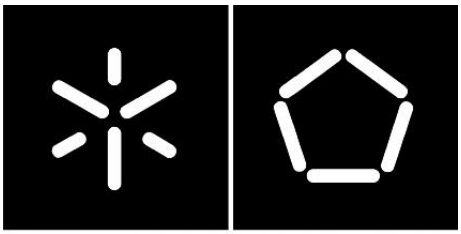


Universidade do Minho

Escola de Engenharia

Departamento de Electrónica Industrial

João Paulo Sousa Silva

Desenvolvimento de sistema de controlo de movimento e percepção de entidades com recurso a visão por computador para equipa de robôs futebolistas

Dissertação submetida à Universidade do Minho para obtenção de grau de Mestre em Engenharia Electrónica Industrial e Computadores

Dissertação realizada sob a orientação científica do Professor Doutor António Fernando Macedo Ribeiro

Outubro de 2011

Resumo

Desde 1997 que a equipa de futebol robótico do Laboratório de Automação e Robótica do Departamento de Electrónica Industrial da Universidade do Minho participa na *Middle Size League* [**1**]. Com o crescente nível de exigência desta liga, surge a necessidade de construir uma nova geração de robôs.

Este trabalho consiste na especificação e desenvolvimento dos seguintes módulos de software: visão, controlo de movimento, e camada de decisão. O objectivo do módulo de visão é fornecer informação sobre o ambiente envolvente de cada robô, como por exemplo a posição de obstáculos e da bola. O módulo de controlo de movimento tem como objectivo determinar quais os comandos a enviar para os actuadores de forma a realizar uma tarefa de jogo, como por exemplo ir atrás da bola ou desviar-se de obstáculos. Por fim, no módulo da camada de decisão pretende-se seleccionar quais as acções que um robô deve realizar em função da informação conhecida.

A extracção de informação do ambiente através do módulo de visão é baseada maioritariamente na cor dos objectos. O controlo do movimento baseia-se numa arquitectura híbrida de comportamentos, em que cada um destes especifica uma acção a realizar no ambiente. O conceito de papel é utilizado no módulo da camada de decisão, e serve para especificar, para uma dada situação, quais os comportamentos a utilizar.

Palavras-Chave: Robótica móvel, *Robocup*, arquitecturas de controlo de robôs móveis, processamento de imagem.

Abstract

The Laboratory Automation and Robotics robotic soccer team participates actively on the *Middle Size League* [**1**] since 1997. This team belongs to the Department of Industrial Electronics from the University of Minho. With the growing level of demand in this league, there was the need to build a new generation of robots

This work describes the solutions and developments of the following software modules: vision, movement control and decision layer. The aim of the vision module consists of providing information of each robot surrounding environment, like the obstacles and ball position. The motion control module is responsible for determining the commands to send to the actuators to accomplish a certain game task, like going after the ball or avoiding obstacles. Finally, the decision layer module is intended to select the actions the robot must perform based on known surrounding information.

The environment information gathering is carried out using a vision module based largely on the objects colours. The motion control is based on a behaviours hybrid architecture, where each behaviour specifies the action to perform on the environment. The role concept is used on the decision layer module, and is used to specify which behaviours to use for a given situation.

Keywords: Mobile robotics, *Robocup*, mobile robots control architectures, image processing.

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao meu pai, Carlos Alberto e à minha mãe, Maria Rosinda por todo carinho e confiança que me proporcionaram durante todo o meu percurso académico.

À minha namorada, Juliana Oliveira pela paciência e compreensão nas horas mais difíceis, e por todo o carinho e amizade dos últimos anos.

Aos meus colegas e amigos de laboratório, Bruno Pereira, João Rodrigues, João Costa, José Vieira, Merylin Santos, Miguel Sousa, Rui Moreira, Rui Pereira, Paulo Rogério e Sérgio Silva, por toda ajuda e companheirismo durante o curso.

Por fim, mas não menos importante, quero prestar os meus sinceros agradecimentos ao meu orientador, Prof. Dr. Fernando Ribeiro e também ao Prof. Dr. Gil Lopes, pela oportunidade de realizar este trabalho, e por todo o apoio e disponibilidade.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

3D	Três Dimensões
API	*Application programming interface*
CAD	*Computer-aided design*
CCD	*Charge coupled device*
CCW	*Conterclockwise*
CPLD	*Complex programmable logic device*
CPU	*Central Processing Unit*
CW	*Clockwise*
FPGA	*Field programmable gate array*
FPS	Frames por segundo
HSL	*Hue Saturation Lightness*
HSV	*Hue Saturation Value*
I&D	Investigação e Desenvolvimento
IMU	*Inertial Measurement Unit*
MBS	MINHO *Basestation*
MSL	*Middle Size League*
NPC	Número píxeis campo
RLA	Relação largura altura
PAL	*Phase Alternate Line*
RAM	*Random Access Memory*
RGB	*Red Green Blue*
RMA	Relação massa altura
SDK	*Software Development Ki*
S.O.	Sistema Operativo

1. Introdução

Nos últimos anos o número de aplicações em robótica móvel tem vindo a aumentar significativamente. Um dos grandes responsáveis desse crescimento é o *Robocup* [**2**], um evento científico a nível mundial que estimula o desenvolvimento de soluções de robótica.

Com o intuito de participar neste tipo de eventos, o Laboratório de Automação e Robótica do Departamento de Electrónica Industrial da Universidade do Minho criou em 1997 a MinhoTeam. Desde então, a equipa tem participado em várias ligas de vários eventos nacionais e internacionais. A aposta principal é na *Middle Size League* (MSL), uma liga de futebol robótico presente neste tipo de eventos.

1.1. Problema

A última geração de robôs futebolistas desenvolvidos até à data do inicio deste trabalho não dispunha de características ao nível das actuais exigências da MSL. Por isso, pretende-se construir uma nova geração de robôs, a qual deve ter em conta a actual dinâmica de jogo. O hardware e software necessários para este projecto são desenvolvidos por cinco alunos do 5º ano do Mestrado Integrado em Engenharia Electrónica Industrial e Computadores da Universidade do Minho.

1.2. Proposta

Pretende-se desenvolver, durante este trabalho, os módulos de software necessários para que os robôs possam interagir com o ambiente. Mais concretamente, software que permita adquirir e processar imagens do sistema visão omnidireccional, gerar trajectórias para o movimento, e determinar quais as acções a realizar.

1.3. Objectivos

Este trabalho está inserido no desenvolvimento dessa nova geração de robôs, e consiste na especificação e desenvolvimento dos seguintes módulos de software: visão, controlo de movimento, e camada de decisão. O objectivo do módulo de visão é fornecer informação sobre o ambiente envolvente de cada robô, como por exemplo a posição de obstáculos e bola. O módulo de controlo de movimento tem como objectivo determinar quais os comandos a enviar para os actuadores de forma a realizar uma tarefa de jogo, como por exemplo ir atrás da bola ou desviar-

se de obstáculos. Por fim, no módulo da camada de decisão pretende-se seleccionar quais as acções que um robô deve realizar em função da informação conhecida. A extracção de informação do ambiente através do módulo de visão é baseada maioritariamente na cor dos objectos. O controlo do movimento baseia-se numa arquitectura híbrida de comportamentos, em que cada um destes especifica uma acção a realizar no ambiente. O conceito de papel é utilizado no módulo da camada de decisão, e serve para especificar, para uma dada situação, quais os comportamentos utilizar.

O sistema deve funcionar a pelo menos 30 ciclos por segundo incluindo todo o processamento.

1.4. Estrutura do Documento

Este documento está dividido em sete capítulos. No capítulo actual é descrito o problema a tratar, e são definidos a proposta e os objectivos deste trabalho. No segundo capítulo, é feita uma análise sobre o estado da arte dos seguintes assuntos: Robótica, Robótica Móvel, *Robocup*, MSL, e MinhoTeam. De seguida, no capítulo três, faz-se uma revisão dos conceitos teóricos necessários para o desenvolvimento do projecto. No capítulo quatro descrevem-se os detalhes de implementação dos vários componentes do sistema desenvolvido, e no quinto capítulo mostram-se os resultados obtidos. O capítulo seis contém a discussão dos resultados, e por fim, no capítulo sete fazem-se as conclusões finais e propõe-se o trabalho futuro.

2. Estado da arte

Este capítulo tem como principal objectivo apurar em que estado se encontra a robótica, para isso é inicialmente feito o estudo etimológico das palavras robô e robótica, depois muito abreviadamente, fala-se do aparecimento do primeiro robô industrial e do impacto socioeconómico adjacente. São posteriormente revistas algumas das vantagens do uso de robôs, e quais as principais áreas de utilização. Numa segunda fase especifica-se esta análise, onde se fala apenas de robôs móveis. Aqui, descrevem-se a características mais importantes dos mesmos, e mostram-se alguns exemplos de aplicação deste tipo de robô. De seguida é feita uma descrição do *Robocup*, das suas diferentes ligas e dos seus principais objectivos. Posto isto, entra-se numa fase mais técnica deste capítulo, onde se analisam algumas das equipas que participam na MSL. Por fim, faz-se uma análise à última versão de robôs da MinhoTeam que participou em eventos oficiais.

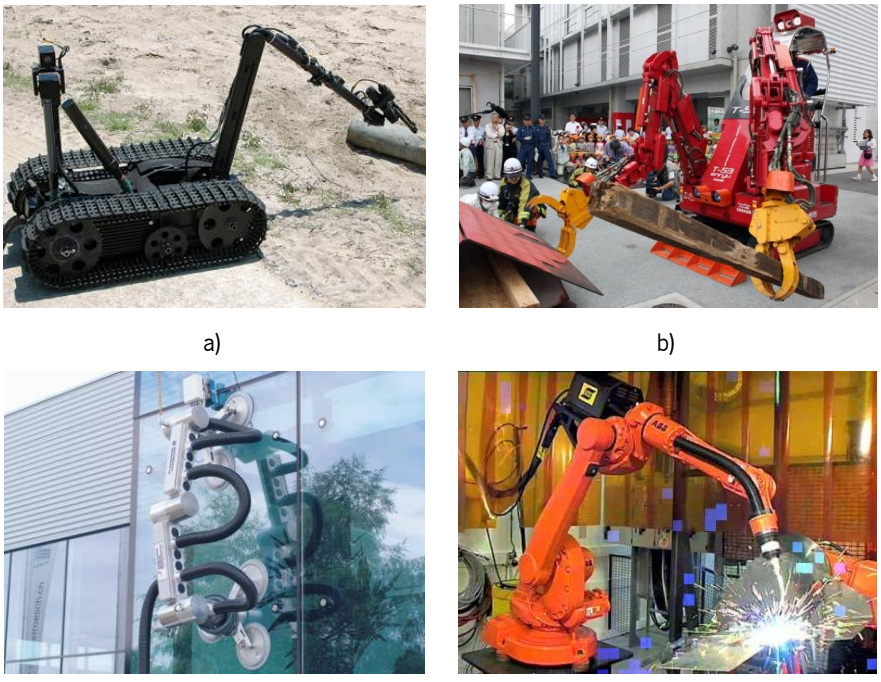
2.1. Robótica

A palavra robô foi pela primeira vez utilizada numa pequena peça do Checo Josef Capek e tornando-se depois popular com a peça Rossum's Universal Robots do seu irmão Karel Capek. Em checo, *robota* tem o significado de trabalho forçado, e a palavra *robotnik* significa escravo. Com isto, surgem as definições que conhecemos nos dias de hoje [**3**] [**4**]. Já a palavra robótica foi utilizada pela primeira vez em Runaround [**5**], uma curta história publicada em 1942 por Isaac Asimov [**6**].

Desde há muito tempo a humanidade procura novas formas de automatizar as suas tarefas. A robótica tem proporcionado enormes avanços nesse sentido tanto em tarefas industriais como em tarefas domésticas. Com o surgimento do primeiro robô industrial em 1961, o UNIMATE (Figura 2.1), e seus sucessores, as linhas de produção industriais ganharam uma nova dinâmica, com isto, a indústria robótica entra numa fase de grande crescimento, criando a necessidade do aparecimento de empresas, cursos de formação, e laboratórios de I&D nesta área, contribuindo assim de forma positiva para um crescimento económico global. A partir dos anos 90 surgem em força aplicações para robôs móveis, correspondendo a mais uma oportunidade de criação de novas empresas e laboratórios de I&D.

Figura 2.1 - Unimate [7]

Para além do desenvolvimento económico são muitas as vantagens que a robótica oferece. Uma das mais importantes é que os robôs podem realizar tarefas que para os humanos são repetitivas, perigosas, ou pouco higiénicas, libertando-os para tarefas mais criativas, e realizam-nas com maior precisão e rapidez. A Figura 2.2 mostra alguns exemplos de robôs a executarem este tipo de tarefas. Outra grande vantagem é a facilidade de reconfiguração de um robô, pois podem ser facilmente reprogramados para efectuarem diferentes tarefas, oferecendo uma elevada flexibilidade.



a) b)

c) d)

Figura 2.2 - Exemplos de robôs: a) Anti-minas [8]; b) Salvamento; c) Limpa vidros [9]; d) Soldador [10]

Como foi dito anteriormente, é possível que um único robô execute diversas tarefas. Essa versatilidade em conjunto com outros factores faz com que os robôs sejam utilizados em enumeras situações. Dependendo do contexto em que se utilizam, podem-se agrupar em conjuntos distintos. A *International Federation of Robotics* começa por dividi-los em dois grandes grupos: robôs industriais e robôs de serviços. Este último é subdividido noutros dois grupos: robôs de serviços para uso profissional e robôs de serviços para uso privado e pessoal [**11**]. Na Tabela 2-1 apresentam-se algumas das principais áreas de aplicação dos robôs.

Tabela 2-1: Tipos de robôs por utilização

Robôs Industriais	Robôs de Serviços	
	Para uso profissional	Para uso privado e pessoal
Automóveis	Segurança	Diversão
Electrodomésticos	Logística	Companhia
Borracha e plásticos	Medicina	Transporte individual
Alimentos	Limpeza	Limpeza
Comunicação	Transportes	Vigilância
Vidro e cerâmica	Construção	
Metal	Turismo	
Semi-condutores		

Pode-se verificar que a robótica está presente nas principais indústrias e serviços, e que todos estes foram fortemente influenciados positivamente pela introdução de robôs a nível de eficácia, rapidez, autonomia, robustez, etc. Na Figura 2.3 pode-se constatar que a população mundial de robôs tem vindo a crescer, o que reflecte o crescente sucesso da introdução de robôs nas várias indústrias e serviços.

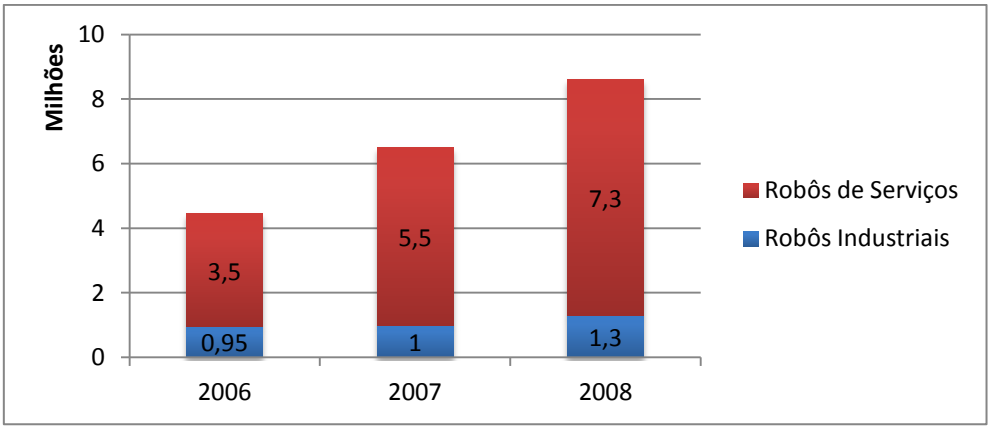


Figura 2.3 - População Mundial de Robôs [12]

2.2. Robótica Móvel

Um subgrupo da robótica é a robótica móvel, que tem como principal característica distintiva o facto de os robôs terem capacidade locomotora, ao contrário de por exemplo um braço manipulador fixo, que tem que ser aparafusado a alguma superfície. Assim sendo, este tipo de robôs não está limitado a um espaço de trabalho reduzido devido à sua capacidade de mobilidade, podendo operar numa maior área.

2.2.1. Tipos de robôs móveis

Os robôs móveis podem ser classificados por várias formas, mas uma das mais utilizadas é a classificação pelo ambiente onde o robô opera, podendo-se então definir três principais classes de robôs móveis: terrestres, aéreos, e aquáticos. Mas este tipo de classificação por si só pode não ser suficiente, sendo então útil classifica-los também pelo tipo de locomoção que utilizam:

Tabela 2-2: Tipos de robôs por locomoção

Tipo de Locomoção				
Rodas	Pernas	Asas ou hélices (aéreas)	Barbatanas ou hélices (marítimas)	Outros
Uma roda Duas rodas Três rodas Quatro rodas Cinco rodas ou mais	Salto (uma perna) Humanóides (duas pernas) Quadrúpedes Outro número de pernas	Duas asas/hélices Quatro asas/hélices Três hélices Outro número de asas ou hélices	Duas barbatanas/hélices Quatro barbatanas/hélices Outro número de barbatanas ou hélices	Lagartas Ventosas

Na Figura 2.4 mostram-se alguns robôs móveis dos vários tipos existentes. É de notar que para além dos listados acima ainda existem mais tipos de robôs, alguns deles sendo combinações dos aqui referidos.

a) Rovio [**13**]

b) Idris [**14**]

c) Ampbot [**15**]

d) DraganFlyer X4 [**16**]

e) Fire Scout [**17**]

f) Scorpion [**18**]

g) Trimares [**19**]

h) Roaz 2 [**20**]

i) Robot fish [**21**]

Figura 2.4 - Exemplos de Robôs Móveis

Outro aspecto de classificação que é comum utilizar é o nível de autonomia de um robô. Aqui, podemos dividir os robôs em três grupos:

- **Controlados Remotamente** – Todos as acções são feitas pelo robô são controladas pelo operador;
- **Semi-autónomos** – O operador dá instruções de alto nível e o robô executa todas as tarefas intermediárias autonomamente. Por exemplo, o operador indica um local para onde o robô se deve deslocar, e o robô terá que se mover nessa direcção evitando ao mesmo tempo obstáculos sem mais intervenção humana.
- **Autónomos** – Aqui todas as acções são decididas pelo próprio robô em função dos dados extraídos do ambiente em que se encontra.

2.2.2. Interacção com o ambiente

Para que um robô possa interagir com o seu meio envolvente, necessita de obter dados sobre o mesmo. Este procedimento dá pelo nome de fase aquisição, onde os dados são capturados através dos sensores do robô e armazenados na sua memória. Depois, é necessário transformar estes dados em informação útil para o robô, para isso e muito mais temos a fase de processamento que é realizada pela unidade de processamento do robô. Por fim, o robô usa essa informação para modificar o ambiente onde se encontra, utilizando os seus actuadores. A Figura 2.5 ilustra este processo.

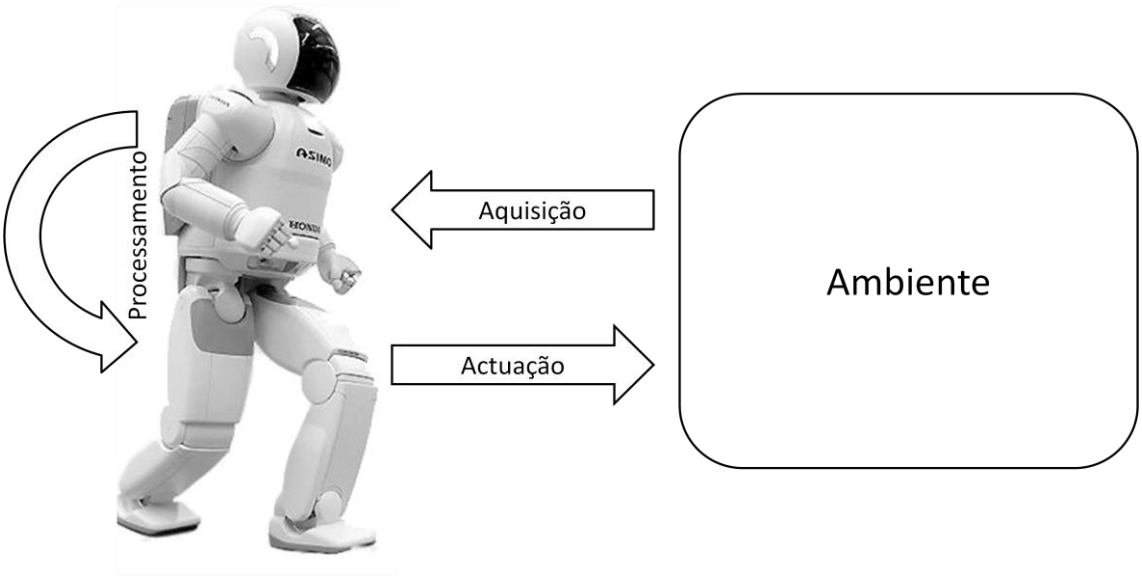


Figura 2.5 - Ciclo de Interacção com o Ambiente

Como foi referido anteriormente a fase de aquisição de dados é feita recorrendo aos sensores do robô, podendo estes serem internos ou externos. Os sensores internos são responsáveis pela medida de grandezas físicas no interior do robô, como por exemplo a velocidade das rodas, ou a temperatura de algum componente. Já os sensores externos obtêm dados do meio envolvente, como por exemplo distâncias a objectos, intensidade luminosa, cor de um objecto, etc [**22**][**23**]. Os sensores podem ainda ser classificados como passivos ou activos. Os sensores passivos medem grandezas físicas do ambiente em que se encontram sem emitirem energia, já os sensores activos emitem uma certa quantidade de energia e medem a reacção do ambiente a esse estímulo. Na tabela 2.2 [**23**] mostram-se alguns dos sensores mais utilizados em robôs móveis e a sua classificação e na Figura 2.6 está um exemplo dos sensores que um robô pode utilizar.

Tabela 2-3: Tipos de Sensores na Robótica

Grupo	Sensor	Interno/Externo	Activo/Passivo
Sensores tácteis (Detecção de contacto físico ou proximidade)	Interruptores de contacto	Externo	Passivo
	Barreiras ópticas	Externo	Activo
Sensores de rodas/motores (velocidade e posição)	Encoders de escova	Interno	Passivo
	Potenciómetros	Interno	Passivo
	Encoders ópticos	Interno	Activo
	Encoders magnéticos	Interno	Activo
	Encoders indutivos	Interno	Activo
	Encoders capacitivos	Interno	Activo
Sensores de orientação (orientação em relação a um referencial fixo)	Compasso/bússola	Externo	Passivo
	Giroscópio	Interno	Passivo
	Inclinómetro	Externo	Activo/Passivo
Alcance dinâmico (reflexão, tempo de voo, e triangulação geométrica)	GPS	Externo	Activo
	Reflectivos	Externo	Activo
	Ultra-sons	Externo	Activo
	Lasers	Externo	Activo
	Triangulação óptica	Externo	Activo
	Luz estruturada	Externo	Activo
Movimento/Velocidade (velocidade relativa a objectos fixos ou em movimento)	Efeito de *Doppler* electromagnético	Externo	Activo
	Efeito de *Doppler* sonoro	Externo	Activo
Sensores de visão (analise de imagem, segmentação, reconhecimento de objectos)	Câmaras CCD/CMOS	Externo	Passivo
	Módulos de detecção de objectos (câmara + software)	Externo	Passivo

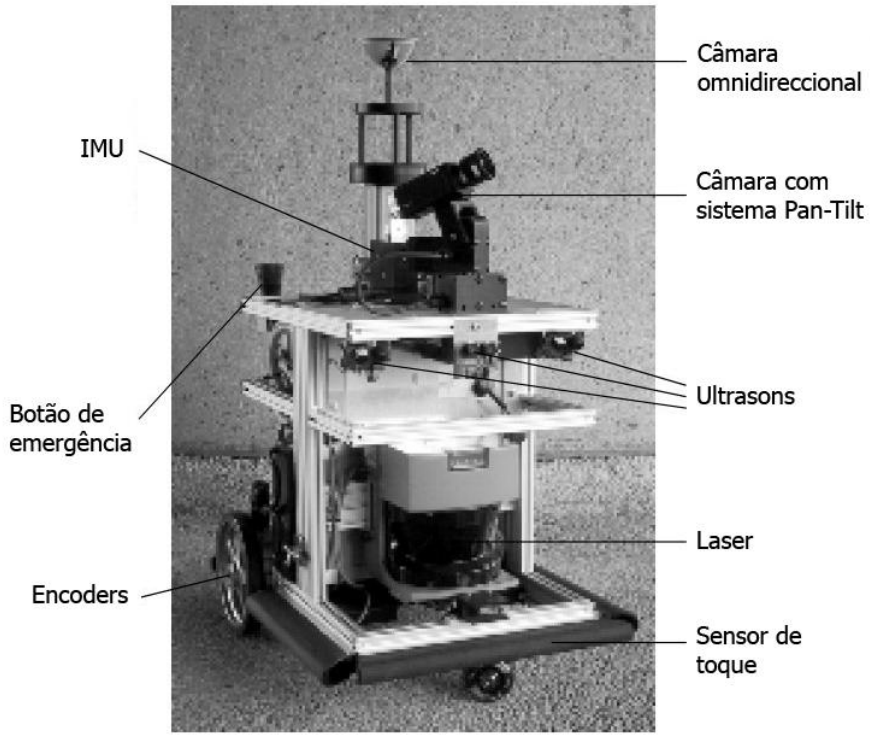


Figura 2.6 - Robô Biba, BlueBotics SA [23**]**

As principais características que se devem ter em conta na escolha de qualquer sensor são as presentes na tabela seguinte:

Tabela 2-4: Pricipais características de um sensor [22**]**

Característica
Sensibilidade
Linearidade
Gama de funcionamento
Tempo de resposta
Precisão
Exactidão
Resolução
Tipo de saída
Histerese
Limite e zona morta

Na fase de processamento são várias as tarefas que a unidade de processamento poderá ter que realizar. A complexidade dessas tarefas varia com os tipos de arquitecturas usadas (Figura 2.7).

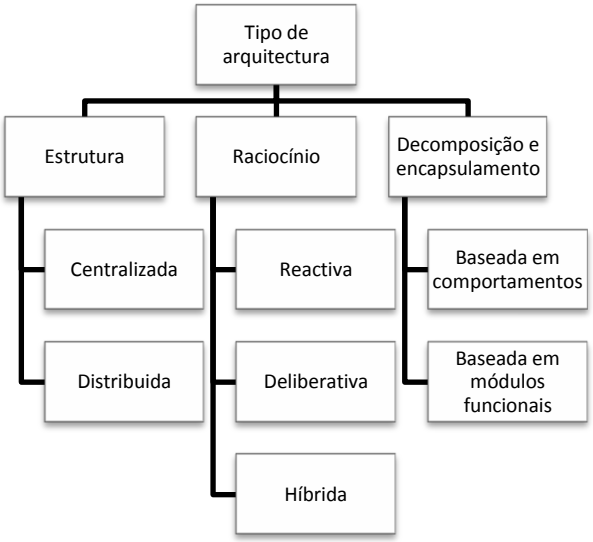


Figura 2.7 - Tipos de arquitecturas utilizadas em robótica móvel [22]

Durante esta fase são resolvidos inúmeros problemas inerentes à robótica móvel. Os mais importantes são o tratamento dos dados provenientes dos sensores, localização, cinemática e/ou dinâmica, modelação do ambiente, e navegação.

Depois dos dados serem adquiridos pelos sensores do robô é necessário transformá-los em informação útil. Aqui entra a unidade de processamento, em que esta trata os dados para que o sistema os possa utilizar para realizar as tarefas que lhe foram atribuídas. Por exemplo, um sensor CCD de uma câmara digital pode não devolver os dados no formato de imagem utilizada no posterior processamento, então é necessário converter esses dados para que contenham informação pronta a utilizar.

Tanto a cinemática como a dinâmica estudam o movimento de um robô, no entanto a cinemática apenas toma em conta as relações geométricas do sistema. Já a dinâmica considera todas as forças associadas ao movimento. O problema da cinemática pode ser dividido em duas partes, directa e inversa. No primeiro caso o problema consiste em determinar a posição do *end-effector*[1] do robô, conhecendo-se a disposição de todas as juntas do robô. O problema da cinemática inversa consiste no contrário, ou seja, dada uma posição desejada do *end-effector* determinar qual o conjunto de soluções da disposição geométrica das juntas.

A modelação do ambiente pode ser feita de diferentes formas e com diferentes níveis de detalhe. Por exemplo num robô seguidor de linha básico, apenas a posição e ângulo relativos entre o robô e a linha a seguir são necessários modelar. Por outro lado no caso de um robô que

[1] *End-effector:* Parte do robô que interage com o ambiente.

navegue livremente numa estrada pública terá que modelar objectos em 3D reconstruindo o ambiente em seu redor de forma a evitar colisões, respeitar sinais de trânsito, etc.

Para que a navegação de um robô móvel num determinado ambiente seja eficaz é necessário saber a sua localização global ou relativa a algum objecto, e em várias situações também será necessário um mapa do ambiente. O mapa pode ser estático ou dinâmico, em que no primeiro caso é carrega-lo para a memória do robô e supõe-se que todos os objectos no ambiente não se mexem, já no caso de um mapa dinâmico o robô actualiza o mapa com a informação proveniente dos seus sensores. A forma de representar os mapas também pode variar, como se pode ver na Figura 2.8.

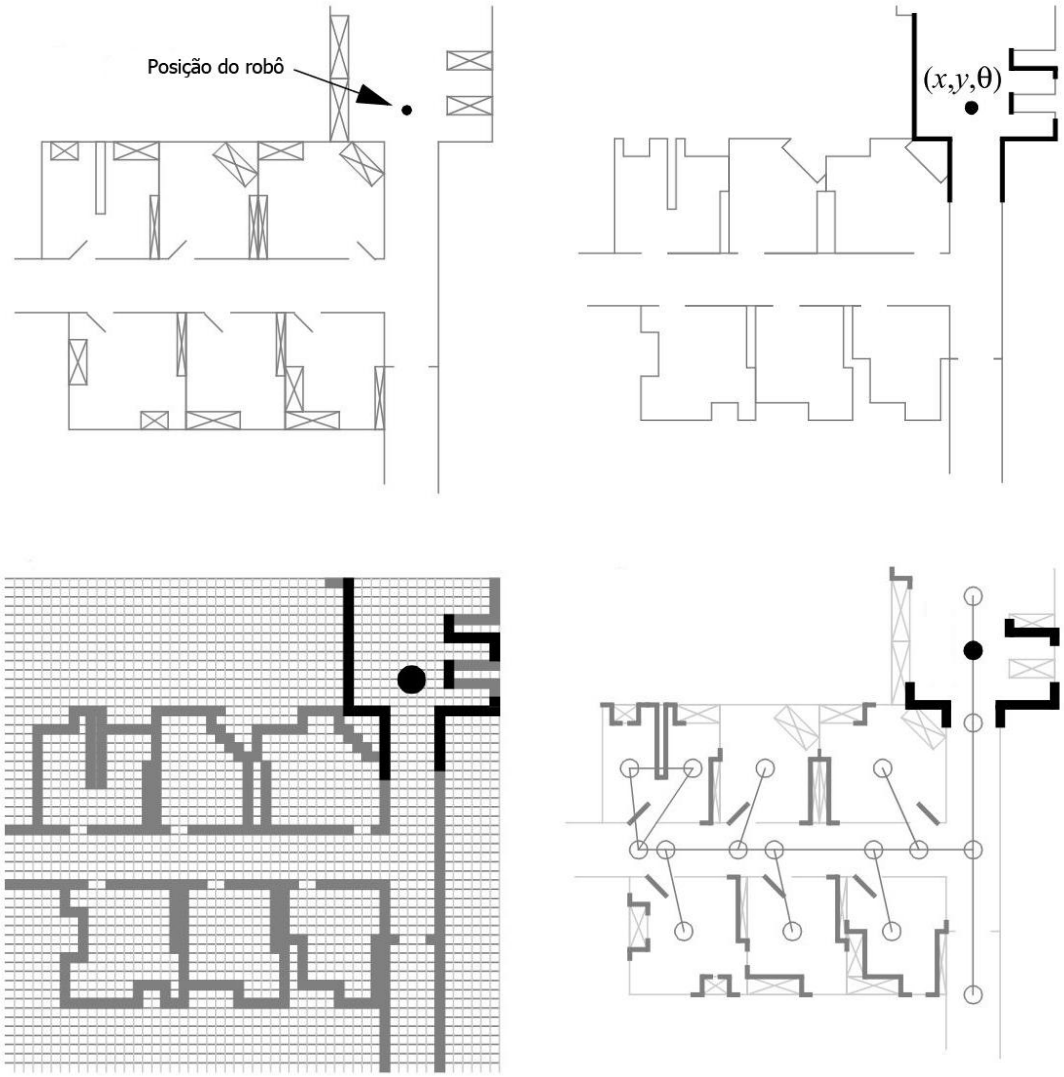


Figura 2.8 - Exemplos de representação de mapas [23]

Por fim, na fase de actuação o robô tem que actuar no meio onde se encontra, e para isso envia comandos para os seus actuadores. Dependendo do tipo de energia que utilizam para funcionar são classificados de formas diferentes, como se pode ver na tabela seguinte.

Tabela 2-5: Classificação dos actuadores segundo o tipo de energia utilizada [22]

Tipo de energia utilizada		
Eléctrica	Hidráulica	Pneumática
Motores síncronos Motores de indução Motores DC Motores de passo Servomotores	Actuadores hidráulicos	Cilindros pneumáticos Motores pneumáticos de aletas rotativas Motores pneumáticos de pistão axial

Segundo [**22**], os accionamentos pneumáticos utilizam ar comprimido e os hidráulicos utilizam por exemplo água ou óleo. Estes dois tipos de accionamento necessitam de mais manutenção que os actuadores eléctricos, fazendo com que estes últimos sejam mais utilizados em robótica. Na encontra-se uma comparação entre os vários tipos de actuadores utilizados em robótica.

Tabela 2-6: Tabela comparativa dos actuadores utilizados na robótica[22]

	Pneumático	Hidráulico	Eléctrico
Energia	Ar comprimido (5 -10 bar)	Óleo mineral (50 – 100 bar)	Corrente eléctrica
Opções	Cilindros Motor de aletas Motor de pistão	Cilindros Motor de aletas Motor de pistão axial	Corrente contínua Corrente alternada Motor de passo
Vantagens	Baratos Rápidos Sensíveis Robustos	Rápidos Alta relação peso -potência Auto-lubrificantes Alta capacidade de carga Estabilidade frente a cargas estáticas	Precisos Confiáveis Fácil controlo Fácil instalação Silenciosos
Desvantagens	Dificuldade de controlo contínuo Instalação especial (filtros, compressores, etc.) Ruidoso	Difícil manutenção Instalação especial (filtros, eliminadores de ar) Frequentes fugas Caros	Potência limitada

Pode-se verificar que os actuadores eléctricos são os que têm menos desvantagens. Isto em conjunto com a sua elevada precisão, fiabilidade, facilidade de instalação, e facilidade de controlo faz com que estes sejam uma boa solução para a maioria dos problemas relacionados com robótica.

2.3. Robocup

Esta secção tem como objectivo descrever o *Robocup*, uma iniciativa científica internacional com o objectivo de estimular o desenvolvimento de robôs inteligentes. O objectivo inicial, aquando da sua criação em 1997, era construir uma equipa de robôs futebolistas capazes de vencer a equipa de humanos campeã do mundo em 2050. Entretanto, os objectivos foram expandidos a outras necessidades da sociedade moderna. Actualmente está dividido em quatro principais temas [**2**]:

- RoboCupSoccer
- RoboCupRescue
- RoboCup@Home
- RoboCupJunior

Cada um destes temas está por sua vez dividido em várias ligas, em que cada uma tem os seus diferentes objectivos.

2.3.1. RoboCupSoccer

Aqui o futebol robótico é o principal desafio. Cada uma das suas ligas aborda o mesmo problema mas de maneiras diferentes. Enquanto umas concentram-se apenas no desenvolvimento de software (*Simulation e Standard Platform*), outras estimulam também o desenvolvimento de componentes electrónicos e mecânicos (*Humanoid, Middle Size, e Small Size*). Tem como principais desafios a modelação do ambiente e a localização, comunicação, e trabalho de equipa de robôs completamente autónomos.

Humanoid League

Nesta liga os robôs têm a forma aproximada de um humano, ou seja, tem cabeça, tronco, e membros. Não existe um formato padrão para os robôs. A Figura 2.9 mostra exemplos deste tipo de robôs.

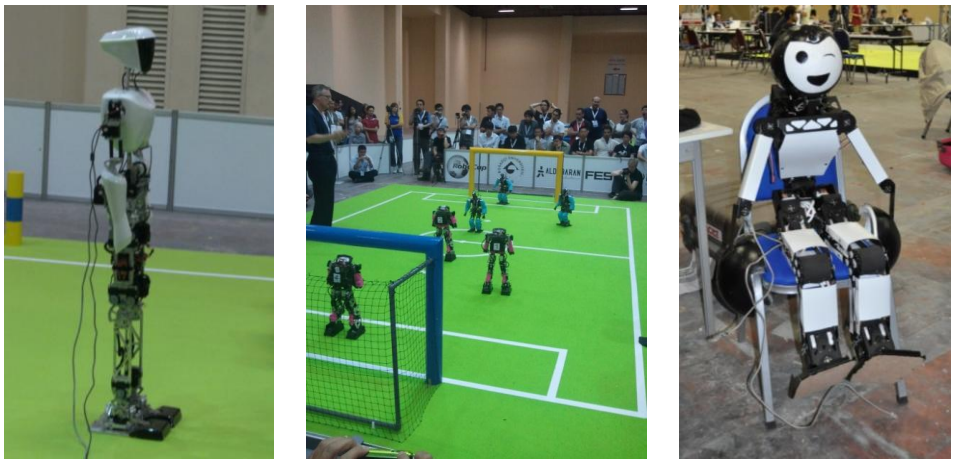
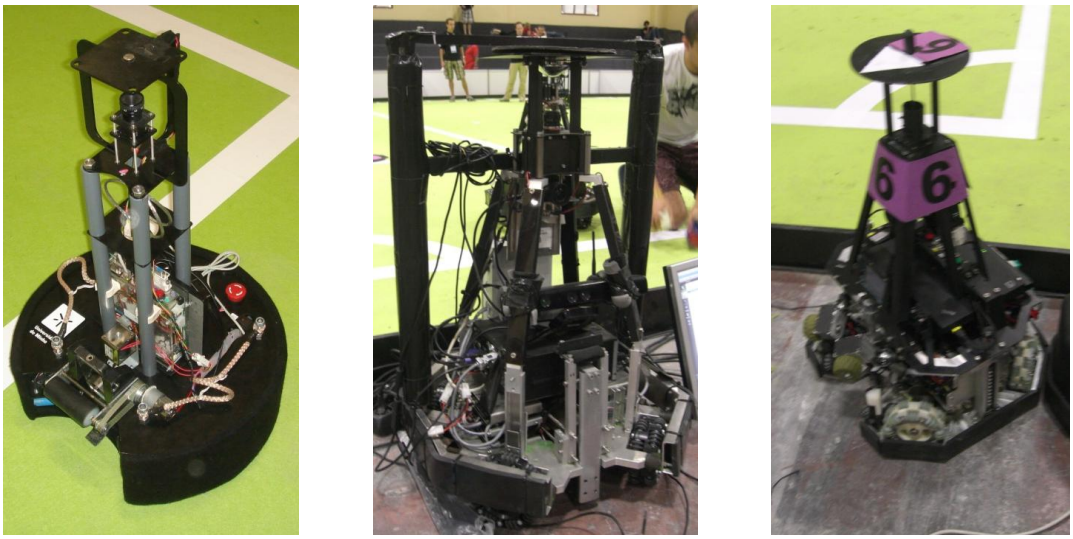


Figura 2.9 - Exemplos de robôs da *Humanoid League*

Um dos principais desafios desta liga é manter o equilíbrio dos robôs, pois a sua estrutura torna esta tarefa bastante difícil. Para além de simplesmente se equilibrarem têm também que andar, chutar, etc.

Middle Size League

Esta liga é constituída por equipas de até cinco robôs cada. Não há sensores exteriores, e cada robô é completamente autónomo. Actualmente utilizam-se robôs com duas, três, ou quatro rodas. Existe partilha de informação entre os elementos de equipa em tempo real, a qual proporciona o trabalho cooperativo entre robôs. A latência dos sensores tem que ser reduzida, pois existem objectos de elevado interesse (bola, adversários) que se deslocam com velocidades de alguns metros por segundo. Na Figura 2.10 estão alguns robôs desta liga.



a) b) c)

Figura 2.10 - Exemplos de robôs da MSL: a) Minho; b) RFC Stuttgart; c) MRL

Para além das elevadas velocidades acima referidas, existem outros importantes desafios nesta liga, como por exemplo a elevada dimensão do campo (comparativamente a outras ligas) onde decorrem os jogos, e o facto da bola utilizada ser do tamanha oficial 5 da FIFA e poder ter qualquer cor.

Simulation League

Esta liga existe em duas modalidades: 2D ou 3D. Foca-se principalmente na inteligência artificial e estratégia de equipa, e os seus jogadores virtuais jogam futebol num campo também virtual [**2**].

a) b)

Figura 2.11 – *Simulation League*: a) *2D;* b) *3D* [24]

A Figura 2.11 mostra os ambientes da *simulation league*2D e 3D. A grande diferença entre as duas é como o próprio nome sugere uma dimensão espacial. Enquanto na *simulation league* 2D trata-se o problema em duas dimensões, na outra trata-se o mesmo problema em três dimensões, o que lhe acrescenta alguma realidade

Small Size League

A *small size league* tem contornos idênticos aos da *middle size league*, com a principal diferença de que nesta as imagens são adquiridas por câmaras que se encontram por cima do campo de jogo, a um elevado *frame rate*[2]. Isto e o peso reduzido dos robôs, permite jogar com grande velocidade.

[2] Número de imagens capturadas por segundo. Normalmente expressa-se em fps ou Hz.

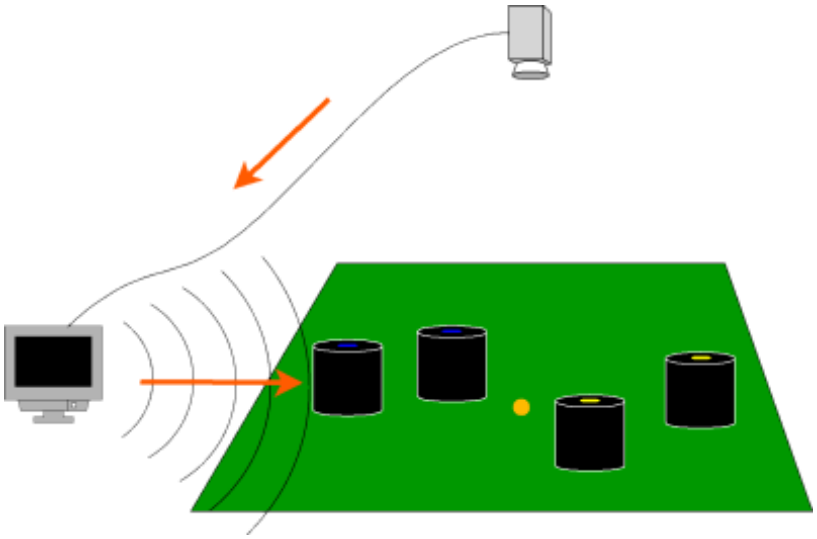
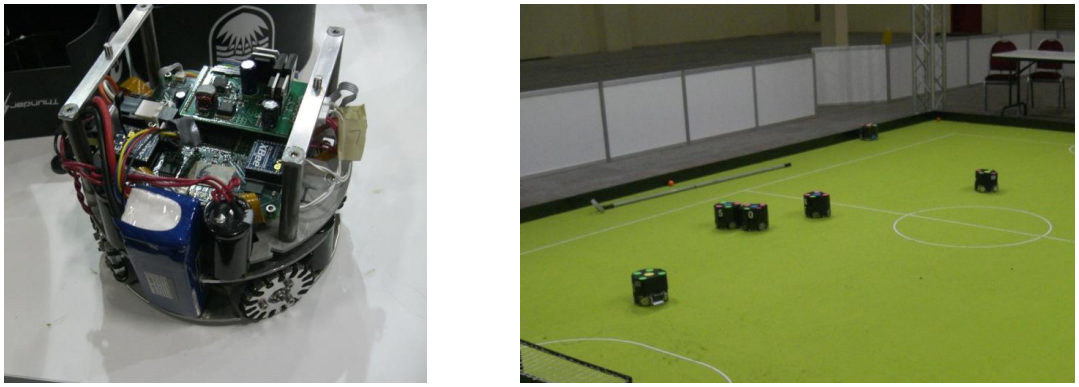


Figura 2.12 - Sistema de Visão (simplificado) da liga *Small Size* [24]

No lado direito da Figura 2.13 podem-se ver vários robôs desta liga, nos quais existem umas marcas coloridas que servem para identificar e posteriormente localizar os robôs no campo através das imagens adquiridas pelas câmaras posicionadas no topo do campo. Do lado esquerdo pode-se ver com maior detalhe um dos robôs utilizados nesta liga.



a) b)

Figura 2.13 - Robôs da *Small Size League*: a) British colombia university; b) Vários

Standard Platform League

Esta liga é semelhante à *Humanoid League*, mas nesta as equipas usam uma plataforma standard. Actualmente a plataforma utilizada é o humanóide Nao (Figura 2.14) da Aldebaran Robotics. O facto de as equipas não terem de desenvolver o hardware permite-lhes dedicarem-se unicamente ao desenvolvimento de software.

Figura 2.14: Nao da Aldebaran Robotics [25**]**

Os jogos desta liga são de quatro contra quatro robôs, em que em cada equipa há um guarda-redes e três jogadores de campo.

Na Tabela 2-7 resumem-se as principais características físicas de cada uma destas ligas (excepto a de simulação por ser virtual). Aqui mostram-se as medidas de cada campo, qual o tipo de bola que cada liga utiliza, o tempo de jogo, e o número e dimensões máximas dos robôs.

Tabela 2-7: Comparação das principais características físicas das ligas do *RoboCupSoccer*

Liga		Campo			Bola	Duração de cada parte (minutos)	Robôs			
		L (mm)	C (mm)	Cor			D (mm)	A (mm)	P (kg)	#
Humanoid	Kid	4000	6000	Verde	Ténis, tamanho standard, laranja	10		60		3
	Teen	6000	9000		FIFA, tamanho 3, laranja			120	20	2
	Adult	4000	6000		FIFA, tamanho 5, laranja			180		1
Middle Size		12000	18000	Verde	FIFA, tamanho 5	15	520	800	40	5
Small Size		4050	6050	Verde	Golfe, tamanho standard, laranja	10	180	150		
Standard Platform		4000	6000	Verde	Hóquei de rua, tamanho standard, laranja	10		570	5	4
L – Largura; C – Comprimento; D – Diâmetro; A – Altura; P – Peso; # - Número de jogadores de uma equipa (Valores máximos)										

2.3.2. RoboCup Rescue

O *RoboCup Rescue* tem como principal objectivo promover o desenvolvimento de robôs para busca e salvamento. Em situações de catástrofe é essencial uma rápida intervenção para ajudar as vítimas, assim sendo, o uso de robôs para auxiliar nas tarefas associadas a este tipo de situação é uma mais-valia. Este tipo de robôs têm de estar dotados de várias capacidades como por exemplo fazer o mapeamento do ambiente em causa, planear o melhor trajecto a seguir conforme a situação, detectar vítimas no terreno, etc.

Robot League

Nesta liga os robôs têm de ser capazes de andar numa zona de catástrofe a procurar vítimas, determinar as suas condições e comunicar com as mesmas, construir um mapa do local, instalar sensores para monitorizar substâncias prejudiciais, entregar fluidos e medicamentos, e marcar os melhores caminhos para as vítimas. O ambiente usado na competição (Figura 2.15) é o de um prédio parcialmente destruído por um tremor de terra [**24**].

Figura 2.15 - Ambiente de busca e salvamento

A maioria dos robôs utilizados neste tipo de usam lagartas como tipo de locomoção para facilitar o acesso a áreas mais danificadas, e uma percentagem mais pequena dos robôs utilizam rodas normais. Na Figura 2.16 a) e b) apresenta-se dois exemplares de robôs usados nesta liga.

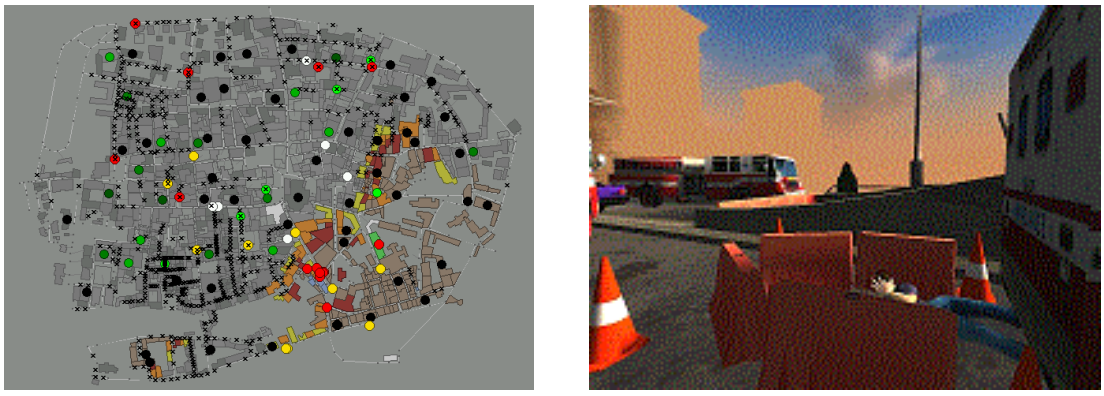
a) b) c)

Figura 2.16 - Robôs de busca e salvamento: a) Com lagartas b) com rodas; C) Estação do operador

A prova tem partes em que os robôs são completamente autónomos e outras em que são semi-autónomos. O robô tem de enviar informações para o exterior, neste caso para uma estação de monitorização (Figura 2.16 c)) em que um operador visualiza essa informação e pode tomar decisões nas partes em que o robô está no modo semi-autónomo.

Rescue Simulation League

Esta liga é semelhante à anterior com a principal diferença de ser virtual. Ela própria está dividida em duas competições: *agent e virtual robot*. Para além de todos os desafios apresentados na anterior liga, aqui a construção dos ambientes virtuais é um desafio acrescido. O facto de não ser necessário ter um robô real para efectuar a prova é uma grande vantagem em relação à anterior. A figura seguinte mostra algumas cenas destas provas.



a) b)

Figura 2.17 - a) Competição *Agent*; b) Competição *Virtual robot competition*

A competição *agent* tem dois principais objectivos: estimular o desenvolvimento de simuladores e mapas para este tipo de situações, e o desenvolvimento de agentes capazes de ajudar em cenas de desastre. Já na competição *virtual robot* é usado um simulador standard onde os utilizadores podem testar vários robôs com vários tipos de sensores e actuadores de uma forma muito realista, aproximando-se assim dos robôs reais [**26**].

2.3.3. Robocup@Home

@Home League

O principal objectivo desta liga é o desenvolvimento de tecnologia para robôs de serviços para uso profissional e pessoal. O tipo de problemas que os robôs desta liga têm de resolver aumenta de dificuldade de ano para ano. Os critérios para a avaliação das provas são os seguintes [**27**]:

- Incluir interacção homem máquina
- Relevância social
- Orientado à aplicação
- Cientificamente desafiador
- Barato e fácil montagem
- Simples e com regras fáceis
- Interessante de ver
- Demorar pouco tempo

O ambiente em que os robôs operam é uma casa simplificada, que com o tempo tem tendência a ganhar maior complexidade. A Figura 2.18 ilustra um destes cenários. Como se pode ver é um ambiente com uma estrutura semelhante ao de uma habitação, onde se encontram cadeiras, mesas, armários, etc. Este facto permite que estes robôs possam operar na maioria das habitações actuais, facilitando a sua aplicação ao dia-a-dia.

Figura 2.18 - Cenário de uma prova do *Robocup@home* [28]

2.3.4. RoboCupJunior

Junior

A liga *Junior* está dividida em três outras ligas:

- *Dance:* Equipas de um ou mais robôs vestidos a rigor têm que fazer coreografias ao som da música (Figura 2.19 a)).
- *Soccer:* Equipas de dois robôs jogam futebol com uma bola com emissores de infravermelhos, num campo fechado (Figura 2.19 b)).
- *Rescue:* Aqui os robôs têm que seguir uma linha marcada no chão plano, identificar vítimas, e evitar obstáculos (Figura 2.19 c)).

a) b) c)

Figura 2.19 - Provas *Junior* do *Robocup*: a) *Dance*; b) *Soccer*; c) *Rescue* [29]

Nesta secção foram brevemente analisadas todas as ligas do *Robocup*, na próxima secção é feita uma análise mais detalhada da MSL, pois é nela que este trabalho se insere.

2.4. Middle Size League

2.4.1. A competição

Esta liga existe desde 1997, e desde então são inúmeros os avanços alcançados, dos quais um dos mais importantes foi a remoção das cores das balizas em 2008. Até então uma baliza era amarela e outra azul, e a maioria dos robôs utilizava essa cor para saber onde a baliza do adversário estava. Outro dos avanços que se foi sentindo progressivamente foi o alongamento das dimensões do campo de jogo, que actualmente tem 18 metros de comprimento e 12 de largura. Inicialmente utilizava-se para todos os jogos uma bola cor-de-laranja lisa, mas com o passar do tempo desenvolveram-se sistemas de visão capazes de detectar bolas de uma cor ou mais cores variadas.

Os jogos têm duas partes de 15 minutos cada, com um intervalo máximo de 5 minutos entre elas. As equipas podem jogar com cinco jogadores no máximo, em que um destes é

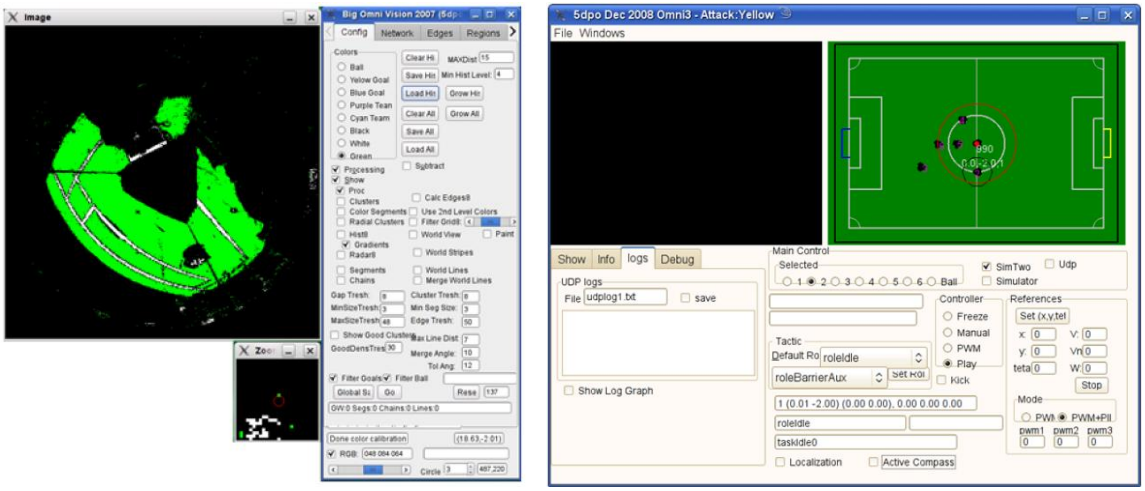
guarda-redes. Existe ainda um computador exterior ao campo de jogo por equipa, que é responsável por receber os comandos do árbitro e enviá-los para os robôs, poderá também ter a estratégia da equipa, tratar a informação proveniente de todos os elementos da equipa, etc.

2.4.2. Casos de estudo

Nesta secção é feita uma análise aos sistemas de visão e aos sistemas de controlo de alto nível de algumas das equipas que participam activamente na MSL.

5DPO

Esta é a equipa da Faculdade de Engenharia da Universidade do Porto, Portugal, participa na MSL desde 2001, e ficou no 2º lugar no Festival Nacional de Robótica em 2009 e 2010. O módulo de visão utilizado dá pelo nome de HAL (*Hardware Abstraction Layer*), é responsável por encontrar as linhas do campo, a bola e os obstáculos através das imagens capturadas pelo sistema de visão omnidireccional (Figura 2.20 a)). A extracção de informação baseia-se principalmente na cor dos objectos [**30**].



a) b)

Figura 2.20 - a) Módulo de visão da equipa 5DPO; b) Software Dec [30]

Cada robô tem um módulo de software chamado Dec [**30**] que é responsável por tomar as principais decisões do sistema baseando-se na informação disponível, e é constituído por três camadas: Papeis, Tarefas, e Acções. Na primeira camada, é atribuída a função que cada robô deve desempenhar, como por exemplo defesa ou guarda-redes. Na segunda, decide-se quais as tarefas que deve desempenhar em função da função atribuída. Por fim, as acções tornam possível a execução das tarefas.

CAMBADA

Começou o seu desenvolvimento em 2003, faz parte do Instituto de Engenharia Electrónica e Telemática da Universidade de Aveiro, Portugal, e foi campeã mundial em 2008. O sistema de visão utilizado é omnidireccional, e o guarda-redes utiliza também uma câmara de perspectiva para detecção da bola em 3D. O software de visão (Figura 2.21) está dividido em três módulos: *Utility Sub-System, Color Processing Sub-System,* e *Morphological Processing Sub-System*. O *Utility Sub-System* serve por exemplo para adquirir e mostrar imagens. *O Color Processing Sub-System* é responsável pela classificação de cores e detecção de objectos por cor. Por fim, o *Morphological Processing Sub-System* é utilizado para detectar a bola, independentemente da sua cor.

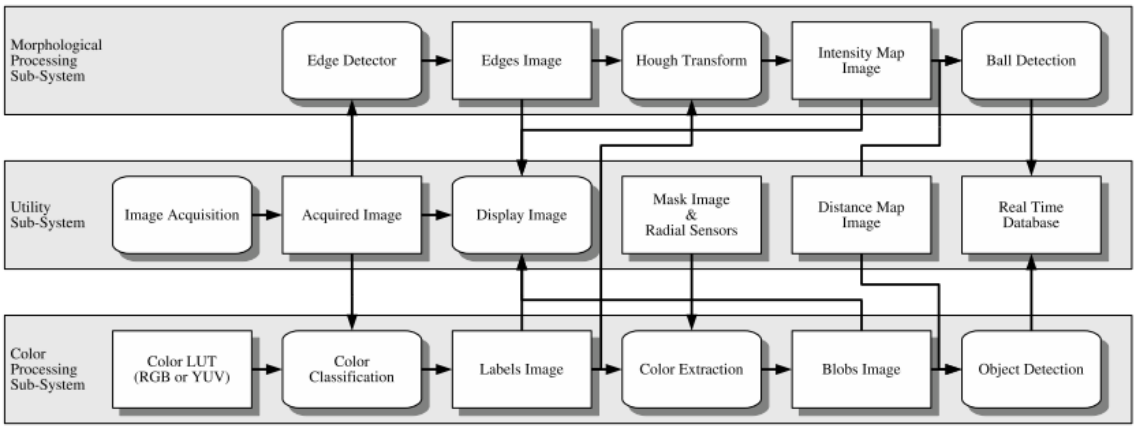


Figura 2.21 - Arquitectura do software de visão da equipa CAMBADA [31**]**

Para controlo e coordenação de alto nível utilizam três módulos de software: *Sensor Fusion*, *Basic Behaviors*, e *High-Level Decision and Cooperation*. Em que o primeiro é responsável por adquirir informação dos sensores e dos outros jogadores de equipa, para actualizar uma base de dados de tempo real. O segundo módulo disponibiliza um conjunto de primitivas que o *High-Level Decision and Cooperation* utiliza para controlar o robô [**32**].

MRL

A equipa MRL começou o seu trabalho em 2003 no Laboratório de Investigação de Mecatrónica da Universidade de Azad, Irão. Obteve o 4º lugar nas edições de 2009 e 2010 do *Robocup*. O seu sistema de visão é omnidireccional, com excepção do guarda-redes que também tem uma câmara de perspectiva para obter visão estéreo. A detecção de objectos é baseada principalmente na sua cor, a definição das classes de cor é feita recorrendo ao espaço HSL e o resultado guardado numa *Lookup Table*. A detecção das linhas do campo é feita através de uma pesquisa radial (Figura 2.22).

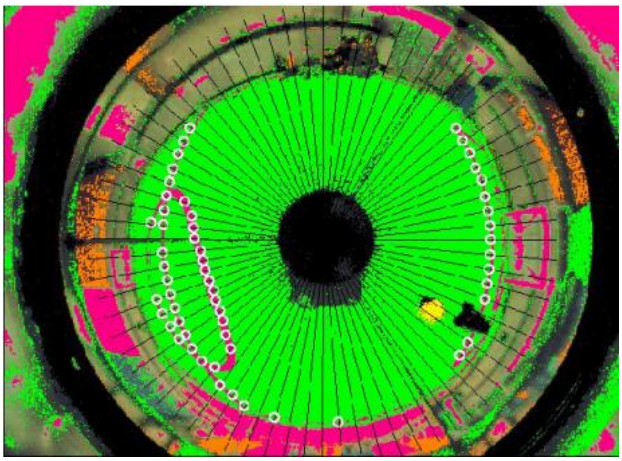


Figura 2.22 - Imagem da pesquisa radial - equipa MRL [33]

O controlo de alto nível é separado em três unidades principais: *Knowledge Unit, Planning Unit,* e *Executing Unit* (Figura 2.23). A tarefa da *Knowledge Unit* é a aquisição e processamento de dados provenientes dos vários sensores. O principal objectivo da *Planning Unit* é tomar decisões de alto nível, como por exemplo seguir a bola, e enviar essas decisões (*commands*) para a *Executing Unit*. Por fim, a *Executing Unit* define quais os comportamentos básicos a executar em função dos *commands* recebidos.

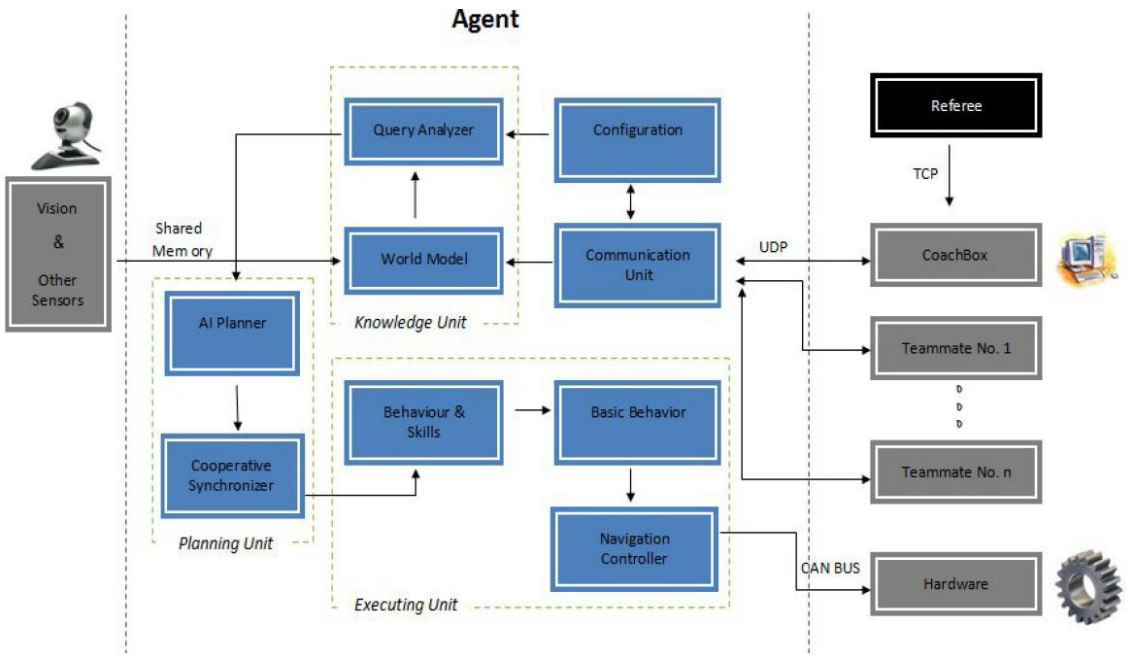
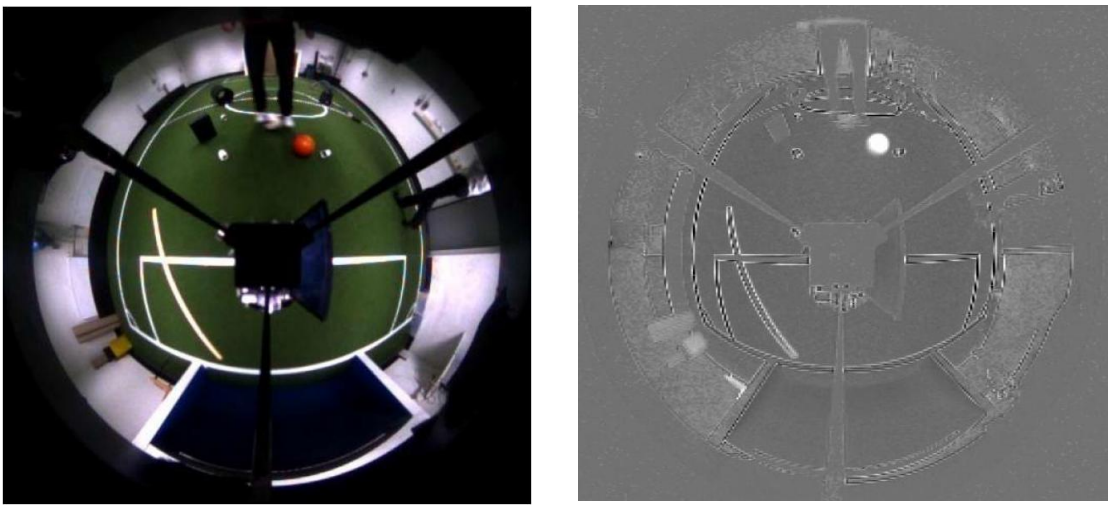


Figura 2.23 - Diagrama de blocos do controlador de alto nível da equipa MRL [33]

Carpe Noctem

Esta equipa faz parte do Departamento de Sistemas Distribuídos da Universidade de Kassel, Alemanha. Da mesma forma que os últimos exemplos, o seu sistema de visão é omnidireccional, com excepção do guarda-redes que também tem uma câmara de perspectiva para obter visão estéreo. O espaço de cores utilizado é o YUV, e a detecção de objectos é feita recorrendo ao canal Y (tons de cinza), com excepção da bola, que se baseia no plano UV (Figura 2.24). De seguida é aplicado um algoritmo de *template matching* para determinar qual das regiões de interesse é a bola.



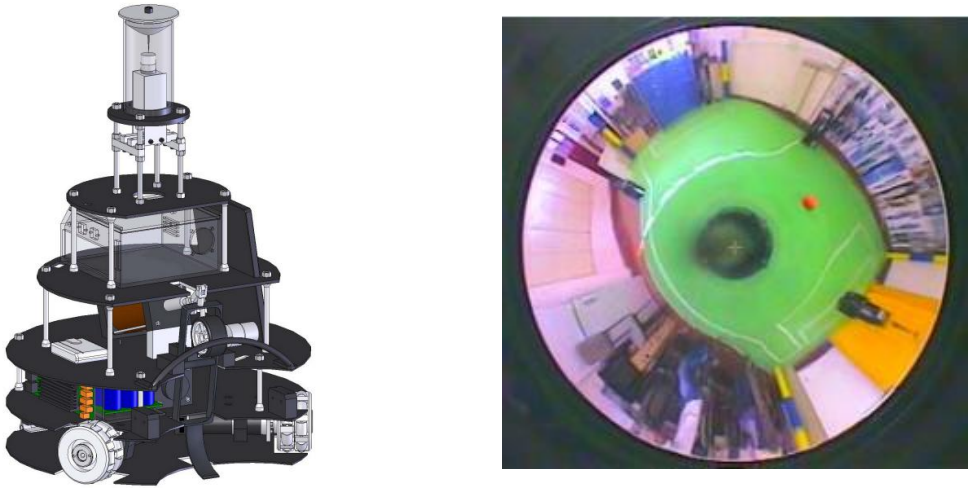
(a) (b)

Figura 2.24 - a) Imagem original; b) Imagem do plano UV [34]

A equipa criou uma linguagem chamada ALICA [**35**] para modelar o comportamento dos seus robôs. O seu software é organizado em vários módulos, em que se destacam os seguintes: *Vision* e *Base*. O módulo *Vision* é responsável por detectar a bola e os obstáculos, e determinar a posição do robô no campo de jogo. O módulo *Base* gere os comportamentos, faz a modelação do espaço de trabalho, e planeia trajectórias.

2.5. MinhoTeam

A MinhoTeam é a equipa de futebol robótico do Laboratório de Automação e Robótica do Departamento de Electrónica Industrial da Universidade do Minho. Foi criada em 1997, e participou intensivamente na MSL até 2007. A última versão de robôs (Figura 2.25 a)) foi criada em 2006.



a) b)

Figura 2.25 - Última versão de robôs da MinhoTeam: a) Desenho CAD da estrutura mecânica [36]; b) Imagem do sistema de visão omnidireccional [37]

Era utilizado um sistema de visão omnidireccional (Figura 2.25 b)), e o espaço RGB para classificar as cores. A detecção de objectos era feita através de histogramas verticais de cor (Figura 2.26).

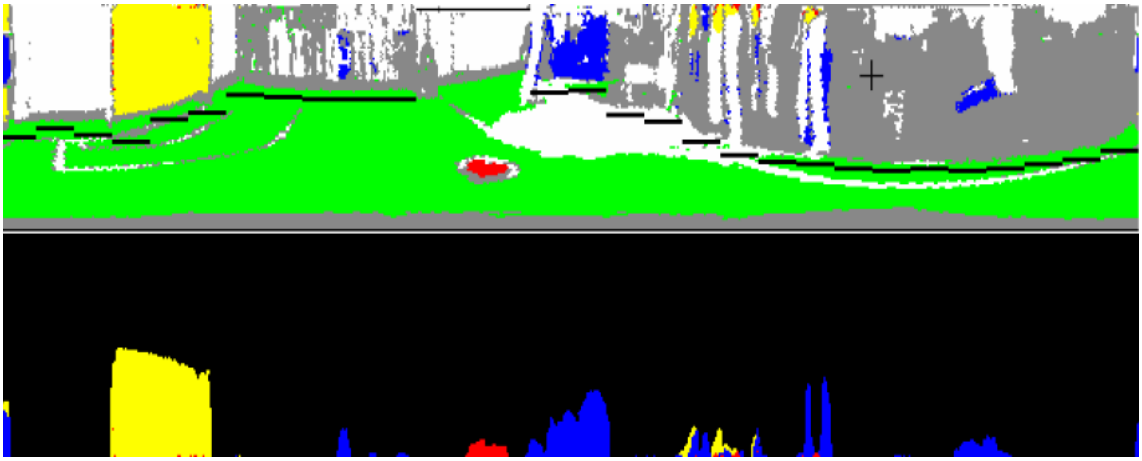


Figura 2.26 - Histogramas verticais de cor [37]

Existiam quatro módulos de software principais: Cores, Hardware, Jogo e Monitor. A função do primeiro era permitir a calibração das cores e da câmara. O módulo Hardware servia para testar e configurar o hardware do robô. O Jogo e o Monitor eram as partes activas durante o tempo de jogo, em que o primeiro processava os dados disponibilizados pelos sensores e executava acções baseadas nas decisões por ele tomadas, e o segundo fazia o interface entre o RefBox e os robôs. A geração da maioria do movimento era realizada utilizando sistemas dinâmicos não lineares.

3. Conceitos Teóricos

Neste capítulo são apresentados a maioria dos conceitos teóricos necessários para a realização deste trabalho.

3.1. Espaços de cor

Um espaço de cor é um sistema para representar a cor numericamente [**38**]. De seguida são revistos os espaços de cores utilizados neste trabalho.

3.1.1. RGB

O espaço de cores RGB é composto por três componentes, as cores primárias: vermelho, verde, e azul. A cor é obtida através da mistura das três cores primárias. As várias combinações resultantes podem ser representadas num cubo tridimensional (Figura 3.1).

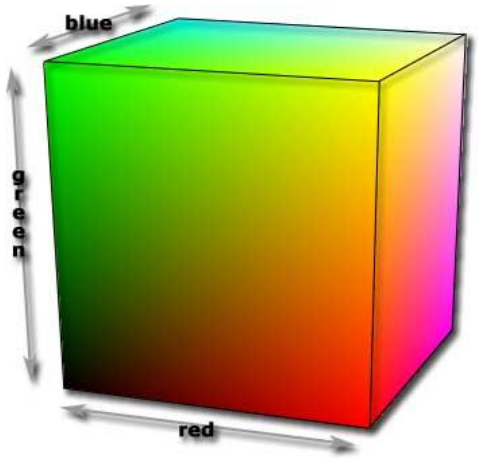


Figura 3.1 - Cubo RGB [39]

É normalmente utilizado em dispositivos electrónicos como computadores, placas gráficas, e monitores [**39**].

3.1.2. HSV

O espaço de cores HSV é também composto por três componentes: *Hue, Saturation, e Value.* Aqui, o *Hue* define o tom da cor, e abrange todas as cores possíveis. Normalmente é representado como um ângulo que varia de 0 a 360 graus. Já a *Saturation* define a quantidade de cinzento presente na cor. Costuma ser representada de 0 a 100% e quanto maior for este valor menor é o nível de cinza da cor. Por fim, o *Value* define o brilho da cor, e depende da

Saturation. Também varia de 0 a 100%, e no caso de ser zero a cor é totalmente preta [**40**]. A Figura 3.2 mostra uma das formas de representação deste espaço de cores.

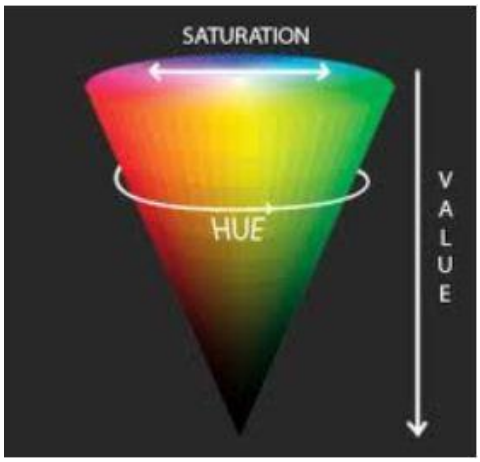


Figura 3.2 - Cone HSV [40]

Uma grande vantagem do HSV em relação a outros espaços de cores é que é bastante parecido com a forma que humanos percebem as cores. Isto permite que uma cor seja definida por um humano com relativa facilidade comparando por exemplo com o espaço de cores RGB [**40**].

3.1.3. YUV

O espaço de cores YUV é definido em termos de luminância e crominância, em que o primeiro é constituído pelo canal Y e define o brilho da cor, já a crominância é constituída pelos canais U e V que definem o tom da cor. Este espaço de cor é utilizado nos sistemas televisivos PAL analógicos. A Figura 3.3 mostra um exemplo um plano U-V para um dados Y.

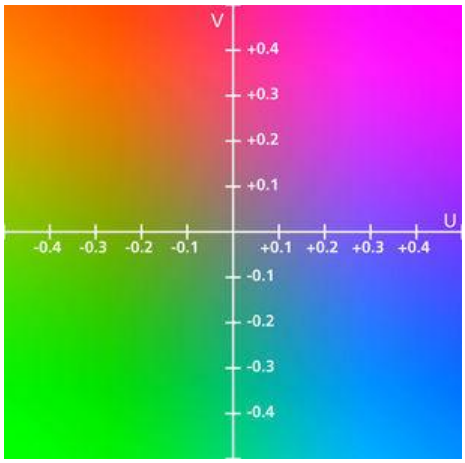


Figura 3.3 - Plano U-V para Y=0.5 [41]

A vantagem deste espaço de cor em relação a outros, é o facto de conter toda a informação do brilho da imagem num só canal.

3.2. Qualidade de imagem

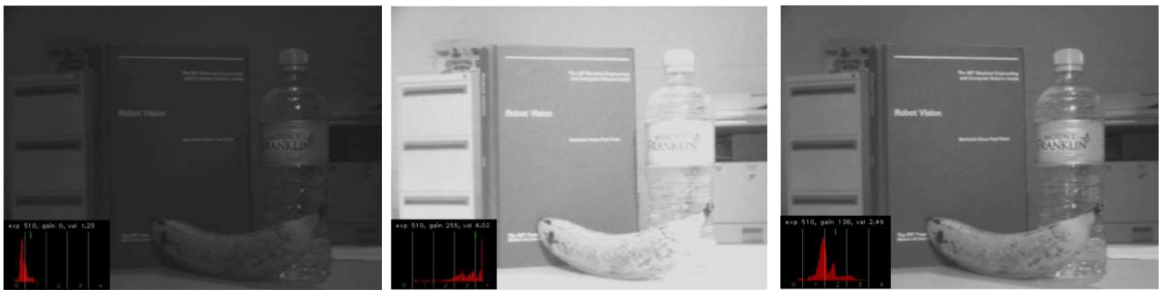
A qualidade de uma imagem é muito importante em aplicações de visão artificial, e são vários os factores de qualidade existentes. Uma imagem com boa qualidade deverá preservar todos os detalhes da cena original. Neste trabalho são abordados alguns dos factores de qualidade de imagem mais importantes: exposição, balanço dos brancos, e brilho. Cada um deles está associado a algumas medidas que se podem fazer através do estudo estatístico da população de píxeis presentes na imagem.

3.2.1. Exposição

Através do estudo da exposição de uma imagem pode-se determinar se esta está sobreexposta, subexposta, ou correctamente exposta. Uma ferramenta bastante utilizada para análise da exposição é o histograma de intensidades da imagem, a partir do qual se podem extrair medidas para definir o nível de exposição de uma imagem. Dividindo o histograma de intensidades em cinco regiões pode-se calcular o *mean sample value* (MSV), o qual permite determinar o balanço da distribuição de tonalidades presentes numa imagem:

$$\boldsymbol{MSV} = \frac{\sum_{j=0}^{4}(\boldsymbol{j}+\mathbf{1})\boldsymbol{x_j}}{\sum_{j=0}^{4}\boldsymbol{x_j}} \tag{3.1}$$

Em que j representa uma região do histograma de intensidades e x_j representa a soma de todos os valores de uma dada região. Pode-se então dizer que uma imagem está correctamente exposta quando MSV≈2,5 [**42**]. Na Figura 3.4 mostram-se alguns exemplos da mesma cena com diferentes níveis de exposição.



a) b) c)

Figura 3.4 - Imagens com diferentes níveis de exposição e correspondentes histogramas de intensidades a) Imagem subexposta MSV=1.23; b) Imagem sobreexposta MSV=4.02; c)Imagem correctamente exposta MSV=2.49 [42**]**

3.2.2. Balanço dos brancos

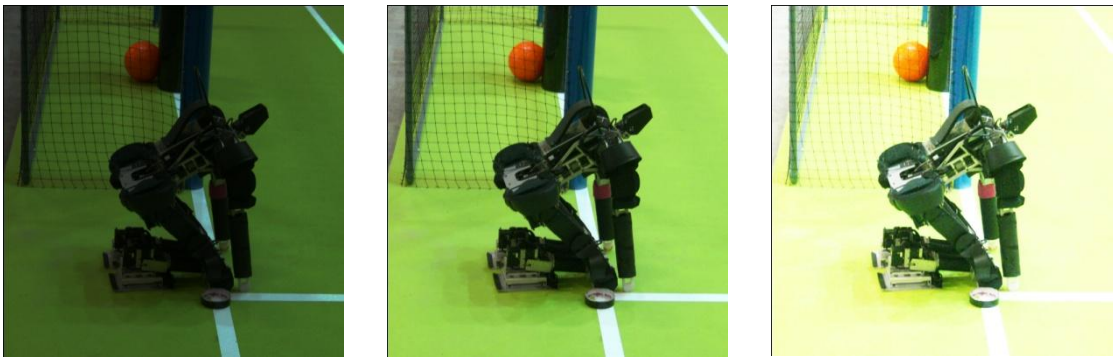
O balanço dos brancos, ou do inglês *white balance*, permite determinar a tonalidade dos brancos numa imagem. Dependendo da fonte luminosa a temperatura da cor[3] é diferente, fazendo com que em certas condições objectos que na realidade são brancos apareçam na imagem com outras cores [**43**]. A Figura 3.5 mostra a mesma cena com diferentes valores de balanço dos brancos.

a) b)

Figura 3.5 - a)Balanço dos brancos incorrecto; b)Balanço dos brancos correcto [43]

3.2.3. Brilho

O brilho de uma imagem permite determinar se esta está demasiadamente clara ou escura, ou se tem um brilho correcto. Neste caso, a intensidade da fonte luminosa utilizada para capturar a imagem influência directamente este parâmetro. Na Figura 3.6 mostra-se a mesma cena com vários valores de brilho.



a) b) c)

Figura 3.6 - Imagem com diferentes brilhos a) Baixo brilho; b) Brilho correcto; c) Alto brilho

[3] A temperatura da cor descreve o espectro de luz que é radiado por um objecto que absorve toda a luz incidente. [**43**]

3.3. Segmentação de imagem

Processo que tem como objectivo dividir uma imagem em regiões (conjuntos de píxeis), separando os objectos de interesse do fundo da imagem. Os objectos distinguem-se do fundo da imagem devido a características específicas da aplicação em questão. As principais características utilizadas em processamento de imagem são:

- Intensidade
- Cor
- Forma
- Textura
- Movimento (vídeos)
- Disparidade (imagem estéreo)

Estas podem se divididas em duas classes: características primárias e características secundárias, em que as primeiras são dadas directamente pelo sensor (p. ex: intensidade, cor), e as segundas têm de ser calculadas a partir das primárias (p. ex: texturas, disparidade estéreo).

3.3.1. Threshold

Uma das técnicas utilizadas para segmentar uma imagem é o uso de um *threshold*. Dada uma determinada característica (p. ex: cor, intensidade), e uma imagem original $f(x,y)$, temos [**44**]:

$$g(x,y) = \begin{cases} 1\ para\ f(x,y) \geq threshold \\ 0\ para\ f(x,y) < threshold \end{cases} \quad (3.2)$$

Esta é uma técnica bastante simples, de baixo custo computacional, e fácil de implementar. Por outro lado, o valor ideal para o *threshold* é difícil de encontrar (Figura 3.7), e o seu desempenho diminui em imagens com baixo contraste entre o fundo e os objectos de interesse.

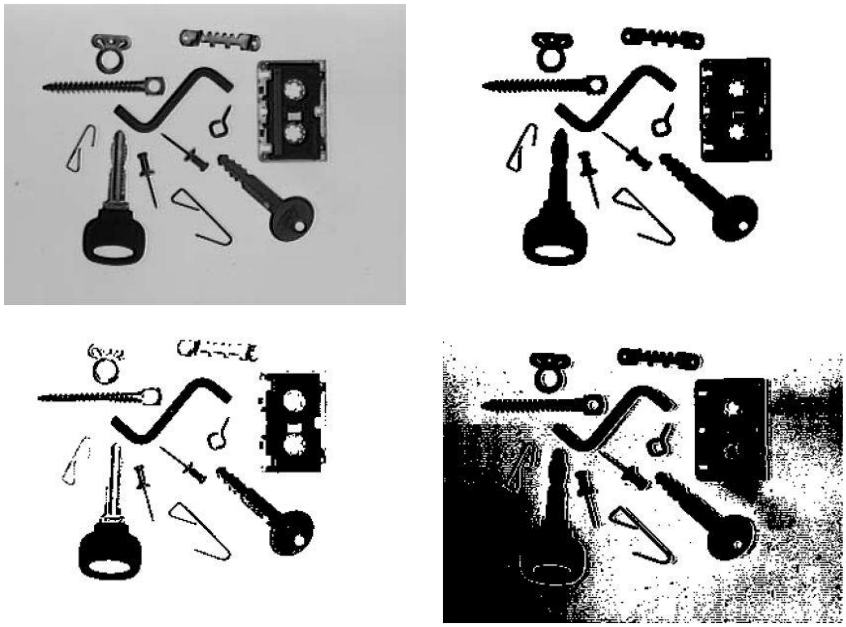


Figura 3.7 - Exemplos de segmentação de imagens por intensidade usando *threshold* simples [44]

3.4. Identificação de componentes ligados baseada em matrizes

A identificação de componentes ligados é uma ferramenta que é utilizada em imagens binárias para identificar grupos de píxeis ligados que possivelmente representem objectos de interesse. Idealmente as imagens binárias contêm a informação relevante no seu primeiro plano, e tudo o resto fica no plano de fundo. Então, o processo de identificação de componentes ligados consiste basicamente em atribuir etiquetas aos píxeis do primeiro plano, para que no final do processo cada uma destas etiquetas represente um objecto diferente na imagem. Agora, a noção de conexão entre píxeis depende do tipo de vizinhança que se está a usar.

3.4.1. Tipos mais comuns de vizinhança entre píxeis

Neste caso só se utilizam vizinhanças baseadas em quadrados. A Figura 3.8 mostra os tipos de vizinhança mais utilizados desta classe [**45**].

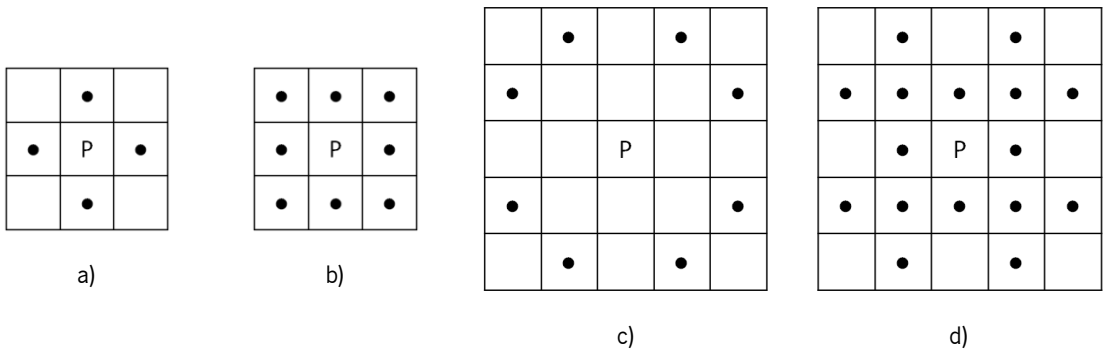


Figura 3.8 - Tipos de ligações entre píxeis: a) 4-Neighbourhood; b) 8-Neighbourhood; c) knight-Neighbourhood; d) 16-Neighbourhood

3.4.2. Algoritmo Clássico

Dada uma imagem binária, pode-se definir o seguinte:

- p – Um pixel da imagem;
- F - O conjunto de píxeis de primeiro plano;
- F^C – O conjunto de píxeis do plano de fundo;
- $Label(p)$ – Matriz do mesmo tamanho que a imagem que guarda as etiquetas de todos os píxeis;
- $Label(q)$ – Matriz que representa as etiquetas dos píxeis vizinhos ao pixel em análise;
- λ – Contador de etiquetas.

Inicializar $Label(p)$ e λ:

$$\begin{cases} Label(p) = +\infty, \forall p \in F \\ Label(p) = 0, \forall p \in F^C \end{cases}$$

$$\lambda = 1$$

Percorrer toda a imagem de forma a preencher $Label(p)$:

Para cada p:

Se $p \in F$:

Se $\min\big(Label(q)\big) \neq +\infty$ e $\min\big(Label(q)\big) > 0$

$Label(p) = \min\left(Label(q)\right)$

Senão:

$Label(p) = \lambda$

$\lambda = \lambda + 1$

Senão:

Ir para próximo p

Repetir o processo até não haver alterações em $Label(p)$.

A Figura 3.9 ilustra este processo no caso particular de uma implementação sequencial com vizinhança do tipo *8-neighbourhood*.

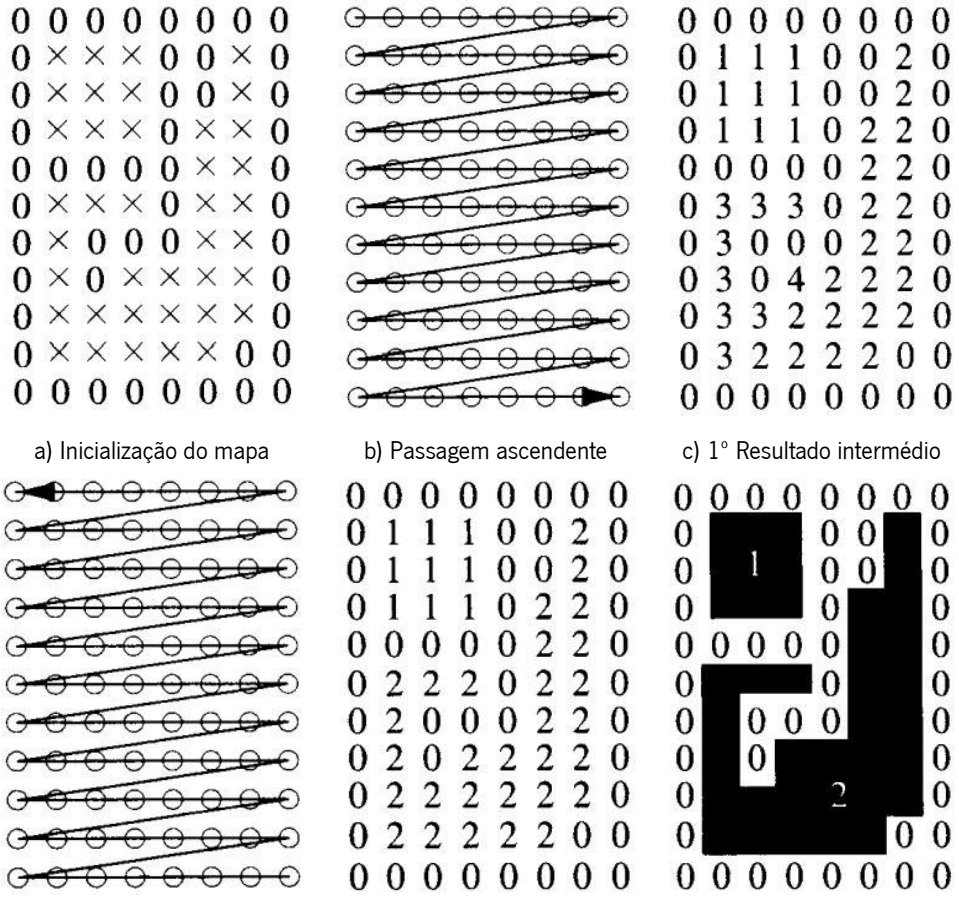


Figura 3.9 - Atribuição sequencial de etiquetas a componentes ligados [45]

Na Figura 3.9 a) pode-se ver o resultado da inicialização de $Label(p)$, onde as cruzes representam infinito. De seguida a imagem é percorrida no sentido crescente (Figura 3.9 b)) e é obtida uma representação temporária de $Label(p)$ visível Figura 3.9 c). Depois repete-se o mesmo processo mas no sentido inverso (Figura 3.9 d)), e $Label(p)$ fica com o aspecto da Figura 3.9 e). Todo este processo terá que ser repetido até que não haja alterações em $Label(p)$. O aspecto final deste exemplo pode-se ver na Figura 3.9 f), em que claramente se detectam dois objectos distintos.

3.5. Operadores Morfológicos Binários

Os operadores morfológicos, quando aplicados a imagens digitais, alteram a forma dos objectos. Estes permitem realizar filtragem e operações geométricas recorrendo apenas funções lógicas.

3.5.1. Elementos estruturantes

As operações morfológicas envolvem pelo menos dois conjuntos. No caso do processamento de imagem esses dois conjuntos são a própria imagem a processar e um elemento estruturante (EE), em que este último é normalmente representado por uma matriz. A forma do elemento estruturante usada tem bastante influência sobre o resultado final da operação. Na Figura 3.10 mostram-se alguns exemplos populares de elementos estruturantes.

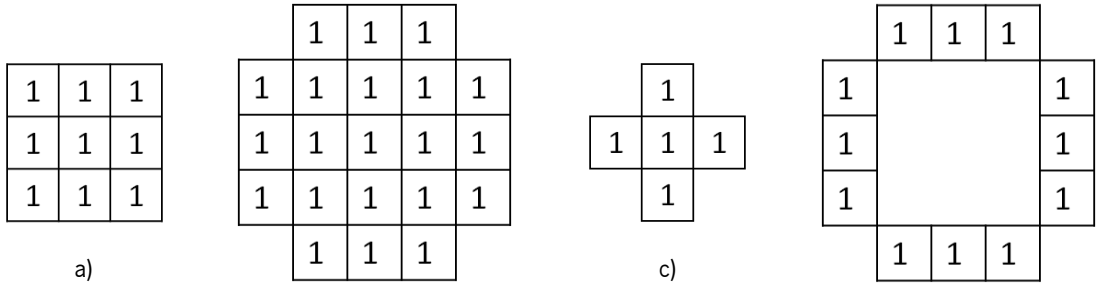


Figura 3.10 - Exemplos de elementos estruturantes: a)Caixa(3,3); b)Disco(5); c)Cruz(3,3); d)Anel(5)

Dada uma imagem binária pode-se então definir os seguintes tipos básicos de operadores morfológicos.

3.5.2. Erode

Este operador encolhe os objectos presentes no plano da imagem a que é aplicado. Como resultado, alguns dos objectos desaparecem devido ao seu reduzido tamanho, e por vezes um só objecto é separado em vários. Matematicamente pode ser definido como a subtracção de Minkowski ($\ominus$). Então, a erosão de uma imagem binária pode ser definida por [**46**]:

$$F \ominus B = \bigcap_{y \in B} F_{-y} \tag{3.3}$$

Em que Brepresenta o EE, F o primeiro plano da imagem, e y o vector de translação. A operação realizada consiste em passar o ES por todos os píxeis da imagem, e sempre que a origem deste coincida com um pixel do primeiro plano é realizado um AND lógico entre o EE e os píxeis da imagem coincidentes.

3.5.3. Dilate

Este operador expande os objectos presentes no plano da imagem a que é aplicado. Como resultado, nos objectos de interesse os contornos são suavizados e alguns buracos tapados. Por vezes, dois ou mais objectos são unidos num só. Matematicamente pode ser definido como a adição de Minkowski ($\oplus$). Desta forma, a dilatação de uma imagem binária pode ser definida por [**46**]:

$$F \oplus B = \bigcup_{y \in B} F_{+y} \tag{3.4}$$

Aqui a operação a realizar é semelhante à anterior, com a diferença que, em vez de um AND lógico usa-se um OR lógico.

A partir da combinação em cascata destes dois operadores básicos podem-se criar os seguintes operadores:

3.5.4. Open

Este operador pode ser definido da seguinte forma [**46**]:

$$F \circ B = (F \ominus B) \oplus B \tag{3.5}$$

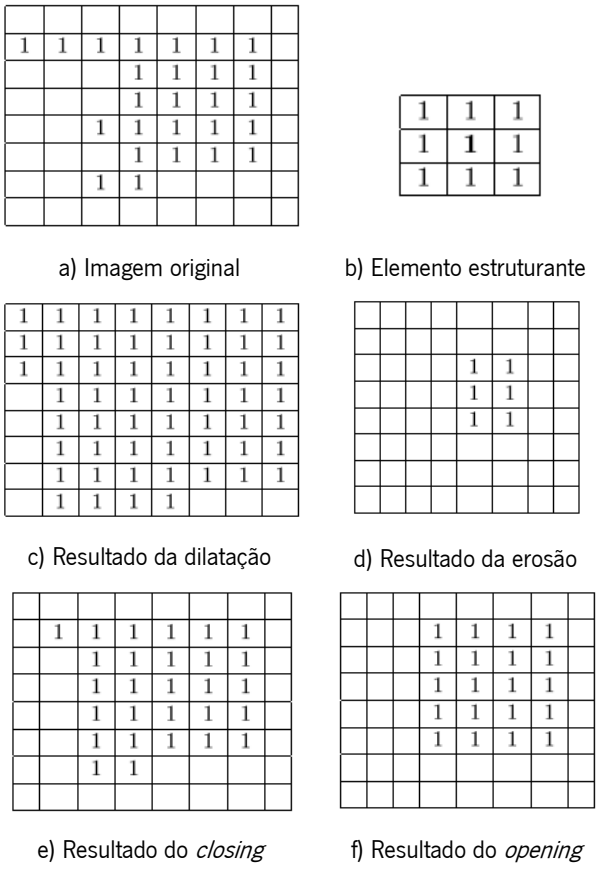
Ou seja, consiste em fazer uma erosão seguida de uma dilatação. É utilizado para remover conjuntos de píxeis que são menores que o EE, remover ligações entre dois ou mais objectos, ou seja, tem efeitos semelhantes à erosão mas preserva tamanho dos objectos.

3.5.5. Close

Este operador pode ser definido da seguinte forma [**46**]:

$$F \bullet B = (F \oplus B) \ominus B \tag{3.6}$$

Ou seja, consiste em fazer uma dilatação seguida de uma erosão. É utilizado para preencher pequenos buracos, e suavizar contornos, ou seja, tem efeitos semelhantes à dilatação mas preserva tamanho dos objectos. A Figura 3.11 mostra o efeito destes operadores sobre uma imagem binária.



a) Imagem original b) Elemento estruturante

c) Resultado da dilatação d) Resultado da erosão

e) Resultado do *closing* f) Resultado do *opening*

Figura 3.11 - Operações morfológicas binárias [47**]**

Pode-se verificar que estes operadores são bastante úteis para pré-processamento de imagens. Por exemplo numa aplicação onde é necessário detectar objectos numa imagem, antes de correr os algoritmos de detecção, torna-se útil a utilização de operadores morfológicos de forma a melhorar as características dos objectos, e a distinção entre eles.

3.6. Transformadas Geométricas

Dados dois sistemas de coordenadas, é importante a capacidade de relacionar as coordenadas de um vector de um sistema para o outro.

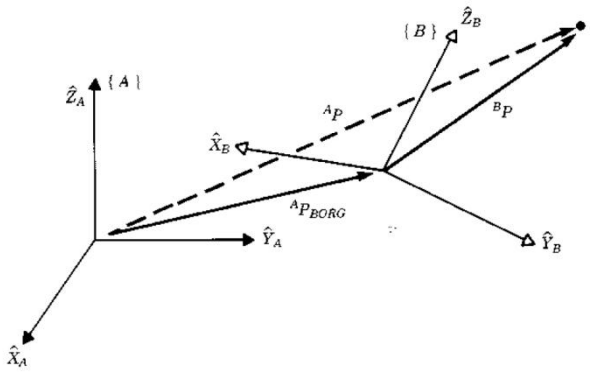


Figura 3.12 - Exemplo de relação entre dois sistemas de eixos [48**]**

A matriz de transformação genérica entre dois sistemas de coordenadas pode ser representada da seguinte forma [**48**]:

$$ {}^{A}_{B}T = \begin{bmatrix} {}^{A}_{B}R & {}^{A}P_{BORG} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{3.7} $$

$$ {}^{A}_{B}R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.8} $$

$$ {}^{A}P_{BORG} = [t_x \quad t_y \quad t_z]^T \tag{3.9} $$

Em que ${}^{A}_{B}T$ significa a transformação do referencial B relativa ao referencial A e está representada na forma homogénea, ${}^{A}_{B}R$ é a matriz de rotação de B em relação a A, e ${}^{A}P_{BORG}$ é o vector que localiza a origem de B relativamente à origem de A.

Relativamente a ${}^{A}_{B}R$, a rotação pode ser efectuada em torno de qualquer um dos eixos de um referencial, logo existe uma matriz de transformação para cada eixo:

Rotação em torno do eixo x de um ângulo θ:

$$ Rot(x,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10} $$

Rotação em torno do eixo y de um ângulo θ:

$$ Rot(y,\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.11} $$

Rotação em torno do eixo z de um ângulo θ:

$$ Rot(z,\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12} $$

Posto isto, dado o vector ${}^{B}P$, pode-se calcular ${}^{A}P$ utilizando a seguinte equação [**48**]:

$$ \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{A}_{B}R & {}^{A}P_{BORG} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix} \tag{3.13} $$

3.7. Arquitecturas de controlo de robôs móveis

Para que um robô móvel tenha uma correcta interacção com o seu espaço de trabalho é necessário que este seja conhecido, quer seja através da leitura de sensores ou através de um modelo, para depois haver um possível planeamento e executado um conjunto de acções resultantes. A forma como a percepção e as acções estão ligadas define a arquitectura de controlo de um robô móvel. Existem três grandes classes de arquitecturas de controlo de robôs móveis: Deliberativa, Reactiva e Híbrida [**49**] [**50**].

3.7.1. Arquitectura Deliberativa

Neste tipo de arquitectura é necessário um mapa global do espaço de trabalho do robô. Sendo este último conhecido, e dado um determinado objectivo, é então construído um caminho óptimo até este, e por fim o robô executa as tarefas necessárias para seguir esse caminho. A Figura 3.13 mostra as partes constituintes deste tipo de sistema.

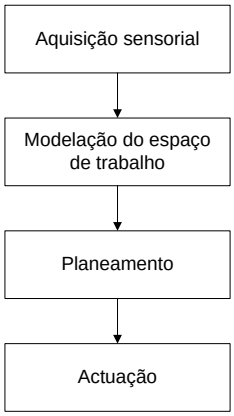


Figura 3.13 - Arquitectura Deliberativa

Como planeamento é totalmente feito antes da execução, este tipo de abordagem não é viável para ambientes dinâmicos. Por outro lado, o mapa global do ambiente é por vezes bastante difícil de obter.

3.7.2. Arquitectura Reactiva

Esta arquitectura surgiu para colmatar as falhas dos sistemas puramente deliberativos em ambientes dinâmicos. Neste tipo de abordagem não é necessário um mapa global do ambiente, pois não é feito nenhum planeamento global. Em vez disso, é utilizada apenas a informação local, obtida através dos sensores, para a selecção dos comportamentos que devem ficar activos. Esses comportamentos são então transformados em acções no espaço de trabalho. Na Figura 3.14 pode-se ver o esquema deste tipo de arquitectura.

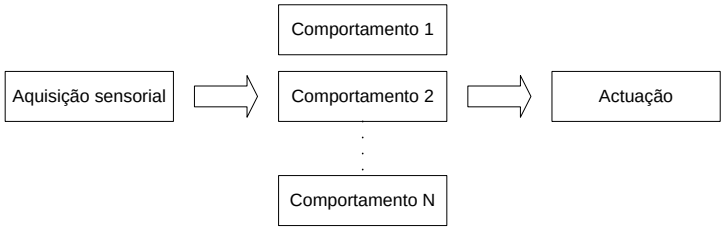


Figura 3.14 - Arquitectura Reactiva

Este tipo de sistema faz com que a aquisição sensorial e a execução fiquem muito próximas uma da outra, permitindo respostas muito rápidas em ambientes altamente dinâmicos. No entanto, a falta do módulo de planeamento, torna por vezes impossível a realização de tarefas mais complexas.

3.7.3. Arquitectura Híbrida

Cada uma das arquitecturas anteriores tem as suas vantagens, uma em relação à outra, mas de forma independente não representam uma solução abrangente. De forma a possibilitar a navegação em ambientes desconhecidos e complexos, surge a arquitectura híbrida, na qual são combinados os pontos fortes das outras duas.

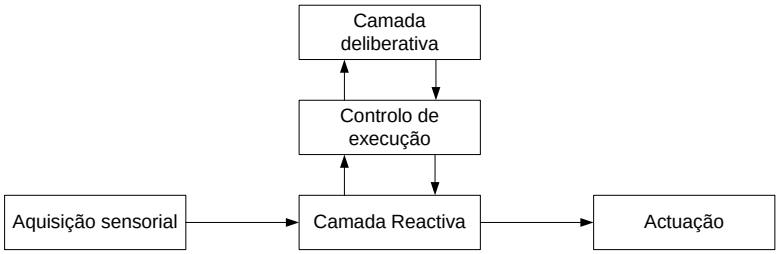


Figura 3.15 - Arquitectura Híbrida

Como se pode ver na Figura 3.15, esta arquitectura é normalmente constituída por três camadas: camada deliberativa, controlo de execução e camada reactiva. A camada deliberativa é responsável por tarefas como a construção de mapas, planeamento, etc. A camada do controlo de execução tem como função coordenar os comportamentos activos, e fazer o interface entre a camada deliberativa e a camada reactiva. A camada reactiva define quais os comportamentos disponiveis no sistema.

Neste capítulo foram abordados os principais conceitos teóricos necessários para a realização deste trabalho. No próximo capítulo é descrita a forma como foi realizada a implementação de toda a parte prática do mesmo.

4. Implementação

Este capítulo está dividido em seis subcapítulos e tem como objectivo descrever os detalhes de implementação deste trabalho. Nos dois primeiros subcapítulos explica-se como os dados são adquiridos e pré-processados (subcapítulo 4.1), e como são depois transformados em informação útil (subcapítulo 4.2). De seguida, no subcapítulo 4.3 são especificados os comportamentos básicos que foram desenvolvidos, e no subcapítulo 4.4 mostra-se e especifica-se quais os papéis que os agentes podem tomar. No subcapítulo 4.5 demonstra-se qual a informação partilhada, e como é transmitida. Por fim, no subcapítulo 4.6 descreve-se a estrutura da camada de decisão da aplicação principal.

Este trabalho é desenvolvido utilizando o SDK Qt da Nokia e linguagem de programação c++. O sistema operativo utilizado é o Ubuntu 10.04.

O programa principal de cada robô tem uma linha de execução principal que pode ser representada pelo fluxograma da Figura 4.1.

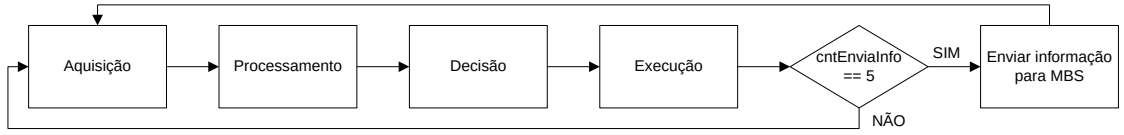


Figura 4.1 - Linha de execução do programa principal

Como se pode ver na figura acima o primeiro passo é a aquisição de dados, onde uma nova imagem é capturada e os sensores de baixo nível são lidos. De seguida, na fase de processamento os dados adquiridos são utilizados para determinar a posição e velocidade da bola, a posição dos obstáculos, e a localização do robô no campo. Em função do estado e papel actuais, na fase de decisão determinam-se quais os comportamentos que estão activos no robô. Depois de decididas quais as acções a realizar, na fase de execução enviam-se os comandos correspondentes para os actuadores. Por fim, a cada cinco ciclos de processamento, envia-se a informação sobre o estado do robô e da bola para a MBS.

A Figura 4.2 ilustra a arquitectura de software do robô. Os módulos que foram definidos nos objectivos deste trabalho são subdivididos em vários sub-módulos para uma melhor gestão da complexidade.

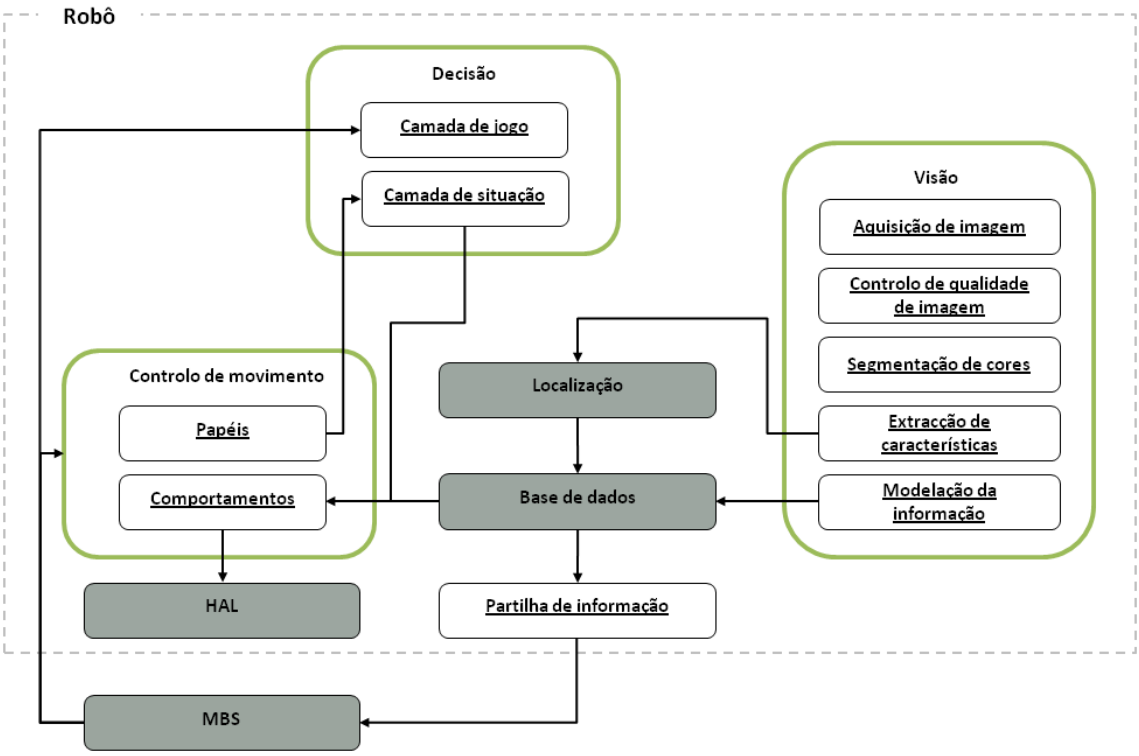


Figura 4.2 - Arquitectura de software do robô (os módulos a cinzento estão fora do âmbito deste trabalho)

No resto deste capítulo são descritos os vários sub-módulos apresentados na Figura 4.2 que fazem parte do âmbito deste trabalho.

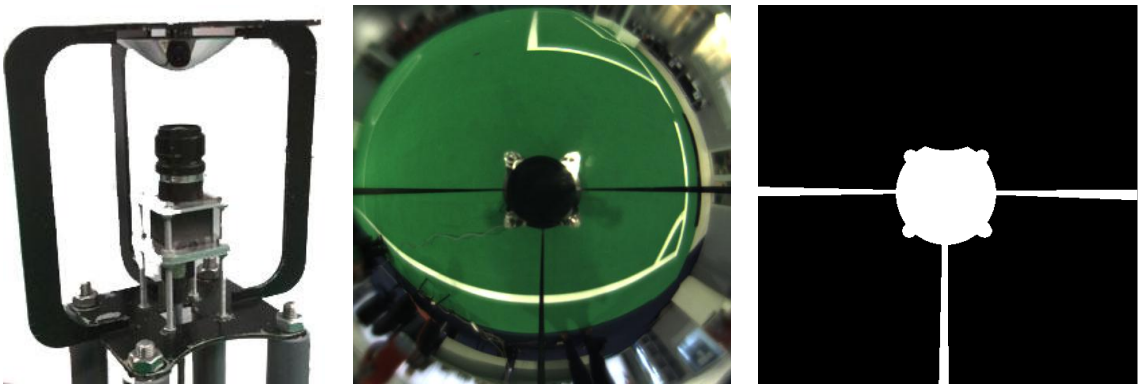
4.1. Aquisição de Dados e Pré-processamento

Neste subcapítulo é descrita a forma como os dados são adquiridos, como esses dados são utilizados para realizar o controlo de exposição das imagens adquiridas, e ainda como esses dados são preparados para a fase posterior de processamento.

4.1.1. Aquisição de Imagem

Para a aquisição de imagens foi utilizada uma câmara Firewire Flea®2 da Point Grey. Esta, permite uma velocidade máxima de transmissão de 800Mb/s (IEEE-1394b), no entanto as motherboards utilizadas nestes robôs apenas permitem uma velocidade máxima de transmissão de 400Mb/s (IEEE-1394a), limitando a comunicação a 400Mb/s. As imagens podem ser transmitidas em vários formatos, mas o formato por nós utilizado é o RGB888. De forma a assegurar um bom compromisso entre velocidade de processamento e qualidade de imagem, definiu-se uma resolução de 480x480 píxeis para as imagens capturadas, utilizando o modo Format_7 da câmara em questão. Para interface com a câmara é utilizada a biblioteca libdc1394 [**51**] que disponibiliza uma boa API para captura de imagens.

O sistema de visão utilizado é omnidireccional (Figura 4.3 a)) o que permite adquirir imagens a 360°, e consiste essencialmente numa câmara mais um espelho convexo. Este tipo de espelho introduz distorção na imagem, como se pode ver na Figura 4.3 b), em que as linhas laterais do campo aparecem nitidamente curvadas e que na realidade são rectas.



a) b) c)

Figura 4.3 - a) Sistema de visão omnidireccional; b) Imagem capturada; c) Máscara da projecção do robô

O círculo e as hastes negros visíveis na Figura 4.3 b) são a projecção do próprio robô na imagem, através do espelho. Para não se processarem os píxeis que representam a projecção do robô, utiliza-se uma máscara semelhante à da Figura 4.3 c). Nesta, a área branca é a que não deve ser processada.

A aquisição das imagens é feita sincronamente com o resto do processamento, ou seja, é feito um pedido de transmissão de imagem por cada ciclo de processamento. O processo de captura de uma imagem é descrito no fluxograma da Figura 4.4.

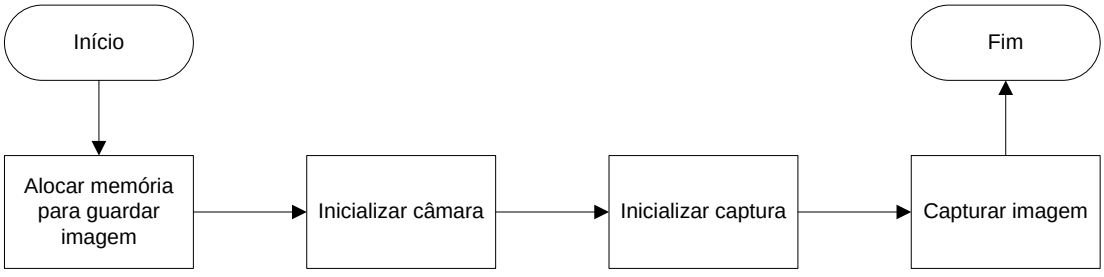


Figura 4.4 - Fluxograma de captura de uma imagem

4.1.2. Controlo de Qualidade da Imagem

As condições luminosas do ambiente onde os robôs operam, afectam significativamente os sistemas de visão. No caso do sistema de visão utilizado, em que a maior parte do processamento é feito pela cor dos objectos, as consequências são bastante significativas. Então, de forma a diminuir os efeitos negativos das variações de luminosidade na imagem foi

implementado um sistema de controlo de qualidade da imagem baseado no trabalho desenvolvido em [**52**].

Os factores de qualidade utilizados são: a exposição, o balanço dos brancos, e o brilho da imagem. Para determinar se a exposição da imagem é correcta utiliza-se o MSV (Figura 4.5), extraído a partir do histograma das intensidades. Como foi dito anteriormente, para que uma imagem esteja correctamente exposta o MSV deve ser igual a 2,5 [**42**][**52**]. No caso do balanço dos brancos, para se extrair uma medida qualitativa, é necessário conhecer à partida uma área da imagem que seja branca, e esta deve estar representada no espaço de cores YUV. Posto isto, para que os brancos estejam correctamente calibrados, a média dos canais U e V deve ser 127 (imagem de 8 bits). De forma semelhante, para o brilho também é necessário conhecer uma área da imagem em avanço, mas neste caso deverá ser preta. Para que o brilho esteja bem calibrado, a média dos píxeis da área conhecida deve ser próxima de zero.

Como a imagem é capturada no formato RGB, é necessário converter a imagem em tons de cinza para o cálculo do histograma das intensidades, e os píxeis da área de interesse do balanço dos brancos em YUV. Utiliza-se para isso a função de conversão de cores *cvCvtColor* da biblioteca *OpenCv* [**53**].

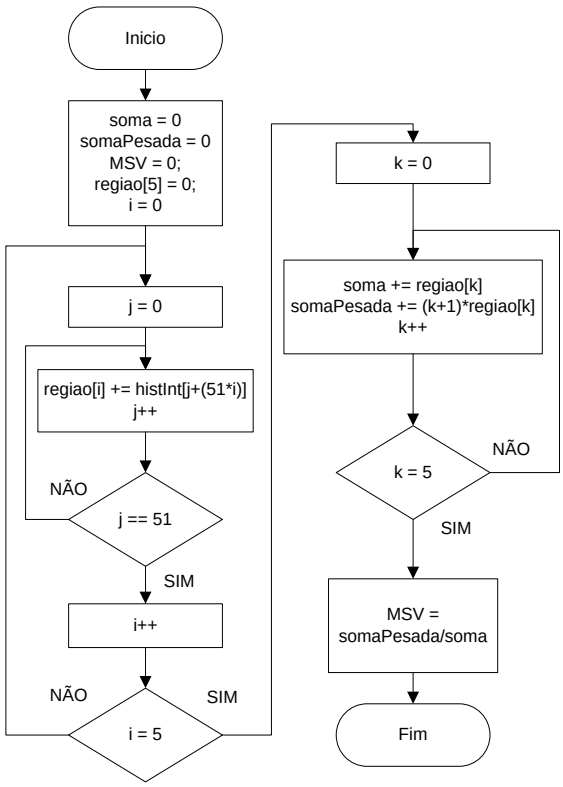


Figura 4.5 - Cálculo do MSV

Da mesma forma que em [**52**], foi implementado um controlador PI para cada um dos parâmetros associados aos factores de qualidade. No caso da exposição, os parâmetros que as nossas câmaras disponibilizam para o seu controlo são o *Shutter speed* e o *Gain*, em que o primeiro permite controlar o tempo de exposição, e o segundo determina a amplificação do sinal de saída do CCD. Para o balanço dos brancos, as nossas câmaras disponibilizam um parâmetro chamado *White balance*, o qual permite ajustar a quantidade de azul e vermelho presentes na imagem. Por fim, o parâmetro offset (*Brigthness*) da nossa câmara permite controlar o brilho da imagem. Todos estes parâmetros são calibrados um de cada vez, antes de cada jogo.

4.1.3. Segmentação de Cores e Extracção de Características

Depois da captura de uma imagem, é necessário dividi-la em diferentes zonas com o intuito de encontrar objectos de interesse. Neste caso a imagem é segmentada pelas suas cores, pois assume-se que os objectos de interesse têm cores suficientemente distintas. Os objectivos desta fase são encontrar possíveis candidatos para a bola, e detectar os contornos dos obstáculos e das linhas de campo. As diversas etapas desta fase são representadas no fluxograma da Figura 4.6.

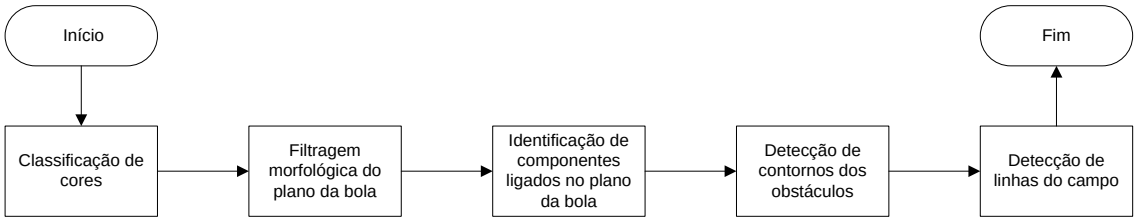


Figura 4.6 - Fluxograma do processo de segmentação de imagem por cor

Classificação de cores

A classificação de cores é feita recorrendo a uma *look-up-table* (LUT) de 2^{16} posições. Numa imagem de 24bits é necessária uma tabela com 2^{24} elementos para conter todas as combinações de cores, e como este é um número bastante elevado para a dimensão de uma tabela, optou-se por reduzir o tamanho da mesma para 2^{16} elementos, à custa da perda de um pouco de resolução no espaço de cor.

O preenchimento da LUT é feito de forma manual, em que um utilizador define um sub-espaço de cor para cada objecto de interesse. O espaço de cores utilizado neste processo é o HSV, pois facilita a escolha do tom da cor ou matiz (canal H), e do ajuste do brilho e quantidade de cinza, de forma independente. Então, para definir uma classe de cor, o utilizador tem de

escolher limites máximos e mínimos para cada canal HSV, e qual o objecto que essa classe representa. Depois, internamente para cada combinação deste sub-espaço HSV é feita a conversão para RGB888 [**54**] e depois para RGB565 (Tabela 4-1). Por fim, é atribuído um caracter que representa essa classe de cor na posição correspondente da LUT (Figura 4.7).

Tabela 4-1: Exemplo de conversão de HSV para RGB565

H	S	V	RGB888	RGB565
100	40	99	0xB9FC97	0x173F12
100	40	100	0xBBFF99	0x173F13
100	41	99	0xB7FC95	0x163F13
100	41	100	0xB9FF96	0x173F12
101	40	99	0xB7FC97	0x163F12
101	40	100	0xB9FF99	0x173F13
101	41	99	0xB6FC95	0x163F12
101	41	100	0xB8FF96	0x173F12

100 <= H <= 101; 40 <= S <= 41; 99 <= V <= 100

No exemplo da Figura 4.7 pode-se ver o caracter 'c' (proveniente do objecto campo) na posição 0x163F12 da LUT, em que esta posição de memória é obtida através do resultado da conversão de HSV para RGB565.

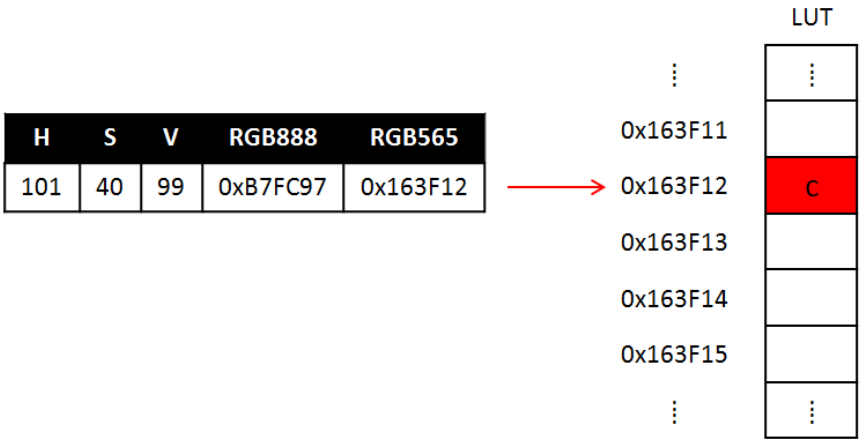


Figura 4.7 - Exemplo de conversão de HSV para RGB565 e consequente preenchimento da LUT

Após o preenchimento da LUT, o sistema está pronto para classificar os píxeis da imagem. Para isso, cada pixel da imagem é convertido do seu formato original (RGB888) para RGB565, e o valor resultante é usado como índice para aceder à LUT e assim determinar a que

classe de cor pertence em função do caracter presente no endereço correspondente. A Tabela seguinte mostra as classes de cores definidas neste sistema.

Tabela 4-2: Classes de cores

Classe	Caracter Associado	Cor representativa
Bola	'b'	Vermelho
Campo	'c'	Verde
Obstáculo	'o'	Amarelo
Linha	'l'	Branco

À medida que os píxeis da imagem vão sendo classificados, o resultado é guardado numa matriz chamada **ImgClassificada** (Figura 4.8 a) e b)).

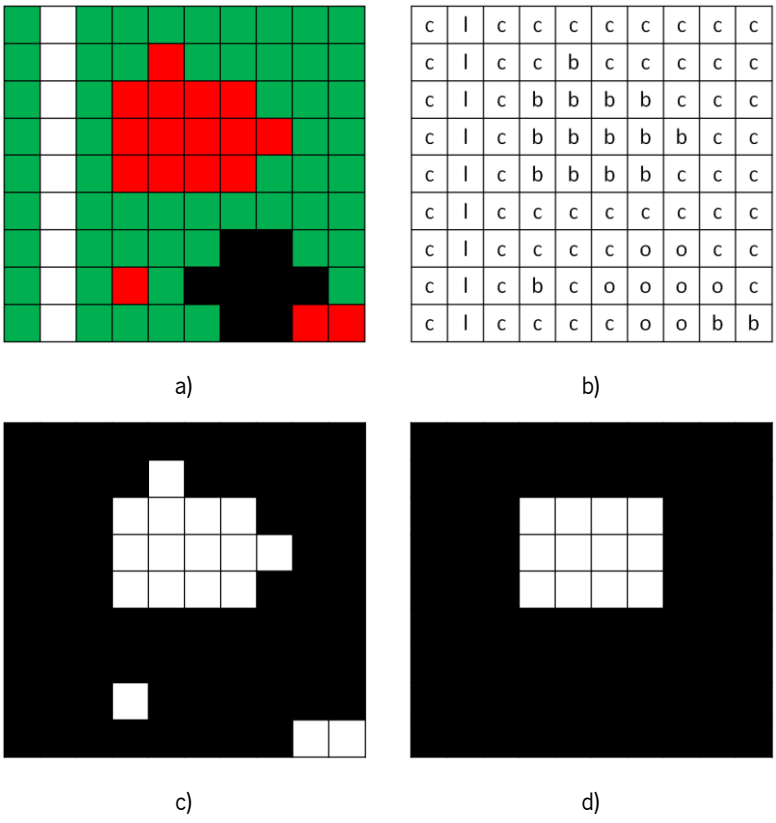


Figura 4.8 - a) Representação visual da Imagem classificada; b) Matriz ImgClassificada; c) Plano da bola; d) Resultado do operador *open* sobre o plano da bola

Filtragem morfológica do plano da bola

Concluída a classificação dos píxeis, passa-se à filtragem morfológica do conjunto de píxeis que foram classificados como bola (plano da bola), e para isso, primeiro constrói-se uma imagem binária a partir desse conjunto como se pode ver na Figura 4.8 c) a qual vai ser então

filtrada. O filtro utilizado é o *open*, e foi implementado utilizando as funções *cvErode* e *cvDilate* da biblioteca *OpenCV* em cascata com um elemento estruturante do tipo caixa 3x3, e com isto obtém-se a Figura 4.8 d).

Identificação de componentes ligados no plano da bola

Após a aplicação do filtro passa-se à identificação de componentes ligados no plano da bola. Para isso utiliza-se uma abordagem semelhante à descrita em [**55**]. Deste processo surge uma lista com os *blobs*[4] candidatos a bola, em que cada elemento contém: massa, coordenadas mínimas e coordenadas máximas do rectângulo envolvente (Figura 4.9).

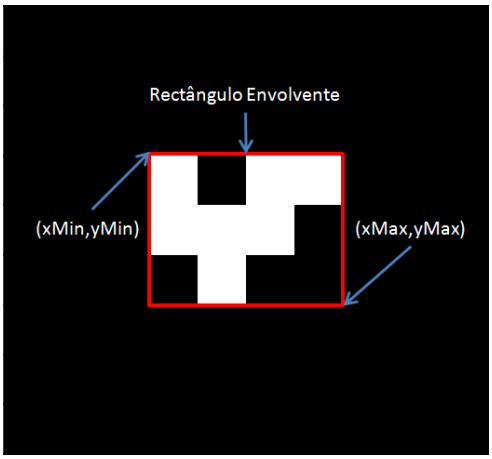


Figura 4.9 - Elementos de um *blob*

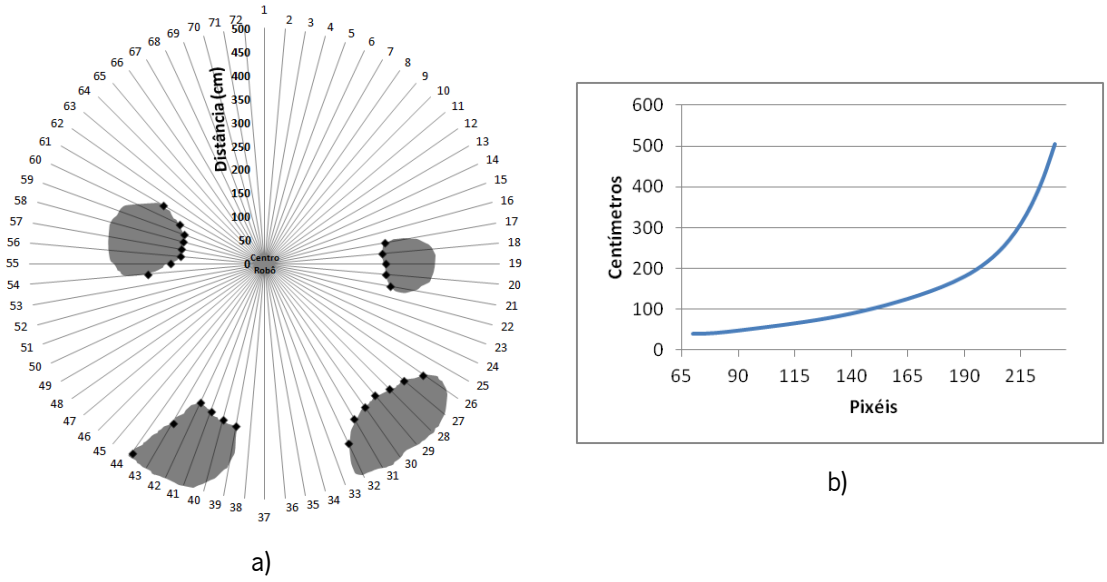
Detecção de contornos dos obstáculos

De seguida pretende-se encontrar os contornos de possíveis obstáculos na imagem classificada, e para isso é feita uma pesquisa radial. Esta pesquisa consiste em projectar linhas na imagem a partir do centro do robô (Figura 4.10 a)), de forma a simular sensores de obstáculos reais. Sempre que se encontra um conjunto seguido de píxeis do tipo obstáculo com mais do que um número predefinido de elementos, na linha de um sensor, é calculada a distância (em píxeis) até ao primeiro pixel desse conjunto, e convertida para a distância em centímetros correspondente (Equação 4.1 e Figura 4.10 b)). No final do varrimento de todos os sensores, o valor de cada um é guardado na base de dados do robô.

$$y \cong 3{,}56E^{-10} * x^6 - 2{,}82E^{-7} * x^5 + 9{,}16E^{-5} * x^4 - 1{,}56E^{-2} * x^3 + 1{,}46 * x^2 - 71{,}07 * x + 1434{,}44 \quad (4.1)$$

[4] *Blob:* conjunto de píxeis vizinhos que partilham características comuns.

A equação da forma do espelho do sistema de visão omnidireccional utilizado não é conhecida, portanto para se determinar a Equação 4.1, utiliza-se uma regressão polinomial recorrendo ao *Microsoft Excel* e algumas amostras retiradas numa das direcções do espelho. Posto isto, para qualquer outro pixel recorre-se a esta equação, dando-se como entrada a distância do pixel em questão ao centro da imagem, e obtém-se como saída a distância em centímetros que ele representa. Assume-se que o espelho está devidamente calibrado de forma a reduzir possíveis erros.



a) b)

Figura 4.10 - a) Exemplo de detecção de contornos de obstáculos; b) Gráfico da relação entre as distâncias em píxeis e as distâncias em centímetros

Detecção de linhas do campo

Por fim, passa-se à detecção das linhas de campo, pois a localização do robô no campo depende essencialmente destes dados. A estratégia utilizada é semelhante à usada nos obstáculos, pois também é feita uma pesquisa radial, mas aqui é acrescentada uma pesquisa axial (Figura 4.11). Em ambas as pesquisas o que se procura são transições entre píxeis de campo e píxeis de linha ou vice-versa. Sempre que é encontrada uma transição válida, as suas coordenadas em centímetros são obtidas através da Equação 4.1 e guardadas na base de dados do robô.

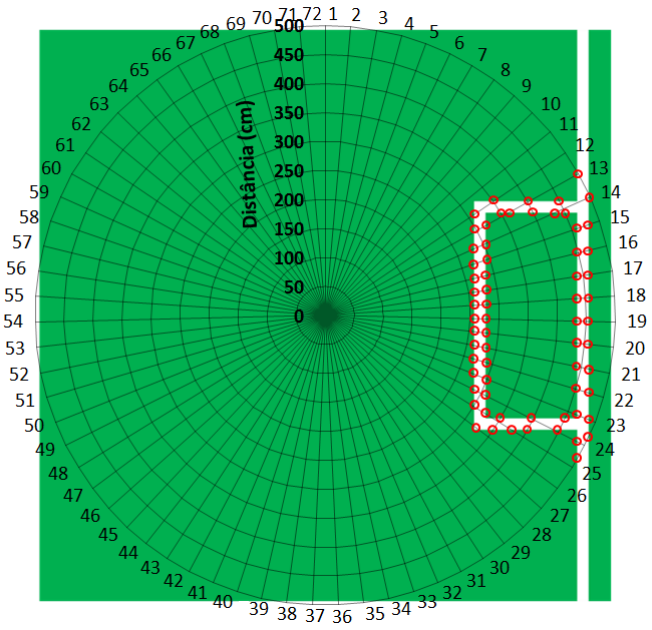


Figura 4.11 - Disposição das linhas de pesquisa radiais e axiais

4.1.4. Sensores de baixo nível

Para adquirir dados dos sensores de baixo nível presentes no robô, é utilizada a abstracção de hardware descrita em [**56**], a qual permite a leitura dos seguintes sensores:

- Infra-vermelhos para detecção da bola
- Bússola electrónica
- ADC de tensão da bateria
- Temperatura da placa de actuação dos motores
- Encoders dos motores

A leitura dos sensores é feita através de uma API C++, a duas cadências diferentes. No caso da bússola e dos infra-vermelhos é feita uma leitura a cada 20 ms, e para a tensão da bateria e temperatura dos motores é feita uma leitura a cada 2s.

4.2. Modelação da Informação

Depois da aquisição dos dados e extracção de algumas das características base, passa-se então para a modelação da informação. Aqui, os dados recolhidos na fase anterior são transformados em informação útil para as fases posteriores.

4.2.1. Bola

Num jogo de futebol, a bola é um dos principais objectos em campo, logo uma das maiores prioridades dos jogadores é saber onde esta se encontra. No futebol robótico o principio

é o mesmo, portanto umas das tarefas mais importantes dos robôs é detectar a posição da bola, e se possível prever a sua trajectória. Neste trabalho, a posição e velocidade da bola são calculadas da forma que a seguir se explica.

Posição

Partindo da lista de *blobs* construída anteriormente, pretende-se agora determinar qual dos candidatos é a bola. Para isso, são analisadas várias características:

- Massa: número de píxeis que constituem o *blob*
- Relação entre massa e área (RMA): massa/área
- Relação entre largura e altura (RLA): largura/altura
- Número de píxeis de campo em torno do *blob* (NPC)

Na Figura 4.12 mostram-se exemplos de *blobs* e as medidas correspondentes

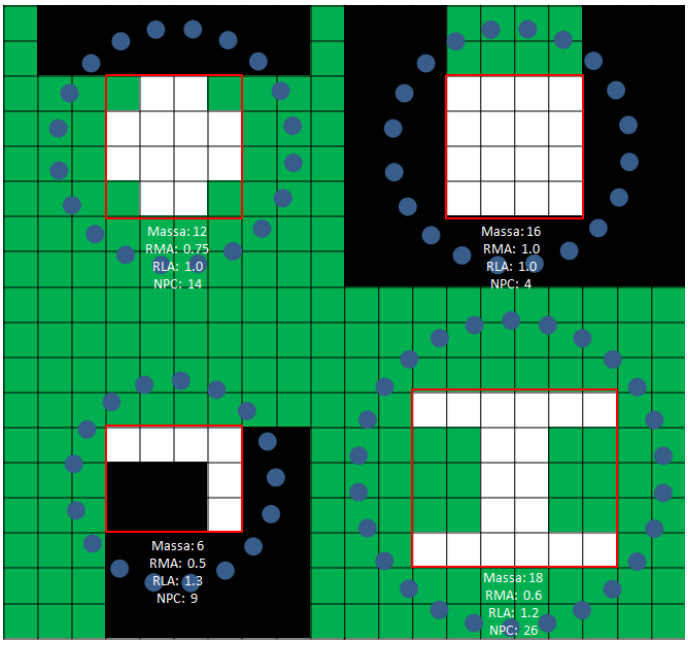


Figura 4.12 - Vários *blobs* e medidas associadas. Os circulos azuis representam as zonas onde se procuram píxeis de campo em torno do *blob*

Para cada uma das características são definidos limites máximos e mínimos de forma a filtrar falsos positivos. Do conjunto de *blobs* que passam este teste é escolhido para bola o que estiver mais próximo do centro da imagem. Para calcular a distância em centímetros do centro da bola ao centro do robô utilizou-se um método semelhante ao da conversão de píxeis para centímetros vista no subcapítulo anterior, e o resultado é a Equação 4.2.

$$y \cong 3{,}56E^{-10} * x^6 - 1{,}45E^{-7} * x^5 + 2{,}33E^{-5} * x^4 - 1{,}79E^{-3} * x^3 + 7{,}29E^{-2} * x^2 - 8{,}24E^{-1} * x + 42{,}26 \quad (4.2)$$

No final, a informação sobre a posição da bola é guardada na base de dados do robô.

Velocidade

Assumindo movimento uniforme para curtos intervalos de tempo e utilizando as últimas n amostras da posição da bola é feita uma regressão linear para cada uma das coordenadas x e y, e as equações resultantes são derivadas de forma a obter as velocidades em cada eixo (vBx e vBy).

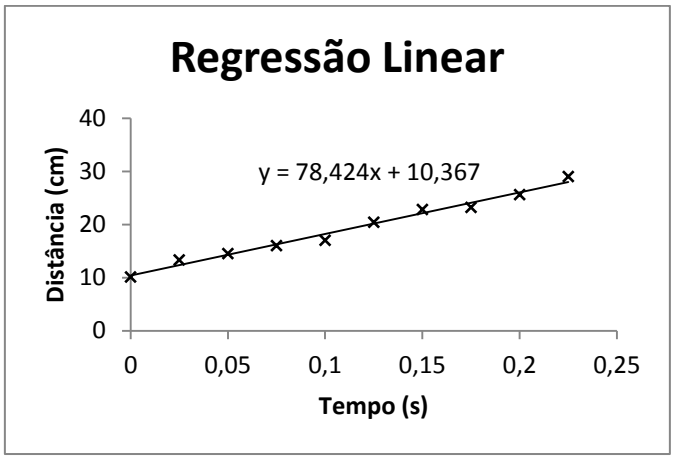


Figura 4.13 - Exemplo de regressão linear para obtenção da velocidade da bola num eixo

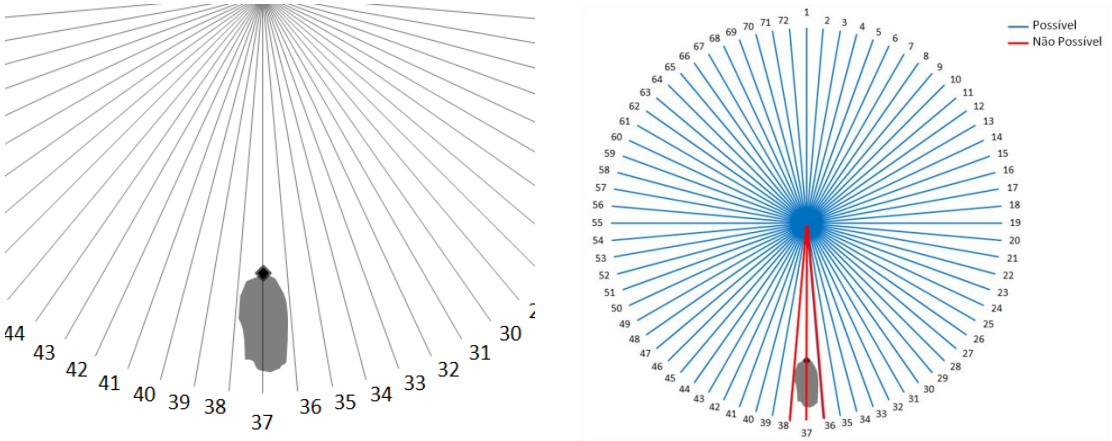
No exemplo da Figura 4.13, podem-se ver os valores de 10 amostras (n = 10) e a regressão linear correspondente. Para obter a velocidade é preciso derivar a equação que se obtém da regressão, que na prática como se trata de uma equação do tipo $y = mx + b$, é directamente o valor do declive da recta (m). Para calcular a regressão, foi adaptado o código disponível em [**57**].

4.2.2. Obstáculos

Para se obter uma representação eficaz dos obstáculos presentes no campo de jogo utilizam-se os dados obtidos anteriormente para construir um mapa de obstáculos. Neste mapa estão representadas quais as direcções de navegação[5] do robô que são possíveis ou não possíveis (tendo em consideração o tamanho do robô).

[5] É de notar que com este tipo de aproximação o número de direcções de navegação possíveis coincide com o número de sensores de obstáculos definidos.

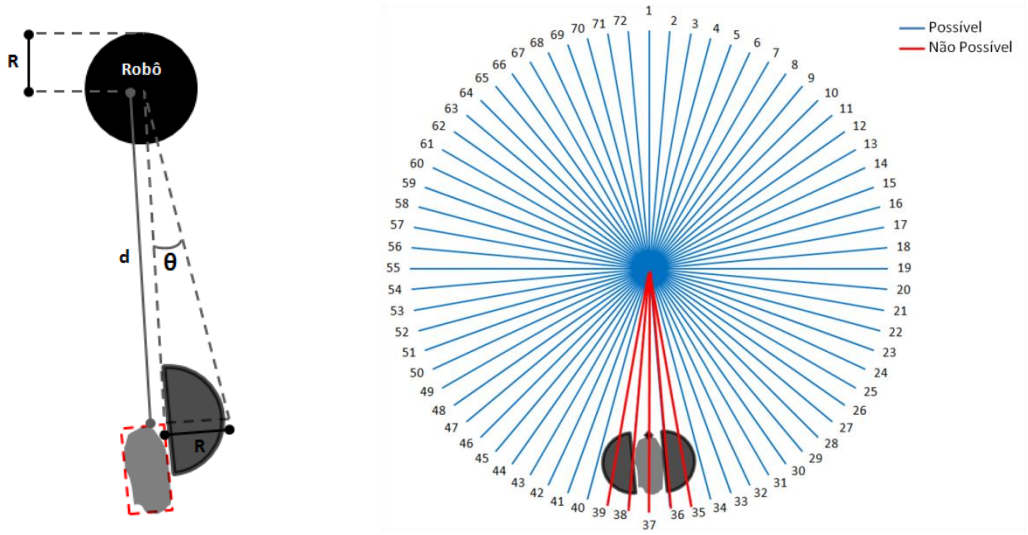
No exemplo da Figura 4.14 a) se apenas se tomasse em atenção as direcções em que se detectam contornos de obstáculos, era possível haver colisão. Para se evitar este tipo de situações, em cada direcção que se detecta o contorno de um obstáculo, marca-se esta e as duas direcções vizinhas como não possíveis de passagem (Figura 4.14 b)).



a) b)

Figura 4.14 - a) Exemplo de obstáculo que ocupa apenas um sensor; b) Mapa de obstáculos inicial

Falta agora levar em consideração as dimensões do próprio robô, e para isso projecta-se o seu raio à distância de cada obstáculo, de forma a se obter o espaço angular ocupado pelo robô a essa distância (Figura 4.15 a)).



a) b)

Figura 4.15 - a) Representação geométrica da projecção do raio do robô em obstáculos; b) Mapa de obstáculos resultante. Os semi-círculos representam a projecção do robô dividido ao meio

Feita a projecção, actualiza-se o mapa de obstáculos com a nova informação. Para o exemplo da Figura 4.14 a) assumindo-se que o obstáculo se encontra a 250 centímetros do

centro do robô, o mapa resultante é o da Figura 4.15 b). Para calcular o espaço angular ocupado pela projecção do raio do robô utilizou-se a seguinte equação:

$$\theta = \tan^{-1}\frac{R}{d} \tag{4.3}$$

$$n = \ inteiro\left(\frac{\theta}{res}\right) \tag{4.4}$$

Na configuração actual os sensores de obstáculos têm 5° de desfasamento entre eles, portanto a resolução para a direcção de navegação é também 5°. Com isto, pode-se calcular quantas direcções ocupa a projecção do robô, dividindo o valor de θ pela resolução (Equação 4.4). Agora, no mapa de obstáculos, marcam-se como não passáveis mais n direcções de cada lado de cada obstáculo. No caso do exemplo da Figura 4.14 b), obtemos θ=5,7° e n=1, o que se traduz em apenas uma direcção ocupada (parte inteira do resultado), e o resultado é o da Figura 4.15 b), na qual foram adicionadas uma direcção de cada lado do obstáculo.

4.3. Comportamentos

As acções que os robôs podem tomar para interagir com o ambiente são especificadas através de comportamentos. Neste subcapítulo são definidos os comportamentos base, e como são implementados.

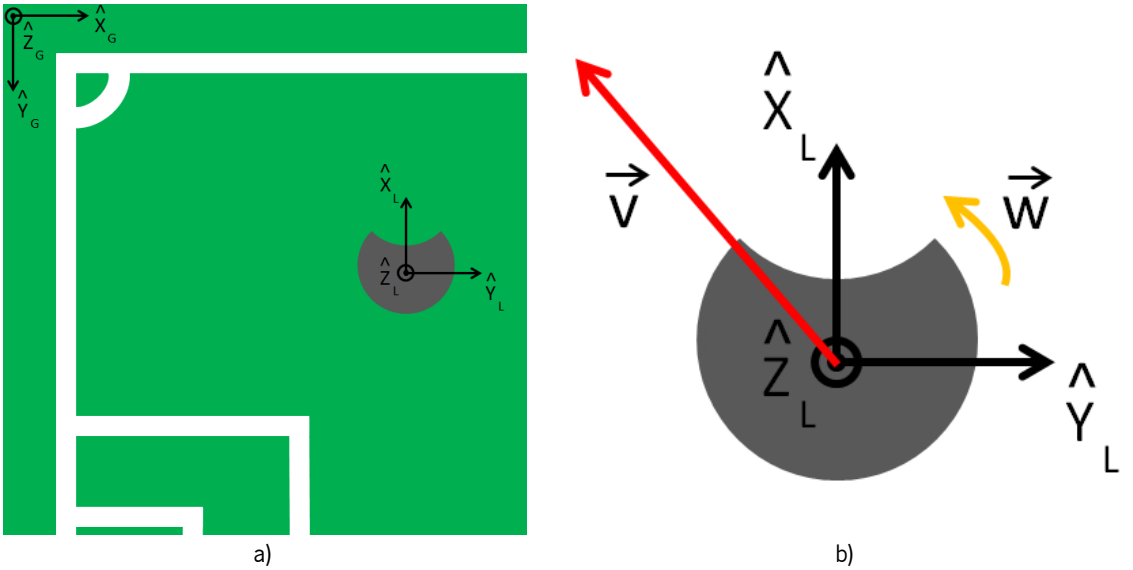
Tal como para a leitura dos sensores, o envio de comandos para os actuadores é feito através da abstracção de hardware descrita em [**56**], a qual permite enviar comandos para a sistema de locomoção omnidireccional, *dribbler*, e dispositivo de chuto. Os comandos disponibilizados são os que estão presentes na Tabela 4-3.

Tabela 4-3: Comandos dos actuadores disponiveis

Actuador	Comandos	Gama
Sistema de locomoção omnidireccional	Velocidade Linear	0 – 100%
	Direcção	0 – 360°
	Velocidade Angular	0 – 100%
Dribbler	Intensidade de drible	0 – 100/
Dispositivo de chuto	Tipo de chuto (vertical ou horizontal)	H/V
	Intensidade de chuto	0 – 100%

Destes comandos, os enviados para o sistema de locomoção omnidireccional são internamente transformados em velocidades para cada uma das três rodas omnidireccionais presentes nos robôs.

Definem-se agora mais dois sistemas de eixos cartesianos, um para o campo de jogo (sistema global) e outro para o próprio robô (sistema local), como se pode ver na Figura 4.16 a).



a) b)

Figura 4.16 - a) Sistemas de eixos cartesianos Global e Local; b) Vectores velocidade linear ($\vec{v}$) e velocidade angular ($\vec{w}$)

Em que são conhecidos à partida o vector ${}^{G}P_{LORG}$ e a orientação θ do robô em relação ao referencial G [**58**]. Com isto, é possível transformar coordenadas de pontos conhecido de um referencial para o outro.

Os comportamentos desenvolvidos durante este trabalho são os definidos de seguida. Nestes, é utilizada a informação presente na base de dados do robô, e pretende-se controlar a postura do mesmo, manipulando os vectores velocidade linear e velocidade angular. A excepção é o comportamento chutar, no qual se pretende controlar o alcance do remate através da intensidade de chuto. Para cada comportamento são definidos quais os seus objectivos, e qual o algoritmo e/ou leis de controlo utilizados.

4.3.1. Ir à bola

Neste comportamento pretende-se fazer com que o robô se dirija para a bola de jogo. Utilizando a informação adquirida anteriormente sobre a posição da bola relativamente ao robô, o que se pretende é minimizar a distância entre o robô e a bola.

As variáveis manipuladas são o módulo e a direcção da velocidade linear, em que se utilizam as seguintes leis de controlo:

$$\begin{cases} v = Vmax * \tanh(\sigma * (d - \Delta)),\ se\ \tanh(\sigma * (d - \Delta)) > 0 \\ v = 0, \qquad se\ \tanh(\sigma * (d - \Delta)) < 0 \end{cases} \tag{4.5}$$

$$d_v = d_b \tag{4.6}$$

Em que v representa a intensidade da velocidade linear, d_v representa a direcção da mesma, e d é a distância à bola. Já σ e Δ são parâmetros da função.

4.3.2. Ir para coordenada no campo

Aqui pretende-se que o robô se desloque de um ponto no campo para outro. Consultada a base de dados do robô para saber a sua posição no campo, pretende-se minimizar a distância entre a posição actual e a posição desejada no campo.

Este comportamento é semelhante ao anterior, com a diferença que, em vez do objectivo ser a bola, é um ponto virtual no campo de jogo, daí as leis de controlo serem as mesmas, alterando-se apenas a direcção da bola pela coordenada do ponto desejado.

4.3.3. Rodar

Este comportamento tem como objectivo orientar a frente do robô com um dado ponto no espaço, manipulando a velocidade angular. Para isso é definida seguinte regra:

$$w = Wmax * \left(\frac{1}{(1 + e^{\Delta\theta * \beta})} - 0{,}5\right) \tag{4.7}$$

Em que w é a velocidade angular, β e Wmax são ganhos, e $\Delta\theta$ é a diferença entre o ângulo desejado e o actual. O sinal de w determina o sentido de rotação.

4.3.4. Evitar obstáculos

Como o nome indica, este comportamento tem como objectivo evitar colisões com os obstáculos presentes no campo de jogo. Parar isso, utiliza-se o mapa de obstáculos construído anteriormente, de forma a escolher uma direcção de navegação livre.

Este comportamento actua sobre a direcção de navegação do robô, que neste caso coincide com a direcção da velocidade linear. A Figura 4.17 descreve o algoritmo implementado, o qual tem como parâmetros de entrada o mapa de obstáculos (listaObs), a distância do

obstáculo mais próximo (minObsDist), a direcção desejada (dirAlvo), e a direcção actual (dirActual).

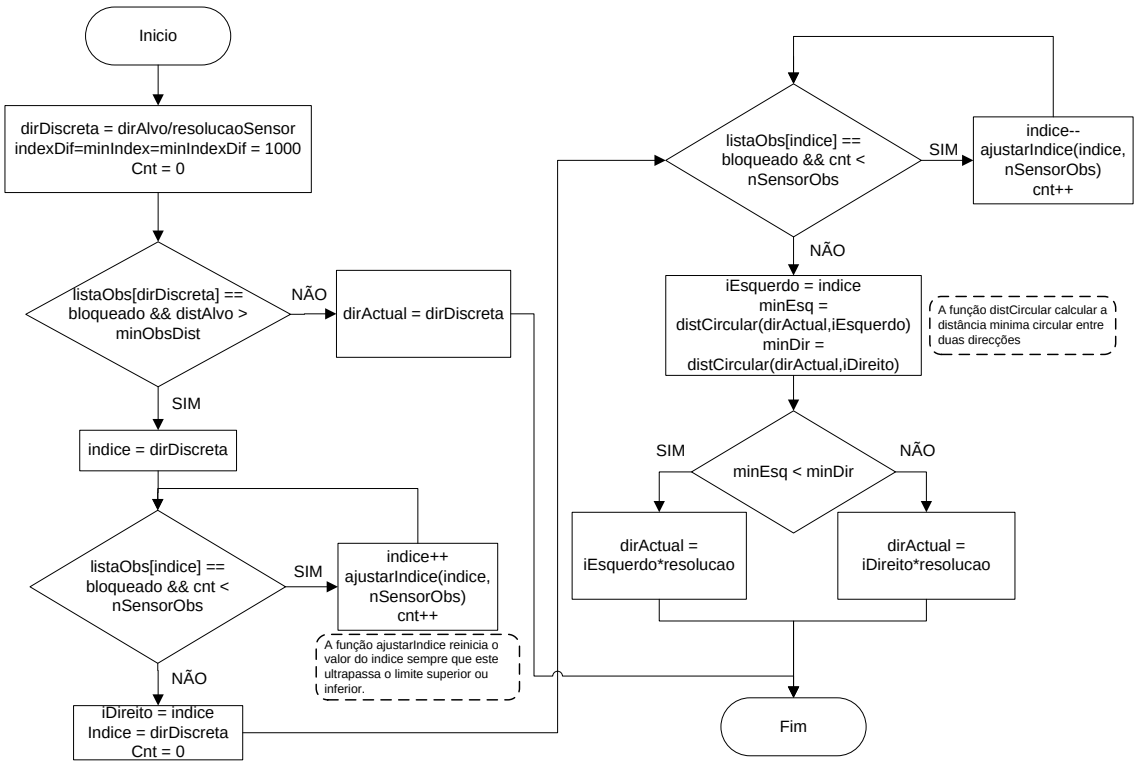


Figura 4.17 - Fluxograma do algoritmo de desvio de obstáculos

A Figura 4.18 demonstra o comportamento pretendido. É de reparar que a orientação do robô mantém-se constante, pois o controlo desta é feito por um comportamento à parte.

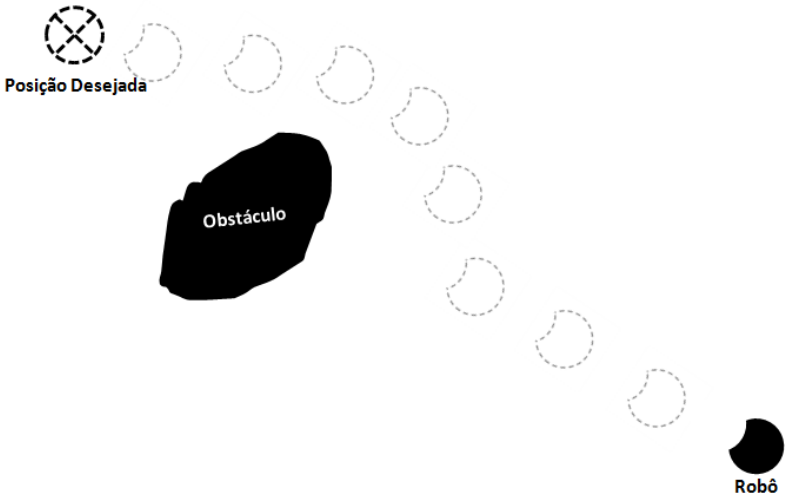


Figura 4.18 - Exemplo de trajectória pretendida para o desvio de obstáculos

4.3.5. Drift

Este comportamento tem como objectivo variar a direcção de navegação sem perder a bola. Para isso são manipuladas tanto a velocidade linear como a velocidade angular. Dada uma direcção de navegação desejada (θ), definem-se as seguintes regras:

$$\begin{cases} w = -Wmax * \sin\theta \\ d_v = \tan^{-1}\left(\frac{-\sin\theta}{\cos\theta}\right), & \theta < 90 \vee \theta \geq 270 \\ w = -Wmax \\ d_v = -90 \end{cases}, \quad 90 \leq \theta < 180 \quad , \quad \begin{cases} w = Wmax \\ d_v = 90 \end{cases}, \quad 180 \leq \theta < 270 \tag{4.8}$$

$$\begin{cases} v = Vref(\alpha * (1 - \tanh(\theta * \varphi)) + 1), & 0 \leq \theta < 90 \\ v = Vref(\alpha * (1 - \tanh(90 * \varphi)) + 1), & 90 \leq \theta < 180 \\ v = Vref(\alpha * (1 - \tanh(270 * \varphi)) + 1), & 180 \leq \theta < 270 \\ v = Vref(\alpha * (1 - \tanh((360 - \theta) * \varphi)) + 1), & \theta \geq 270 \end{cases} \tag{4.9}$$

Em que α, φ, Vref, e Wmax são ganhos. Para uma melhor compreensão, na Figura 4.19 estão os gráficos correspondentes destas equações.

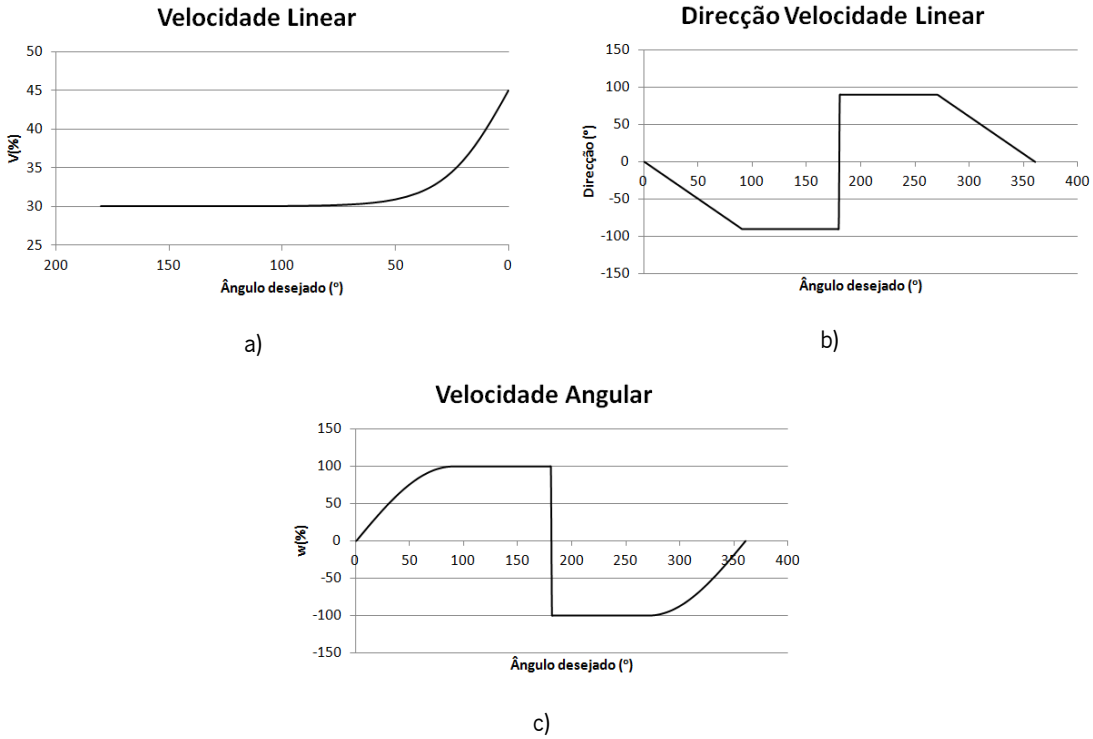


Figura 4.19 - Gráficos das equações 4.9 e 4.10: a)Velocidade linear para $\alpha = 0,5$, $\varphi = 0,035$, e Vref = 30%; b)Direcção da velocidade linear; c) Velocidade angular para Wmax = 100%

4.3.6. Driblar

O objectivo deste comportamento é semelhante ao comportamento **evitar obstáculos**, com o acréscimo de o robô ter a bola em sua posse, e não a devendo perder.

Está dividido em duas partes, em que a primeira corresponde a um algoritmo para encontrar a melhor direcção livre, e a segunda parte consiste em executar um **drift** para essa direcção de forma a evitar o obstáculo sem perder a bola. O algoritmo em questão está representado no fluxograma da Figura 4.20.

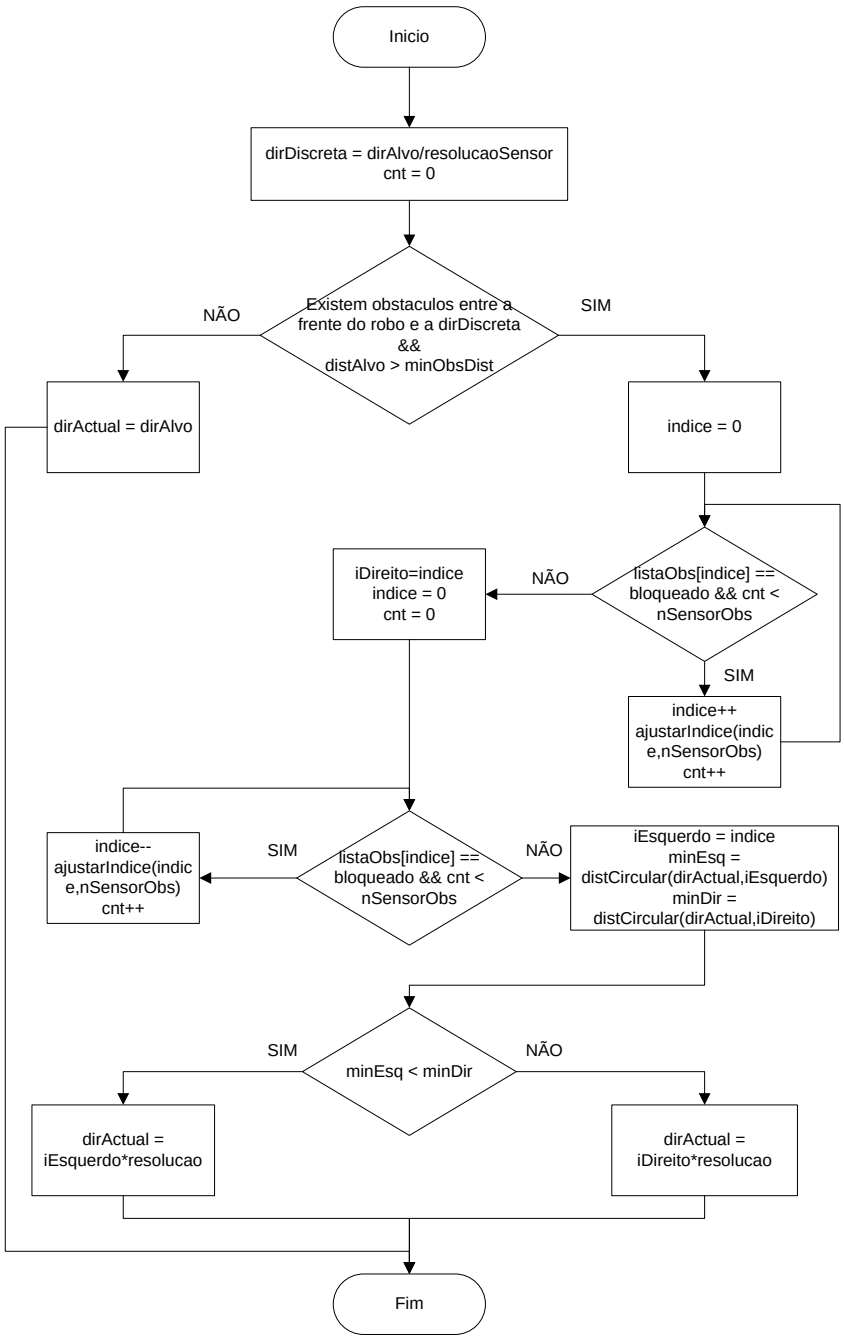


Figura 4.20 - Fluxograma do algoritmo de drible

Na prática o que se pretende com este comportamento é algo semelhante ao representado na Figura 4.21. O movimento deste comportamento tem de ser bastante mais suave do que o evitar obstáculos para o robô não perder a bola.

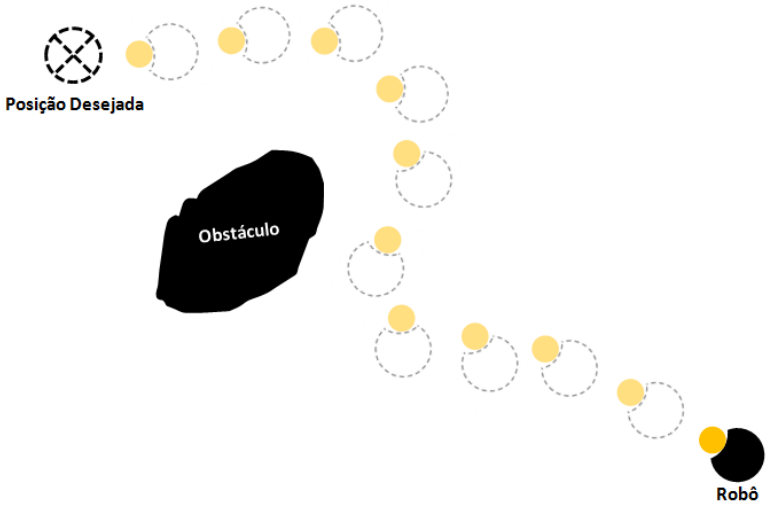


Figura 4.21 - Exemplo de trajectória pretendida com o comportamento drible

4.3.7. Orbitar

Este comportamento serve para que o robô descreva trajectórias circulares em torno de um ponto. Para isso, definem-se duas forças virtuais a actuar no robô (Figura 4.22).

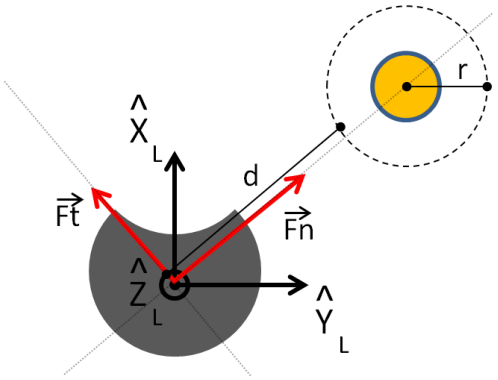


Figura 4.22 - Forças virtuais de órbita

As forças virtuais $\overrightarrow{Fn}$ e $\overrightarrow{Ft}$ são respectivamente a força normal e a força tangencial que o robô sofre sobre o efeito do objecto em questão, d é a distância do centro do robô à tangente mais próxima da linha de órbita, e por fim, r é o raio de órbita. As equações que descrevem $\overrightarrow{Fn}$ e $\overrightarrow{Ft}$ são as seguintes:

$$\begin{cases} Fn = |d * \beta| \\ \theta_{Fn} = \theta_{pt} \end{cases} \quad (4.10)$$

$$\begin{cases} Ft = \gamma \\ \theta_{Ft} = \theta_{pt} + 90, & dir = CCW \\ \theta_{Ft} = \theta_{pt} + 270, & dir = CW \end{cases} \quad (4.11)$$

Em que β é um ganho, γ o valor atribuído ao modulo de $\overrightarrow{Ft}$, θ_{Fn} e θ_{Ft} são as direcções das duas forças, θ_{pt}é a direcção do ponto sobre o qual a órbita é realizada, e por fim CW e CCW definem o sentido de movimento.

Por fim, a direcção do vector velocidade linear é obtida a partir da soma vectorial de $\overrightarrow{Fn}$ e $\overrightarrow{Ft}$ (Equação 4.13), e a sua intensidade é um valor constante predefinido.

$$d_v = \tan^{-1}\left(\frac{Fn_y + Ft_y}{Fn_x + Ft_x}\right) \quad (4.12)$$

4.3.8. Chutar

O objectivo deste comportamento é chutar a bola para um alvo. Sabendo-se a distância d (cm) a que se encontra esse alvo, e o tipo de chuto (horizontal ou vertical), calcula-se a sua força f (%) através das equações seguintes, que foram obtidas através de interpolação:

Chuto Horizontal:

$$f \cong 1{,}16E^{-4} * d^2 + 2{,}13E^{-1} * d + 1{,}01 \quad (4.13)$$

Chuto Vertical:

$$f \cong 2{,}58E^{-7} * d^3 + 3{,}73E^{-4} * d^2 - 9{,}76E^{-2} * d + 70 \quad (4.14)$$

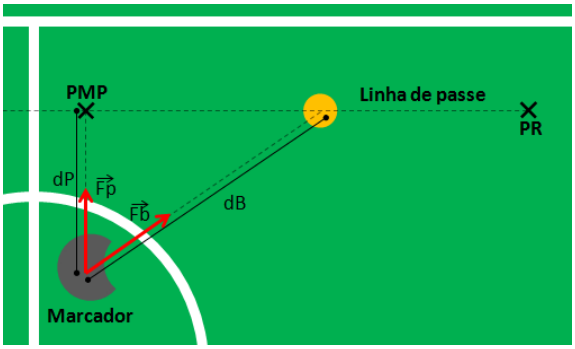
4.3.9. Aproximação de passe

O que se pretende com este comportamento é uma aproximação suave do robô à bola em situações de bola parada. De forma semelhante ao comportamento **orbitar**, aqui também se definem duas forças virtuais a actuar sobre o robô (Figura 4.23 a)). A direcção da força resultante da soma destas duas define a direcção da velocidade linear do robô. O tipo de movimento pretendido é o ilustrado na Figura 4.23 b), e para isso são definidas as seguintes regras:

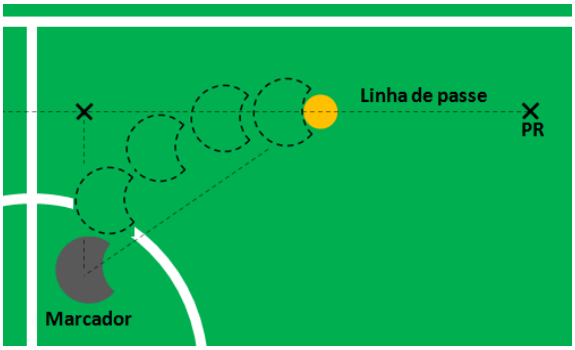
$$\begin{cases} Fp = d_P * \alpha \\ Fb = d_B * \beta \end{cases} \tag{4.15}$$

$$\begin{cases} d_v = \tan^{-1}\left(\dfrac{Fp_y + Fb_y}{Fp_x + Fb_x}\right) \\ v = \tanh\left(\varphi * \left(\sqrt{\left((Fp_x + Fb_x)^2 + \left(Fp_y + Fb_y\right)^2\right)} - \delta\right)\right) \end{cases} \tag{4.16}$$

Em que Fp e Fb são os módulos dos vectores $\overrightarrow{Fp}$ e $\overrightarrow{Fb}$. Os parâmetros α, β, e φ são ganhos, e δ é um offset. As direcções de $\overrightarrow{Fp}$ e $\overrightarrow{Fb}$ são dadas pela semi-recta que une o centro do robô ao ponto mais próximo à linha de passe, e pela semi-recta que une o centro do robô ao centro da bola, respectivamente.



a)



b)

Figura 4.23 - Aproximação de passe: a) Forças virtuais; b) Movimento pretendido

Este comportamento é sempre acompanhado do comportamento **rodar** para orientar a frente do robô para a bola.

Concluída a descrição dos comportamentos base, pode-se verificar que a maioria trata de acções relativamente simples, mas onde a combinação destas pode levar a um comportamento global complexo, que é o que se verifica durante uma situação de jogo.

4.4. Papéis

Neste subcapítulo são definidos e descritos os papéis que os robôs podem assumir. O conceito de papel neste contexto está associado ao tipo de tarefas que um agente tem de realizar. No caso concreto do futebol robótico, podem-se identificar imediatamente dois papéis distintos: guarda-redes e jogador de campo. No caso desta equipa foram definidos os seguintes papéis para situações de bola corrida:

- Guarda-redes (r_Keeper)
- Defesa (r_Defender)
- Avançado (r_Striker)

Depois, para os lances de bola parada surgiram mais dois papéis:

- Marcador (r_Marker)
- Receptor (r_Receiver)

Cada um destes utiliza os comportamentos definidos anteriormente para efectuar as suas tarefas. Já a atribuição destes papéis é feita pela MBS (MINHO *Basestation* [**59**]) durante o tempo de jogo de forma dinâmica. De seguida é feita uma descrição de cada um dos papéis definidos acima.

4.4.1. Guarda-Redes

Sempre que é atribuído este papel a um robô, este tem como principal tarefa evitar o máximo de golos na nossa baliza. O diagrama de estados da Figura 4.24 descreve as acções a realizar pelo guarda-redes.

Figura 4.24 - Diagrama de estados do guarda-redes

O ponto de intersecção é calculado através da projecção do vector velocidade da bola sobre a linha de fundo do campo (Figura 4.25). Sempre que o guarda-redes vê a bola, move-se

para o ponto de intersecção de forma a evitar que esta entre na baliza. Caso o ponto de intersecção esteja fora dos limites da baliza mais uma histerese, o robô não se mexe.

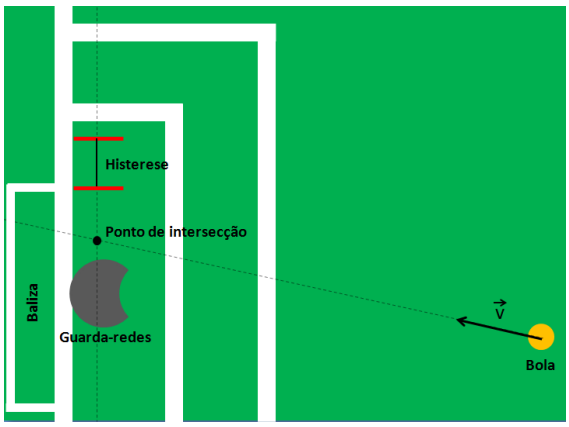
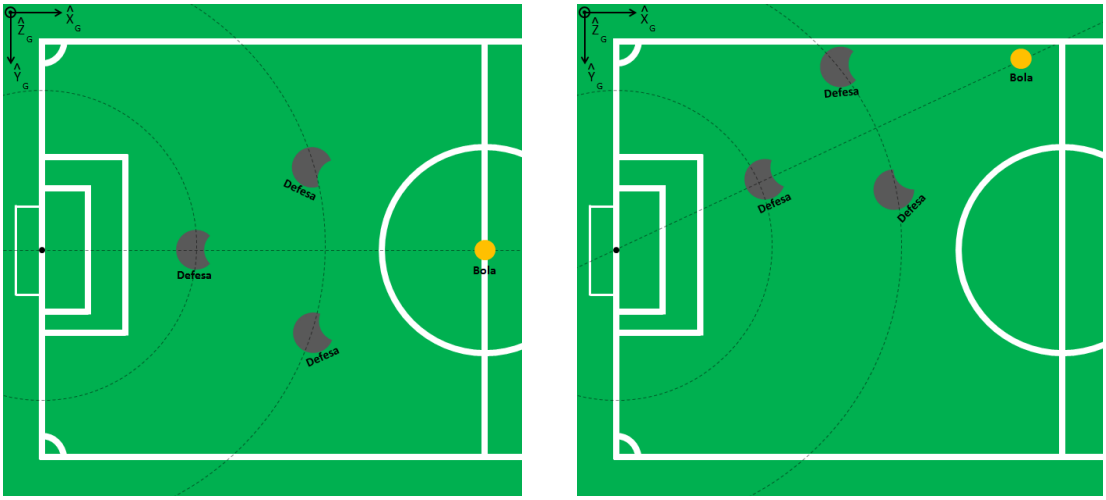


Figura 4.25 - Interceptação de bola

Para a movimentação de um ponto para outro, é utilizado o comportamento **ir para coordenada no campo**, e para a orientação é utilizado o **rodar**.

4.4.2. Defesa

O papel de um defesa é obstruir o caminho aos adversários para evitar possíveis remates. A forma de agir dos defesas é semelhante à do guarda-redes, pois deve inicialmente deslocar-se para uma posição base atribuída pela MBS. Depois, sempre que a posição da bola é conhecida, ajusta-se a posição base do defesa para que este acompanhe o deslocamento da linha imaginária formada pela bola e o centro da própria baliza (Figura 4.26).



a) b)

Figura 4.26 - Posição dos defesas: a) Situação inicial; b) Posição da bola modificada

Ao mesmo tempo que se move, caso a posição da bola seja conhecida, orienta-se sempre para esta. Os comportamentos utilizados são os mesmos do guarda-redes.

4.4.3. Atacante

Quando um robô está no papel de atacante deve tentar tirar a bola ao adversário, e quando na posse da mesma, deve tentar marcar golo. A máquina de estados da Figura 4.27 descreve o funcionamento do papel de atacante.

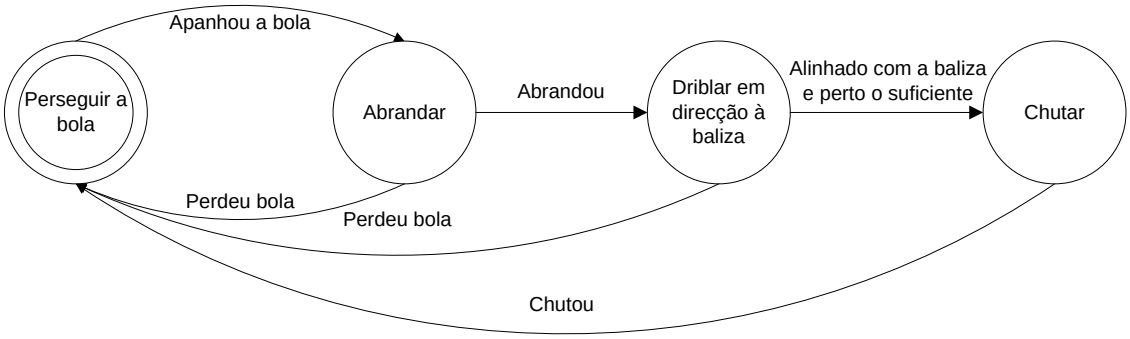


Figura 4.27 - Diagrama de estados do atacante

No estado inicial o robô tem de perseguir a bola, utilizando para isso os comportamentos **ir à bola** e **rodar**. Sempre que apanha a bola abranda suavemente, de forma a obter um melhor domínio sobre esta, e de seguida utiliza o comportamento **driblar** para se dirigir para a baliza do adversário evitando obstáculos, e tentar rematar. Para que seja possível efectuar o remate é necessário que o robô esteja alinhado com a baliza, e se encontre a uma distância alcançável (Figura 4.28). O remate é feito utilizando um chuto vertical, porque é o mais difícil para o guarda-redes adversário defender.

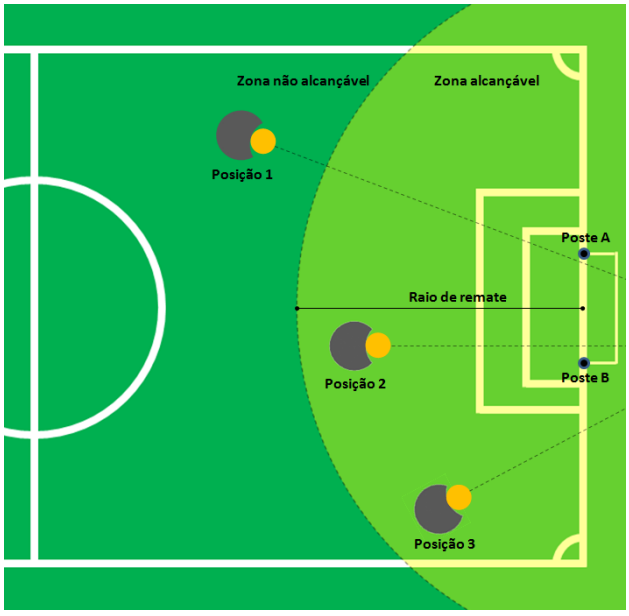


Figura 4.28 - Situações de remate

Na figura acima, o robô que se encontra na posição 1 não pode rematar, por não se encontrar na zona alcançável, mesmo estando alinhado com a baliza. O que se encontra na posição 3, está na situação oposta, ou seja, tem alcance de remate, mas não está alinhado com a baliza. Por fim, o robô da posição 2 pode rematar, pois encontra-se correctamente alinhado com a baliza, e tem alcance de remate.

4.4.4. Marcador

Sempre que ocorre uma situação de bola parada é atribuído o papel de marcador a um dos robôs da equipa. Este deverá posicionar-se atrás da bola, virado para o robô com papel de receptor, utilizando para isso o comportamento de aproximação de passe. De seguida, quando o árbitro der sinal, tem de passar a bola para o colega de equipa, seleccionando a força de chuto em função da distância entre os dois.

4.4.5. Receptor

O papel do receptor é posicionar-se num ponto do campo dado pela MBS, e esperar pelo sinal de passe do marcador.

A Figura 4.29 exemplifica uma situação de bola parada. Trata-se da marcação de um fora a favor da própria equipa. Primeiro, os robôs posicionam-se nas posições desejadas PM e PR, e quando o árbitro sinaliza, o marcador passa a bola para o receptor.

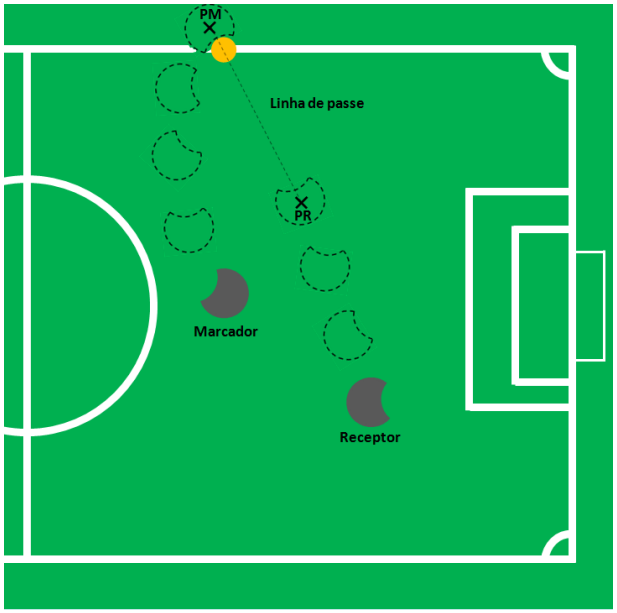


Figura 4.29 - Marcação de uma fora; PM - Posição desejada para o marcador; PR - Posição desejada para o receptor

Neste subcapítulo foram descritos os papéis que os robôs podem assumir. A escolha deste tipo de abordagem oferece alguma modularidade no desenvolvimento do sistema, pois a introdução, alteração, ou até remoção de papéis poderá ser feita facilmente.

4.5. Partilha de Informação

Para que cooperação entre os agentes seja eficiente, é necessário que estes partilhem alguma informação. A partilha é feita sempre através da MBS, pois não pode haver comunicação directa entre robôs. A Figura 4.30 ilustra como é feita a troca de informação, e quais as principais acções deste processo. A parte superior da figura descreve o processo de envio de comandos do árbitro para os robôs, no qual é feito o envio de um dos comandos listados em [**60**], fazendo com que os robôs tenham de mudar o seu estado interno. Na parte do meio, pode-se ver que cada robô da equipa envia a informação que adquire localmente, ou seja, através dos seus sensores, para a MBS, na qual essa informação é fundida [**59**] de forma a se obter uma melhor representação do ambiente, e enviada de volta para todos os robôs. Por fim, no fundo da imagem, ilustra-se o envio de papéis para os robôs, em que a MBS envia um papel para cada robô.

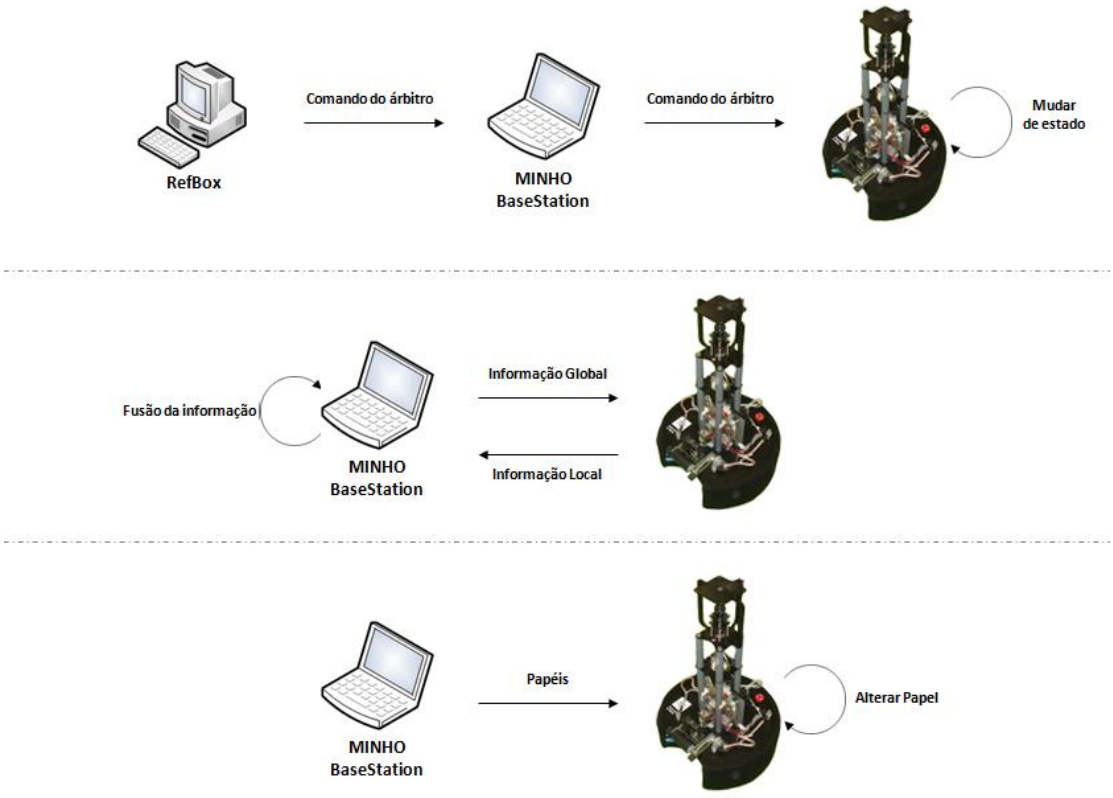


Figura 4.30 - Partilha de informação

A informação trocada entre a MBS pode ser representada através das tramas definidas em [**59**]. A trama 1 é utilizada para cada robô enviar a sua informação local para a MBS, e a trama 3 é utilizada para enviar para todos os robôs o resultado da fusão da informação. Já a trama 2 serve para a MBS enviar papéis, posições, e comandos do árbitro para os robôs.

Tabela 4-4: Tramas de comunicação [59**]**

Trama 1													
ID	Cod	Tipo	PosX	PosY	Dir	BolaX	BolaY	BolaV	BolaD	Estado	Papel	Bats	Vars
Descrição	Código da trama	Tipo de trama	Posição e orientação do robô no campo			Posição da bola no campo		Velocidade e direcção da bola		Estado do robô	Papel actual	Tensão das baterias	Variáveis internas

Trama 2								
ID	Cod	Tipo	Comando	PosX	PosY	Dir	Papel	PosX_Rec	PosY_Rec
Descrição	Código da trama	Tipo de trama	Comando da RefBox	Posição e orientação desejadas para o Robô			Papel do robô	Posição para receptor, só para bolas paradas	

Trama 3						
ID	Cod	Tipo	BolaX	BolaY	BolaV	BolaD
Descrição	Código da trama	Tipo de trama	Posição da bola		Velocidade e direcção da bola	

As tramas são enviadas utilizando o protocolo de comunicação UDP, pois pretende-se que a informação seja enviada o mais rapidamente possível, e a perda de um ou outro pacote não é crítico. O envio de informação à MBS é feito por cada robô de forma síncrona, a uma frequência cinco vezes inferior à de processamento, o que na configuração actual resulta numa frequência de aproximadamente 8 vezes por segundo. Já o envio de informação da MBS para os robôs é feito em média 10 vezes por segundo.

4.6. Camada de Decisão

Este subcapítulo explica a estrutura da camada de decisão da aplicação principal de cada robô.

A camada de decisão pode ser dividida em duas: camada de jogo e camada de situação. Em que na primeira, é decidido qual o estado do robô (Figura 4.31) em função do seu estado anterior e dos sinais provenientes da MBS, e na segunda, decidem-se quais os comportamentos que o robô deve executar em função do seu papel e/ou estado.

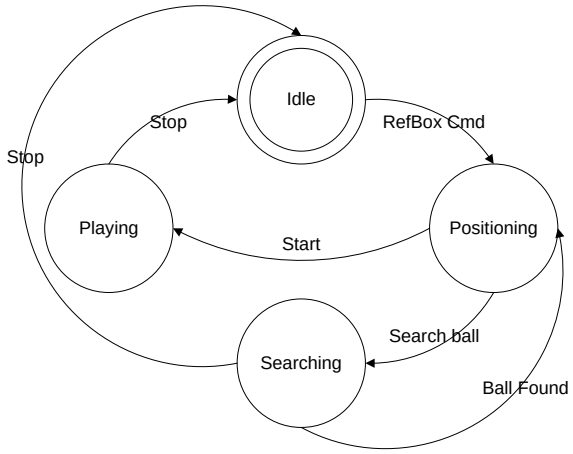


Figura 4.31 - Diagrama de estados da camada de jogo

A primeira camada de decisão é representada pelo diagrama de estados da Figura 4.31. Cada um dos seus estados contém internamente o seu próprio bloco de decisão, que em conjunto formam a camada de situação.

4.6.1. Idle

O estado **Idle** pode ser caracterizado como sendo neutro, isto é, aqui o robô realiza todo o processamento mas não se mexe, tendo de esperar por um comando da RefBox para transitar de estado. A entrada neste estado dá-se caso o robô esteja em **Searching** ou em **Playing**, e haja um sinal de **Stop**.

4.6.2. Positioning

No estado **Positioning**, o robô deve posicionar-se numa posição do campo de jogo dada pela MBS, utilizando para isso os comportamentos **ir para coordenada no campo**, **rodar** e **evitar obstáculos**. No caso de ter o papel **Marcador** utiliza também o comportamento **aproximação de passe**. O fluxograma da Figura 4.32 ilustra os passos que são efectuados neste estado.

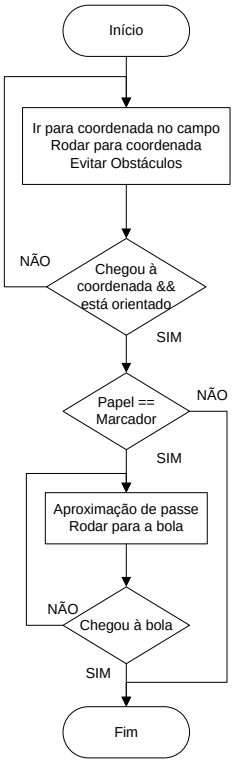


Figura 4.32 - Fluxograma do estado *Positioning*

Sempre que um robô se encontra neste estado e recebe o comando **SearchBall** da MBS, transita para o estado **Searching**. Por outro lado, se receber o comando **Start** transita para o estado **Playing**.

4.6.3. Playing

No estado **Playing** o robô terá que executar as tarefas associadas ao papel que lhe está actualmente atribuído. A permuta entre papéis é feita incondicionalmente, aquando da recepção de um comando da MBS para esse efeito (Figura 4.33). A entrada neste estado dá-se sempre que o árbitro envia o sinal de **Start** e o robô se encontra no estado **Positioning**. Se receber o sinal **Stop** transita para o **Idle**.

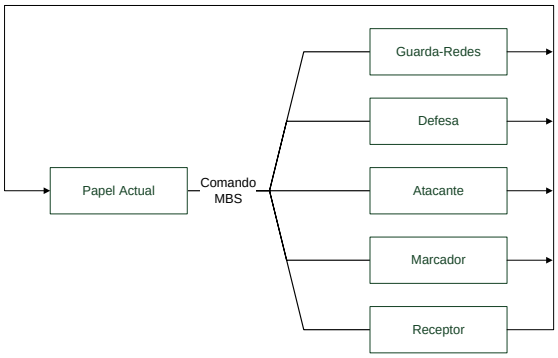


Figura 4.33 - Diagrama de blocos da alteração de papéis

4.6.4. Searching

Por fim, o estado **Searching** serve para que o robô procure a bola caso a posição desta não seja conhecida por nenhum robô da equipa. Esta procura é feita através de pontos-chave predefinidos, utilizando o comportamento **ir para coordenada no campo** (Figura 4.34). Durante o tempo de pesquisa, se a posição da bola voltar a ser conhecida, o robô volta para o estado **Positioning**.

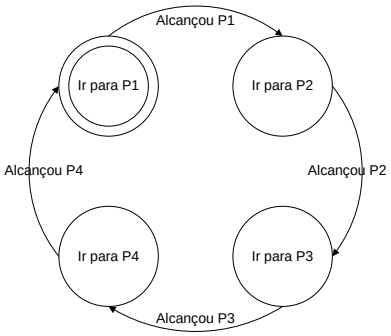


Figura 4.34 - Diagrama de estados de do estado *Searching,* P1, P2, P3, e P4 são os pontos-chave predefinidos

5.Análise de Resultados

Neste capítulo faz-se a análise quantitativa e qualitativa dos resultados obtidos com este trabalho. O espaço utilizado para a realização dos testes é uma das salas do Laboratório de Automação e Robótica da Universidade do Minho, a qual contém metade de um campo de jogo com medidas 14x10 m. A medição dos tempos de processamento dos vários algoritmos testados é feita utilizando as funções *cvGetTickCount* e *cvGetTickFrequency* da biblioteca *OpenCv*. O tempo final para cada algoritmo testado é obtido através do valor médio de 100 amostras. Os testes são realizados num sistema com um CPU dual core a 2GHz, 4GB de memória RAM, a correr o S.O. Ubuntu 10.04 de 32bits.

5.1. Aquisição de Dados e Pré-processamento

5.1.1. Aquisição de Imagem

De forma a escolher uma resolução que garanta um bom compromisso entre velocidade de processamento e qualidade de imagem, realizaram-se capturas com várias resoluções diferentes, e mediu-se o *frame rate* máximo de cada uma das configurações. As imagens adquiridas têm profundidade de oito bits, três canais de cor, e a velocidade máxima de transmissão é de 400Mb/s.

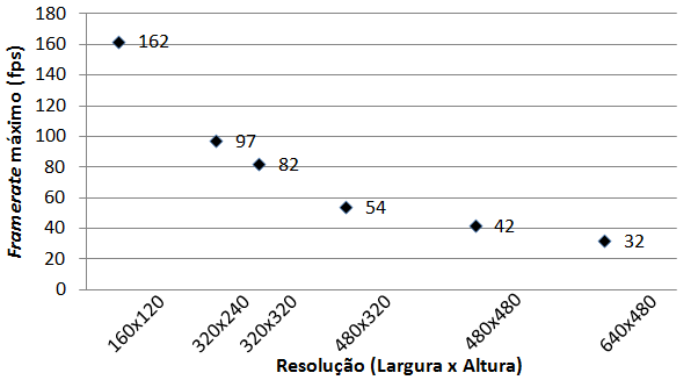
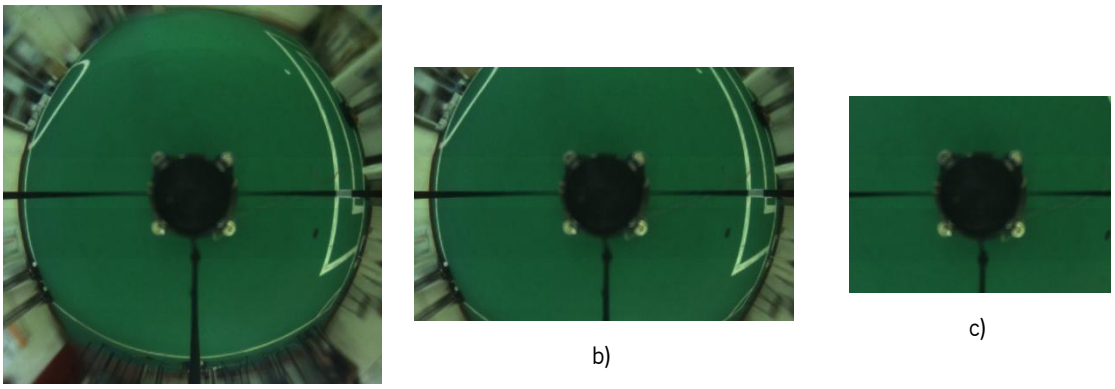


Figura 5.1 - *Frame rate* máximo de capturas com diferentes resoluções

Como se pode ver no gráfico da Figura 5.1, o *frame rate* máximo diminui de forma exponencial com o aumento da resolução da imagem, tornando as resoluções menores mais atractivas à partida. No entanto, quanto menor a resolução escolhida, menos informação se obtém (Figura 5.2).



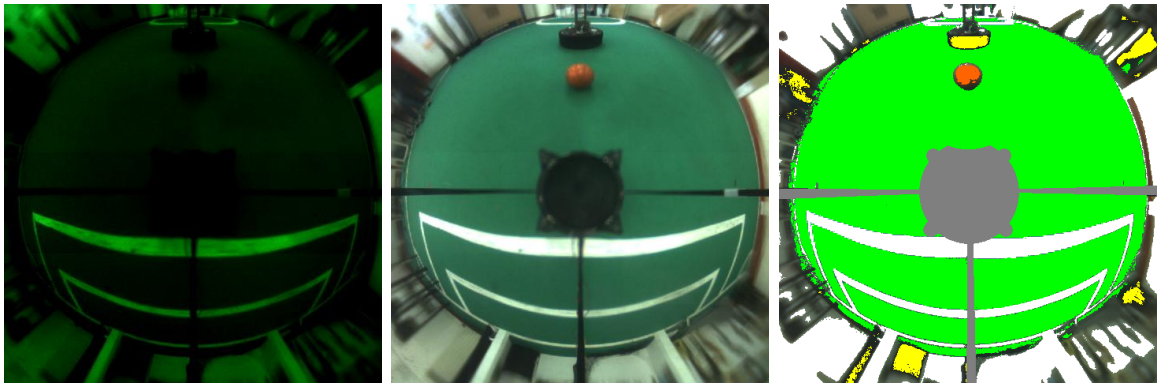
a) b) c)

Figura 5.2 - Exemplos de imagens capturadas com diferentes resoluções: a)480x480; b)480x320; c)320x320

A resolução escolhida foi a de 480x480 píxeis, pois consegue-se obter bastante mais informação do que nas imagens com resoluções inferiores, e ainda alcançar um *frame rate* razoável para a aplicação em questão.

5.1.1. Controlo de Qualidade da Imagem

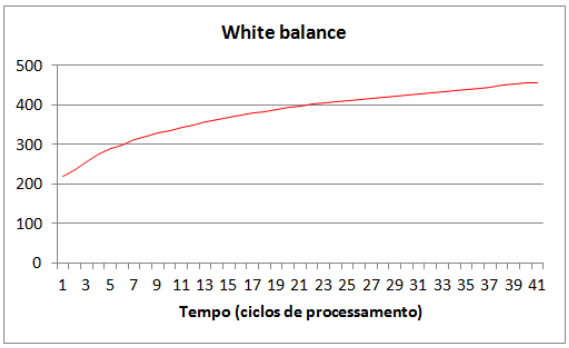
Para testar o sistema de controlo de qualidade da imagem, os parâmetros da câmara do robô foram inicializados de forma a se obter uma imagem com fraca qualidade (Figura 5.3 a)). Na Figura 5.3 b) vê-se a imagem que se obtém com os parâmetros corrigidos automaticamente, e na Figura 5.3 c) a correspondente segmentação.



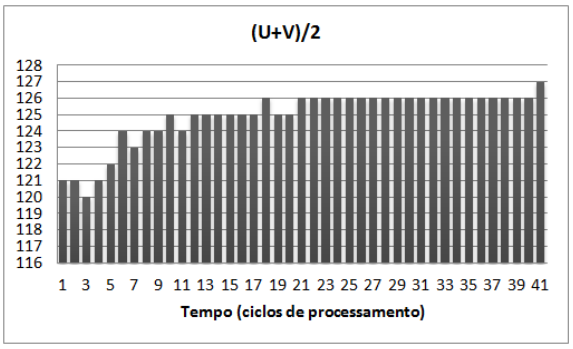
a) b) c)

Figura 5.3 - Teste do controlo de qualidade da imagem: a) Imagem com fraca qualidade; b) Imagem com boa qualidade; c) Imagem segmentada

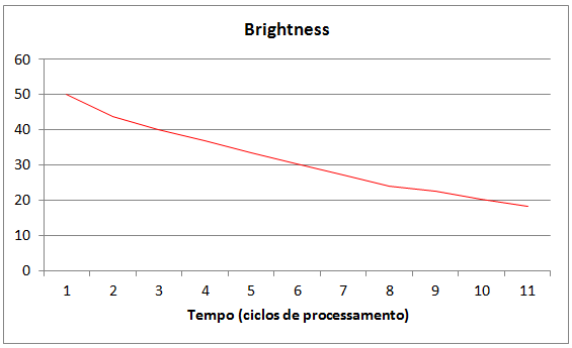
A seguir, na Figura 5.4 mostra-se a evolução temporal dos vários parâmetros controlados e das medidas associadas.



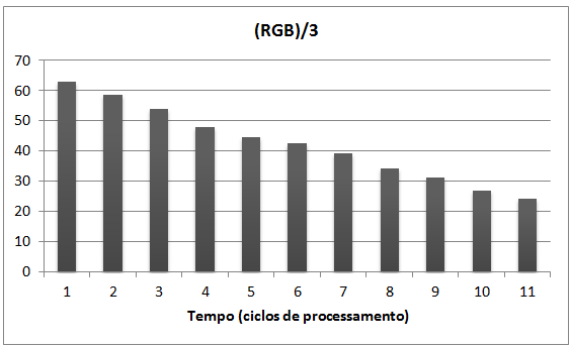
a) Evolução do *White balance*



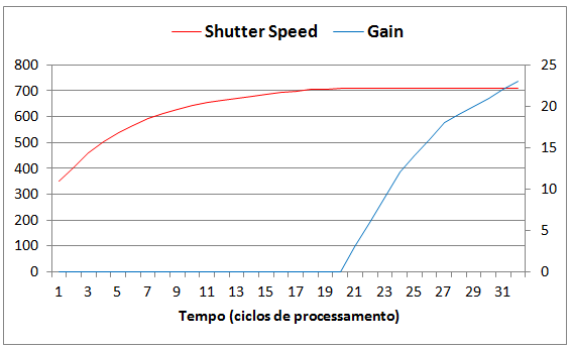
b) Média dos canais U e V da amostra de branco



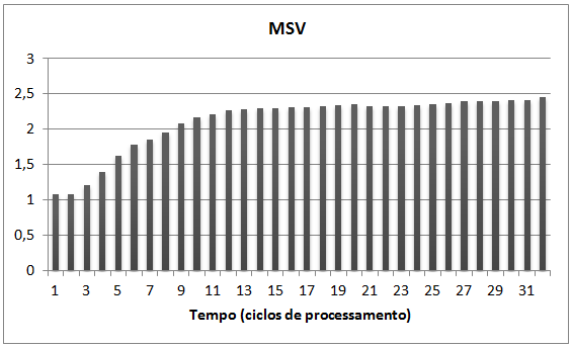
c) Evolução do *Brightness*



d) Média dos canais RGB da amostra de preto



e) Evolução do Shutter *speed* e do *Gain*



f) *Mean Sample Value*

Figura 5.4 - Gráficos do teste do controlo de qualidade da imagem

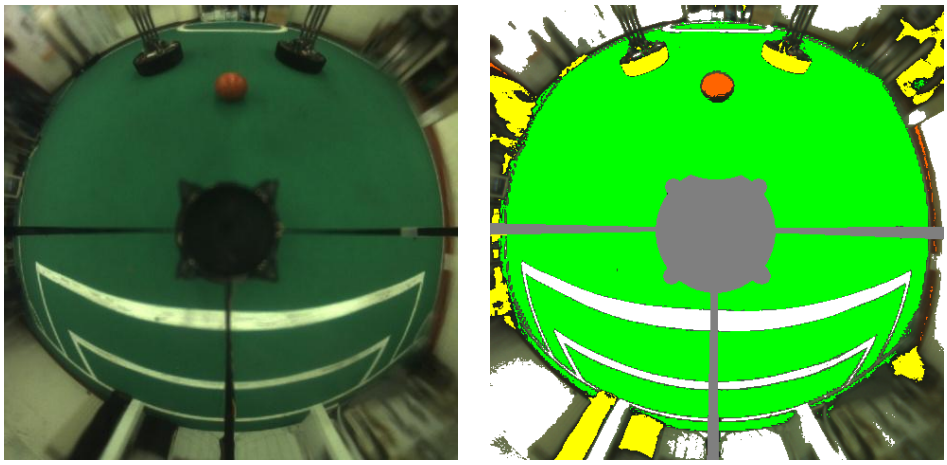
Para o parâmetro *brightness,* teoricamente o valor desejado para a média dos canais RGB é zero, mas neste sistema de visão isso origina imagens relativamente escuras. Para evitar estas situações foi definida uma histerese de 25 unidades. Na Figura 5.4 e) pode-se verificar que o *Shutter speed* satura antes que o MSV chegue ao valor desejado, nesse momento o valor do *Gain* começa a subir para corrigir essa situação.

Com esta calibração automática, em ambientes *indoor*, as cores das imagens são correctamente segmentadas.

5.1.2. Segmentação de Cores e Extracção de Características

Classificação de cores

A Figura 5.5 mostra o resultado da classificação de cores de uma imagem obtida pelo sistema de visão de um dos robôs. Do lado esquerdo da figura vê-se a imagem original capturada, e do lado direito, o resultado da classificação.



a) b)

Figura 5.5 - Classificação de cores: a) Imagem Original; b) Imagem segmentada

Os intervalos do espaço de cores HSV utilizados para a classificação de cada classe de cor estão presentes na Tabela 5-1. O tempo médio de processamento do algoritmo implementado é 2,1 ms.

Tabela 5-1: Intervalos do espaço de cores HSV

Classe	H mínimo	H máximo	S mínimo	S máximo	V mínimo	V máximo
Bola	0	32	64	100	25	100
Campo	120	163	41	70	21	70
Obstáculos	0	359	0	33	0	15
Linha	0	359	0	43	49	100

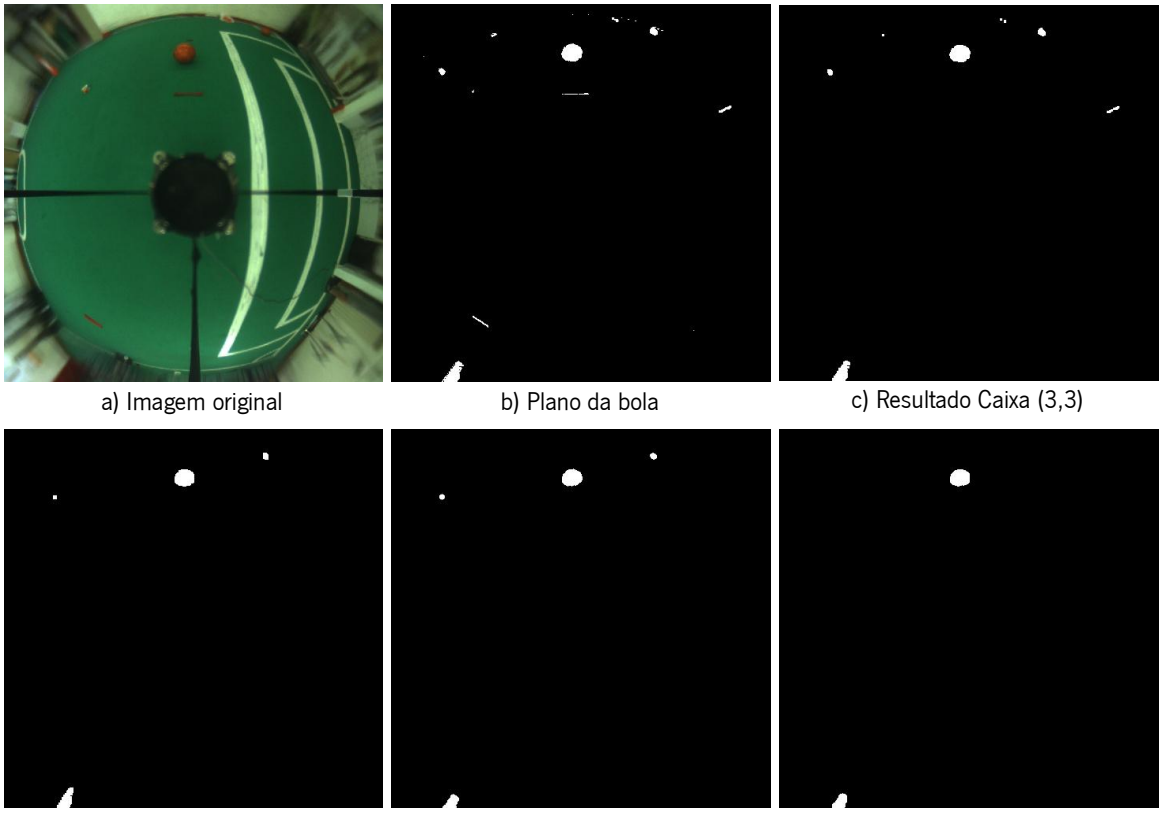
Filtragem morfológica do plano da bola

De forma a testar o filtro *open*, foi utilizada uma imagem com vários objectos da mesma cor da bola (Figura 5.6 a)). Ao ser extraído o plano da bola obtém-se a Figura 5.6 b). A Tabela 5-2 mostra o tempo de processamento do filtro aplicado ao plano da bola, com vários elementos estruturantes.

Tabela 5-2: Tempo de processamento do filtro *open*

Elemento Estruturante	Tempo (ms)
Caixa (3,3)	6,5
Disco (3)	6,6
Caixa (5,5)	8,8
Disco (5)	22
Caixa (7,7)	11
Disco (7)	42
Caixa (11,11)	16
Disco (11)	115

As Figura 5.6 c), d), e) e f) mostram o resultado do filtro para vários elementos estruturantes. Como se pode verificar, os E.E. caixa (3,3) e disco (3), não são adequados para esta situação, pois não filtram a maioria do ruído (Figura 5.6 c)). Os restantes oferecem resultados razoáveis, mas o caixa (5,5) é o mais rápido, e portanto a melhor escolha para este tipo de aplicação.



a) Imagem original b) Plano da bola c) Resultado Caixa (3,3)

d) Resultado Caixa (5,5) e) Resultado Disco (3,3) f) Resultado Caixa (11,11)

Figura 5.6 - Filtragem morfológica do plano da bola com filtro *open*

Identificação de componentes ligados no plano da bola

Para testar o algoritmo de detecção de *blobs* implementado, foram geradas imagens binárias com 0%, 25%, 50%, 75% e 100% de píxeis de plano de bola. Como seria de esperar, verifica-se que o tempo de processamento aumenta de forma linear com o aumento da percentagem de píxeis de plano de bola (Figura 5.7).

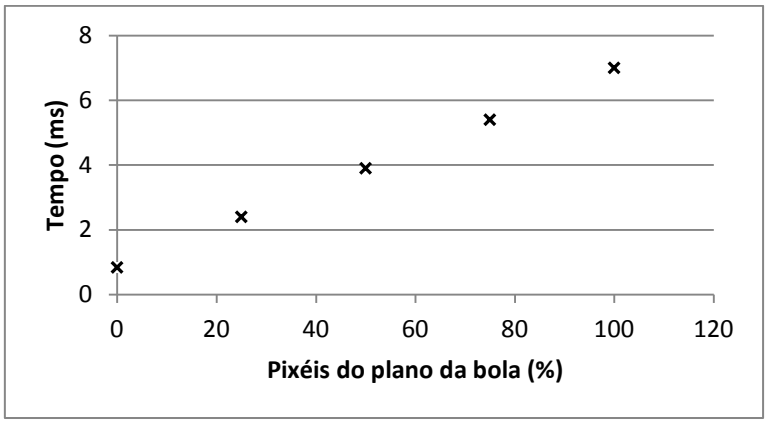
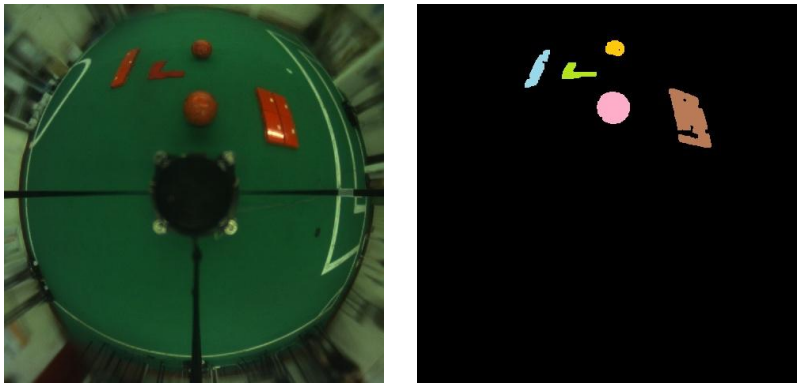


Figura 5.7 - Tempo de processamento do algoritmo de detecção de *blobs*

O resultado de um exemplo concreto deste algoritmo pode ser visto na Figura 5.8 b), na qual foram detectados cinco *blobs* distintos a partir do cenário da Figura 5.8 a), em cerca de 1 ms.

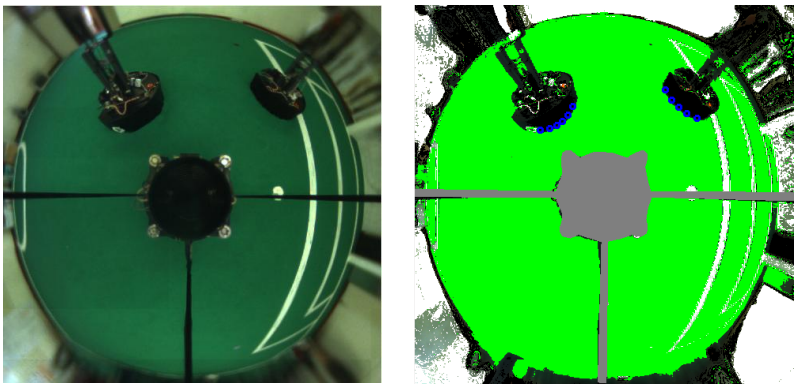


a) b)

Figura 5.8 - Detecção de *blobs*: a) Imagem original; b) *Blobs* encontrados

Detecção de contornos dos obstáculos

Na Figura 5.9 encontra-se o resultado do processo de detecção de contornos dos obstáculos. Os círculos azuis da Figura 5.9 b) marcam os contornos dos obstáculos detectados.



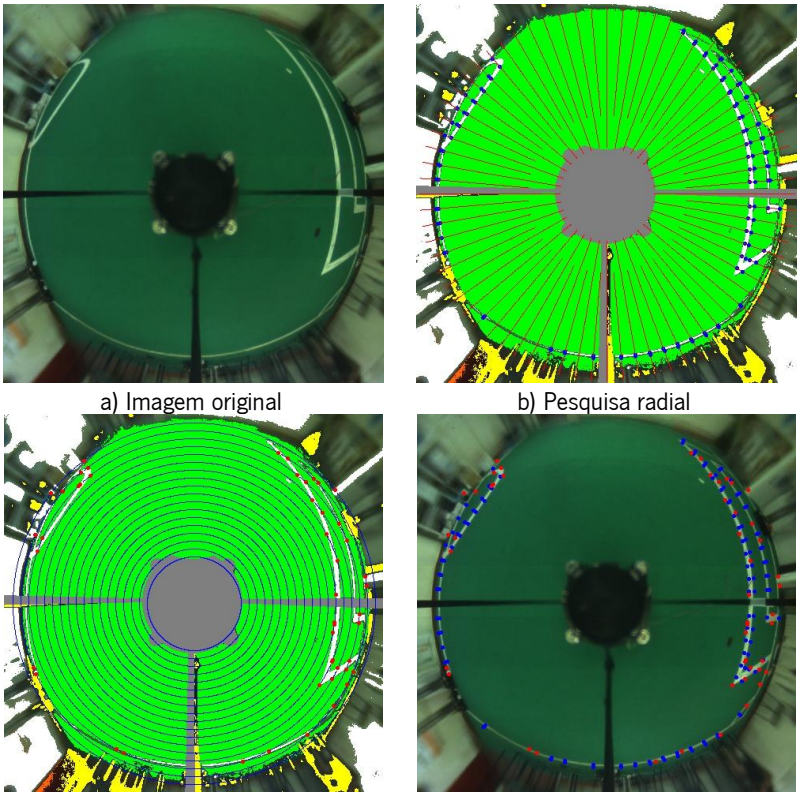
a) b)

Figura 5.9 - Detecção de contornos dos obstáculos: a) Imagem original; b) Imagem com os contornos dos obstáculos

Uma das limitações desta técnica é o facto de cada sensor detectar apenas o contorno do obstáculo mais próximo. No entanto, a sua simplicidade e velocidade de processamento (alguns microsegundos) fizeram com que fosse escolhida.

Detecção de linhas do campo

Para a Figura 5.10 a), o resultado da pesquisa radial das linhas de campo é mostrado na Figura 5.10 b), e o da pesquisa axial na Figura 5.10 c). A junção das transições obtidas através destas duas pesquisas está ilustrada na Figura 5.10 d).



a) Imagem original b) Pesquisa radial

c) Pesquisa axial d) Transições detectadas

Figura 5.10 - Detecção das linhas do campo

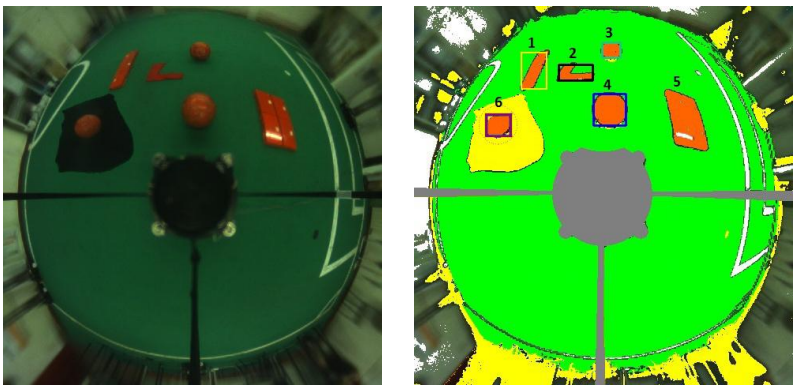
Os dados obtidos com este processo são bastante satisfatórios para entrada no sistema de localização, pois a maioria das linhas de campo visíveis na imagem são detectadas.

5.2. Modelação da Informação

5.2.1. Bola

Posição

Para demonstrar os resultados do algoritmo de detecção de bola, foi montado um cenário com vários objectos da cor da bola (Figura 5.11 a)). Para cada regra do algoritmo, existe pelo menos um objecto que não a satisfaz.



a) b)

Figura 5.11 - Detecção de bola: a) Imagem original; b) Imagem segmentada, com os possíveis candidatos a bola

Cada um dos objectos foi numerado, e à medida que passa cada regra do algoritmo, é desenhado o seu rectângulo envolvente de uma cor diferente. Na tabela seguinte mostram-se quais as regras que cada objecto deste exemplo satisfaz. Tanto o objecto nº 3 como o nº 4 respeitam todas as regras, mas como o nº 4 se encontra mais próximo do robô, é ele o escolhido para representar a bola de jogo.

Tabela 5-3: Regras a que obedecem os objectos representados na Figura 5.11

Nº Objecto	Regras				
	Massa	RMA	RLA	NPC	Mais Próximo
1	✔	✘	✘	✘	✘
2	✔	✔	✘	✘	✘
3	✔	✔	✔	✔	✘
4	✔	✔	✔	✔	✔
5	✘	✘	✘	✘	✘
6	✔	✔	✔	✘	✘

De forma a verificar a qualidade dos resultados da transformação da distância à bola em píxeis para centímetros, foi realizado um teste, no qual se faz com que a bola se desloque em linha recta a uma determinada distância do robô, e este guarda as distâncias a que vê a bola. A Figura 5.12 ilustra este tipo de teste feito para as distâncias de 1m, 2m e 3m.

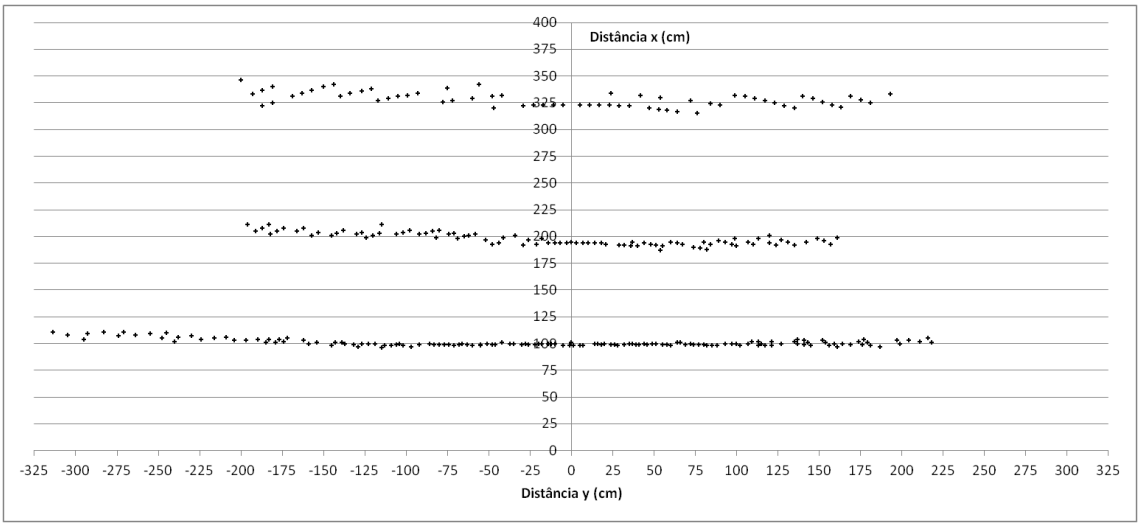
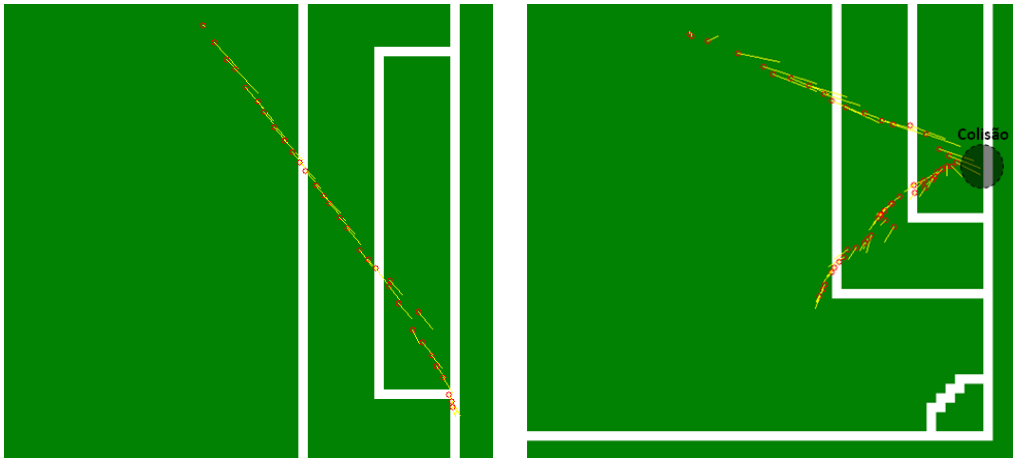


Figura 5.12 - Teste da posição da bola

A 3m, o erro chega a cerca de 15%, mas na prática não interfere de forma significativa nos comportamentos que utilizam esta informação.

Velocidade

A Figura 5.13 mostra o resultado de dois testes do algoritmo de detecção de velocidade da bola. O primeiro teste corresponde a uma situação em que a bola não colide com nenhum objecto, e o outro corresponde a uma situação em que a bola colide com um dos postes da baliza. Os dados foram obtidos por um robô colocado no centro da pequena área.



a) b)

Figura 5.13 - Teste velocidade bola: a) Movimento sem colisão; b) Movimento com colisão

De momento a informação sobre a velocidade da bola é utilizada para o guarda-redes calcular um ponto para interceptar a bola. Esta informação apenas é válida quando os robôs se encontram parados, pois estes ainda não dispõem de um sistema de medição da sua própria velocidade.

5.3. Comportamentos

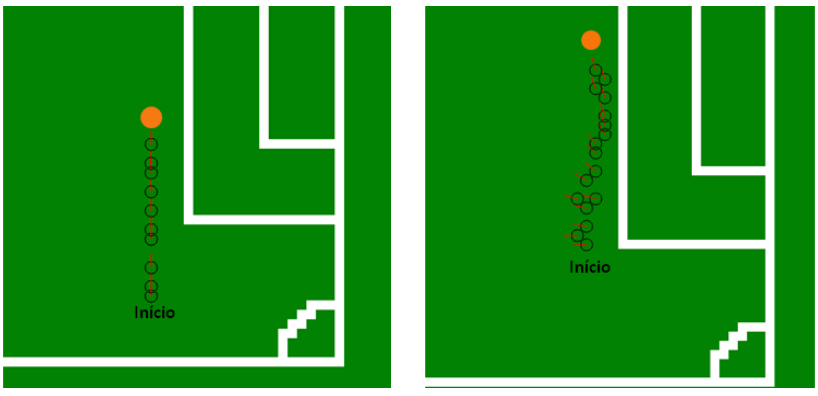
De forma a mostrar os resultados dos comportamentos desenvolvidos, foi guardada a posição e a orientação do robô em relação ao referencial do campo, utilizando o sistema de localização desenvolvido em [**58**].

As amostras da posição, orientação e velocidade desejada do robô foram adquiridas a cada cinco ciclos de processamento. O facto da resolução do sistema de localização ser 10 cm para a posição, e a existência de alguns campos magnéticos presentes no laboratório onde foram realizados os testes os quais interferem nas leituras da bússola, faz com que na ilustração de alguns dos resultados haja pequenos saltos na posição do robô.

Nas ilustrações seguintes, o trajecto percorrido pelo robô é representado por um conjunto de circunferências pretas (centro do robô). Sempre que necessário, a frente do robô é representada por um traço vermelho, a direcção de navegação desejada por um traço amarelo, a bola por um círculo laranja, e os obstáculos por círculos pretos.

5.3.1. Ir à bola

As Figura 5.14 a) e b) mostram duas situações de teste do comportamento **ir à bola**, em que em a) foi testado exclusivamente este comportamento, e em b), foi testado este em conjunto com o comportamento **rodar**.



a) b)

Figura 5.14 - a) Teste do comportamento *Ir à bola*; b) Combinação do comportamento *Ir à bola* com o *Rodar*

No teste em que se utilizou exclusivamente este comportamento, os parâmetros são os seguintes: σ=0,04, Δ=11 cm, e Vmax=35%. Para o teste em que se combinam os dois comportamentos, os parâmetros são os seguintes: σ=0,04, Δ=11 cm, Vmax=45%, e β=0,045,Wmax=30%.

No caso da Figura 5.14 a), o trajecto realizado é o mais curto, no entanto quando se junta a rotação (Figura 5.14 b), a trajectória deixa de ser rectilínea, fazendo com que a distância percorrida seja maior.

5.3.2. Ir para coordenada no campo

Para testar o comportamento **ir para coordenada o campo** são definidos à partida quatro pontos no campo de jogo, tendo o robô que se deslocar sequencialmente através de P1, P2, P3 e P4.

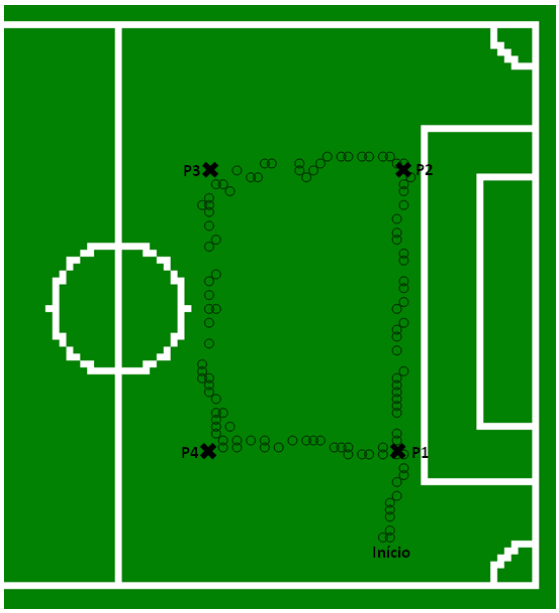


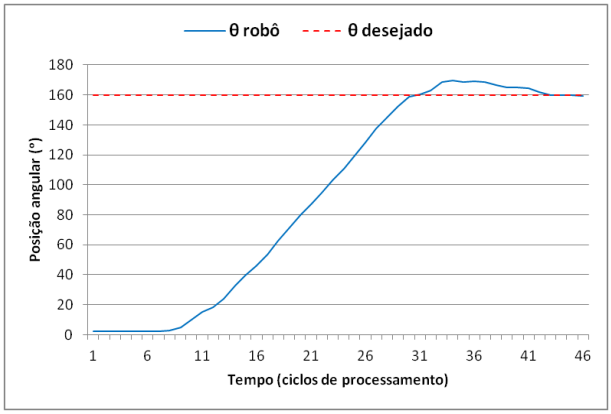
Figura 5.15 - Teste do comportamento *Ir para coordenada no campo*

Para este teste foram utilizados os seguintes parâmetros: σ=0,06, Δ=20 cm, e Vmax=50%.

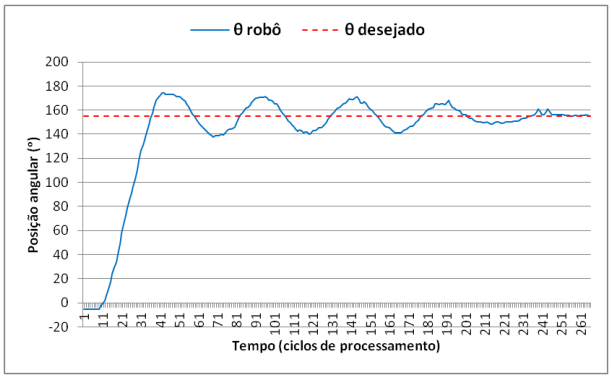
As posições calculadas pelo sistema de localização são mostradas na Figura 5.15. Na realidade, a trajectória descrita pelo robô durante este teste é bastante mais suave do que o ilustrado na figura acima.

5.3.3. Rodar

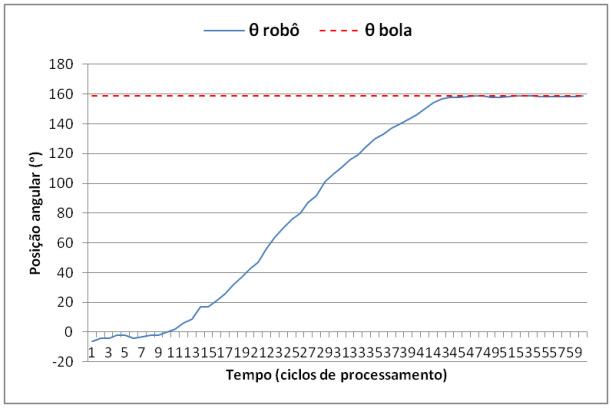
De forma a testar este comportamento, coloca-se a bola numa posição angular conhecida, e tem-se como objectivo alinhar a frente do robô com esta. Nos gráficos da Figura 5.16 a posição da bola é representada pelo θ desejado.



a)



b)



c)

Figura 5.16 - Testes do comportamento rodar: a) Situação 1; b) Situação 2; c) Situação 3

Na primeira situação, os parâmetros utilizados são: Wmax=50%, β=0,055; na segunda: Wmax=60%, β=0,04; e na terceira: Wmax=50%, β=0,04.

No teste da Figura 5.16 a) o tempo de estabelecimento é de aproximadamente 30 ciclos, mas a saída apresenta um pouco de *overshoot.* De forma a tentar diminuir o *overshoot* e o tempo de estabelecimento reduz-se ao ganho β e aumenta-se Wmax. O resultado é visível na Figura 5.16 b), em que a saída oscila. Para diminuir as oscilações diminui-se Wmax, e o resultado é o da Figura 5.16 c).

5.3.4. Evitar obstáculos

Para testar este comportamento, são colocados vários obstáculos no caminho entre duas coordenadas do campo conhecidas. Coloca-se o robô numa dessas coordenadas, e utiliza-se o comportamento i**r para posição no campo** para que este se desloque até à outra coordenada. Durante o percurso, o comportamento **evitar obstáculos** manipula a direcção de navegação de forma a evitar colisão. Na Figura 5.17 podem-se ver quatro situações deste tipo.

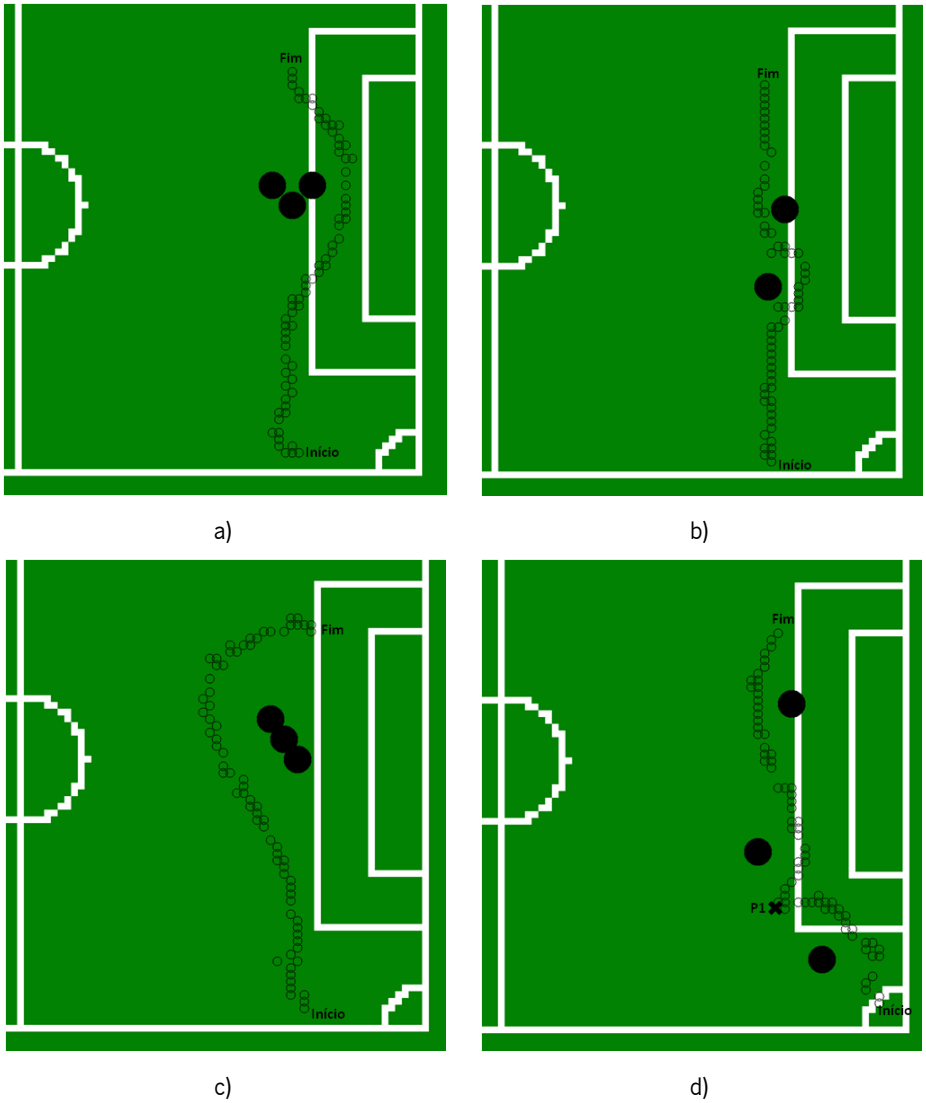


Figura 5.17 - Teste do comportamento *Evitar Obstáculos*: a) Três obstáculos em "v"; b) Dois obstáculos seguidos; c) Três obstáculos em diagonal; d) Três obstáculos aleatórios

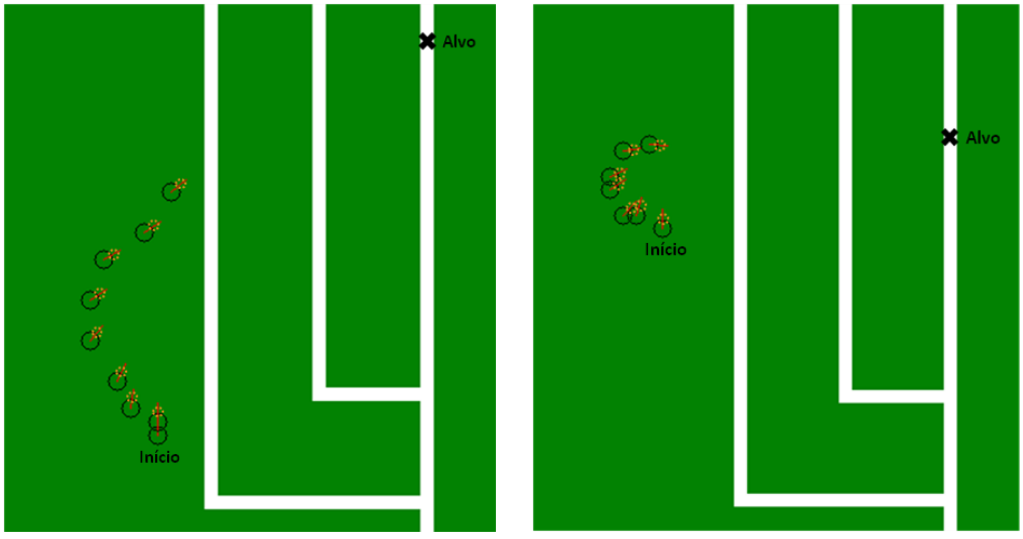
Os parâmetros utilizados nos quatro testes foram: σ=0,07, Δ=15 cm, e Vmax=45%.

Na primeira situação, são colocados três obstáculos em forma de "v" entre a posição inicial e a posição final do percurso. De seguida, na segunda situação, são colocados dois obstáculos bastante próximos um do outro, e o robô passa entre estes. Na terceira situação, são colocados três obstáculos em diagonal, entre a posição inicial e a posição final do percurso. No último teste, são colocados três obstáculos no meio do percurso, e existe um ponto intermédio pelo qual o robô tem forçosamente de passar.

Em todas as situações testadas, o robô foi capaz de se mover até à coordenada destino, sem colidir com os obstáculos.

5.3.5. Drift

O objectivo deste comportamento é fazer com que o robô mude de direcção de navegação na posse da bola sem a perder. Como se pode ver na Figura 5.18, é colocada a bola na posse do robô, e pretende-se que este se oriente na direcção do alvo.



a) b)

Figura 5.18 - Teste do comportamento *Drift*: a) Situação 1; b) Situação 2

Os parâmetros utilizados na primeira situação são: α=0,25, φ=0,045, Vref=40%, e Wmax=25%. E na segunda: α=0,25, φ=0,045, Vref=30%, e Wmax=40%.

Em ambos os casos, o robô consegue orientar-se para a direcção do alvo sem perder a posse da bola. Com o aumento das velocidades linear e angular, em determinadas situações não

é possível dominar a bola, pois o sistema de retenção de bola actual não suporta a força centrífuga exercida sobre esta.

5.3.6. Driblar

O cenário para testar este comportamento é o da Figura 5.19. Nesta, são ilustradas quatro situações nas quais o robô começa o movimento em posse da bola.

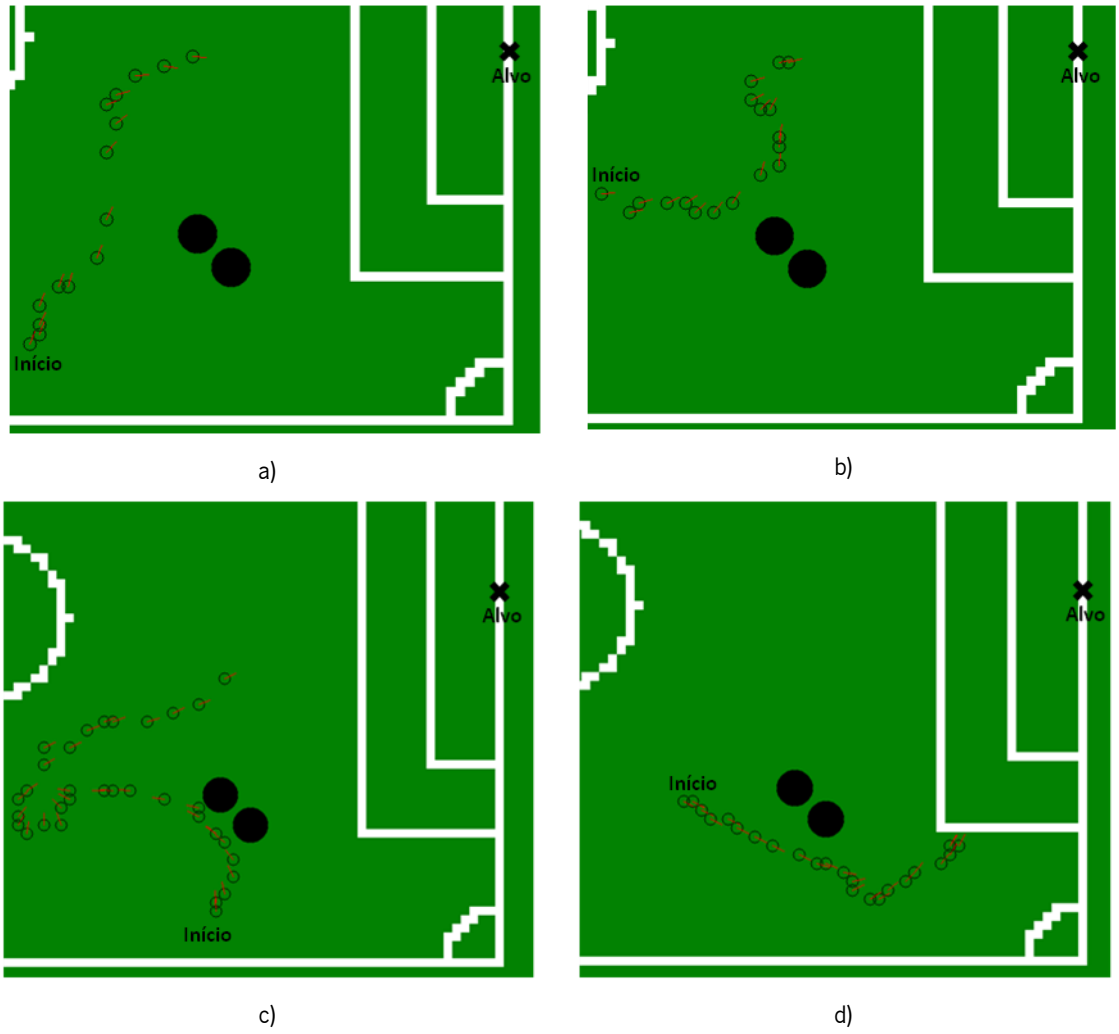


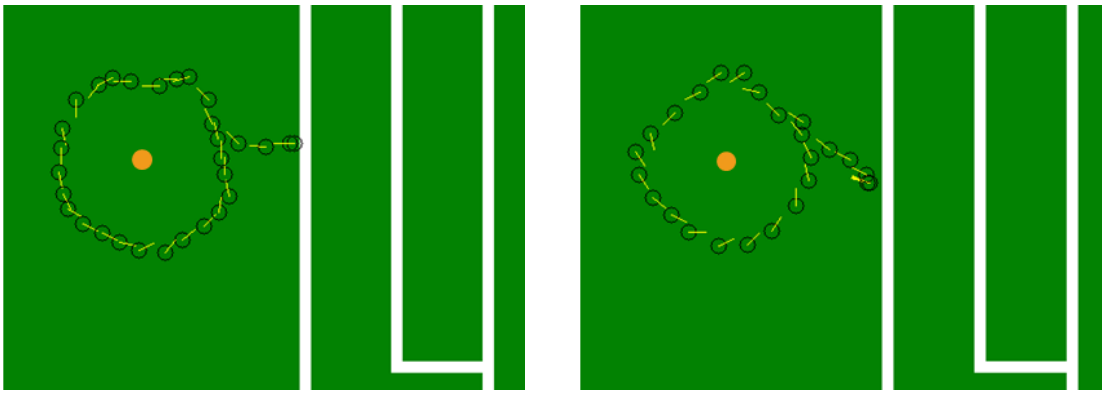
Figura 5.19 - Teste do comportamento *Driblar*: a) Situação 1; b) Situação 2; c) Situação 3; d) Situação 4

Os parâmetros utilizados nas quatro situações são os seguintes: α=0,25, φ=0,045, Vref=35%, e Wmax=40%.

Em todas as situações, verifica-se que o robô se desloca em direcção à baliza, desviando-se dos obstáculos sem perder a bola. No entanto, na situação 3 o trajecto realizado não é o ideal, devido ao facto do algoritmo não levar em conta a distância a percorrer.

5.3.7. Orbitar

Na Figura 5.20 mostra-se o resultado de duas situações de teste do comportamento **orbitar**.



a) b)

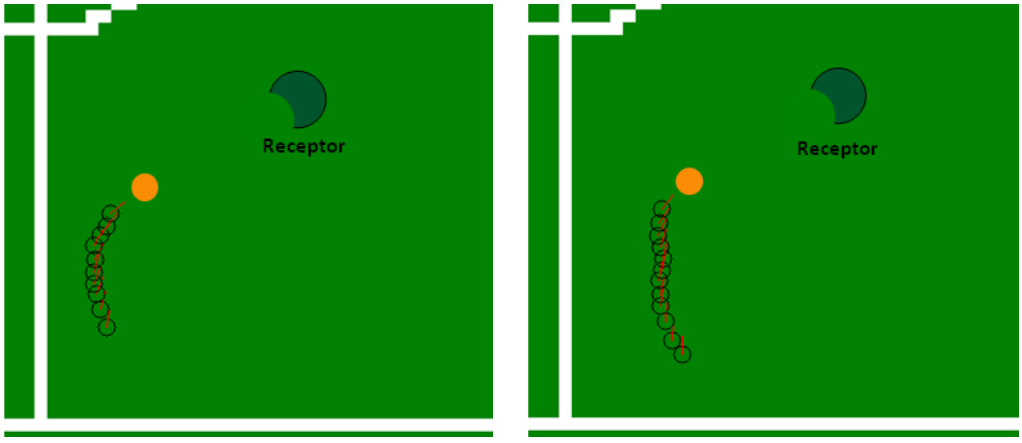
Figura 5.20 - Teste do comportamento *Orbitar*: a) Situação 1; b) Situação 2

Os parâmetros utilizados na primeira situação são: Vmax=40%, d=70cm, β=0,15, γ=0,1 e dir=*CCW.* E na segunda situação utilizam-se os seguintes: Vmax=40%, d=70cm, β=0,25, γ=0,1 e dir=*CCW.*

O aumento de β do primeiro para o segundo teste, faz com que a trajectória se torne menos circular, como se pode ver na Figura 5.20 b).

5.3.8. Aproximação de passe

A Figura 5.21 ilustra duas situações de teste deste comportamento. As coordenadas do receptor são transmitidas pela MBS para o robô que deve passar a bola, para que este possa aproximar-se da bola virado para o receptor.



a) b)

Figura 5.21 - Teste do comportamento *Aproximação de passe* a) Situação 1; b) Situação 2

No primeiro teste os parâmetros utilizados são: Vmax = 40%, α=1,5, β=1,0, φ=0,04, e δ=0,2. E no segundo são: Vmax = 50%, α=0,9, β=1,3, e φ=1,0, e δ=0,2.

Em ambos os casos o robô posiciona-se correctamente atrás da bola orientado para o receptor.

6. Discussão

Neste capítulo são discutidos os resultados obtidos com este trabalho

O sistema de controlo de qualidade da imagem implementado, tal como relatado na fonte [**52**] mostrou ser bastante fiável em ambientes *indoor*. O tempo de cálculo dos parâmetros, para condições iniciais nulas (pior caso), é de cerca de 1,5s, o que é perfeitamente aceitável visto que o algoritmo é executado em situações que o jogo se encontra parado.

A utilização de uma LUT para a classificação da cor dos píxeis é uma boa solução, pois permite acelerar bastante este processo. O espaço de memória ocupado é cerca de 1,56% da RAM do computador utilizado nos testes.

Para a bola ser correctamente detectada, esta deve ter uma cor predominante que contraste com o campo de jogo. A filtragem do plano da bola é feita recorrendo ao filtro morfológico *open*, o qual remove a maioria do ruído com sucesso. Devido ao baixo custo computacional e ao facto de a relação entre tempo de processamento e percentagem de píxeis do plano principal ser linear, a técnica para a identificação de componentes ligados utilizada revela-se uma boa solução para este tipo de aplicação. A detecção de obstáculos utilizando os sensores radiais virtuais permite baixar a área da imagem a analisar para cerca de 8%. No caso das linhas de campo, à pesquisa radial acrescenta-se uma pesquisa axial, e a soma dos píxeis a analisar pelas duas dá um total de 12,4%. Em ambos os casos, o tempo de processamento baixa drasticamente em relação à análise completa da imagem.

A posição da bola no campo de jogo é determinada com precisão suficiente para os robôs poderem interagir de forma correcta com esta. O erro máximo desta medida depende essencialmente da calibração do sistema omnidireccional de visão, e durante os testes realizados foi 15% a 3 m de distância. Já a velocidade da bola estimada é utilizada apenas pelo guarda-redes para obter o ponto de intersecção com a baliza, e actualmente este último é o único elemento que permite testar a exactidão da estimativa. Nos testes realizados, o guarda-redes moveu-se sempre para pontos do campo que se encontravam na trajectória da bola, no entanto, caso a bola fosse rematada com demasiada força, não chegava a tempo de a interceptar. A ausência de um sistema de visão estéreo no guarda-redes impossibilita estimar correctamente a velocidade da bola quando esta apresenta movimento vertical.

As trajectórias geradas envolvem baixo custo computacional comparativamente a algoritmos como o A*, no entanto, não é garantido que o percurso percorrido seja o mais curto. A bola pode ser incluída nos movimentos recorrendo ao comportamento *drift*, mas com algumas limitações de velocidade, nomeadamente, a velocidade linear não deve ultrapassar os 45%, e a velocidade angular os 50%. Para configurações acima destes valores existe uma grande probabilidade de o domínio de bola ser perdido. Isto deve-se principalmente ao facto de o sistema de retenção de bola actualmente utilizado não ser tão desenvolvido como o de outras equipas.

Na geração anterior de robôs da MinhoTeam, todos os robôs tinham a estratégia de jogo em memória. Nesta nova geração a estratégia de alto nível é implementada na MBS, o que retira a independência dos jogadores de campo, mas permite uma maior cooperação de equipa e diminui algum custo computacional de cada jogador.

7. Conclusões e Trabalho Futuro

7.1. Conclusões

Todos os objectivos propostos para este trabalho foram concluídos com sucesso.

O módulo de visão desenvolvido permite detectar de forma eficiente obstáculos, linhas de campo, e bolas com uma cor predominante. Com controlador de qualidade de imagem implementado é possível calcular automaticamente, em ambientes *indoor*, os parâmetros da câmara aproximadamente ideais para a posterior segmentação de imagem.

A arquitectura do módulo de controlo de movimento apresenta uma grande flexibilidade, pois é possível adicionar comportamentos e papéis de forma modular, sem afectar o resto do sistema. As trajectórias geradas permitem ao robô perseguir e driblar a bola, mover-se de um ponto do campo para outro, e desviar-se dos obstáculos, de forma eficiente.

Os papéis definidos, com o auxílio da MBS, tornam possível a cooperação entre robôs, e o conjunto de comportamentos desenvolvidos permitem a realização de todas as tarefas necessárias para efectuar um jogo da MSL.

As duas camadas de software que constituem o módulo de decisão, asseguram a correcta transição entre estados de jogo, e a selecção dos comportamentos apropriados para cada situação.

A execução dos três módulos implementados, mais o de localização, resulta em aproximadamente 40 ciclos de processamento por segundo.

7.2. Trabalho Futuro

O espelho do sistema de visão omnidireccional poderá ser redimensionado para se obter um maior alcance na detecção dos objectos de interesse. A médio prazo, o sistema de visão do guarda-redes deve ser expandido para estéreo, de forma a permitir a detecção do movimento da bola em 3D.

É importante o desenvolvimento de um sistema que permita detectar qualquer tipo de bola da FIFA (tamanho 5), através de técnicas como por exemplo a transformada de *Hough*.

Devido ao alto custo computacional deste tipo de algoritmos, sugere-se que sejam implementados num FPGA ou num CPLD.

De forma a tornar o drible de bola mais robusto, o sistema de retenção de bola deve ser melhorado, seguindo uma metodologia semelhante à descrita em [**61**].

A bússola utilizada pelo sistema de localização é bastante influenciada por campos magnéticos, daí é importante adicionar uma IMU de forma a reduzir o ruído através de fusão sensorial.

A informação disponibilizada pela IMU, em conjunto com a da odometria, poderá também ser utilizada para estimar a velocidade do robô.

Por fim, será interessante fazer a partilha da informação sobre os obstáculos para possibilitar a construção de um mapa global sobre as zonas do campo que estão ocupadas. Essa informação pode ser utilizada como entrada para algoritmos de planeamento de trajectórias mais complexos.

Referências

[1] The RoboCup Federation. (2010, Outubro) Middle Size League - RoboCup Federation Wiki. [Online]. http://wiki.robocup.org/wiki/Middle_Size_League

[2] The Robocup Federation. (2010, Outubro) Robocup website. [Online]. http://www.robocup.org

[3] (2010, Outubro) capek brothers website. [Online]. http://www.capekbrothers.net

[4] (2010, Outubro) Karel Capek website. [Online]. http://www.karelcapek.com

[5] (2010, Outubro) www.rci.rutgers.edu. [Online]. http://www.rci.rutgers.edu

[6] (2010, Outubro) Isaac Asimov webpage. [Online]. http://www.asimovonline.com

[7] (2010, Outubro) Robot Hall of Fame website. [Online]. http://www.robothalloffame.org

[8] www.almc.army.mil. (2010, Outubro) www.almc.army.mil. [Online]. http://www.almc.army.mil/alog/issues/MayJun08/bomb_eodpersonnel.html

[9] www.serbot.ch. (2010, Outubro) www.serbot.ch. [Online]. http://www.serbot.ch/images/cleanant_profi_climbing.jpg

[10] asia-robot.com. (2010, Outubro) http://asia-robot.com. [Online]. http://asia-robot.com/arcwelding.htm

[11] International Federation of Robotics. (2010, Outubro) International Federation of Robotics website. [Online]. http://www.ifr.org

[12] Erico Guizo. (2010, Outubro) IEEE Spectrum's robotics blog. [Online]. http://spectrum.ieee.org

[13] GeekAlerts. (2010, utubro) Rovio - The Wi-Fi Spy Robot. [Online]. http://www.geekalerts.com/rovio-the-wi-fi-spy-robot/

[14] Aberystwyth University. (2010, Outubro) Aberystwyth University - Idris. [Online]. http://www.aber.ac.uk/en/cs/research/ir/robots/idris/

[15] Thomas Ricker. (2010, Outubro) Engadget. [Online]. http://www.engadget.com/2008/06/20/meet-hasbros-ampbot-the-mother-of-all-rollys/

[16] Draganfly Innovations Inc. (2010, Outubro) Draganfly Innovations Inc website. [Online]. http://www.draganfly.com/uav-helicopter/draganflyer-x4/gallery/pictures/picture-39.php

[17] NowPublic. (2010, Outubro) MQ-8 Fire Scout Helicopter. [Online]. http://www.nowpublic.com/tech-biz/mq-8b-fire-scout-helicopter-uav

[18] sUAS News. (2010, Outubro) sUAS News website. [Online]. http://www.suasnews.com/wp-content/uploads/2011/02/scorpion-complete-1.jpg

[19] Diário de Noticias Ciência. (2010, Outubro) Página web do Diário de Noticias Ciência. [Online]. http://www.dn.pt/inicio/ciencia/interior.aspx?content_id=1782323&seccao=Tecnologia

[20] Instituto Superior de Engenharia do Porto - Lab. Sistemas Autónomos. (2010, Outubro) LSA - ROAZ II. [Online]. http://www.lsa.isep.ipp.pt/pages/roaz_roaz2.html

[21] Ben Hirschler and Tim Pearce. (2010, Outubro) Reuters website. [Online]. http://www.reuters.com/article/2009/03/20/us-robotfish-idUSTRE52J1RY20090320

[22] PROF. EDSON ROBERTO DE PIERI. (2002, Março) Curso de Robótica Móvel. Documento.

[23] Roland Siegwart and Illah R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Massachusetts, United States of America: Bradford, 2004.

[24] The Robocup Federation. (2010, Novembro) Robocup Wiki. [Online]. http://wiki.robocup.org

[25] Aldebaran Robotics. (2010, Novembro) Aldebaran Robotics website. [Online]. http://www.aldebaran-robotics.com/

[26] Robocup Rescue. (2010, Novembro) Robocup Rescue website. [Online].

http://www.robocuprescue.org

[27] Robocup@Home. (2010, Novembro) Robocup@Home website. [Online]. http://www.ai.rug.nl/robocupathome

[28] Porfírio Silva. (2010, Novembro) website do CiênciaHoje. [Online]. http://www.cienciahoje.pt/index.php?oid=32915&op=all

[29] Robocup 2010 Singapura. (2010, Novembro) Website do Robocup 2010 Singapura. [Online]. http://www.robocup2010.org/home_Gallery.php?id=9

[30] T. P. Nascimento et al., "5DPO'2011: Team Description Paper," no. Robocup, Janeiro 2011.

[31] A. J. R. Neves et al., "CAMBADA'2010: Team Description Paper," no. Robocup, 2010.

[32] CAMBADA Soccer Team. (2010, Dezembro) CAMBADA - Robots. [Online]. http://www.ieeta.pt/atri/cambada/robots.htm

[33] M. Gholipour et al., "MRL Middle Size Team: RoboCup 2011 Team Description Paper," no. Robocup, 2011.

[34] Carpe Noctem. (2010) Carpe Noctem Software Architecture 2010. Documento.

[35] H., Wagner, M., Reichle, R. Skubch. (2009) A language for interactive cooperative. Documento.

[36] José Fernandes, "Desenvolvimento de hardware e software para robô móveis," Universidade do Minho, Guimarães, Relatório de estágio 2006.

[37] Nuno Peixoto, "Desenvolvimento de Software aplicado ao Futebol Robótico Médio," Universidade do Minho, Guimarães, Relatório de estágio 2006.

[38] Farlex. (2010, Dezembro) The Free Dictionary. [Online]. http://encyclopedia2.thefreedictionary.com/color+space

[39] ChaosPro. (2010, Dezembro) ChaosPro website. [Online]. http://www.chaospro.de/documentation/html/paletteeditor/colorspace_rgb.htm

[40] Tech-FAQ. (2010, Dezembro) Tech-FAQ website. [Online]. http://www.tech-faq.com/hsv.html

[41] EncyclopediaPro. (2010, Dezembro) EncyclopediaPro website. [Online]. http://www.encyclopediapro.com/mw/YUV

[42] Navid Nourani-Vatani and Jonathan Roberts. (2010, Dezembro) Automatic Camera Exposure Control. Documento.

[43] Cambrige in Colors. (2010, Dezembro) www.cambridgeincolour.com. [Online]. http://www.cambridgeincolour.com/tutorials/white-balance.htm

[44] Václav Hlaváč. (2011, Janeiro) Image Segmentation. Documento.

[45] Stéphane Marchand-Maillet and Yazid M. Sharaiha, *BINARY DIGITAL IMAGE PROCESSING A Discrete Approach*.: Academic Press, 2000.

[46] Al Bovik, *The Essencial Guide to Image Processing*. United States of America: Academic Press, 2009.

[47] Linda Shapiro and George Stockman, *Computer Vision*., 2000.

[48] John J. Craig, *Introduction to Robotics Mechanics and Control*, 2nd ed.: Addison Wesley, 1989.

[49] D. Nakhaeinia, S. H. Tang, S. B. Mohd Noor, and O. Motlagh, "A review of control architectures for autonomous navigation of mobile robots," *International Journal of the Physical Sciences*, vol. 6, no. 2, pp. 169-174, Janeiro 2011.

[50] Arkin Ronald C., *Behavior-Based Robotics*. Cambridge, Massachusetts: The MIT Press, 1998.

[51] Damien Douxchamps. (2011, Janeiro) libdc1394: The API for IEEE1394 / Firewire camera.

[Online]. http://damien.douxchamps.net/ieee1394/libdc1394/

[52] António J. R. Neves, Bernardo Cunha, Armando J. Pinho, and Ivo Pinheiro, "Autonomous configuration of parameters in robotic digital cameras," in *4th Iberian Conference on Pattern Recognition and Image Analysis*, Póvoa de Varzim, 2009.

[53] OpenCVWiki. (2011, Fevereiro) OpenCV webpage. [Online]. http://opencv.willowgarage.com/wiki/

[54] www.cs.rit.edu. (2011, Fevereiro) Color Conversion Algorithms. [Online]. http://www.cs.rit.edu/~ncs/color/t_convert.html

[55] Dr. Andrew Greensted. (2011, Fevereiro) The Lab Book Pages. [Online]. http://www.labbookpages.co.uk/software/imgProc/blobDetection.html#code

[56] Fernando Ribeiro et al. (2011) Minho MSL-NG - a New Generation of soccer robots. [Online]. http://www.robotica.dei.uminho.pt/robocup/TDP2011.pdf

[57] David C. Swaim II. (2011, Feveiro) LinearRegression Class Implementation. [Online]. http://david.swaim.com/cpp/linregc.htm

[58] Bruno Pereira, "Localização de robô futebolista MINHO TEAM," Universidade do Minho, Guimarães, Tese de Mestrado 2011.

[59] Sérgio Silva, "Controlo e Monitorização da equipa de robôs futebolistas MINHO TEAM," Universidade do Minho, Guimarães, Tese de Mestrado 2011.

[60] MSL Technical Committee 1997–2010. (2010, Março) Middle Size Robot League Rules and Regulations for 2010. Documento.

[61] Jeroen de Best and René van de Molengraft, "An Active Ball Handling Mechanism for RoboCup," no. Robótica, Robocup, Controlo em tempo real, 2008.