]
    idx = [idx[i] for i in indices]
    skp_final = []
    for i, dis in itertools.izip(idx, dist):
        if dis < distance:
            skp_final.append(skp[i])

    flann = cv2.flann_Index(td, flann_params)
    idx, dist = flann.knnSearch(sd, 1, params={})
    del flann

    dist = dist[:,0]/60.0
    dist = dist.reshape(-1,).tolist()
    idx = idx.reshape(-1).tolist()
    indices = range(len(dist))
    indices.sort(key=lambda i: dist[i])
    dist = [dist[i] for i in indices]
    idx = [idx[i] for i in indices]
    tkp_final = []

    for i, dis in itertools.izip(idx, dist):

        if dis < distance:
            tkp_final.append(tkp[i])

    return skp_final, tkp_final
```

Figura 4.32 - Excerto da função findKeyPoints() do módulo Deteção de Sinais

Após determinar se existe ou não sinalização na imagem recolhida, essa informação é fornecida ao **Algoritmo Central**, o que irá influenciar o comportamento posterior do robot.
No caso de ser detetado o sinal indicativo de uma passagem para peões, uma passadeira, o **Algoritmo Central** chamará um módulo específico para lidar com essa situação, o módulo **Deteção de Peões**.

De referir que os métodos de extração de características foram implementados com recurso à biblioteca SimpleCV.

4.2.8 Deteção de Peões

Como referido anteriormente, quando é detetada sinalização de uma passadeira, recorre-se a este módulo de forma a aguardar a travessia do peão.

Ao ser chamada a função **Espera()** pelo **Algoritmo Central**, inicia-se uma rotina em que é verificado através dos sensores de ultrasons se existe algum peão pronto a iniciar o atravessamento. Caso exista, o robot fica imóvel até á sua passagem. Se nenhum peão for detetado, a rotina termina automaticamente, sendo o **Algoritmo Central** imediatamente retomado.

```
def Espera(velo_atual):

    time.sleep(5) ## Tempo de espera entre a deteção do sinal e a chegada à passadeira

    Sonares.read_dist() ## Lê os sonares


    while Sonares.distFD < 20 or Sonares.distFE < 20 or Sonares.distLFD < 20 Sonares.distLFE < 20:

            Motordireito.loadV(0) ## Para motor direito
            Motordireito.start()
            Motoresquerdo.loadV(0) ## Para motor esquerdo
            Motoresquerdo.start()
            time.sleep(3) ## Tempo de espera até nova leitura dos sinais
            Sonares.read_dist()


    Motordireito.loadV(velo_atual) ## Após a passagem do peão carrega a velocidade anterior do motor
    Motordireito.start()
    Motoresquerdo.loadV(velo_atual) ## Após a passagem do peão carrega a velocidade anterior do motor
    Motoresquerdo.start()


    return
```

Figura 4.33 - Excerto da função Espera() do módulo Deteção de Peões

Este algoritmo recorre também aos módulos **Motores** e **Sonares**.

4.2.9 Algoritmo Central

O **Algoritmo Central** é a base de todo o sistema de condução autónoma desenvolvido. É responsável pela interligação entre os vários módulos, sendo também o garante do seu adequado funcionamento.

É este algoritmo que implementa o mecanismo base de controlo da movimentação do robot. Após a recolha e processamento das imagens adquiridas pelas *webcams*, identifica o desvio do robot face à trajetória a seguir e recorre a um controlador *Fuzzy* de forma a determinar as alterações a efetuar na velocidade das rodas motoras para que o desvio possa ser anulado. Durante a sua execução, este algoritmo recorre também aos módulos de **Deteção de Obstáculos** e **Deteção de Sinalização**. Sempre que alguma destas situações se verifica, a sua execução é interrompida, ficando nesse momento o controlo do robot dependente diretamente desses módulos.

Este algoritmo é iniciado com o carregamento dos módulos desenvolvidos e das bibliotecas utilizadas. Posteriormente é efetuado também o carregamento das configurações do controlador *Fuzzy* a utilizar. Isto é conseguido com recurso a uma função da biblioteca PyFuzzy que adquire todas as parametrizações necessárias a partir de um ficheiro escrito previamente em *Fuzzy Control Language (FCL)* [58].

Este ficheiro é escrito segundo uma sintaxe própria e caracteriza-se por apresentar de forma estruturada, todas as definições necessárias à implementação de um controlador *Fuzzy*.

```
system = fuzzy.storage.fcl.Reader.Reader().load_from_file("Controlador_Fuzzy.fcl")
```

Figura 4.34 - Excerto do Algoritmo Central - carregamento do ficheiro em FCL

Ficheiro – Fuzzy Control Language (FCL)

Este ficheiro consiste essencialmente em quatro blocos, que correspondem basicamente às quatro etapas do funcionamento de um controlador *Fuzzy*.

Definição de Variáveis

Neste bloco são definidas as variáveis de entrada e saída do controlador bem como os seus limites.

```
VAR_INPUT
    Desvio :     REAL; (* RANGE(-250 .. 250) *)
    Var_Velocidade:    REAL; (* RANGE(-1 .. 1) *)
END_VAR

VAR_OUTPUT
    Motor_Esquerdo : REAL; (* RANGE(-10 .. 10) *)
    Motor_Direito : REAL; (* RANGE(-10 .. 10) *)
END_VAR
```

Figura 4.35 - Excerto do ficheiro de configuração em FCL - definição de variáveis

Antes de mais, é de referir que neste trabalho não se recorreu à determinação de distâncias reais através das imagens. Assim, os valores apresentados correspondem a distâncias de *pixéis*.

Para a variável de entrada **Desvio**, que corresponde à distância entre o eixo da faixa de rodagem e o eixo central do robot, foi determinado que a gama de valores a utilizar seria [-250,250]. Estes valores devem-se à largura da faixa de rodagem captada pela *webcam*, com o ângulo definido (45º), a uma resolução de 640x480 *pixéis*, corresponder aproximadamente a 500 *pixéis*, como se verifica na Figura 4.36.

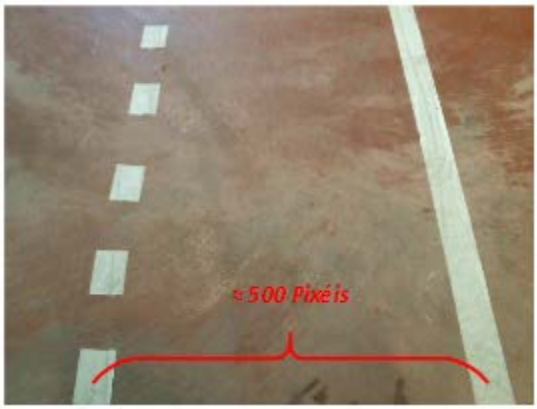


Figura 4.36 - Largura da faixa de rodagem à resolução de 640x480 pixéis

Relativamente à variável ***Var_Velocidade***, esta apenas fornecerá o sentido de variação, bastando para isso a indicção se o valor é positivo ou negativo. Daí, a gama utilizada ser [-1,1].

As variáveis de saída, que correspondem às velocidades a aplicar aos motores, apresentam uma gama de velocidades compatível com o limite máximo de velocidade selecionável pelo utilizador, 10 cm/s.

***Fuzzificação* das Entradas**

Nesta etapa é parametrizada a conversão das entradas numéricas em termos *Fuzzy*, com um certo grau de pertinência.

Para isso, é necessário definir previamente as funções de pertinência para cada um dos termos considerados.

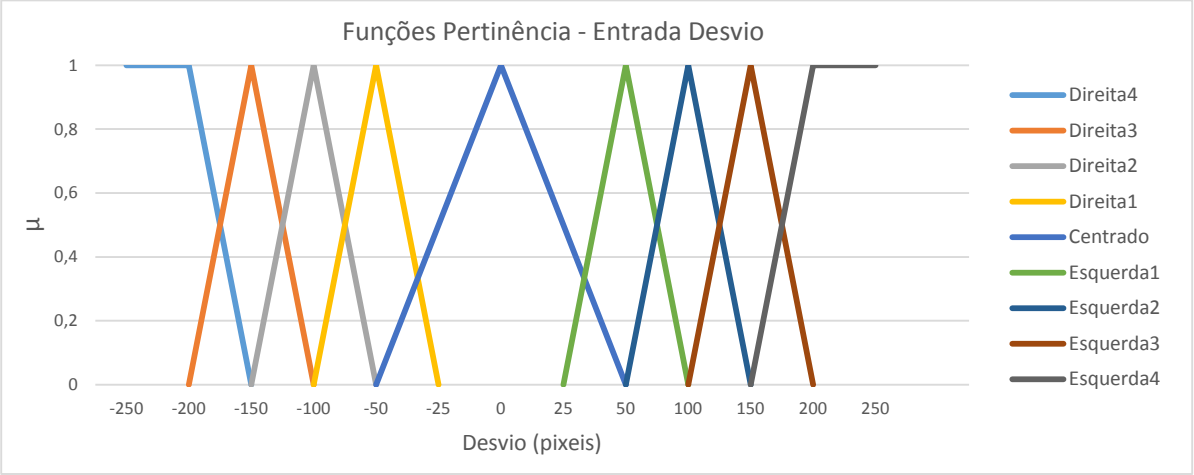


Figura 4.37 - Funções Pertinência - entrada Desvio

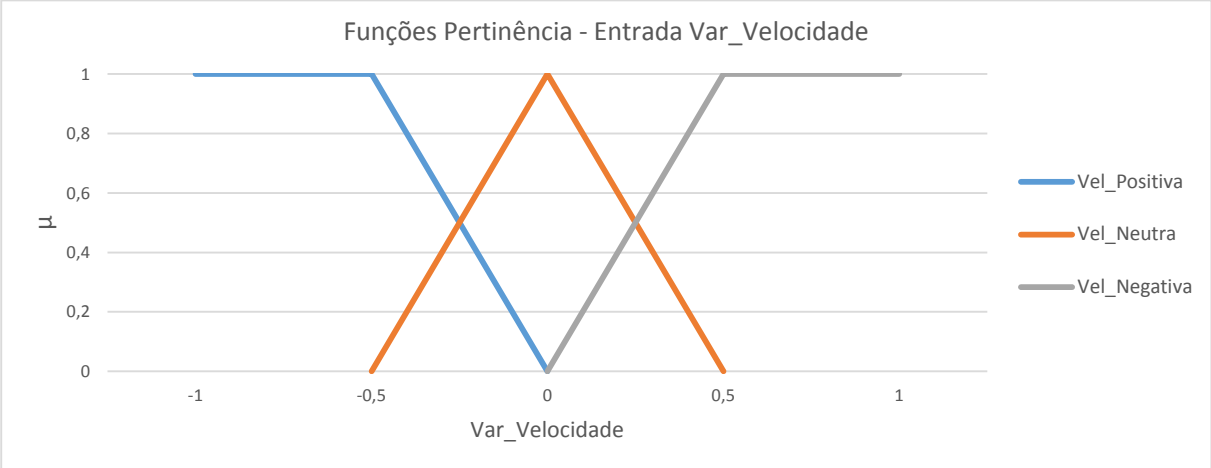


Figura 4.38 - Funções Pertinência - entrada Var_Velocidade

A definição destas funções baseou-se nas diretrizes presentes no trabalho de Hill, Horstkotte & Teichrow [28] e na resposta do sistema aos vários testes efetuados. Estas funções privilegiam uma boa estabilidade quando em equilíbrio. A sua parametrização no ficheiro FCL consiste apenas em indicar os seus pontos representativos.

```
FUZZIFY Desvio
    TERM Direita4 := (-250, 1) (-200, 1) (-150, 0) ;
    TERM Direita3 := (-200, 0) (-150, 1) (-100, 0) ;
    TERM Direita2 := (-150, 0) (-100, 1) (-50, 0) ;
    TERM Direita1 := (-100, 0) (-50, 1) (-25, 0) ;
    TERM Centrado := (-50, 0) (0, 1) (50, 0) ;
    TERM Esquerda1 := (25, 0) (50, 1) (100, 0) ;
    TERM Esquerda2 := (50, 0) (100, 1) (150, 1) ;
    TERM Esquerda3 := (100, 0) (150, 1) (200, 0) ;
    TERM Esquerda4 := (150, 0) (200, 1) (250, 1) ;
END_FUZZIFY

FUZZIFY Var_Velocidade
    TERM vel_positiva := (-1, 1) (0.5, 1) (0, 0) ;
    TERM vel_neutra := (-0.5, 0) (0, 1) (0.5, 0) ;
    TERM vel_negativa := (0, 0) (0.5, 1) (1, 1) ;
END_FUZZIFY
```

Figura 4.39 - Excerto do ficheiro de configuração em FCL - Fuzzificação das entradas

Base de Regras

Esta é a etapa fundamental da parametrização do controlador, sendo aqui estabelecida a sua capacidade de decisão. Representativa de conhecimento empírico acerca do funcionamento e reação do robot às várias situações a que pode ser sujeito, consiste num conjunto de regras na forma "Se… Então…" que avalia as entradas e com base nestas determina as saídas adequadas.

```
RULEBLOCK first
    AND:MIN;
    (*ACCU:MAX;*)
    RULE 0: IF (Desvio IS Esquerda4) THEN (Motor_Esquerdo IS Virar_Direita4);
    RULE 1: IF (Desvio IS Esquerda3) AND (Var_Velocidade IS vel_positiva) THEN (Motor_Esquerdo IS Virar_Direita3);
    RULE 2: IF (Desvio IS Esquerda3) AND (Var_Velocidade IS vel_negativa) THEN (Motor_Esquerdo IS Virar_Direita4);
    RULE 3: IF (Desvio IS Esquerda2) AND (Var_Velocidade IS vel_positiva) THEN (Motor_Esquerdo IS Virar_Direita2);
    RULE 4: IF (Desvio IS Esquerda2) AND (Var_Velocidade IS vel_negativa) THEN (Motor_Esquerdo IS Virar_Direita3);
    RULE 5: IF (Desvio IS Esquerda1) AND (Var_Velocidade IS vel_positiva) THEN (Motor_Esquerdo IS Virar_Direita1);
    RULE 6: IF (Desvio IS Esquerda1) AND (Var_Velocidade IS vel_negativa) THEN (Motor_Esquerdo IS Virar_Direita2);
    RULE 7: IF (Desvio IS Centrado) THEN (Motor_Esquerdo IS Manter_Posicao);
    RULE 8: IF (Desvio IS Centrado) THEN (Motor_Direito IS Manter_Posicao);
    RULE 9: IF (Desvio IS Esquerda4) THEN (Motor_Direito IS Virar_Direita4);
    RULE 10: IF (Desvio IS Esquerda3) AND (Var_Velocidade IS vel_positiva) THEN (Motor_Direita IS Virar_Direita3);
    RULE 11: IF (Desvio IS Esquerda3) AND (Var_Velocidade IS vel_negativa) THEN (Motor_Direita IS Virar_Direita4);
    RULE 12: IF (Desvio IS Esquerda2) AND (Var_Velocidade IS vel_positiva) THEN (Motor_Direita IS Virar_Direita2);
    RULE 13: IF (Desvio IS Esquerda2) AND (Var_Velocidade IS vel_negativa) THEN (Motor_Direita IS Virar_Direita3);
    RULE 14: IF (Desvio IS Esquerda1) AND (Var_Velocidade IS vel_positiva) THEN (Motor_Direita IS Virar_Direita1);
    RULE 15: IF (Desvio IS Esquerda1) AND (Var_Velocidade IS vel_negativa) THEN (Motor_Direita IS Virar_Direita2);
    RULE 16: IF (Desvio IS Direita4) THEN (Motor_Esquerdo IS Virar_Esquerda4);
    RULE 17: IF (Desvio IS Direita3) AND (Var_Velocidade IS vel_positiva) THEN (Motor_Esquerdo IS Virar_Esquerda4);
    RULE 18: IF (Desvio IS Direita3) AND (Var_Velocidade IS vel_negativa) THEN (Motor_Esquerdo IS Virar_Esquerda3);
    RULE 19: IF (Desvio IS Direita2) AND (Var_Velocidade IS vel_positiva) THEN (Motor_Esquerdo IS Virar_Esquerda3);
    RULE 20: IF (Desvio IS Direita2) AND (Var_Velocidade IS vel_negativa) THEN (Motor_Esquerdo IS Virar_Esquerda2);
    RULE 21: IF (Desvio IS Direita1) AND (Var_Velocidade IS vel_positiva) THEN (Motor_Esquerdo IS Virar_Esquerda2);
    RULE 22: IF (Desvio IS Direita1) AND (Var_Velocidade IS vel_negativa) THEN (Motor_Esquerdo IS Virar_Esquerda1);
    RULE 23: IF (Desvio IS Direita4) THEN (Motor_Direito IS Virar_Esquerda4);
    RULE 24: IF (Desvio IS Direita3) AND (Var_Velocidade IS vel_positiva) THEN (Motor_Direito IS Virar_Esquerda4);
    RULE 25: IF (Desvio IS Direita3) AND (Var_Velocidade IS vel_negativa) THEN (Motor_Direito IS Virar_Esquerda3);
    RULE 26: IF (Desvio IS Direita2) AND (Var_Velocidade IS vel_positiva) THEN (Motor_Direito IS Virar_Esquerda3);
    RULE 27: IF (Desvio IS Direita2) AND (Var_Velocidade IS vel_negativa) THEN (Motor_Direito IS Virar_Esquerda2);
    RULE 28: IF (Desvio IS Direita1) AND (Var_Velocidade IS vel_positiva) THEN (Motor_Direito IS Virar_Esquerda2);
    RULE 29: IF (Desvio IS Direita1) AND (Var_Velocidade IS vel_negativa) THEN (Motor_Direito IS Virar_Esquerda1);
END_RULEBLOCK
```

Figura 4.40 - Excerto do ficheiro de configuração em FCL - Base de Regras

Nesta são também definidas as formas de agregação e acumulação necessárias ao sistema de inferência.

Defuzzificação das Saídas

Esta etapa é responsável pela ponderação do peso das diversas respostas resultantes da avaliação das regras. Consoante estas, atribui um valor numérico às saídas que possa ser utilizado externamente ao controlador.

Para isso, são definidas nesta etapa as funções de pertinência das saídas (tal como efetuado para a entrada), o modo de acumulação e o método de cálculo utilizado.

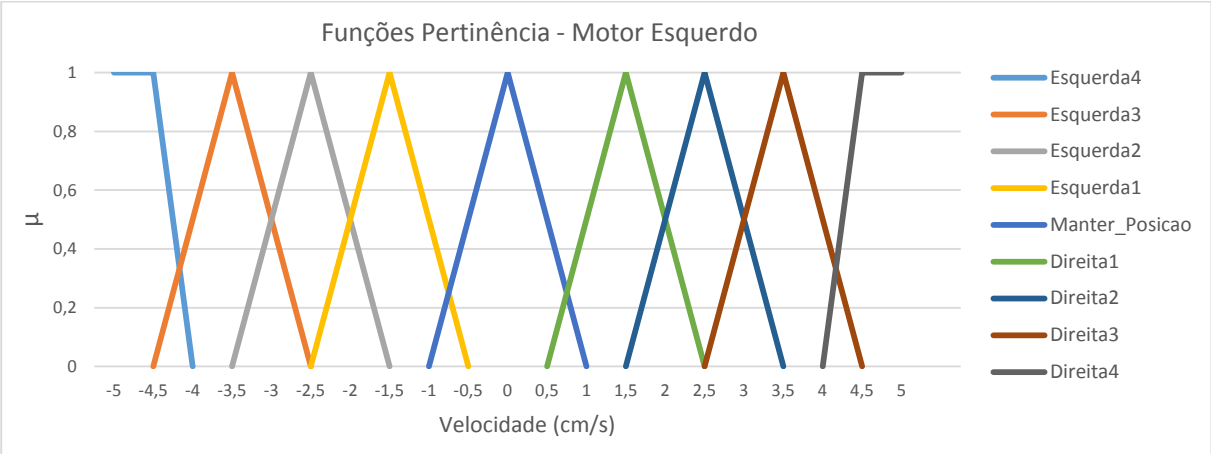


Figura 4.41 - Função Pertinência Motor Esquerdo

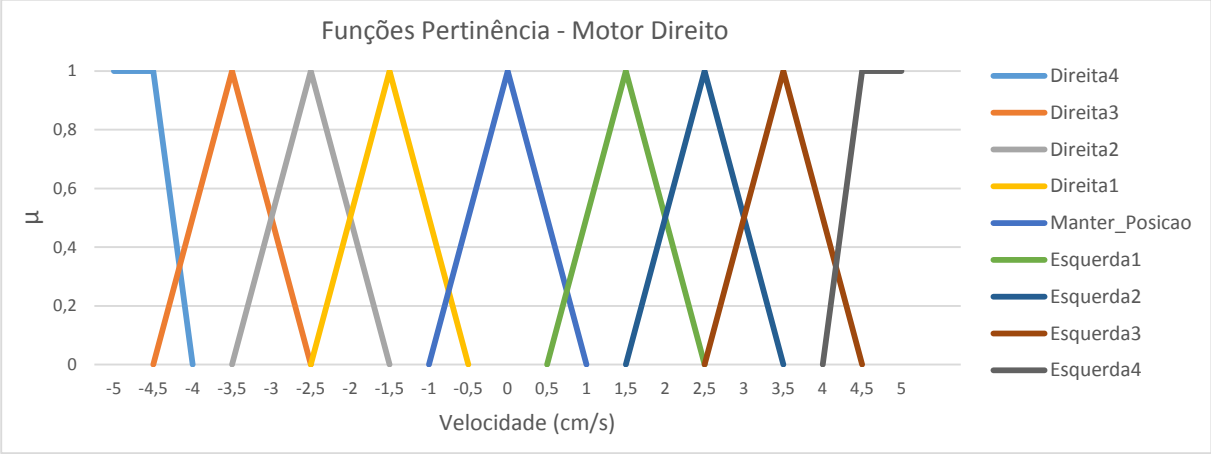


Figura 4.42 - Função Pertinência Motor Direito

Tal como no caso das entradas, a parametrização destas funções no ficheiro FCL consiste apenas em indicar os pontos representativos das funções apresentadas.

```
DEFUZZIFY Motor_Esquerdo
    TERM Virar_Esquerda4 := (-5, 1) (-4.5, 1) (-4, 0) ;
    TERM Virar_Esquerda3 := (-4.5, 0) (-3.5, 1) (-2.5, 0) ;
    TERM Virar_Esquerda2 := (-3.5, 0) (-2.5, 1) (-1.5, 0) ;
    TERM Virar_Esquerda1 := (-2.5, 0) (-1.5, 1) (-0.5, 0) ;
    TERM Manter_Posicao := (-1, 0) (0, 1) (1, 0) ;
    TERM Virar_Direita1 := (0.5, 0) (1.5, 1) (2.5, 0) ;
    TERM Virar_Direita2 := (1.5, 0) (2.5, 1) (3.5, 0) ;
    TERM Virar_Direita3 := (2.5, 0) (3.5, 1) (4.5, 0) ;
    TERM Virar_Direita4 := (4.0, 0) (4.5, 1) (5, 1) ;
    ACCU:MAX;
    METHOD: COG;(*MoM;*)
    DEFAULT := 0;
END_DEFUZZIFY

 DEFUZZIFY Motor_Direito
    TERM Virar_Direita4 := (-5, 1) (-4.5, 1) (-4, 0) ;
    TERM Virar_Direita3 := (-4.5, 0) (-3.5, 1) (-2.5, 0) ;
    TERM Virar_Direita2 := (-3.5, 0) (-2.5, 1) (-1.5, 0) ;
    TERM Virar_Direita1 := (-2.5, 0) (-1.5, 1) (-0.5, 0) ;
    TERM Manter_Posicao := (-1, 0) (0, 1) (1, 0) ;
    TERM Virar_Esquerda1 := (0.5, 0) (1.5, 1) (2.5, 0) ;
    TERM Virar_Esquerda2 := (1.5, 0) (2.5, 1) (3.5, 0) ;
    TERM Virar_Esquerda3 := (2.5, 0) (3.5, 1) (4.5, 0) ;
    TERM Virar_Esquerda4 := (4.0, 0) (4.5, 1) (5, 1) ;
    ACCU:MAX;
    METHOD: COG;(*MoM;*)
    DEFAULT := 0;
END_DEFUZZIFY
```

Figura 4.43 - Excerto do ficheiro de configuração em FCL - Defuzzificação das saídas

Neste caso, o modo de acumulação selecionado foi o **Máximo** e o método de cálculo escolhido foi o **COG – Centro de Gravidade**, ambos descritos com mais detalhe no capítulo anterior. Para o tipo de controlador desenvolvido, são os que apresentam melhores resultados.

Concluído o carregamento do ficheiro de parametrização do controlador, procede-se à inicialização de variáveis, motores e dispositivos de captação de imagem, dando-se assim inicio à rotina principal do algoritmo. Esta consiste numa sequência de análise de imagens e comando de motores que apenas é interrompida quando é detetada sinalização ou um obstáculo.

Após a captação de imagens das duas câmaras é chamada a função **calculo()** do módulo de **Posicionamento da Zona de Interesse** de forma a determinar a posição das zonas da imagem a considerar consoante a velocidade.

```
"""Determinação das Zona de Interesse com base na velocidade"""

pos=Posicao_Referencia.calculo(velocidade) ## Cálculo com recurso ao módulo destinado à função
```

Figura 4.44 - Excerto do Algoritmo Central - determinação da zona de interesse

Seguidamente, é efetuada a filtragem Gaussiana e a divisão da imagem nas zonas de interesse através das funções **smooth()** e **crop()** da biblioteca SimpleCV.

```
img_Faixas=img_Faixas_original.smooth(algorithm_name='gaussian', aperature='', sigma=1, spatial_sigma=3, grayscale=False)

img_Faixas_gray=img_Faixas.grayscale() ## Conversão para Escala de Cinzentos

img_FaixaEsquerda = img_Faixas_gray.crop(0,pos,320,80) ## Extração Zona de Interesse da Faixa Esquerda

img_FaixaDireita = img_Faixas_gray.crop(320,pos,320,80) ## Extração Zona de Interesse da Faixa Direita
```

Figura 4.45 - Excerto do Algoritmo Central - filtragem e divisão em zonas de interesse

Sendo também nesse momento chamada a função **calculo()** do módulo **Auto *Threshold*** para determinar para cada zona de interesse definida anteriormente o seu t*hreshold* otimizado:

```
threshold_esquerda=AT.calculo(img_FaixaEsquerda) ## Determinação Threshold Faixa Esquerda

threshold_direita=AT.calculo(img_FaixaDireita) ## Determinação Threshold Faixa Direita

print "threshold esquerda",threshold_esquerda
print "threshold direita",threshold_direita
```

Figura 4.46 - Excerto do Algoritmo Central - Auto Threshold

Com base nos valores obtidos, é efetuada de seguida a binarização das imagens através da função **binarize()**. Posteriormente, recorre-se à função **findBlobs()** que permite a deteção de contornos de formas (neste caso da faixa de rodagem) presentes nas zonas de interesse definidas. Após esta deteção, podem ser extraídas as coordenadas do ponto central das formas identificadas a partir da função **coordinates()**. As três funções referidas pertencem à biblioteca SimpleCV e implementam a metodologia apresentada no capítulo anterior.

```
img_FaixaEsquerdaTotal=img_Faixas_gray.crop(0,0,320,480)

thresholdtotal_esquerda=AT.calculo(img_FaixaEsquerdaTotal)

img_FaixaEsquerdaTotal_bin=img_FaixaEsquerdaTotal.binarize(thresholdtotal_esquerda).invert()

img_FaixaEsquerdaTotal_Blobs=img_FaixaEsquerdaTotal_bin.findBlobs()

if len (img_FaixaEsquerdaTotal_Blobs) > 2:

      tipo_faixa="tracejada"
else:
      tipo_faixa="continua"


"""Análise Faixa Esquerda"""

img_FaixaEsquerda_bin=img_FaixaEsquerda.binarize(threshold_esquerda).invert()
img_FaixaEsquerda_Blobs=img_FaixaEsquerda_bin.findBlobs()


"""Análise Faixa Direita"""

img_FaixaDireita_bin=img_FaixaDireita.binarize(threshold_direita).invert()
img_FaixaDireita_Blobs=img_FaixaDireita_bin.findBlobs()
```

Figura 4.47 - Excerto do Algoritmo Central - binarização, contornos e coordenadas

Tendo conhecimento das coordenadas do eixo central da faixa de rodagem, torna-se possível calcular a posição objetivo (centro da faixa de rodagem) e consequentemente, o desvio da posição real do robot (o seu eixo central) em relação a esta.
Este cálculo pode ser efetuado na ausência de uma das faixas de rodagem. As faixas de rodagem detetadas são representados no ecrã com recurso à função **dl().circle()** do SimpleCV.

```
"""Calculo Objectivo na Ausencia da Faixa Esquerda"""

if Faixa_Esquerda_Presente==0 and Faixa_Direita_Presente==1 :
    centro_FaixaDireita=centro_FaixaDireita+320 ## Coordenadas do Centro da Faixa Direita
    centro_FaixaEsquerda=centro_FaixaDireita-Largura_Faixa ## Coordenadas do Centro da Faixa
    desvioesq=320-centro_FaixaEsquerda ## Cálculo Desvio Esquerda
    desviodir=centro_FaixaDireita-320 ## Cálculo Desvio Direita
    objectivo=desviodir-desvioesq ## Cálculo Objetivo
    img_Faixas_gray.dl().circle((centro_FaixaDireita,450),10, Color.GREEN, filled=True)
    img_Faixas_gray.dl().circle((centro_FaixaEsquerda,450),10, Color.GREEN, filled=True)



    """Calculo Objectivo na Ausencia da Faixa Direita"""

elif Faixa_Esquerda_Presente==1 and Faixa_Direita_Presente==0:
    centro_FaixaDireita=centro_FaixaEsquerda+Largura_Faixa ## Coordenadas do Centro da Faixa
    desvioesq=320-centro_FaixaEsquerda ## Cálculo Desvio Esquerda
    desviodir=centro_FaixaDireita-320 ## Cálculo Desvio Direita
    objectivo=desviodir-desvioesq ## Cálculo Objetivo
    img_Faixas_gray.dl().circle((centro_FaixaDireita,450),10, Color.GREEN, filled=True)
    img_Faixas_gray.dl().circle((centro_FaixaEsquerda,450),10, Color.GREEN, filled=True)



    """Calculo Objectivo na Presenca das Duas Faixas"""

elif Faixa_Esquerda_Presente==1 and Faixa_Direita_Presente==1:
      desvioesq=320-centro_FaixaEsquerda ## Cálculo Desvio Esquerda
      desviodir=centro_FaixaDireita-320+320 ## Cálculo Desvio Direita
      objectivo=desviodir-desvioesq ## Cálculo Objetivo
      img_Faixas_gray.dl().circle((centro_FaixaDireita+320,450),10, Color.GREEN, filled=True)
      img_Faixas_gray.dl().circle((centro_FaixaEsquerda,450),10, Color.GREEN, filled=True)
      Largura_Faixa=centro_FaixaEsquerda+320-centro_FaixaDireita ## Atualização da Largura
```

Figura 4.48 - Excerto do Algoritmo Central - determinação do desvio face ao objetivo

Antes do envio desta informação ao controlador *Fuzzy*, é verificada a existência de algum tipo de obstáculo ou sinalização nas imediações do robot. Recorre-se para isso às funções **Deteccao()** dos módulos **Deteção de Obstáculos** e **Deteção de Sinalização** respetivamente.

```
"""Deteção e contorno de Obstáculos"""

Deteccao_Obstaculos.Deteccao(tipo_faixa,velo_atual)


"""Deteção e Reconhecimento de Sinais"""

velocidade_n=velo_atual

lim_movimento,lim_velocidade=Deteccao_Sinais.match(img_sinais,velocidade_n)
```

Figura 4.49 - Excerto do Algoritmo Central - deteção de obstáculos e sinalização

Se for detetado algum tipo de obstáculo ou sinalização, são executados os algoritmos especificos dos respetivos módulos. Caso isto não se verifique, é imediatamente fornecida ao controlador *Fuzzy* a informação relativa ao desvio calculado e à sua taxa de variação.

Após isso, a saída do controlador é determinada com base nas entradas fornecidas e na parametrização definida no ficheiro FCL descrito anteriormente.
A função **calculate()** utilizada pertence à biblioteca PyFuzzy.

```
system.calculate(my_input, my_output) ## Determina a saida do controlador com base nas entradas fornecidas

Velo_Motor_Esquerdo=my_output["Motor_Esquerdo"] ## Define as variáveis de saida do controlador
Velo_Motor_Direito=my_output["Motor_Direito"] ## Define as variáveis de saida do controlador
```

Figura 4.50 - Excerto do Algoritmo Central - controlador Fuzzy

Por fim, estando na posse das velocidades adequadas para cada roda motora (saídas do controlador) e conhecendo a sinalização existente, que condiciona as ações a efetuar, enviam-se aos motores as velocidades a considerar. O envio dos comandos é conseguido com recurso ao módulo **Motores**.

```
if lim_velocidade >= velocidade: ## Se a velocidade atual for inferior ao limite

      Motordireito.loadV(Velo_Motor_Direito+velocidade)
      Motordireito.start()
      Motoresquerdo.loadV(Velo_Motor_Esquerdo+velocidade)
      Motoresquerdo.start()
      velo_atual=velocidade

else: ## Se a velocidade atual for superior ao limite

      Motordireito.loadV(Velo_Motor_Direito+lim_velocidade)
      Motordireito.start()
      Motoresquerdo.loadV(Velo_Motor_Esquerdo+lim_velocidade)
      Motoresquerdo.start()
      velo_atual=lim_velocidade
```

Figura 4.51 - Excerto do Algoritmo Central - comando dos motores

Este algoritmo funciona ininterruptamente, permitindo que o robot se mantenha em movimento, num percurso ou pista fechada, evitando obstáculos e cumprindo a sinalização existente.

5. Resultados

No sentido de avaliar o desempenho do algoritmo de condução autónoma implementado, foi construída uma pista onde este poderia ser testado no controlo do robot móvel descrito anteriormente.

Apresenta-se de seguida a pista construída e os resultados obtidos nos testes efetuados aos principais elementos deste sistema e ao seu funcionamento global. De referir que para cada aspeto em análise foram efetuadas 15 voltas de teste à pista.

5.1 ***Pista de Testes***

A pista de testes construída visa proporcionar um ambiente de condução semelhante ao experienciado numa via real de tráfego. Para isso, foi criada uma estrutura que incluía zonas de curva, linhas contínuas e descontínuas, sinalização vertical, passadeiras e obstáculos. É a capacidade de lidar com todas estas situações que ditam a validade dos algoritmos propostos.

Figura 5.1 - Vista geral da pista

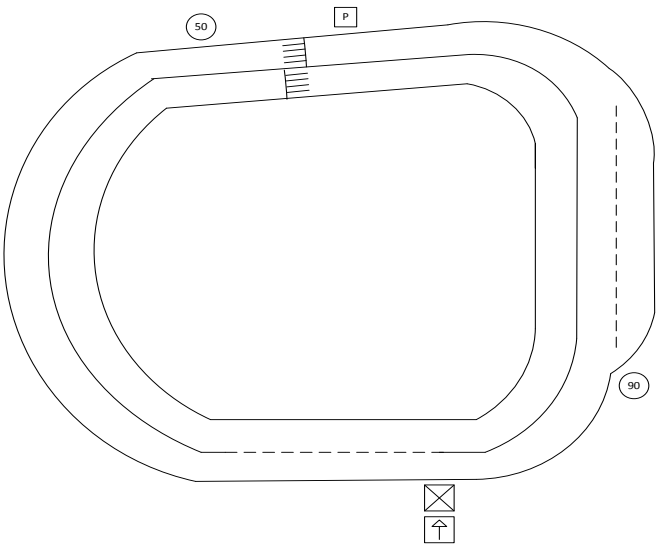


Figura 5.2 - Representação esquemática da pista

De referir que apesar de se pretender tornar esta experiencia o mais real possível, dois dos sinais utilizados (avanço e paragem) não são encontrados em vias públicas.

O uso destes sinais deveu-se ao facto do desenvolvimento deste sistema visar também uma possível participação nas provas de condução autónoma do Festival Nacional de Robótica. Assim, a sinalização vertical presente na pista é a vulgarmente utilizada neste tipo de provas.

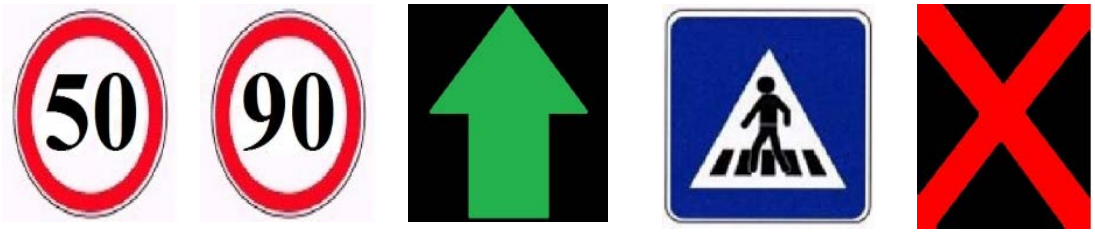


Figura 5.3 - Sinalização presente na pista

5.2 ***Processamento de imagem***

O processamento de imagem, que culmina na deteção das faixas de rodagem e consequente determinação do desvio do robot face ao objetivo, é uma etapa fundamental no funcionamento deste sistema, o que justifica a sua avaliação com particular atenção.

Para isso, efetuaram-se testes funcionais às várias fases desta etapa, cujos principais resultados se resumem de seguida.

5.2.1 Posicionamento das zonas de Interesse

A utilidade do algoritmo responsável pela determinação da posição da zona de interesse em função da velocidade foi avaliada pela comparação da capacidade do robot em concluir o percurso convenientemente, com e sem recurso a este.

Conclusão do Percurso com Sucesso (%)				
	2 cm/s	***4 cm/s***	***6 cm/s***	***8 cm/s***
Sem Algoritmo de Posicionamento	*87 %*	*73 %*	*60 %*	*53 %*
Com algoritmo de Posicionamento	*93 %*	*87 %*	*80 %*	*73 %*

Tabela 5.1 - Conclusão do percurso com sucesso

Como é percetível na tabela anterior, para baixas velocidades, a influência do algoritmo de determinação do posicionamento da zona de interesse não é muito significativa. Porém, com o aumento da velocidade de teste, verifica-se que a sua inclusão representa uma maior capacidade do robot em completar com sucesso a pista construída.

É de referir que a mais-valia deste algoritmo se verifica sobretudo em zonas de curvas ou de aproximação a estas.

Ilustra-se na Figura 5.4 a influência da velocidade na zona onde é efetuada a deteção da faixa de rodagem. Os pontos a verde indicam o local da deteção.

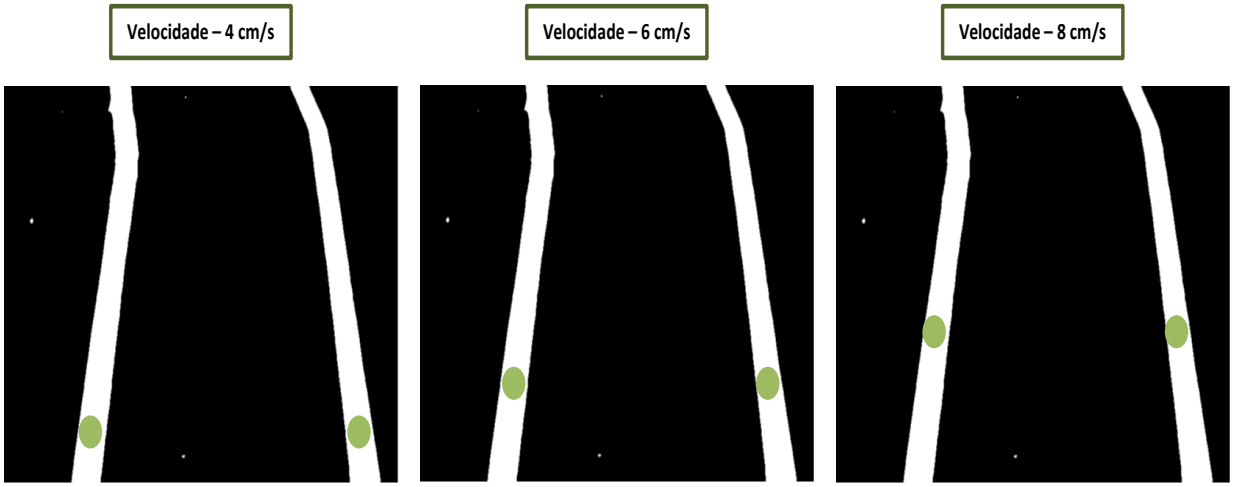


Figura 5.4 - Posicionamento das zonas de interesse consoante a velocidade

Confirma-se que o aumento da velocidade de teste implica uma zona de interesse mais afastada da base da imagem captada, o que significa também mais afastada do robot.

5.2.2 Determinação Automática de Limiar para Binarização

A determinação de um limiar ótimo para a binarização é essencial para que este processo não degrade informação pertinente para a extração de contornos e consequente deteção das faixas de rodagem.

Para esta tarefa, foi desenvolvido um algoritmo automático de cálculo, sustentado no método de Otsu [80] , que torna este processo praticamente imune a variações nas condições de captação da imagem.

Apresentam-se na Figura 5.5, três conjuntos de imagens obtidas com o algoritmo implementado. É possível verificar que apesar das condições da imagem original serem bastante diferentes, o *threshold* determinado permite que o resultado da binarização seja bastante bom e semelhante entre os três.

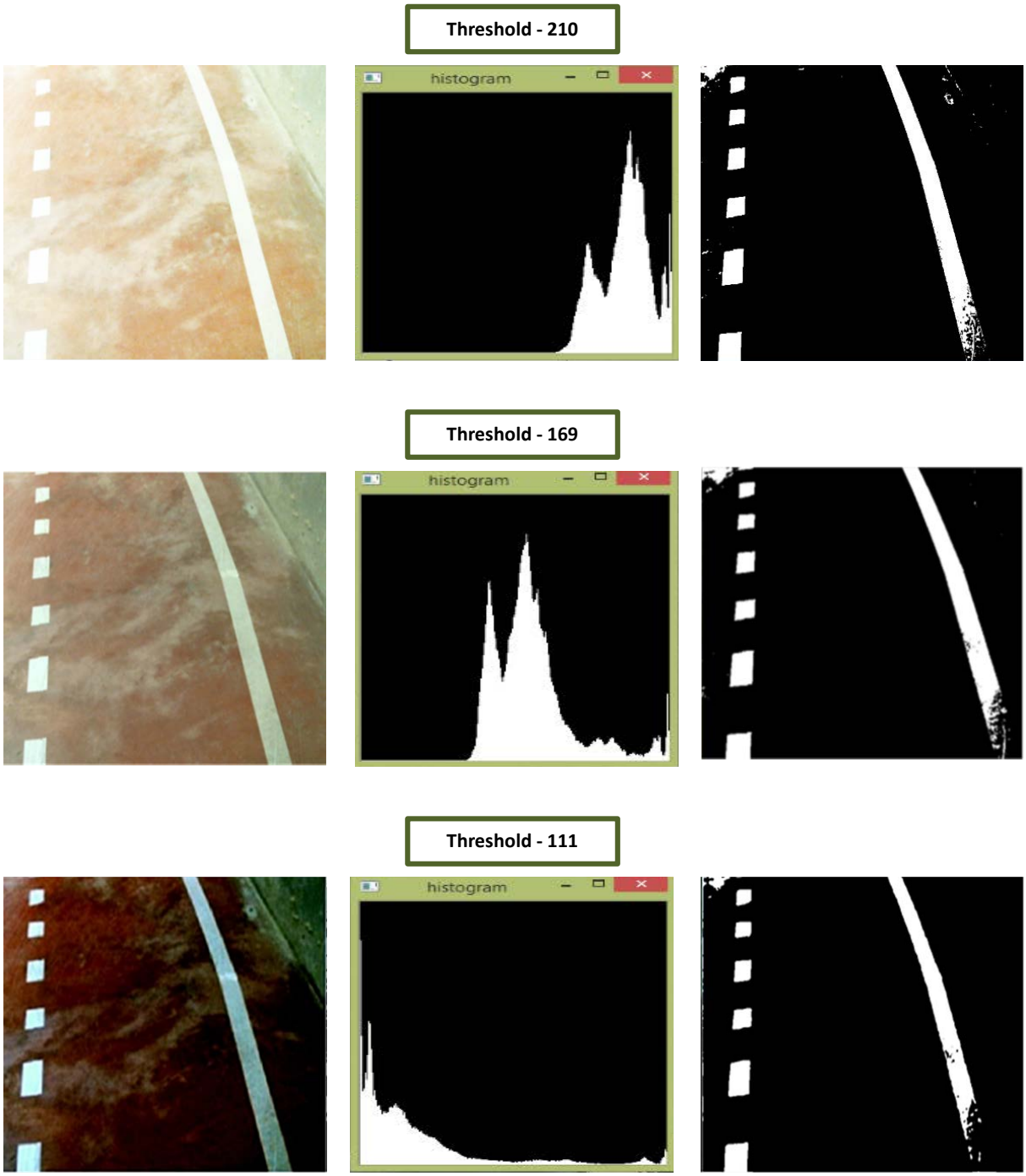


Figura 5.5 - Testes binarização com determinação automática de threshold

Como se verifica nos três exemplos, as faixas de rodagem foram salientadas em relação à restante imagem. Isto torna o seu processo de deteção bastante eficaz.

5.2.3 Análise Global do Processamento

O processamento de imagem culmina na deteção das faixas de rodagem. Todas as etapas anteriores visam proporcionar as condições ideais para o algoritmo de deteção extrair as informações necessárias ao controlo do robot.

Na Figura 5.6 apresentam-se os resultados de três testes de deteção das faixas de rodagem, bem como das várias etapas do processamento que a possibilitam.

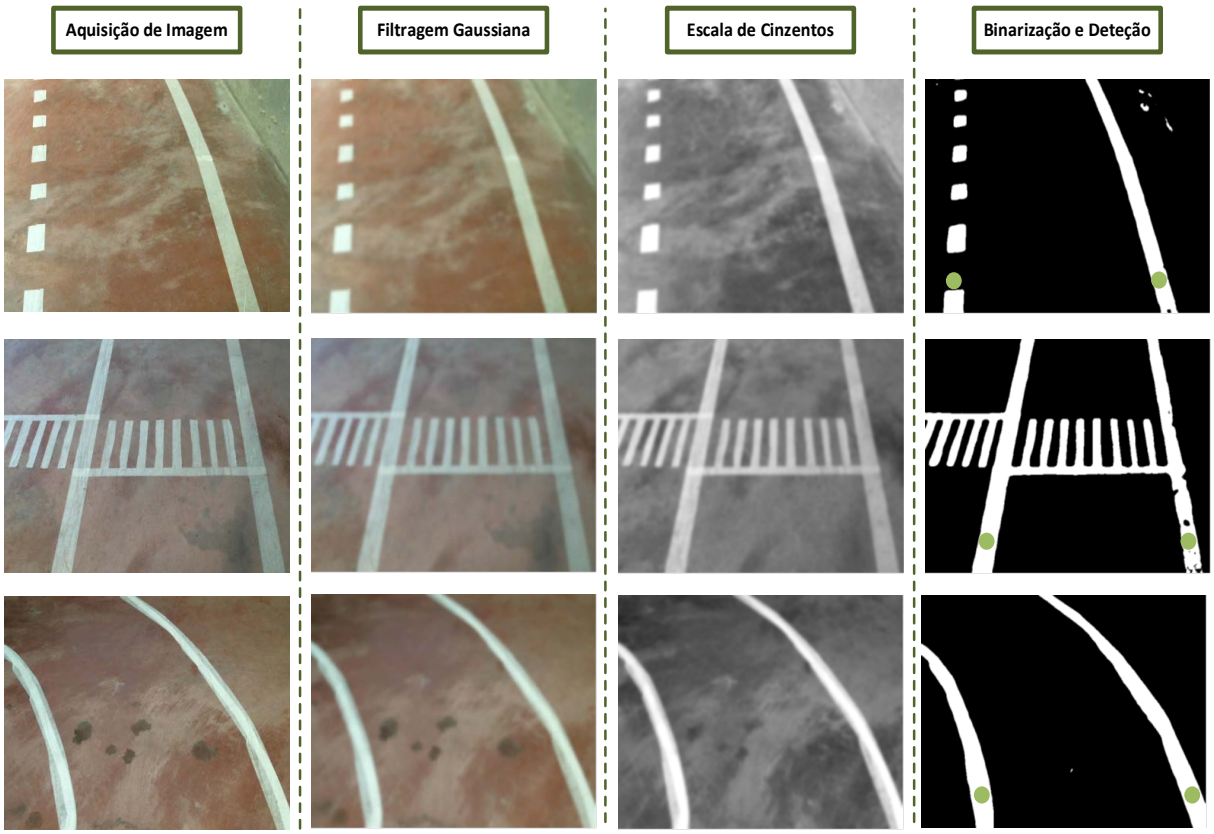


Figura 5.6 - Etapas da deteção das faixas

As zonas em teste na imagem anterior foram escolhidas por serem as mais suscetíveis a causar problemas de deteção. Contudo, como se verifica para os três casos, a posição das faixas de rodagem nas zonas de interesse foi corretamente detetada e identificada.

Avaliar a percentagem de falhas de deteção de 1 faixa ou mesmo das 2 em simultâneo durante as voltas de teste do robot na pista permite aferir a robustez do algoritmo implementado.

Deteção das Faixas de Rodagem (%)				
Velocidade	***2 cm/s***	***4 cm/s***	***6 cm/s***	***8 cm/s***
Falha de Deteção - 1 Faixa	*0 %*	*7 %*	*7 %*	*13 %*
Falha de Deteção - 2 Faixas em Simultâneo	*0 %*	*0 %*	*7 %*	*7 %*

Tabela 5.2 - Deteção das faixas de rodagem

De referir que o maior número de falhas de deteção se verificou em zonas de transição para o traço descontínuo e em zonas de curva.

5.3 ***Deteção de Obstáculos***

A deteção de obstáculos permite que o robot evite o embate e transponha os obstáculos que possam surgir na pista de teste.

Figura 5.7 - Exemplo de transposição de obstáculo

Resumem-se de seguida, os resultados dos testes de eficácia da deteção e contorno dos obstáculos presentes na pista.

Deteção e Contorno de Obstáculos (%)		
	Deteção do Obstáculo	***Contorno Adequado do Obstáculo***
Eficácia	*100 %*	*87 %*

Tabela 5.3 - Deteção e contorno de obstáculos

Como se verifica na tabela anterior, em todos os testes efetuados o obstáculo foi detetado convenientemente, porém, apenas em 87 % das situações, a transposição do mesmo foi efetuada corretamente.

O algoritmo desenvolvido contempla ainda uma funcionalidade extra de deteção de faixa continua. Nesta situação, na presença de um obstáculo, a transposição inicia-se apenas após cinco segundos.

Foi possível verificar que este aspeto do algoritmo também funciona satisfatoriamente.

Desvio de Obstáculos – Identificação de Faixa Contínua (%)	
Eficácia da Deteção	*80 %*

Tabela 5.4 - Desvio de obstáculos - Identificação de faixa contínua

5.4 ***Deteção de Sinais***

A inclusão da deteção de sinais neste sistema visa condicionar o movimento do robot consoante a sinalização vertical presente em determinado local da pista.

O algoritmo que sustenta a deteção, desenvolvido com base no método SIFT [65] de extração de características, revelou-se bem-sucedido, porém, bastante sensível à velocidade de circulação do robot e relativamente pesado do ponto de vista computacional.

Eficácia de Deteção da Sinalização (%)	
Sinal **50**	*87%*
Sinal 90	*80%*
Sinal Avanço	*87%*
Sinal Stop	*93%*
Sinal Passadeira	*87%*
Falsos Positivos	*27%*

Tabela 5.5 - Eficácia da deteção da sinalização

Tempo Total de Execução do Algoritmo de Controlo (s)		
	Sem Deteção de Sinais	***Com Deteção de Sinais***
Tempo	*0,1 s*	*0,7 s*

Tabela 5.6 - Tempo total de execução do algoritmo de controlo

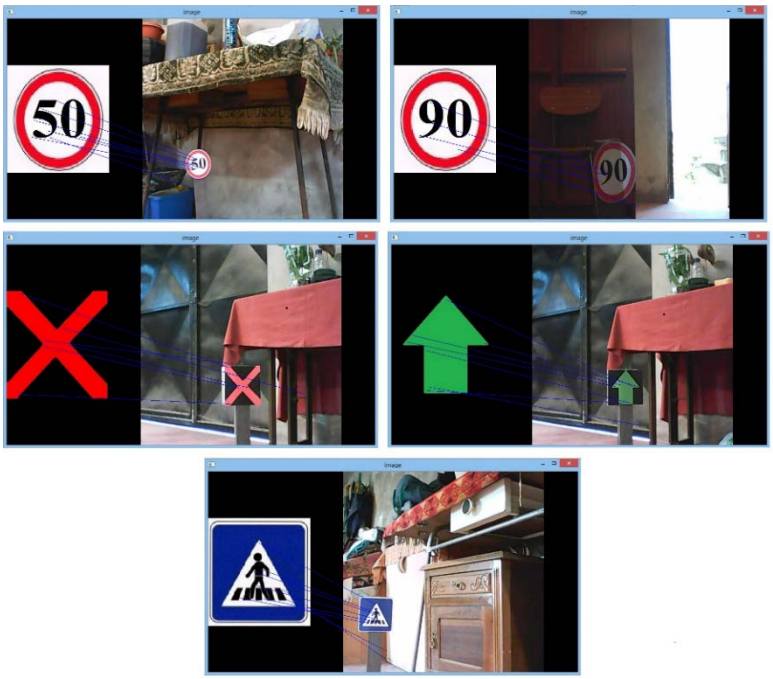


Figura 5.8 - Exemplos de deteção da sinalização

Como é percetível na Tabela 5.5, as percentagens de deteção são bastante satisfatórias, sendo contudo de salientar que para o sinal que apresentava maiores dificuldades ao algoritmo, o sinal de 90, a eficácia de deteção cifra-se apenas em 80%. Esta situação deve-se à reduzida quantidade de características identificativas no sinal extraíveis pelo método SIFT [65].

Relativamente à Tabela 5.6, verifica-se que a inclusão do módulo de deteção de sinais conduz a um tempo de execução bastante superior. Esta situação limita a velocidade de circulação do robot devido ao aumento do seu tempo de reação.

5.5 ***Movimentação do Robot***

Globalmente, a movimentação do robot depende da eficácia das etapas analisadas anteriormente e do desempenho do controlador responsável pela orientação do mesmo na faixa de rodagem.

Embora as etapas anteriores apresentem resultados bastante satisfatórios, a capacidade do robot seguir um determinado percurso, neste caso uma faixa de rodagem, em última análise é condicionada pelo controlador *Fuzzy* dimensionado. É a sua boa parametrização que garantirá um sistema de condução autónoma funcional e utilizável.

Assim, de forma a validar o desempenho global do sistema, e indiretamente do controlador, foram efetuadas tal como para os restantes testes, 15 voltas à pista que conduziram aos seguintes resultados:

Eficácia Global do Sistema Implementado (%)			
2 cm/s	***4 cm/s***	***6 cm/s***	***8 cm/s***
87 %	*80%*	*73 %*	*67 %*

Tabela 5.7 - Eficácia global do sistema implementado

A tabela anterior resume para cada velocidade em teste, a percentagem de voltas concluídas pelo robot seguindo convenientemente a trajetória, respeitando a sinalização e evitando o embate em obstáculos.

Da sua análise, verifica-se que o aumento da velocidade, como seria de esperar, representa uma diminuição da eficácia do sistema.

Quando comparada com a Tabela 5.1, torna-se percetível que a inclusão dos módulos de deteção de sinalização e obstáculos conduz a uma descida na sua eficácia global. Esta situação deve-se à precisão destas etapas e ao maior tempo de execução do algoritmo, que implica um maior tempo de reação do robot. Esta situação acentua-se para velocidades mais elevadas.

Durante os testes efetuados, foi também possível constatar que as zonas mais propícias a falha no seguimento da trajetória se encontravam em curvas e zonas de faixas tracejadas. Representa-se na imagem seguinte a localização dessas zonas na pista construída.

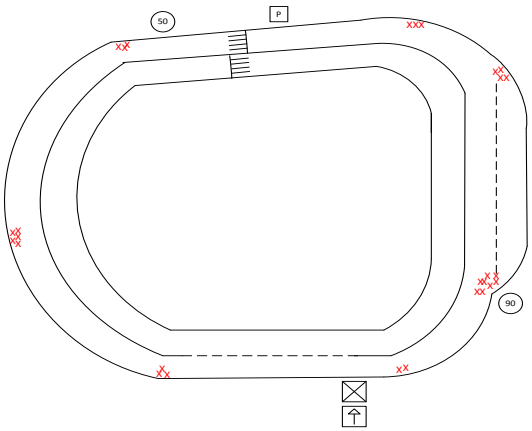


Figura 5.9 - Estrutura da pista com indicação das zonas de falhas

As imagens seguintes ilustram duas dessas zonas:

Figura 5.10 - Zona de falha na bifurcação

Figura 5.11 - Zona de falha em curva

Globalmente pode-se considerar que o sistema desenvolvido é eficaz, seguro e poderá futuramente servir de base a sistemas reais de condução autónoma.

6. Conclusões e Trabalho Futuro

Este capítulo é o culminar do trabalho de pesquisa e desenvolvimento que permitiu a implementação de um sistema de condução autónoma num robot móvel diferencial.

Aqui são apresentadas as principais conclusões sobre os resultados obtidos, é feito um balanço das limitações da metodologia adotada e são avaliadas possíveis perspetivas de desenvolvimento futuro do trabalho realizado.

6.1 *Conclusões*

O desenvolvimento de um sistema de condução autónoma é uma tarefa complexa que, naturalmente, envolve várias etapas intermédias. A obtenção de bons resultados implica que essas etapas sejam eficazes nas funções que de si dependem.

A metodologia proposta para a implementação deste sistema, consiste essencialmente em quatro módulos funcionais, interligados entre si, o processamento de imagem, o reconhecimento de sinais, a deteção de obstáculos e o sistema de controlo.

O processamento de imagem tem como principal tarefa fornecer ao sistema de controlo a posição do eixo central da faixa de rodagem. Consistindo numa sequência de processos de filtragem, conversão de espaço de cor, binarização e extração de características, os resultados a que conduziu são bastante positivos, tendo a identificação do eixo central da faixa de rodagem sido efetuada corretamente em 93 % dos testes (Tabela 5.2). É de salientar neste processamento, a determinação automática do posicionamento da zona de interesse e a utilização de um método automático de determinação do *threshold* ótimo sustentado no método de Otsu [80]. Estes dois aspetos, contribuíram para o bom desempenho deste processo, tornando a deteção das faixas de rodagem praticamente imune a variações da velocidade ou das condições de captação da imagem.

O reconhecimento de sinais visa condicionar o movimento do robot consoante a sinalização existente na pista. Suportado pelo método SIFT [65] de extração de características, revelou-se funcional, porém, tornou a execução do algoritmo de controlo bastante mais lenta, o que implica um aumento do tempo de reação do robot. De referir também que o método utilizado apresentou algumas limitações na deteção de sinais com estruturas pouco angulares. Por exemplo, para o sinal de 90 apenas se obteve uma percentagem de deteção de 80 % (Tabela 5.5), em contraponto com a média global de deteção dos restantes sinais de cerca de 89 % (Tabela 5.5).
A deteção de falsos positivos foi também um aspeto a ter em conta. Nos testes efetuados verificou-se um valor de 27 % (Tabela 5.5).

A deteção dos obstáculos presentes na faixa de rodagem permite ao robot garantir a sua preservação e minimizar o risco de acidente. Foi possível verificar que mesmo com obstáculos de diferentes dimensões, a deteção era corretamente efetuada. Nos testes efetuados obteve-se 100 % (Tabela 5.3) de eficácia nesta tarefa.
O contorno dos obstáculos pode também ser considerado bem-sucedido, apenas um reduzido número de testes, 13% (Tabela 5.3) não foi concluído com êxito.

O funcionamento do sistema de controlo resulta da interação dos módulos referidos anteriormente com o controlador *Fuzzy* implementado.
Este controlador permitiu alcançar resultados bastante satisfatórios, tornando o movimento do robot na faixa de rodagem bastante robusto. Para a velocidade de teste mais elevada, 8 cm/s, obteve-se uma eficácia global do sistema de 67 % (Tabela 5.7), o que comprovou a validade das regras de controlo e parâmetros de "*fuzzificação"* e *"defuzzificação"* utilizados.
Além da capacidade de controlo demonstrada, a simplicidade da sua implementação, que não obriga a uma complexa modelização do sistema a controlar, foi outro dos aspetos que atestaram como acertada a opção por este tipo de controladores.

Relativamente à linguagem de programação utilizada, Python, foi possível verificar que é bastante expedita e versátil, sobretudo devido à grande quantidade de bibliotecas que lhe podem ser adicionadas. Durante as experiencias efetuadas na fase de desenvolvimento do sistema, constatou-se que as bibliotecas SimpleCV e OpenCV que lhe serve de base, oferecem capacidades importantes para a criação de aplicações de visão computacional. Estas bibliotecas possuem potencial para alavancar pesquisas na área de visão computacional, pois além de serem ricas em funcionalidades, são de uso gratuito e de código-fonte aberto.

Globalmente, pode-se considerar que os resultados obtidos neste trabalho comprovam que as metodologias adotadas, a estrutura robótica implementada e o *software* utilizado permitem criar um robot móvel, capaz de desempenhar convenientemente tarefas inerentes à condução autónoma.

6.2 ***Trabalho Futuro***

Os resultados obtidos neste trabalho tornam-no um bom ponto de partida para desenvolvimentos futuros na área da robótica móvel. Durante a sua execução, esteve sempre presente a preocupação de utilizar equipamentos de baixo custo e desenvolver algoritmos simples e organizados modelarmente, de forma a tornar a sua reutilização viável.

A estrutura mecânica utilizada, embora tenha apresentado resultados satisfatórios, poderá ser melhorada, por exemplo, com a adição de um eixo direcional. Esta nova estrutura, permitiria uma maior manobrabilidade do robot e torná-lo-ia mais semelhante a um veículo automóvel.

A estrutura modular adotada na programação, permite que o algoritmo de deteção de sinais possa ser melhorado, por exemplo através da utilização de métodos de extração de características mais eficientes e menos pesados do ponto de vista computacional, sem que isso obrigue a alterações significativas no restante sistema de controlo.

O algoritmo de contorno de obstáculos poderá também de forma relativamente simples ser melhorado. Com recurso à odometria, seria possível tornar o contorno mais célere e capaz de com grande eficácia ultrapassar veículos em movimento.

Um dos objetivo deste trabalho era tornar o sistema desenvolvido capaz de ser utilizado num veículo automóvel. Assim, a implementação deste sistema num veículo e a avaliação das suas capacidades em condições reais, sujeito às condições de trafego e climatização, poderá ser um excelente ponto de partida para um trabalho nesta área.

7. Referências Bibliográficas

[1] **Alves de Araújo, S. and Henriques Librantz, A. F.**, "Visão e Inteligência computacionais aplicadas a navegação autónoma de Robot", *Exacta*, pp. 343-352, 2006

[2] **Aurich, V. and Weule, J.**, "Non-Linear Gaussian Filters Performing Edge Preserving Diffusion" in *Mustererkennung 1995*, Springer Berlin Heidelberg, 1995, pp. 538-545

[3] **Bauzil, G., Briot, M., and Ribes, P.**, "A Navigation Sub-System Using Ultrasonic Sensors for the Mobile Robot HILARE," in *1st Int. Conf. on Robot Vision and Sensory Controls*, Stratfor-upon-Avon, UK, 1981

[4] **Bay, H., Tuytelaars, T., and Gool, L.**, "SURF: Speeded Up Robust Features" in *Computer Vision – ECCV 2006*, vol. 3951, Springer Berlin Heidelberg, 2006, pp. 404-417

[5] **Beis, J. S. and Lowe, D. G.**, "Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces," presented at the Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97), Puerto Rico, 1997

[6] **Blackwell, M.**, "The URANUS Mobile Robot," Robotics Institute, Carnegie Mellon University, 1990

[7] **Bonin-Font, F., Ortiz, A., and Oliver, G.**, "Visual Navigation for Mobile Robots: A Survey", *Journal of Intelligent and Robotic Systems*, vol. 53, pp. 263-296, 2008

[8] **Borenstein, J., Everett, H. R., and Feng, L.**, *Navigating Mobile Robots: Systems and Techniques*, A K Peters, 1996

[9] **Boyle, R. and Thomas, R. C.**, "Computer Vision: A First Course", B. S. Publications, Ed., Blackwell Scientific Publications, 1988, pp. 35-41

[10] **Brooks, R.**, "A Robust Layered Control System for a Mobile Robot" in *IEEE Transactions of Robotics and Automation*, 1986, pp. 14-23

[11] **Bundy, A. and Wallen, L.**, "Difference of Gaussians" in *Catalogue of Artificial Intelligence Tools*, Springer Berlin Heidelberg, 1984, pp. 30-35

[12] **Cai, N. and al, e.**, "Traffic Sign Recognition Based on SIFT Features" in *Advanced Materials Reserarch*, vol. 121-122, 2010, pp. 596-599

[13] **Calonder, M., Lepetit, V., Strecha, C., and Fua, P.**, "BRIEF: Binary Robust Independent Elementary Features" in *Computer Vision – ECCV 2010*, vol. 6314, Springer Berlin Heidelberg, 2010, pp. 778-792

[14] **Canny, J.**, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp. 679-698, 1986

[15] **Celebi, M. E., Kingravi, H. A., Iyatomi, H., Aslandogan, Y. A., Stoecker, W. V., Moss, R. H., *et al.*,** "Border detection in dermoscopy images using statistical region merging", *Skin Res Technol*, vol. 14, pp. 347-353, Aug 2008

[16] **Cernic, S., Jezierski, E., Britos, P., Rossi, B., and García Martínes, R.**, "Genetic Algorithms Applied to Robot Navigation Controller Optimization," Buenos Aires Institute of Technology, 1999

[17] **Cirstea, M. N., Dinu, A., Khor, J. G., and McCormick, M.**, "Neural and Fuzzy Logic Control of Drives and Power Systems", Woburn, Newnes, 2002, pp. 160-164

[18] **Demaagd, K., Oliver, A., Oostendorp, N., and Scott, K.**, *Pratical Computer Vision With SimpleCV*, O'Reilly Media, 2012

[19] **DeSouza, G. N. and Kak, A. C.**, "Vision for Mobile Robot Navigation: A Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 237-267, 2002

[20] **Endriss, U.**, "Lecture Notes - An Introduction to Prolog Programming", Institute for Logic, Language and Computation, 2013

[21] **Filho, O. and Neto, H.**, "Processamento Digital de Imagens", Brasport, Rio de Janeiro, 1999, pp. 83-99

[22] **Flusser, J.**, "Moment Invariants in Image Analysis" in *World Academy of Science, Engineering and Technology*, McGraw-Hill Education (India) Pvt Limited, 2005, pp. 190-231

[23] **Fu, K. S., Gonzales, R. C., and Lee, C. S. G.**, "Robotics - Control, Sensing, Vision and Intelligence", New York, McGraw-Hill Book Inc., 1987, pp. 190-213

[24] **Gal, R. and Cohen, D.**, "Salient Geometric Features for Partial Shape Matching and Similariy", *ACM Trans. Graph.*, pp. 130-150, 2006

[25] **Gudwin, R.**, "Linguagens de Programação", Universidade Estadual de Campinas - Faculdade de Engenharia Elétrica e de Computação, 1997

[26] **Han, K. and DeSouza, G.**, "A Feature Detection Algorithm for Autonomous Camera Calibration" in *Proceedings of the 2007 IFAC International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, France, 2007, pp. 286-291

[27] **He, Z. and Zhu, Y.**, "A Fast Object Detection Method with Rotation Invariant Features" in *World Academy of Science, Engineering and Technology*, vol. 5, 2011, pp. 808-813

[28] **Hill, G., Horskotte, E., and Teichrow, J.**, "Fuzzy-C Development System User's Manual - Release 2.1" in *Togai InfraLogic Inc.*, Irvine, CA, 1989

[29] http://atlas.web.ua.pt/

[30] http://cimg.sourceforge.net/

[31] http://definitions.uslegal.com/r/robotics/

[32] http://jusb.sourceforge.net/

[33] http://mars.jpl.nasa.gov/MPF/rover/sojourner.html

[34] http://paginas.fe.up.pt/~ee99206/orientadores.html

[35] http://pyserial.sourceforge.net/

[36] http://ria.ua.pt/handle/10773/5034

[37] http://sourceforge.net/apps/trac/pyusb/

[38] http://www.argo.ce.unipr.it/argo/english/index.html

[39] http://www.axis.com/files/whitepaper/wp_ccd_cmos_40722_en_1010_lo.pdf

[40] http://www.cs.cmu.edu/~minerva/

[41] http://www.darpa.mil/About/History/Archives.aspx

[42] http://www.eautoexperts.com/mercedes-benz-history/

[43] http://www.ime.usp.br/~slago/slago-pascal.pdf

[44] http://www.libusb.org/

[45] http://www.lsa.isep.ipp.pt/pages/lsa_home.html

[46] http://www.mikechiafulio.com/RIDE/history.htm

[47] http://www.mobilerobots.com/ResearchRobots/Seekur.aspx

[48] http://www.oracle.com/technetwork/java/index-jsp-141752.html

[49] http://www.oracle.com/technetwork/java/javase/tech/jai-142803.html

[50] http://www.pythonware.com/products/pil/

[51] http://www.ri.cmu.edu/pub_files/pub2/bares_john_1999_1/bares_john_1999_1.pdf

[52] http://www.robothalloffame.org/inductees/03inductees/unimate.html

[53] http://www.tannerhelland.com/3643/

[54] http://www.vivelesrobots-education.dk/english/mobile-robots/history

[55] **Huang, H. and Chang, C.**, "A Novel Efficient Threshold Proxy Signature Scheme", *Information Sciences*, vol. 176, pp. 1338-1349, 2006

[56] **Huang, Z. and Leng, J.**, "Analysis of Hu's Moment Invariants on Image Scaling and Rotation," presented at the 2nd International Conference on Computer Engineering and Technology, Chengdu, China, April, 2010

[57] **Ihara, A., Fujiyoshi, H., Takagi, M., Kumon, H., and Tamatsu, Y.**, "Improved Matching Accuracy in Traffic Sign Recognition by Using Differente Feature Subspaces," in *Machine Vision Applications 2009 (MVA2009)*, Aichi, Japan, 2009, pp. 130-133

[58] **International Electrothecnical Comission**, "Fuzzy Control Programming," 1997

[59] **Jia, W., Zhang, H., He, X., and Wu, Q.**, "A Comparison on Histogram Based Image Matching Methods" in *Computer Vision Research Group*, University of Technology, Sidney, 2007

[60] **Kemeny, J. and Kurtz, T.**, "True BASIC Reference Manual", True BASIC, 1990

[61] **Kim, J., Bok, Y., and Kweon, I. S.**, "Robust Vision-Based Autonomous Navigation against Environment Changes," presented at the International Conference on Intelligent Robots and Systems, Nice, France, 2008

[62] **Kundur, S. and Raviv, D.**, "A Vision-Based Pragmtic Strategy for Autonomous Navigation ", *Pattern Recognition*, vol. 31, pp. 1221-1239, 1998

[63] **Laganière, R.**, *OpenCV 2 Computer Vision Application Programming Cookbook: Over 50 Recipes to Master this Library of Programming Functions for Real-time Computer Vision*, Packt Open Source Pub., 2011

[64] **Li, C. and Tam, P.**, "An Iterative Algorithm for Minimum Cross Entropy Thresholding" in *Pattern Recognition Letters 18*, 1998, pp. 771-776

[65] **Lowe, D.**, "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, vol. 60, pp. 91-110, 2004

[66] **Lutz, M.**, "Learning Python", 5th, O'Reilly Media, 2013

[67] **Marchi, J.**, "Navegação de Robôs Móveis Autónomos: Estudo e Implementação de Abordagens", UFSC - Universidade Federal de Santa Catarina, Tese de Mestrado, 2001

[68] **McNeil, D. and Freiberger, P.**, "Fuzzy Logic: The Discovery of a Revolutionary Computer Technology - and how it is Changing our World", Simon & Shuscter, New York, 1993

[69] **Mikolajczyk, K. and Schmid, C.**, "A Performance Evaluation of Local Descriptors", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1615-1630, October 2005

[70] **Milgram, D.**, "Constructing Tree for Region Description" in *Computational Graphics Image Process*, 1979, pp. 88-99

[71] **Ming-Kuei, H.**, "Visual Pattern Recognition by Moment Invariants", *IRE Transactions on Information Theory*, vol. 8, pp. 179-187, 1962

[72] **Mitchell, R. J. and Keating, D. A.**, "Neural Network Control of Simple Mobile Robot" in *Concepts for Neural Networks*, Springer London, 1998, pp. 95-107

[73] **Moravec, H.**, "Visual Mapping by a Robot Rover," in *Proceedings of the 6th international joint conference on Artificial intelligence*, Tokyo, 1979, pp. 598-600

[74] **Nascimento, C.**, "Inteligência Artificial em Controle e Automação" in *Edgard Blucher*, FAPESP, 2004, pp. 170-214

[75] **Ng, K. C. and Trivedi, M. M.**, "A Neuro-Fuzzy Controller for Mobile Robot Navigation and Multirobot Convoying", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 28, pp. 829-840, 1998

[76] **Niemeyer, P. and Leuck, D.**, "Learning Java", 4th, O'Reilly Media, 2013

[77] **Nilsson, N. J.**, "Shakey the Robot," in *Technical Note of SRI International*, , Menlo Park, CA, 1984

[78] **Nishida, M.**, "Experimental Study on On-Off Control Systems for flexible quadruped locomotion robot" in *Advanced Intelligent Mechatronics (AIM)*, IEEE/ASME International Conference on Budapest, 2011, pp. 880-885

[79] **Normey-Rico, J. E., Alcalá, I., Gómez-Ortega, J., and Camacho, E. F.**, "Mobile robot path tracking using a robust PID controller", *Control Engineering Practice*, vol. 9, pp. 1209-1214, 11// 2001

[80] **Otsu, N.**, "A Threshold Selection Method from Gray-Level Histograms" in *IEEE Transactions on Systems, Man and Cybernetics*, vol. 1, 1979, pp. 62-66

[81] **Pezeshki, S., Badalkhani, S., and Javadi, A.**, "Performance Analysis of a Neuro-PID Controller Applied to a Robot Manipulator", *International Journal of Advanced Robotics Systems*, vol. 9, 2012

[82] **Pio, J. L. d. S.**, "Navegação Robótica Aérea Baseada em Visão - Requesitos de Processamento de Imagens para Projeto e Implementação" in *III Workshop em Tratamento de Imagens*, UFMG, Belo Horizonte, 2002, pp. 1-9

[83] **Pires, J. N.**, "Codex Automaticus - Os Primórdios da Robótica Moderna", 1, 2003, pp. 1-5

[84] **Prade, H., Zimmermann, H., Dubois, D., Konolig, K., Ruspini, H., and Saffiotti, A.**, "Using Fuzzy Logic for Mobile Robot Control" in *International Handbook of Fuzzy Sets and Possibility Theory*, vol. 5, Kluwer Academic Publishers Group, 1997

[85] **Ramella, G. and Sanniti di Baja, G.**, "Color Histogram-Based Image Segmentation" in *Computer Analysis of Images and Patterns*, vol. 6854, Springer Berlin Heidelberg, 2011, pp. 76-83

[86] **Reilly, E. D.**, *Milestones in Computer Science and Information Technology*, Greenwood Publishing Group ed., Greenwood Press, 2003

[87] **Ribeiro, M. I.**, "Uma Viagem ao Mundo dos Robots", Ciclo Colóquios Despertar para a Ciência, Universidade do Porto, 2004, p. 1 e 2

[88] **Rosenfeld, A. and Kak, A.**, "Digital Picture Processing" in *Academic Press*, vol. 2, New York, 1982, pp. 349-360

[89] **Rosten, E. and Drummond, T.**, "Machine Learning for high-speed corner detection," presented at the European Conference on Computer Vision, Graz, Austria, 2006

[90] **Rother, C., Minka, T., Blake, A., and Kolmogorov, V.**, "Cosegmentation of Image Pairs by Histogram Matching - Incorporating a Global Constraint into MRFs," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York, USA, 2006, pp. 993-1000

[91] **Russell, S. and Norvig, P.**, "Artificial Intelligence: A Modern Approach", Prentice Hall International, 2010, pp. 407-609

[92] **Samet, H.**, "The Design and Analysis of Spatial Data Structures - 2 ed.", Addison Wesley, 1990, pp. 97-143

[93] **Santos-Victor, J., Sandini, G., CurOtto, F., and Garibaldi, S.**, "Divergent Stereo for Robot Navigation: Learning from Bees," in *Conference on Computer Vision and Pattern Recognition*, New York, USA, 1993, pp. 434-439

[94] **Siegwart, R. and Nourbakhsh, I. R.**, "Introduction to Autonomous Mobile Robots", Bradford Book, 2004, pp. 257-298

[95] **Sobreira, H.**, "Clever Robot," Tese de Mestrado, FEUP, 2009

[96] **Souza, C., Capovilla, F., and Eleotério, T.**, "Criptografia", *Centro Paulo Sousa, Taquaritinga, São Paulo,* 2010

[97] **Stankovic, R. and Falkowski, B.**, "The Haar Wavelet Transform: Its Status and Achievements" in *Computers and Electric Engineering 29*, Pergamon, 2003, pp. 25-44

[98] **Stroustrup, B.**, *The C++ Programming Language - Fourth Edition,* Addison-Wesley ed., 2013

[99] **Suzuki, S. and Abe, K.**, "Topological Structural Analysis of Digitized Binary Images by Border Following", *Computer Vision, Graphics and Image Processing,* vol. 30, pp. 32-46, 1985

[100] **Thrun, S., Bucken, A., Bugard, W., Fox, D., Frohlinghaus, T., Henning, D., *et al.*,** "Map Learning and High-Speed Navigation in RHINO" in *Case Studies of Successful Robot Systems,* Cambridge, MA, MIT Press, 2008, pp. 21-52

[101] **Thrun, S., Montemerlo, H. D., and Al, e.**, "Stanley: The Robot that Won the DARPA Grand Challange", *Journal of Field Robotics,* pp. 661-692, 2006

[102] **Tomatis, N., Nourbaksh, I., Arras, K. O., and Siegwart, R.**, "A Hybrid Approach for Robust and Precise Mobile Robot Navigation with Compact Environment Modeling," in *International Conference on Robotics & Automation*, Seoul, Korea, 2001

[103] **Twardowski, T., Cyganek, B., and Borgosz, J.**, "Gradient Based Dense Stereo Matching" in *Image Analysis and Recognition*, vol. 3211, Springer Berlin Heidelberg, 2004, pp. 721-728

[104] ***Recomentadion ITU-R BT.709-5, Parameter values for the HDTV standards for production and international programme exchange***, BT Series - Broadcasting Service (Television), 2002

[105] **Vidal, A.**, "Feupcar 2.0 - Condução Autónoma no Festival Nacional de Robótica ", Tese de Mestrado, FEUP, 2011

8. Anexo I

Lista de comandos do controlador LM629N

Neste ponto descrevem-se alguns dos comandos disponibilizados pelo controlador LM629N, necessários à realização de diversas operações de controlo, tais como, inicialização do sistema, interrupção de controlo, carregamento dos parâmetros do filtro de controlo, controlo de trajetória e verificação de dados. A tabela seguinte resume o conjunto de instruções do controlador LM629N.

Conjunto de Instruções do Controlador				
Comando	***Tipo***	***Descrição***	***Código***	***Bytes***
RESET	Inicialização	Reset ao LM629N	00	0
DFH	Inicialização	Define a posição de absoluto	02	0
SIP	Interrupção	Define a posição do sinal de indexação	03	2
LPEI	Interrupção	Interrupção por erro de posição	IB	2
LPES	Interrupção	Paragem por erro de posição	IA	2
SBPA	Interrupção	Define ponto de referência Absoluto	20	4
SBPR	Interrupção	Define ponto de referência Relativo	21	4
MSKI	Interrupção	Definição de potenciais interrupções	IC	2
RSTI	Interrupção	Reset de interrupções	ID	2
LFIL	Filtro	Carregamento dos parâmetros do filtro	IE	2 a 10
UDF	Filtro	Atualização do filtro	04	0
LTRJ	Trajetória	Carregamento de trajetória	IF	2 a 14
STT	Trajetória	Início de movimento	01	0
RDSTAT	Leitura	Leitura do byte estados	----	1
RDSIGS	Leitura	Leitura de registo de sinais	0C	2
RDIP	Leitura	Leitura da posição de indexação	09	4
RDDP	Leitura	Leitura da posição de referência	08	4
RDRP	Leitura	Leitura da posição real	0A	4
RDDV	Leitura	Leitura da velocidade de referência	07	4
RDRV	Leitura	Leitura de velocidade real	OB	2
DRSUM	Leitura	Leitura do erro integral acumulado	0D	2

Tabela 8.1 - Conjunto de instruções do controlador

Comandos de inicialização do sistema

- ***Comando RESET***

A execução deste comando resulta em igualar a zero os seguintes fatores do controlador: coeficientes de filtro, parâmetros de trajetória, e o sinal de saída de controlo do motor.

Este comando desativa ainda todos os comandos de interrupção, com exceção dos comandos SBPA e SBPR que mantêm o seu estado, configura a porta de saída de dados para 8 bits, e define a posição absoluta atual como origem posição de zero absoluto ou "home". Este comando que pode ser executado com os motores parados ou em funcionamento é completado em menos de 1.5 ms

- ***Comando DFH: Define Home***

Este comando declara a posição atual do motor como posição de zero absoluto ou " home". Podendo ser executado com o motor em funcionamento, apenas afetará a posição de paragem do motor, após o envio do comando STT (start motion control).

Comandos de interrupção de controlo

Estes comandos estão associados a condições que podem ser usadas para interromper a execução do programa de controlo. Para que qualquer um dos comandos de interrupção efetivamente atue sobre o programa de controlo, o utilizador deve colocar o seu bit correspondente no comando MSKI (MaSK Interrupts) com estado lógico 1. Os estados lógicos dos diversos comandos de interrupção de controlo são dados a conhecer ao computador de controlo via leitura e análise do byte de estados.

- ***Comando SIP: Set Index Position***

Após a execução deste comando a posição absoluta que corresponde á ocorrência do próximo impulso de indexação, será gravada no registo de indexação e o bit 3 do byte de estados será posto a nível lógico 1. A posição é gravada quando os dois sinais em quadratura do encoder e o sinal de indexação estiverem com um nível lógico baixo. Este registo pode então ser lido, facilitando a definição da posição de zero absoluto (home) com impulso de indexação

- ***Comando LPEI: Load Position Error for Interrupt***

Esta instrução permite ao utilizador, impor um valor limite (0000 a 7FFF hex) para a deteção do erro de posição. A deteção do erro ocorre quando a amplitude absoluta do erro de posição excede o valor limite imposto, o que resulta em colocar o bit 5 do byte de estados com um nível lógico alto. O envio dos bytes de dados do valor limite de erro é feito do mais significativo (bits 15 a 8) para o menos significativo (bits 7 a 0):

- ***Comando LPES: Load Position Error for Stopping***

Esta instrução adiciona á Instrução anterior a capacidade de parar o motor após a deteção de um erro de posição. Tal como o comando LPEI o primeiro byte de dados a enviar é o mais significativo e o bit 5 do byte de estados é colocado com um nível lógico alto.

- ***Comando SBPA: Set Break Point Absolute***

Este comando permite ao utilizador definir um ponto de referência (C0000000 a 3FFFFFFF hex) em termos de posição absoluta. O bit 6 do byte de estados é colocado com um nível lógico 1 quando o ponto de referência é atingido, podendo se programado interromper o programa de controlo.

- ***Comando SBPR: Set Break Point Relative***

Este comando diferencia-se do anterior na medida em que permite definir um ponto de referência (C0000000 a 3FFFFFFF hex) em termos de posição relativa. O valor carregado deve ser tal que quando adicionado à posição alvo o resultado se mantenha dentro da gama de valores de posição do sistema (C0000000 a 3 FFFFFFF hex). Assim como no comando anterior, também o bit 6 do byte de estados é colocado com um nível lógico alto quando o ponto de referência é atingido.

- ***Comando MSKI: Mask Interrupts***

Este comando através dos dois bytes de dados que se seguem imediatamente ao código do comando MSKI, permite programar quais das possíveis condições de interrupção deveram interromper a execução do programa de controlo. Após uma interrupção a leitura do byte de estados permite ao utilizador saber quais das condições previamente definidas pelo comando MSKI provocaram tal interrupção.

Bytes de dados do comando MSKI	
Bit	***Tipo***
Bit 15 a 7	Não utilizados
Bit 6	Interrupção de ponto de referência
Bit 5	Interrupção de erro posição
Bit 4	Interrupção de endereço de posição excedido
Bit 3	Interrupção de sinal de indexação adquirido
Bit 2	Interrupção de trajetória completada
Bit 1	Interrupção de erro de comando
Bit 0	Não utilizado

Tabela 8.2 - Bytes de dados do comando MSKI

De referir que este comando apenas controla o processo de interrupção do programa de controlo, a leitura do byte de estados reflete em qualquer instante as condições atuais independentemente dos bits do comando MSKI.

- ***Comando RSTI: Reset Interrupts***

Após a ocorrência de uma das potenciais condições de interrupção, este comando permite proceder ao reset dos bits do byte de estados (status byte)

- ***Comando LFIL: Load Filter Parameters***

Este comando permite definir os coeficientes (kp, ki, kd e il) do filtro e o intervalo de amostragem derivativo. Após carregar o código do comando LFIL (0x1E), os próximos dois bytes a serem enviados ao LM629N constituem uma palavra de 16-Bit de controlo do filtro que, com base na tabela 0.3, informam o controlador da natureza e do número de bytes de dados que serão posteriormente enviados.
O envio destes dois bytes deve ser feito, do mais significativo (bit 15 a 8) para o menos significativo (bit 7 a 0).

Bytes de dados do filtro PID	
Bit	***Tipo***
Bit 15	Intervalo de amostragem derivativo Bit 7
Bit 14	Intervalo de amostragem derivativo Bit6
Bit 13	Intervalo de amostragem derivativo Bit 5
Bit 12	Intervalo de amostragem derivativo Bit4
Bit 11	Intervalo de amostragem derivativo Bit 3
Bit 10	Intervalo de amostragem derivativo Bit 2
Bit 9	Intervalo de amostragem derivativo Bit 1
Bit 8	Intervalo de amostragem derivativo Bit 0
Bit 7	Não utilizado
Bit 6	Não utilizado
Bit 5	Não utilizado
Bit 4	Não utilizado
Bit 3	Carregar kp
Bit 2	Carregar ki
Bit 1	Carregar kd
Bit 0	Carregar il

Tabela 8.3 - Bytes de dados do filtro PID

Bit 8 a 15, permitem selecionar o intervalo de amostragem do termo derivativo.

Bit 0 a 3, informam o LM629N quais dos parâmetros do filtro estão prestes a ser carregados. Pode optar-se por carregar penas alguns dos parâmetros do filtro. Aqueles escolhidos para serem carregados são indicados pelo estado lógico 1 na posição do bit correspondente da palavra de controlo do filtro.

Os bytes de dados enviados ao LM629N imediatamente após a palavra de controlo do filtro, constituem os parâmetros a carregar. Devem ser enviados aos pares compreendendo palavras de 16-bit e pela ordem decrescente indicada na palavra de controlo, ou seja, primeiro o coeficiente kp, depois o ki, seguido do kd e por fim o limite de integração. Cada coeficiente do filtro (kp, ki, kd e il) é composto duas palavras (32-bit), sendo o seu envio realizado bit mais significativo para o menos significativo.

- ***Comando UDF: Update Filter***

O comando UDF é utilizado para atualizar os parâmetros do filtro, programados pelo comando LFIL. Todos ou apenas alguns dos parâmetros do filtro podem ser carregados, no entanto, só após a execução do comando UDF estas alterações irão afetar a o controlo realizado pelo filtro.

Comandos de controlo de trajectória

- ***Comando LTRJ: Load Trajectory Parameters***

Este comando possibilita definir os parâmetros de controlo de trajetória (posição, velocidade e aceleração), o modo de operação (posição ou velocidade) e a direção de rotação (apenas no modo de velocidade). A definição destes parâmetros permite gerar o perfil de velocidade pretendido.

Após carregar o código do comando LTRJ (0x1F,os próximos dois bytes a serem enviados ao LM629N constituem uma palavra de 16 – Bit de controlo de trajetória que, com base na tabela 0.4, informam o controlador da natureza e do número de bytes de dados que serão posteriormente enviados

O envio destes dois bytes deve ser feito pela ordem decrescente, ou seja, do mais significativo (bit 15) para o menos significativo (bit 0)

Bytes de dados do comando LTRJ	
Bit	***Tipo***
Bit 15	Não utilizado
Bit 14	Não utilizado
Bit 13	Não utilizado
Bit 12	Sentido de rotação (modo de velocidade apenas)
Bit 11	Modo de velocidade
Bit 10	Paragem suave (Desacelera como programado)
Bit 9	Paragem abrupta (Desaceleração máxima)
Bit 8	Desligar motor
Bit 7	Não utilizado
Bit 6	Utilizado Não
Bit 5	Carregar Aceleração
Bit 4	Considerar a aceleração como relativa
Bit 3	Carregar velocidade
Bit 2	Considerar a Velocidade como relativa
Bit 1	Carregar Posição
Bit 0	Considerar a Posição como relativa

Tabela 8.4 - Bytes de dados do comando LTRJ

Bit 12, permite definir o sentido de rotação do motor quando selecionado o modo de velocidade. O estado lógico 1 indica o sentido de direção direto

Bit 11, determina se o LM629N opera no modo de velocidade (estado lógico 1) ou no modo de posição (estado lógico “zero”).

Bits 8 a 10, permitem definir o tipo de paragem que se pretende que o motor execute. Estes bits têm um funcionamento exclusivo, ou seja, em qualquer instante apenas um dos bits pode ter o estado lógico 1.
Bits 5 a 0, indicam ao LM629N quais dos parâmetros de trajetória serão posteriormente enviados palavras.

Qualquer dos parâmetros é suscetível de ser alterado com o motor em movimento, no entanto se a aceleração for modificada, o próximo comando STT (start) só deve ser enviado após o motor ter completado o movimento atual estar parado.
Os bytes de dados enviados ao LM629N imediatamente após apalavra de controlo de trajetória, constituem os parâmetros de trajetória a carregar. Devem ser enviados aos pares compreendendo palavras de 16-bit e pela ordem decrescente indicada na palavra de controlo, ou seja, primeiro a aceleração, depois a velocidade e por fim a posição. Cada parâmetro de trajetória (posição, velocidade e aceleração) é composto duas palavras (32-bit), sendo o seu envio realizado do bit mais significativo para o menos significativo.

Comando de início de movimento

O comando STT (Start Motion Control) é utilizado para executar a trajetória previamente carregada pelo comando LTRJ. Caso o valor de aceleração tenha sido alterado o envio do comando STT só deverá ser efetuado após o motor ter completado o movimento atual ou estar parado. Se o comando STT for executado nestas condições será gerado um erro de interrupção e o comando ignorado.

Comando de leitura de dados:

OS seguintes comandos do LM629N, podendo ser executados com o motor em movimento ou com o motor parado, são utilizados para a aquisição de dados de diversos registos do controlador tais como, posição e velocidade.

Com exceção do comando RDSTAT, os dados são lidos do LM629N a partir da porta de dados, após o envio do comando correspondente para a porta de comandos.

- ***Comando RDSTAT: Read Status Byte***

A leitura e análise do byte de estados, permite monitorizar o funcionamento do controlador de motores. Os oito bits que constituem são definidos na tabela seguinte.

Byte de estados do comando RDSTAT	
Bit	***Tipo***
Bit 7	Motor desligado
Bit 6	Ponto de referência atingido (interrupção)
Bit 5	Erro de posição excessivo (interrupção)
Bit 4	Endereço de posição excedido (interrupção)
Bit 3	Sinal de indexação adquirido (interrupção)
Bit 2	Trajetória completada (interrupção)
Bit 1	Erro do comando (interrupção)
Bit 0	Busy bit

Tabela 8.5 - Byte de estados do comando RDSTAT

Bit 7, o estado lógico 1 indica a paragem do motor. Podendo ser "ativado" pelos seguintes comandos: RESET, LPES ou quando é selecionada a paragem manual do motor através do comando LTRJ.

Bit 6, o estado lógico 1 indica que o ponto de referência previamente carregado pelos comandos SBPA e ou SBPR foi excedido. O seu funcionamento é independente do estado do bit de interrupção correspondente.

Bit 5, o estado lógico 1 indica a existência de uma condição de interrupção do movimento por erro de posição excessivo. Esta situação verifica-se quando o valor de erro de posição carregado pelos comandos LPEI e ou LPES é exercido.

Bit 4, o estado lógico 1 indica que o espaço de endereçamento de posição do LM629N foi excedido. Apenas se verifica quando a operar no modo de velocidade.

Bit 3, o estado lógico 1 sinaliza a ocorrência de um impulso de indexação se o comando SIP tiver sido executado. Indica também a atualização do registo de posição de indexação.

Bit 2, o estado lógico 1 indica que a trajetória especificada pelo comando LTRJ e inicializada pelo comando STT foi completada. O estado deste bit resulta da operação lógica OR entre o bit7 e o bit 10 do sinal de registo (comando RDSIGS).

Bit 1, o estado lógico 1 indica que ocorreu uma tentativa de leitura de dados quando o procedimento correto seria o de escrita de dados (ou vice versa).

Bit 0, o estado lógico "zero" indica que uma operação de leitura ou de escrita pode ser executada. Qualquer comando enviado com o busy bit ocupado (estado lógico 1) será ignorado não gerando no entanto qualquer erro de interrupção.

- ***Comando RDSIGS: Read Signals Register***

Podendo ser executado a qualquer instante, a leitura do registo interno de "sinais" do LM629N pode ser feita usando este comando. O primeiro byte é o mais significativo. O byte menos significativo deste registo, com exceção do bit 0, duplica o byte de estados.

Byte de dados do comando RDSIGS	
Bit	***Tipo***
Bit 15	Interrupção do programa de controlo
Bit 14	Aceleração Carregada
Bit 13	UDF executado
Bit 12	Sentido de rotação direto
Bit 11	Modo de Velocidade
Bit 10	Posição alvo
Bit 9	Desligar motor após excesso de erro de posição
Bit 8	Modo de 8-bits de saída
Bit 7	Motor desligado
Bit 6	Ponto de referência atingido (interrupção)
Bit 5	Erro de posição excessivo (interrupção)
Bit 4	Endereço de posição excedido (interrupção)
Bit 3	Sinal de indexação adquirido (interrupção)
Bit 2	Trajetória completada (interrupção)
Bit 1	Erro de comando (interrupção)
Bit 0	Adquirir sinal de indexação (SIP executado)

Tabela 8.6 - Byte de dados do comando RDSIGS

Bit 15, o estado lógico 1 indica que ocorreu uma operação de interrupção. Com a execução do comando RSTI é colocado com o estado lógico "zero".

Bit 14, o estado lógico 1 indica que os dados referentes à aceleração foram escritos para o LM629N. Com a execução do comando STT é colocado com o estado lógico "zero".

Bit 13, o estado lógico 1 indica a execução do comando UDF.

Bit12, atualizado após a execução do comando STT, é utilizado para monitorizar o sentido de rotação do motor quando a operar no modo de velocidade. O estado lógico 1 indica o sentido de rotação direto.

Bit 11, atualizado após a execução do comando STT, o estado lógico 1 indica o modo de operação em velocidade.

Bit 10, o estado 1 indica que o gerador de trajetória completou o movimento referente ao último comando STT enviado.

Bit 9, o estado lógico 1 indica a execução do comando LPES. Com a execução do comando LPEI é colocado com estado lógico "zero".

Bit 8, o estado lógico 1 indica a execução do comando RESET.

Bit 0, o estado lógico 1 indica a execução do comando SIP mantendo-se ativo até novo impulso de indexação.

- ***Comando RDIP: Read Index Positon***

Este comando lê a posição guardada no registo de indexação. Sempre que o comando SIP é executado, a nova posição de indexação menos a anterior posição de indexação, dividida pela resolução do encoder incremental (nº de linhas vezes quatro), deve ser sempre um número inteiro (C0000000 a 3 FFFFFFF Hex). O comando pode também ser usado para identificar ou verificar qualquer outra posição do motor. Os bytes são lidos do mais significativo para o menos o menos significativo.

- ***Comando RDDT: Read Desired Position***

Este comando lê a posição de referência instantânea naquele intervalo de amostragem de acordo com o perfil de trajetória, ou seja, o valor (C0000000 a 3FFFFFFF hex) utilizado como referência de posição no ponto de soma do sistema de controlo de posição. Os bytes são lidos do mais significativo para o menos significativo.

- ***Comando RDRP: Read Real Position***

Este comando lê a posição atual do motor (C0000000 to 3FFFFFFF Hex). Este é o sinal de retorno de posição fornecido pelo encoder incremental para o ponto de soma do sistema de controlo de posição. Os bytes são lidos do mais significativo para o menos significativo.

- ***Comando RDDV: Read Desired Velocity***

Este comando lê a parte inteira e fracionária da velocidade de referência instantânea, usada para gerar o perfil de velocidade pretendido. Os bytes são lidos do mais significativo para o menos significativo. Os valores lidos são utilizados para realizar a comparação numérica com a velocidade especificada, contudo e porque os dois bytes menos significativos representam a parte fracionária da velocidade, apenas os dois bytes significativos são apropriados para comparação com os dados obtidos via o comando RDRV (Read Real Velocitos dados recebidos osy). De notar também, que apesar da entrada de velocidade ser restringida a números positivos, os dados recebidos a partir do comando RDDV representam uma quantidade especificada em que os números negativos representam movimentos realizados em sentido inverso.

- ***Comando RDRV: Read Real Velocity***

Este comando lê a parte inteira da velocidade não é devolvida uma vez que os valores recebidos derivam de um encoder incremental que apenas fornece valores inteiros. Para comparação com os resultados obtidos com a execução do comando RDDV ou com o valor inserido pelo utilizador, o valor (C000 a 3FFF Hex) devolvido pelo comando RDRV deve ser multiplicado por 2^{16} , ou seja, deslocado 16 posições para a esquerda. Também, os dados recebidos a partir do comando RDRV representam uma quantidade especificada em que os números negativos representam movimentos realizados em sentido inverso.

- ***Comando* RDSUM: Read Integration-Term –Summation Value**

Este comando lê o valor acumulado pelo termo de integração (00000 hex ao valor corrente do limite de integração).