

Ciência.Inovação
2010

Programa Operacional Ciência e Inovação 2010

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO



Dog's Life 2:

Robô Vigilante

Joana Isabel Loureiro Carneiro

Laboratório de Inteligência Artificial e Ciência de Computadores

Rua do Campo Alegre, 823, 4150 Porto, Portugal

Fevereiro de 2006

621.3(047.3)/
LEEC
2006/CARNj

Dog's Life 2:

Robô Vigilante

Joana Isabel Loureiro Carneiro

Trabalho de Final de Licenciatura em Engenharia Electrotécnica e de Computadores, da Faculdade de Engenharia da Universidade do Porto, realizado no âmbito do Programa Operacional Ciência e Inovação 2010, sob a orientação do Professor Doutor Luís Paulo Reis e do Mestre António Manuel Pereira.

Laboratório de Inteligência Artificial e Ciência de Computadores
Rua do Campo Alegre, 823, 4150 Porto, Portugal

Fevereiro de 2006

621 2(07.3) /CCCC 2006/CAWJ

Unit
Facultate de Psihologie

Nº 105170

CDU

Data 24 02 10

Resumo

Este trabalho propõe uma abordagem ao problema da utilização de um robô móvel como um elemento de segurança, sendo o robô ERS-7 da Sony a figura principal deste sistema. O ERS-7 funcionará como uma espécie de “cão de guarda” de uma área bem delimitada, tendo contudo algumas capacidades em nada semelhantes às encontradas nos cães comuns.

O ERS-7 é capaz de navegar autonomamente de um ponto para outro, tendo sempre noção se sabe ou não onde está, se precisa ou não de o saber e que medidas deve tomar no caso de não saber mas precisar. Para além disso, se o ERS-7 estiver a deslocar-se para um determinado ponto e deparar-se com um obstáculo, evita-o contornando-o de maneira a ter o caminho para o objectivo desimpedido. Mas a principal função de um cão de guarda é a de detectar a presença de pessoas, tarefa que o ERS-7 também executa. Agora, o que diferencia o ERS-7 dos cães comuns, quando se trata desta tarefa, é que, para além de detectar e assinalar esse facto, o robô envia a imagem do intruso captada com o auxílio da sua câmara, para uma aplicação exterior a si (a ser executada por exemplo num PC) que a armazena para futura consulta. Esta aplicação, que comunica com o ERS-7, permite também enviar comandos básicos para o robô, tais como iniciar e parar, e ver o que a câmara está a captar em tempo real.

Abstract

This paper proposes an approach to the problem of using a mobile robot as an element of security, being the ERS-7 Sony robot the main figure of this system. The ERS-7 will function as a “watchdog”, of a well delimited area, although having some capabilities that have nothing in common with the ones found in common dogs.

The ERS-7 is capable of autonomously navigate from a point to another, always having the notion if he knows or not where he is, if he needs to know it and what measures are needed to be taken in case he doesn't know where he is but needs to. Besides that, if the ERS-7 is going to a certain point and detects an obstacle, he avoids it turning around it so it will have the path to the objective clear. But the main function of a watchdog is to detect people, task that the ERS-7 executes. Now, what differentiates the ERS-7 from the common dog when it comes to this task, is that besides detect and signalize, the robot sends the collected intruder image to an exterior application (executed, for example, in a PC) that stores it for future consultation. This application, that communicates with the ERS-7, also allows to send basic commands to the robot, such as start and stop, and to see what the robot camera is receiving, in real time.

Aos Meus Avós

Índice

Lista de Figuras	xix
Lista de Tabelas	xxi
1. Introdução	1
1.1 Enquadramento.....	1
1.2 Motivação.....	2
1.3 Objectivos.....	2
1.4 Estrutura do Relatório.....	3
2. Robô ERS-7	5
2.1 Introdução.....	5
2.2 Hardware.....	6
2.3 Plataformas de Programação.....	7
2.4 Conclusões.....	9
3. ERS-7 como Sistema de Vigilância	11
3.1 Visão Geral.....	11
3.2 O Conceito.....	11
3.2.1 O Ambiente.....	12
3.2.2 As Tarefas.....	14
3.2.3 O Agente.....	14
3.2.4 A Aplicação Central de Vigilância.....	15
3.3 Perspectiva Global da Arquitectura.....	15
3.3.1 Código da equipa rUNSWift.....	15
3.3.2 Especificações das Características Globais da Arquitectura.....	16
3.4 Conclusões.....	17
4. Visão	19
4.1 Introdução.....	19

4.2	Princípios e conceitos do Módulo de Visão.....	19
4.3	Detecção e reconhecimento de postes	22
4.3.1	Encontrar poste.....	23
4.3.2	Verificações internas entre postes reconhecidos.....	25
4.4	Detecção de intruso.....	26
4.5	Detecção de movimento	27
4.6	Conclusões.....	29
5.	Localização	30
5.1	Estado da Arte	30
5.2	Visão Geral.....	31
5.3	Localização Relativa.....	32
5.4	Localização Global.....	33
5.5	Conclusões.....	35
6.	Comportamento	36
6.1	O Conceito.....	36
6.2	Localização Activa	38
6.3	Sistema de Navegação	39
6.4	Detecção e Desvio de Obstáculos.....	40
6.4.1	Detecção de obstáculos	40
6.4.2	Desvio de obstáculos.....	41
6.5	Conclusões.....	43
7.	Aplicação Central de Vigilância	45
7.1	Visão Geral.....	45
7.2	Conceito.....	45
7.3	Interface.....	47
7.4	Conclusões.....	49
8.	Testes e Análise de Resultados	51
8.1	Detecção de intruso.....	51
8.1.1	Calibração de cor.....	51
8.1.2	Rostos.....	53

8.2 Navegação	54
8.2.1 Sem obstáculos.....	54
8.2.2 Desvio de obstáculos.....	55
8.2.3 Determinação de coordenadas	58
8.2.4 Localização Activa.....	60
9. Conclusões e Perspectivas de Desenvolvimento	63
9.1 Conclusões.....	63
9.2 Perspectivas de Desenvolvimento	64
Referências Bibliográficas	66
Informações técnicas gerais	69
A.1 Lista de Hardware utilizado.....	69
A.2 Instalação de Software.....	69
A.2.1 OPEN-R SDK.....	70
A.2.2 Java 2 Platform, Standard Edition	70
A.3 Configuração do acesso à rede sem fios	71
Especificações externas do ERS-7	72
B.1 Medidas externas	72
B.2 Limitações dos movimentos das juntas a nível de Hardware	74
B.3 Limitações dos movimentos das juntas a nível de Software.....	76
Aibo, nota histórica	78
Calibração da cor	82
D.1 Preparação	82
D.2 Como calibrar	83
D.3 Classificação Manual.....	84
D.4 Ficheiro obtido.....	84
Vigilante.cc	85

Lista de Figuras

FIGURA 1: ROBÔ ERS-7.....	5
FIGURA 2: ANATOMIA DO ERS-7: (A) VISTO DE LADO, (B) VISTO DE CIMA E (C) LEGENDA.....	6
FIGURA 3: MEMORY STICK.....	6
FIGURA 4: ESQUEMA DO SISTEMA DE VIGILÂNCIA.....	12
FIGURA 5: ESTÍMULOS QUE O ERS-7 RECEBE DO MEIO AMBIENTE.....	13
FIGURA 6: ARQUITECTURA DA APLICAÇÃO.....	16
FIGURA 7: MUNDO VISTO PELO ERS-7.....	20
FIGURA 8: OBJECTO IDENTIFICADO PELO MÓDULO DE VISÃO.....	21
FIGURA 9: ESQUEMA GENÉRICO DOS POSTES UTILIZADOS.....	22
FIGURA 10: ASPECTO E DENOMINAÇÃO DE CADA UM DOS POSTES.....	22
FIGURA 11: PSEUDO-CÓDIGO DA FUNÇÃO FINDBEACONS().....	23
FIGURA 12: PSEUDO-CÓDIGO DA FUNÇÃO BEACONSANITIES().....	25
FIGURA 13: PSEUDO-CÓDIGO DO ALGORITMO DE DETECÇÃO DE MOVIMENTO.....	28
FIGURA 14 INPUTS E OUTPUTS DO MÓDULO DE LOCALIZAÇÃO.....	31
FIGURA 15 LOCALIZAÇÃO RELATIVA A UM POSTE.....	32
FIGURA 16 LOCALIZAÇÃO GLOBAL.....	33
FIGURA 17 DETERMINAÇÃO DE COORDENADAS X E Y.....	34
FIGURA 18 ESTÍMULOS VERSUS RESPOSTAS.....	36
FIGURA 19 COMPORTAMENTO DO ERS-7.....	37
FIGURA 20 MÁQUINA DE ESTADOS.....	38
FIGURA 21: PSEUDO-CÓDIGO DA LOCALIZAÇÃO ACTIVA.....	39
FIGURA 22 THE FAR SIDE DE GARY LARSON.....	39
FIGURA 23 SEQUÊNCIA DE PONTOS OBJECTIVOS NO PERCURSO DO ERS-7.....	40
FIGURA 24 ALGORITMO DE DESVIO DE OBSTÁCULO.....	42
FIGURA 25 ERS-7 CONTORNA OBSTÁCULO.....	43
FIGURA 26 TRANSFERÊNCIA DE DADOS.....	46
FIGURA 27 CASOS DE USO DA APLICAÇÃO CENTRAL DE VIGILÂNCIA.....	46
FIGURA 28 LAYOUT DA APLICAÇÃO CENTRAL DE VIGILÂNCIA.....	47
FIGURA 29 LAYOUT DA JANELA DE LIGAÇÃO À VISÃO DO ERS-7.....	48
FIGURA 30 LAYOUT DA JANELA CONTROLADA PELO ERS-7.....	48
FIGURA 31 ARMAZENAMENTO DO FICHEIRO QUE CONTÉM A IMAGEM ENVIADA PELO ERS-7.....	49

FIGURA 32 FORMA E TONALIDADES UTILIZADAS NO TESTE.....	52
FIGURA 33 AMOSTRA VISTA PELO ERS-7.....	52
FIGURA 34 OBSTÁCULOS UTILIZADOS, DA ESQUERDA PARA A DIREITA: FÁCIL, MÉDIO, DIFÍCIL DE CONTORNAR	56
FIGURA 35 ESQUEMA DAS CONDIÇÕES DOS TESTES	56
FIGURA 36 ERS-7 DESVIA-SE DE UM OBSTÁCULO DO TIPO FÁCIL.....	57
FIGURA 37 ERS-7 DESVIA-SE DE UM OBSTÁCULO DO TIPO MÉDIO	57
FIGURA 38 ERS-7 DESVIA-SE DE UM OBSTÁCULO DO TIPO DIFÍCIL.....	58
FIGURA 39 VISTA DE CIMA DO ERS-7	72
FIGURA 40 VISTA DE LADO DO ERS-7.....	72
FIGURA 41 VISTA DE FRENTE DA CABEÇA DO ERS-7.....	73
FIGURA 42 VISTA DE FRENTE DO ERS-7	73
FIGURA 43 VISTA DE LADO DA CABEÇA DO ERS-7	73
FIGURA 44 LIMITAÇÕES A NÍVEL DO MOVIMENTO DA BOCA, TILT 1, TILT 2 E CAUDA.....	74
FIGURA 45 LIMITAÇÕES A NÍVEL DO MOVIMENTO DAS PATAS.....	74
FIGURA 46 LIMITAÇÕES A NÍVEL DO MOVIMENTO DA CABEÇA EM RELAÇÃO AO CORPO E CAUDA	75
FIGURA 47 LIMITAÇÕES A NÍVEL DO MOVIMENTO DAS ORELHAS E DA PARTE INFERIOR DAS PATAS	75
FIGURA 48 PRIMEIRO PROTÓTIPO	78
FIGURA 49 PROTÓTIPO.....	79
FIGURA 50 MODELO ERS-110.....	79
FIGURA 51 MODELO ERS-210.....	80
FIGURA 52 MODELOS ERS-311 E ERS-312 (DA ESQUERDA PARA A DIREITA).....	80
FIGURA 53 MODELO ERS-220.....	80
FIGURA 54 ROBÔ ERS-7 DA SONY	81

Lista de Tabelas

TABELA 1: CONDIÇÕES APLICADAS A UM AGLOMERADO INDIVIDUAL	24
TABELA 2: CONDIÇÕES APLICADAS A DOIS AGLOMERADOS QUE POTENCIALMENTE FORMAM UM POSTE	25
TABELA 3: CONDIÇÕES APLICADAS A AGLOMERADOS QUE POTENCIALMENTE PERTENCEM A UM ROSTO.....	26
TABELA 4: CARACTERÍSTICAS DOS SENSORES DE DISTÂNCIA DO ERS-7.....	41
TABELA 5: RESULTADOS DO TESTE GAMAS DE TONALIDADE	52
TABELA 6: RESULTADOS DOS TESTES A PRIMEIRO CONJUNTO DE ROSTOS	53
TABELA 7: RESULTADOS DOS TESTES A SEGUNDO CONJUNTO DE ROSTOS	54
TABELA 8: RESULTADOS DOS TESTES AO ALGORITMO DE DESVIO DE OBSTÁCULOS	55
TABELA 9: RESULTADOS DOS TESTES AO ALGORITMO DE DESVIO DE OBSTÁCULOS	57
TABELA 10: RESULTADOS DO TESTE DE COORDENADAS PARA O PONTO (85, 85).....	59
TABELA 11: RESULTADOS DO TESTE DE COORDENADAS PARA O PONTO (40, 85).....	59
TABELA 12: RESULTADOS DO TESTE DE COORDENADAS PARA O PONTO (85, 40).....	59
TABELA 13: RESULTADOS DOS TESTES AO ALGORITMO DE LOCALIZAÇÃO ACTIVA	61
TABELA 14: LIMITAÇÕES DAS JUNÇÕES ÚNICAS.....	76
TABELA 15: RELAÇÃO ENTRE AS DUAS JUNÇÕES (J1 E J2) DAS PERNAS DA FRENTE E DE TRÁS.....	77
TABELA 16: RELAÇÃO ENTRE O MOVIMENTO DA BOCA E DAS JUNÇÕES DO PESCOÇO.....	77
TABELA 17: ARQUIVOS NECESSÁRIOS PARA PODER INICIAR A CALIBRAÇÃO DE CORES.....	82

Capítulo 1

1. Introdução

1.1 Enquadramento

O trabalho descrito neste projecto liga um agente¹ artificial, autónomo e móvel ao desempenho de uma tarefa complexa – vigiar um dado perímetro, como se tratasse de um cão de guarda.

Neste contexto, as temáticas de maior relevância estudadas e abordadas neste relatório são:

- Navegação autónoma em ambientes não controlados;
- Detecção e reconhecimento de objectos;
- Detecção de pessoas;
- Localização espacial;
- Comunicação entre robô e PC.

Os tópicos, acima apresentados, podem ser por si só alvo de prolongado estudo e tornar-se tão complexos e morosos no seu desenvolvimento conforme os objectivos e o grau de robustez que se pretendam, contudo, no contexto deste projecto, teve-se em linha de vista estudar e implementar uma aplicação que não incidisse especialmente em nenhum dos pontos anteriores mas que permitisse perceber a importância de cada um deles no contexto global.

¹Apesar da definição de Agente não ser consensual podemos considerar tratar-se de “uma entidade computacional (vulgarmente *software*), situado num dado ambiente, que tem a percepção desse ambiente através de sensores, tem capacidade de raciocínio e age de forma autónoma nesse ambiente através de actuadores, de forma a desempenhar uma dada função para a qual foi projectado” [Reis, 2003].

1.2 Motivação

As questões ligadas à segurança têm vindo a adquirir cada vez mais importância, num mundo em que as pessoas se sentem cada vez mais insatisfeitas com as medidas tomadas para a protecção dos seus bens tanto intelectuais como materiais. Paralelamente, têm-se vindo a assistir a uma crescente atribuição de tarefas, tradicionalmente desempenhadas pelo homem, a robôs capazes de agir autonomamente, pelas vantagens que podem ter, como por exemplo:

- Desempenho interrompido por longos períodos de tempo;
- Desempenho metódico mas capaz de adaptação a situações de excepção;
- Possibilidade de melhoria constante do desempenho;
- Comunicação quase instantânea com outros dispositivos.

Nomeadamente no caso da tarefa de vigiar uma dada área, todas estas vantagens podem ser uma mais valia, podendo ainda ser uma solução de baixo custo para este problema. Assim sendo, parece fazer todo o sentido estudar a possibilidade de apresentar como alternativa à vigilância efectuada por humanos a vigilância realizada por um robô móvel.

1.3 Objectivos

O objectivo deste projecto é desenvolver um sistema de segurança cujo principal interveniente seja o modelo ERS-7, do conhecido robô AIBO da Sony.

Para o modelo ERS-7 deverá ser desenvolvida e implementada uma aplicação que lhe permita actuar como vigia de um dado perímetro. O sucesso do desempenho desta tarefa irá implicar a satisfação de outros objectivos bem mais concretos, que serão:

- Implementação de um **sistema de navegação**, de modo a que o ERS-7 seja capaz de se deslocar autonomamente de um objectivo para outro, detectando e evitando os obstáculos que lhe apareçam pelo caminho;
- Ajuste de algoritmos de processamento de imagem já existentes de modo a permitir a **deteção e reconhecimento de objectos** necessários ao bom funcionamento da aplicação, e à **deteção de intrusos**;
- Implementação de algoritmos de **deteção de movimento** numa imagem, funcionando como outra forma de deteção de intrusos;
- Implementação de algoritmos de **localização espacial** (tanto relativa como global), com o mínimo de erro possível e que sejam um bom compromisso entre simplicidade e eficiência;

- Inclusão no ERS-7 de **capacidades de comunicação com o exterior, enviando e recebendo informação**. Espera-se que o ERS-7 seja capaz de enviar imagens para uma aplicação a ser executado num PC externo, e que seja capaz de receber comandos da mesma.

Este último objectivo, pressupõe a implementação de uma outra aplicação denominada Aplicação Central de Vigilância e que tem os seus próprios objectivos, que se passam a especificar:

- Monitorização, utilizando a câmara do ERS-7, da área a vigiar;
- Comando do robô ERS-7 à distância (comandos básicos de parar e iniciar/reiniciar);
- Armazenamento das imagens enviadas pelo ERS-7 referentes aos intrusos detectados.

Ao longo dos capítulos seguintes vão ser apresentadas em pormenor as soluções desenvolvidas para que o projecto fosse ao encontro destes objectivos, tendo-se procurado sempre estudar as soluções apresentadas noutros estudos ou projectos, de modo a perceber por onde dada temática tem evoluído, não esquecendo nunca as características específicas do ERS-7 (que se tentaram potenciar).

1.4 Estrutura do Relatório

O relatório está dividido em 9 capítulos, sendo este o primeiro dos capítulos, constituindo uma breve introdução ao trabalho desenvolvido.

O segundo capítulo é dedicado à apresentação resumida das características físicas e electrónicas do ERS-7 e algumas das ferramentas consideradas como possíveis para utilizar na programação da aplicação a desenvolver para o ERS-7.

O terceiro capítulo realiza a junção entre o ERS-7 e a possibilidade de o tornar num sistema de vigilância. É apresentado o conceito geral proposto para o sistema de vigilância a implementar, concretizando-se em seguida a ideia, esquematizando-se a sua arquitectura. O código da equipa de futebol robótico rUNSWift² é referenciado neste capítulo, como o código base utilizado para a implementação do projecto presente.

² Este código inclui funcionalidades básicas de locomoção e processamento de imagem. A equipa rUNSWift da Universidade de New South Wales, Sidney, Austrália, foi campeã do mundo de futebol robótico (liga de robôs com pernas) em 2000, 2001 e 2003.

O quarto capítulo é inteiramente dedicado ao módulo de Visão, uma vez que se trata de um dos módulos mais importantes para o bom funcionamento desta aplicação. Este capítulo é iniciado pelos principais conceitos envolvidos no reconhecimento de objectos, seguindo-se os detalhes de implementação da detecção e reconhecimento de objectos de interesse neste projecto, da detecção de pessoas e do algoritmo de detecção de movimento.

No quinto capítulo é apresentado o módulo de Localização em detalhe. Existem dois métodos de localização a serem utilizados na aplicação e que são o alvo deste capítulo: a localização global e a localização relativa (em que o robô sabe qual a sua posição em relação a um dado poste de referência).

O sexto capítulo discute o módulo de Comportamento responsável pelas tomadas de decisão do robô. As principais acções do robô, neste módulo, são sem dúvida a navegação, detecção e desvio de obstáculos, localização activa e a sinalização da detecção de presença humana.

O sétimo capítulo é dedicado em exclusivo à comunicação do ERS-7 com a Aplicação Central de Vigilância. Para além de se abordar o modo como se processa transferência de dados e que dados estão envolvidos, é apresentada, em detalhe, a Aplicação Central de Vigilância.

O oitavo capítulo apresenta os testes efectuados aos algoritmos desenvolvidos para este projecto e a análise dos resultados. O capítulo procura validar a abordagem utilizada de modo a que se possa perceber quais as potencialidades e limitações dos algoritmos implementados. Procura também ajudar, através da análise dos resultados obtidos, a definir qual o trabalho futuro que poderá fazer sentido neste projecto com vista à sua melhoria.

O nono e último capítulo contém as conclusões gerais deste trabalho e apresenta algumas perspectivas de desenvolvimentos futuros.

2.2 Hardware

O ERS-7 não é disponível numa variedade de acessórios e actuadores que lhe permitem interagir com o meio ambiente (ver figura 2). Algumas das características mais relevantes a nível de hardware do ERS-7 são apresentadas em seguida:

Capítulo 2

2. Robô ERS-7

"The first decade of the 21st century will be dominated by robots"

Toshitada Doi, Vice Presidente Executivo, inventor do AIBO, Sony para repórteres da Reuters

2.1 Introdução

O ERS-7 foi lançado em 2003 e deixou para trás quatro modelos base e dois protótipos³ [Sony, 2005]. Apesar de toda a evolução sofrida ao longo dos inúmeros modelos desenvolvidos, todos têm em comum a denominação comercial de AIBO, que deixa pressupor que foram criados para algo mais do que mero entretenimento. AIBO é o resultado da junção de A.I. (*Artificial Intelligence*), *Eye* e *Robot*. Em japonês a palavra AIBO quer também dizer “companheiro”.



Figura 1: Robô ERS-7

Ao longo deste capítulo serão analisadas as características do ERS-7, que fazem dele um robô cheio de potencialidades e que fizeram com que fosse tão interessante para o desenvolvimento deste projecto.

³ Para mais dados referentes à história do AIBO, modelos anteriores e evolução, consultar o Anexo C.

2.2 Hardware

O ERS-7 põe à disposição uma considerável variedade de sensores e actuadores que lhe permitem interagir com o meio ambiente (ver figura 2). Algumas das características mais relevantes a nível de hardware do ERS-7 são apresentadas em seguida.

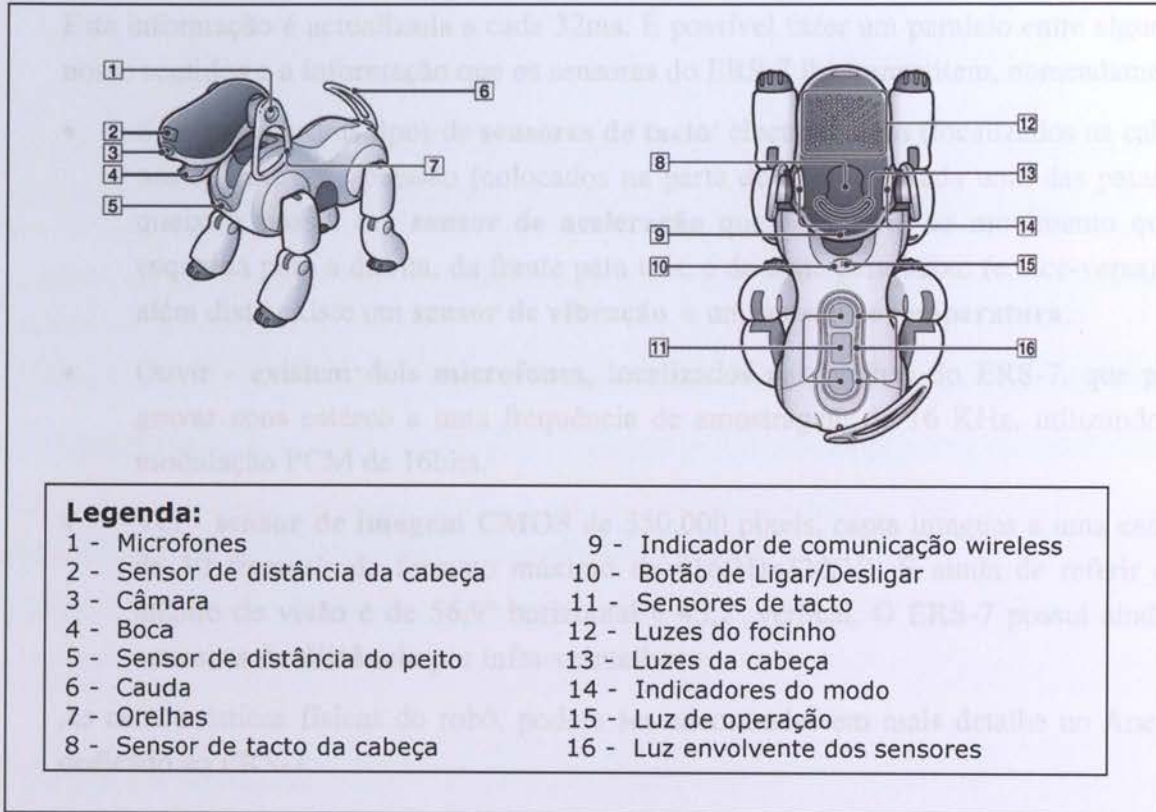


Figura 2: Anatomia do ERS-7: (a) visto de lado, (b) visto de cima e (c) legenda⁴

O robô ERS-7 possui um **processador** interno de 64 bits que opera a uma velocidade de 576 MHz.

Possui dois tipos de **memória**: uma memória interna e uma memória removível. A memória interna é do tipo SDRAM e pode armazenar até 64Mb de dados. A memória removível existe sob a forma de um cartão de memória, denominado *memory stick* (de 8, 16 ou 32Mb). O robô é programado através do seu *memory stick* (apresentado na figura 3).



Figura 3: Memory Stick

⁴ As imagens (a) e (b) foram retiradas do Manual Resumido do ERS-7 disponibilizado pela Sony [AIBO, 2005]

A nível de partes móveis, que os **actuadores** permitem controlar, o ERS-7 possui 20 graus de liberdade. Os graus de liberdade estão distribuídos ao longo da estrutura física do ERS-7 da seguinte maneira: 3 na cabeça; 1 na boca; 3 em cada perna o que perfaz um total de 12 nas quatro pernas; 1 em cada orelha; 2 na cauda.

Os **sensores** que o ERS-7 possui, permitem-lhe recolher informação do mundo exterior. Esta informação é actualizada a cada 32ms. É possível fazer um paralelo entre alguns dos nossos sentidos e a informação que os sensores do ERS-7 lhe transmitem, nomeadamente:

- Sentir - tem dois tipos de **sensores de tacto**: electrostáticos (localizados na cabeça e nas costas) e de pressão (colocados na parte de baixo de cada uma das patas e no queixo); possui um **sensor de aceleração** que é sensível ao movimento quer da esquerda para a direita, da frente para trás, e de cima para baixo (e vice-versa); para além disto existe um **sensor de vibração** e um **sensor de temperatura**.
- Ouvir - existem dois **microfones**, localizados nas orelhas do ERS-7, que podem gravar sons estéreo a uma frequência de amostragem de 16 KHz, utilizando uma modulação PCM de 16bits.
- Ver - **sensor de imagem CMOS** de 350,000 pixels, capta imagens a uma cadencia de 30 frames/s de formato máximo de 416(H)x320(V). É ainda de referir que o ângulo de visão é de 56,9° horizontal e 45,2° vertical. O ERS-7 possui ainda três **sensores de distância** por infra-vermelhos.

As características físicas do robô, podem ser encontradas em mais detalhe no Anexo B, dedicado ao ERS-7.

2.3 Plataformas de Programação

Existem diversas plataformas de programação que podem ser utilizadas no desenvolvimento de aplicações para o ERS-7, sendo algumas delas totalmente vocacionadas para esse fim.

Para transformar o ERS-7 num robô vigilante, consideraram-se algumas alternativas, umas mais profundamente do que outras, analisando as vantagens que poderiam trazer ao projecto e quais as suas limitações. Algumas das possibilidades encontradas foram:

- **R-CODE SDK**

Encontra-se dentro das alternativas mais simples, oferecendo um ambiente com comandos pré-definidos. É assim possível programar o AIBO em poucas linhas de código para andar ou por exemplo dançar. Trata-se de uma linguagem extremamente simples e acessível, mais voltada para o uso de quem programa o AIBO como passatempo. O código não necessita de ser compilado uma vez que se tratam de scripts. A desvantagem do R-CODE

é a de impor limitações ao grau de controlo que o programador pode ter sobre o ERS-7. Esta é portanto uma alternativa, mas que para o grau de controlo do robô exigido neste projecto não pode ser seriamente considerada. É apresentada aqui esta alternativa mais a título de exemplo da variedade de plataformas existentes, e de como algumas delas são acessíveis a quem não tenha grande experiência de programação.

- **Tekkotsu**⁵

Desenvolvido pela Universidade de *Carnegie Mellon*, é, à semelhança do OPEN-R (apresentado em seguida), uma plataforma orientada a objectos, baseada em C++. Permite que o programador não se tenha que preocupar com questões de mais baixo nível, podendo ser adicionados novos comportamentos a um menu do sistema e ligados ou desligados conforme se preferir.

- **OPEN-R**⁶

OPEN-R é a interface standard que a Sony tem vindo a promover para o AIBO e que permite expandir consideravelmente as capacidades deste robô.

OPEN-R SDK é o ambiente de desenvolvimento baseado em C++ que permite desenvolver aplicações que funcionam nos diferentes modelos do AIBO (ERS-7, ERS-210, ERS-220).

As características do software OPEN-R incluem:

- **Software modularizado** - os módulos de software são denominados “objectos”. Os objectos são executados paralelamente;
- **Comunicação entre objectos** - os objectos podem comunicar entre si. A conexão entre objectos está definida num ficheiro externo, que é carregado quando o sistema é iniciado, alocando e configurando os caminhos para a comunicação entre objectos. As portas que conectam os objectos estão identificadas pelo nome do serviço, fazendo com os objectos sejam extremamente modulares e que o software possa ser facilmente alterado.
- **Estruturado em camadas** - camada de sistema fornece uma série de serviços (*input* de som, *output* de som, *input* de imagem, *output* de valores para dos actuadores e

⁵ É mantida e actualizada uma página web pelos responsáveis do projecto Tekkotsu, donde foi retirada a informação [Tekkotsu]

⁶ As informações relativas ao OPEN-R foram retiradas da sua página oficial [OPENR]

input dos valores dos diversos sensores) que serve como interface para a camada de aplicação. Essa interface é implementada através de comunicação entre objectos. A camada de sistema disponibiliza ainda a interface com o protocolo TCP/IP o que permite aos utilizadores o uso da rede sem fios para comunicar com o robô.

O OPEN-R fornece uma API extremamente completa e poderosa (de forma livre e aberta) que disponibiliza as ferramentas para implementar tudo o que é possível dentro das limitações de hardware do robô, sendo para isso necessários bons conhecimentos de C++.

2.4 Conclusões

O robô AIBO da Sony, criado originalmente como uma mascote, ao não se ter cingido a um software fixo sem possibilidade de alteração ou evolução, deixou que o seu enorme potencial pudesse ser aproveitado não só pelos seus utilizadores menos imaginativos, como por aqueles que pretendem testar o que é possível fazer do ERS-7. Um robô bailarino? Um jogador de futebol? E porque não um cão de guarda? As hipóteses são inúmeras.

Não é por acaso que se desenvolveu uma Liga, dentro do concurso *RoboCup* [RoboCup], denominada *Four Legged League* [Aibo_League, 2005], que coloca equipas de quatro robôs Aibo a disputar jogos de futebol. Isto porque o ERS-7 combina um conjunto de dispositivos que o tornam extremamente completo e desafiante, sendo impossível que duas equipas distintas resolvam o mesmo problema de maneiras iguais, com uma diversidade de hipóteses de plataformas de programação e linguagens de programação (pode-se usar várias linguagens interligadas). Esta combinação, faz com que os problemas que podem ser propostos ao Aibo (em particular ao ERS-7 por ser o mais evoluído dos modelos) existam num número astronómico. Dado o caso de sucesso do futebol robótico, e outros de menos visibilidade, a escolha do robô, para a execução deste projecto recaiu sobre o ERS-7 como a opção mais lógica.

Em termos de plataforma de programação, escolheu-se a plataforma OPEN-R, utilizando-se como linguagem de programação o C++. Interessa neste projecto ir ao fundo do ERS-7, tendo como preocupação controla-lo desde o mais baixo nível. Esta é por isso a plataforma ideal. O C++, sendo uma linguagem orientada a objectos, adequa-se ao OPEN-R.

Nos próximos capítulos poder-se-á avaliar a adequação das escolhas efectuadas, mas a questão mais importante será perceber se as soluções apresentadas para os problemas que forem surgindo utilizam o potencial oferecido pelo ERS-7 e a plataforma de programação.

Capítulo 3

3. ERS-7 como Sistema de Vigilância

3.1 Visão Geral

A última década tem vindo a assistir a uma crescente utilização de câmaras como método de assegurar a segurança de espaços, como centros comerciais, parques de estacionamento, interior de instalações, etc. O trabalho de monitorizar as imagens, normalmente atribuído a um operador humano, tem tendência a tornar-se um trabalho aborrecido, onde a falta de concentração do operador poderá significar a não detecção de um evento importante nas imagens recebidas pela(s) câmara(s). Para colmatar esta falha, estes sistemas de vigilância, tem tido tendência a evoluir no sentido de se tornarem sistemas autónomos [Remagnino et al., 2004], dotados de capacidade para processar as imagens sem que seja necessária intervenção humana (ver [Remagnino et al., 1998]).

Mas se os robôs móveis têm vindo a desempenhar tantas tarefas, com tanto sucesso, que muito têm em comum com este projecto, como navegar autonomamente por ambientes de escritório reais [Nourbakhsh, 1998] e algumas incursões na temática da vigilância como [Rybsky et al., 2000], porque não tentar atribuir ao ERS-7 que, como foi visto no capítulo anterior, tem tantas potencialidades, a tarefa de tentar criar uma alternativa aos sistemas de vigilância tradicionalmente usados? E se este sistema móvel seguisse a tendência de evolução dos sistemas tradicionais, e aliasse a mobilidade à autonomia?

3.2 O Conceito

Pretende-se neste projecto criar um Agente Autónomo capaz de sentir o ambiente que o rodeia, estando preparado para se movimentar em ambientes reais, e que consiga agir e planear de acordo com ele, tendo em conta a tarefa ou missão que se pretende que seja efectuada.

A figura 4 apresenta um esquema genérico do Sistema de Vigilância a implementar. Passa-se agora à caracterização mais pormenorizada do Sistema de Vigilância, nomeadamente do ambiente em que o ERS-7 se terá de deslocar e das tarefas que se espera que sejam realizadas, para depois nos debruçarmos sobre o agente e a Aplicação Central de Vigilância.

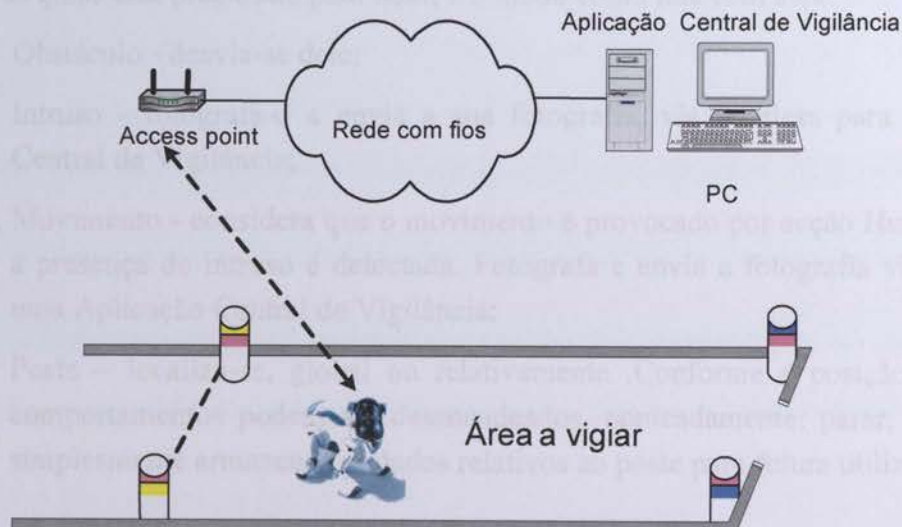


Figura 4: Esquema do Sistema de Vigilância

3.2.1 O Ambiente

No desenvolvimento de aplicações para robôs móveis, é essencial ter em linha de vista o ambiente em que o robô se terá de deslocar.

Considera-se, neste caso, como ambiente, a área delimitada pelos quatro postes (que se podem observar na figura 4) que deve ser quadrada ou rectangular. Os postes que contêm uma banda de cor igual, para além da cor de rosa, definem o comprimento da área, enquanto que o conjunto de postes, com bandas de cores diferentes para além da rosa, definem a largura.

No caso do projecto presente, consideram-se como ambientes para os quais o robô está vocacionado, ambientes:

- Planos;
- Conhecidos ou desconhecidos;
- Com obstáculos móveis e/ou estáticos;
- Iluminação mínima de 300 lux;

- Em que os postes possam estar, por norma, a descoberto em relação ao ERS-7, ou seja não tapados constantemente por algum obstáculo.

Estando a caracterização sumária do ambiente efectuada, é agora de mencionar os estímulos que o meio ambiente pode fornecer ao robô, que são mostrando na figura 5, e com os quais está preparado para lidar, e o modo como lida com eles:

- Obstáculo - desvia-se dele;
- Intruso - fotografa-o e envia a sua fotografia, via wireless para uma Aplicação Central de Vigilância;
- Movimento - considera que o movimento é provocado por acção Humana e por isso a presença de intruso é detectada. Fotografa e envia a fotografia via wireless para uma Aplicação Central de Vigilância;
- Poste – localiza-se, global ou relativamente .Conforme a posição obtida, vários comportamentos podem ser desencadeados, nomeadamente: parar, virar, andar ou simplesmente armazenar os dados relativos ao poste para futura utilização.

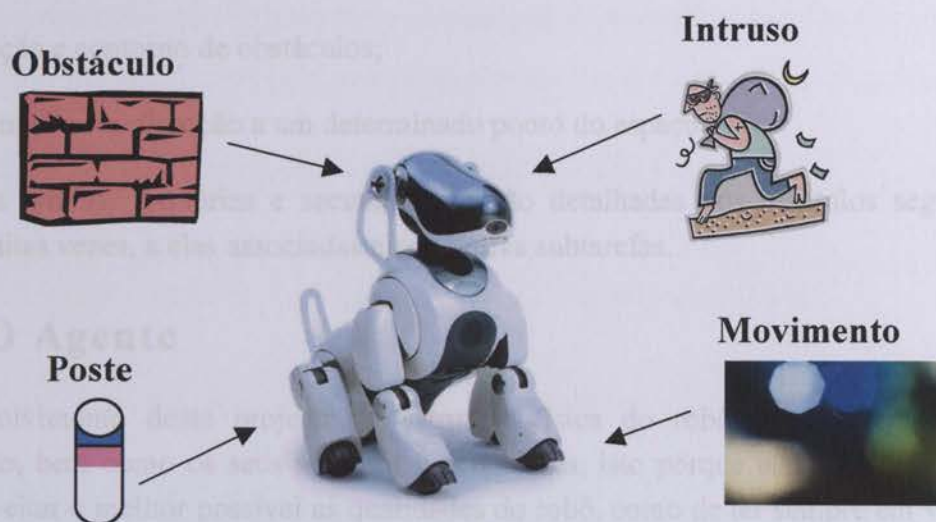


Figura 5: Estímulos que o ERS-7 recebe do meio ambiente

É ainda de referir que para cada novo ambiente em que o ERS-7 possa ser colocado é recomendável recalibrar a sua visão⁷, nomeadamente as cores, para que o software possa funcionar a 100%. A mudança de ambiente pode, portanto, implicar um certo tempo de preparação.

⁷ Para mais informações acerca do processo de calibração consultar o Anexo D.

3.2.2 As Tarefas

Pretende-se que o ERS-7 seja capaz de executar um certo número de tarefas, umas mais secundárias que outras, conforme os estímulos do mundo exterior e daquilo que é esperado que faça em determinada altura da execução do software.

A **tarefa primária** do ERS-7 é a de detectar intrusos (fotografá-los e enviar a sua foto para a Aplicação Central de Vigilância) enquanto efectua um percurso, parando em pontos predefinidos.

A realização com sucesso da tarefa primária implica a realização de uma série de outras tarefas, secundárias, mas de extrema importância para o sucesso deste sistema de vigilância. Passa-se a enumerar de forma resumida as **tarefas secundárias** que se espera que o ERS-7 realize oportunamente e de forma o mais optimizada possível, de modo a que o máximo tempo de operação do robô esteja à disposição da realização da tarefa primária:

- Reconhecimento de postes e sua identificação;
- Localização passiva ou activa;
- Detecção e contorno de obstáculos;
- Locomoção em direcção a um determinado ponto do espaço.

Todos estas tarefas, primárias e secundárias serão detalhadas nos capítulos seguintes, estando, muitas vezes, a elas associadas ainda outras subtarefas.

3.2.3 O Agente

No desenvolvimento deste projecto, a estrutura física do robô foi tida em grande consideração, bem como os seus sensores e actuadores, isto porque não só é importante tentar aproveitar o melhor possível as qualidades do robô, como de ter sempre em vista as suas limitações.

Do estudo do ERS-7 chegaram-se às características e dispositivos que melhor queremos aproveitar neste projecto e que são uma mais valia para o sistema de vigilância a implementar:

- **Câmara a cores** - irá permitir recolher informação bem precisa e complexa sobre o mundo exterior ao robô.
- **Sensores de distância** - serão responsáveis por fazer a detecção e contorno de obstáculos no caminho do robô.

- **Quatro patas cada uma com 3 graus de liberdade** - o robô terá que ser capaz de se movimentar com agilidade em várias direcções, nomeadamente em frente e virar.
- **Cabeça com 3 graus de liberdades** - uma vez que a câmara se encontra na cabeça do robô, todos estes graus de liberdade serão postos ao serviço de fazer o melhor varrimento possível do espaço.
- **CPU** - possibilitará que o robô seja capaz de tomar decisões autonomamente, baseadas na informação recolhida do mundo exterior pela câmara e sensores, se programado para isso.
- **Acesso à rede sem fios** - o robô poderá enviar informação por exemplo para um PC (que esteja também ligado à rede sem fios), nomeadamente fotografias das pessoas detectadas.

3.2.4 A Aplicação Central de Vigilância

A Aplicação Central de Vigilância foi concebida como um sistema de apoio à aplicação desenvolvida para o ERS-7, sendo executada numa máquina exterior ao robô. Contudo, este software acaba por dar sentido a todo o trabalho desenvolvido, uma vez que realiza importantes tarefas num sistema de vigilância que se possa dizer satisfatório.

Pretende-se que a Aplicação Central de Vigilância funcione como uma interface agradável que permita à distância comandar, monitorizar o ERS-7 e armazenar as imagens enviadas pelo ERS-7. Mais pormenores podem ser encontrados no Capítulo 7.

3.3 Perspectiva Global da Arquitectura

3.3.1 Código da equipa rUNSWift⁸

A arquitectura do sistema desenvolvido para o ERS-7 é baseada na arquitectura do código de 2004 da equipa rUNSWift. Passemos então a conhecer um pouco melhor esta equipa e o código por ela desenvolvido.

rUNSWift é uma equipa de futebol robótico estabelecida pela Universidade de Nova Gales do Sul (UNSW) localizada em Sydney, Austrália, que tem vindo a participar na competição *Robocup*, inserindo-se na *Four Legged League*.

Apesar das diferenças entre os objectivos para o qual o código da equipa rUNSWift foi criado e os objectivos deste projecto, existem alguns pontos comuns, ou com necessidade

⁸ Toda a informação referente a esta equipa, nomeadamente um pouca da sua história, código e relatórios, foi obtida na sua página online [rUNSWift, 2004]

de apenas alguns ajustes, o que tornou possível a utilização deste código como base para o desenvolvimento deste projecto. Exemplo disso é a estrutura do código rUNSWift, que é constituído por 5 módulos, nomeadamente: visão, localização, locomoção, comportamento e wireless, que pode facilmente ser adaptada.

De modo a que seja possível fazer uma pequena ressalva do que foi utilizado na íntegra deste código, o que foi adaptado ou totalmente refeito, ao longo do relatório a utilização de algoritmos desenvolvidos pela equipa rUNSWift será assinalada.

3.3.2 Especificações das Características Globais da Arquitectura

“Ao nível mais geral, a arquitectura, tem como principal função fornecer meios para que o sistema alcance os seus múltiplos objectivos de forma eficiente. Deve ainda ser capaz de satisfazer as restrições de tempo, e ser segura e tolerante a falhas” [Neves, 1999]

Tendo em linha de conta que diversas questões devem ser ponderadas aquando da escolha da arquitectura, passa-se a apresentar a arquitectura da aplicação do sistema de vigilância (figura 6), para em seguida analisá-la.

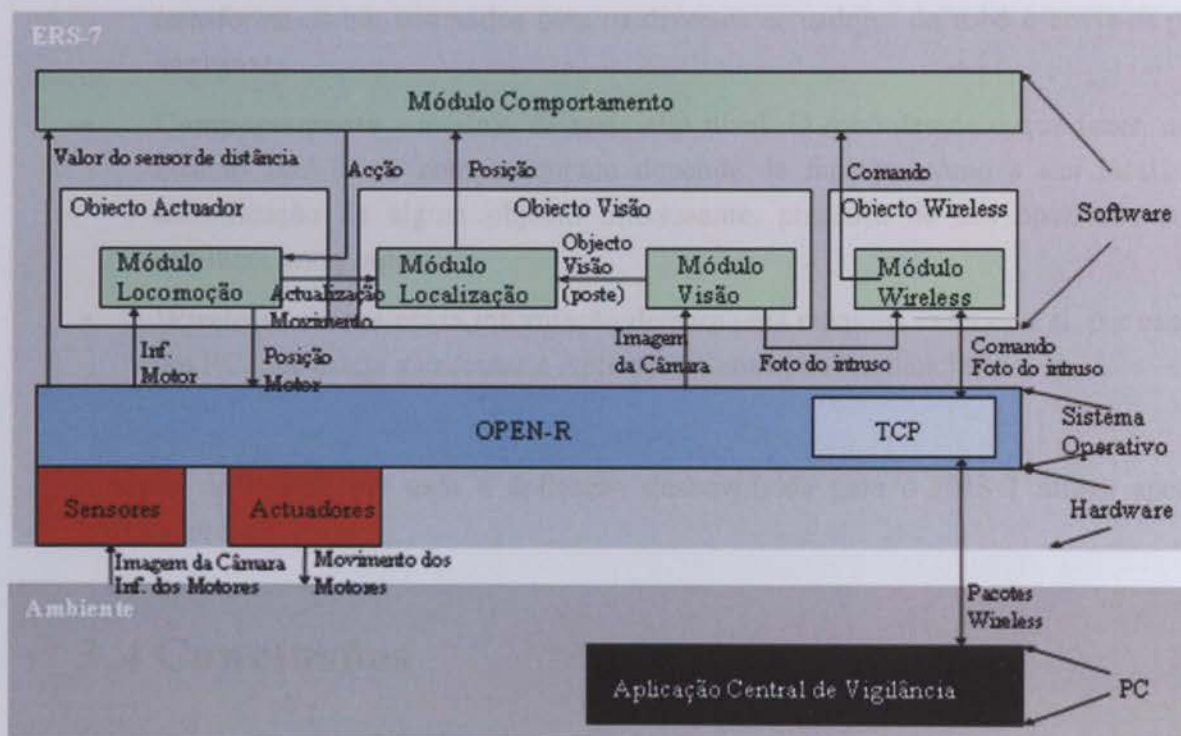


Figura 6: Arquitectura da Aplicação

A arquitectura tem como principais características:

- Estruturação por módulos (5 módulos);
- Módulos executados em modo paralelo;
- Possibilidade de troca de informação entre os diferentes módulos.

Os 5 módulos anteriormente referidos são:

- **Visão** - processa as imagens da câmara e atribui a cada píxel uma cor (caracterizada pelas 3 componentes Y, Cr e Cb). Píxeis da mesma cor, que sejam vizinhos, são aglomerados. Alguns destes aglomerados são identificados como objectos. Objectos que sejam reconhecidos podem ser utilizados nos módulos de alto nível como localização e comportamento. Para além disso, o módulo de visão permite ao ERS-7, quando parado, detectar movimento na imagem.
- **Localização** - recebe informação do módulo de visão. As imagens da câmara fornecem informação extremamente útil, como identificação de um poste, a que distância se encontra e qual o ângulo que faz com o centro da imagem do robô. A partir destas informações, o robô pode localizar-se relativamente, ou cruzando-as com informações anteriores localizar-se globalmente.
- **Locomoção** - recebe instruções do módulo comportamento, interpreta-as, transforma-as em comandos para os diversos actuadores do robô e envia-as para o hardware.
- **Comportamento** - módulo de mais alto nível. O robô decide o que fazer, onde e quando fazê-lo. O comportamento depende de factores como a sua localização, identificação de algum objecto interessante, presença de um obstáculo no seu caminho, entre outros.
- **Wireless** - recebe/envia informação de/para uma máquina exterior a si, por exemplo um PC, que esteja a executar a Aplicação Central de Vigilância.

É ainda de referir que toda a aplicação desenvolvida para o ERS-7 utiliza apenas a linguagem C++.

3.4 Conclusões

De acordo com as conclusões do capítulo anterior, esta fase do projecto, desenvolve o conceito de utilizar o ERS-7 como figura principal de um sistema de vigilância de modo a utilizar as potencialidades por ele oferecidas.

Para além do código da equipa rUNSWift, cujas problemáticas que aborda tem muito a ver com o presente projecto, foram tomados como exemplos, para a concretização do conceito do ERS-7 como sistema de vigilância, outros projectos, que utilizassem o ERS-7 ou outros robôs. Contudo, dadas as características especiais do ambiente em que o ERS-7 se terá de movimentar as ideias retiradas da equipa de futebol robótico apenas podem servir de base. Problemáticas como a localização terá de ter toda uma nova resposta, uma vez que as marcas no terreno que o ERS-7 tem à disposição já não são tantas como no futebol robótico (postes, linhas e balizas). O comportamento nada terá a ver com o de um jogador de futebol, e a navegação tem uma sequencia de fases bem definidas, onde evitar que o ERS-7 se esbarre com algum obstáculo tem um novo significado - não basta evitá-lo, é preciso contorná-lo de forma lógica de modo a que a navegação possa prosseguir sem que o robô volte a ter de lidar com o mesmo obstáculo. Novos algoritmos terão de ser adicionados ao módulo de visão, nomeadamente a detecção de movimento e a detecção de objectos adaptada para que o ERS-7 detecte também pessoas. O módulo wireless terá também de ser adaptado de modo a enviar a informação pretendida na altura devida.

Se este capítulo apresenta essencialmente o projecto do sistema de vigilância, os capítulos seguintes contêm a sua implementação, onde cada um dos conceitos e ideias apresentadas aqui serão detalhados.

Capítulo 4

4. Visão

4.1 Introdução

Qual a importância de um sensor de imagem num robô? A melhor forma de o analisar consiste em analisar como este sentido é utilizado tanto nos Humanos como em alguns animais, transformando-se no mais poderoso dos meios de percepção.

No início dos anos 80, alguns investigadores começaram a investigar os animais como referência nos seus projectos, nomeadamente na área da percepção. Descobriram, por exemplo que os sapos, usam uma simples detecção de movimento visual para apanharem as suas presas. As abelhas, centram-se em aspectos específicos do espectro de cores na recolha do pólen. No caso Humano, estudos indicam que utilizamos a nossa área periférica de visão, de baixa resolução, para detectar movimento (que permitem por exemplo, evitar colisões com objectos em movimento). Já a área de mais alta resolução é utilizada para recolher informação para, por exemplo, reconhecer um objecto. Pode-se assim concluir que tanto Humanos como animais não fazem uso de tudo ao mesmo tempo (cor, movimento, dimensão temporal) mas antes direccionam a sua atenção para um pequeno aspecto do seu campo de visão conforme o objectivo que queiram atingir.[Kortenkamp et al., 1998]

O Módulo de Visão apresentado neste capítulo e utilizado neste projecto, utiliza o mesmo princípio de selecção de toda a informação disponível no campo de visão, recolhida pelo sensor de imagem, conforme a tarefa desempenhada e o objectivo que se pretende atingir, seja ele reconhecer um objecto, perceber se esse reconhecimento é coerente com os resto dos dados ou simplesmente detectar movimento.

4.2 Princípios e conceitos do Módulo de Visão

Os algoritmos de reconhecimento de objectos utilizados foram essencialmente os disponibilizados no código da equipa rUNSWift, tendo-se apenas procedido, no entanto a

diversas alterações. O código foi adaptado para os objectos que é necessário reconhecer neste projecto, nomeadamente potenciais pessoas e foi introduzido o algoritmo de detecção de movimento na imagem (quando o ERS-7 se encontra parado) neste módulo de visão.

É de referir que o módulo de visão recebe 30 imagens a cada segundo, o que significa ter aproximadamente 0,033 segundos para processar a informação contida em cada imagem. A eficiência ao processar a informação é por isso um aspecto importante e crítico para este módulo.

Identificação de objectos

O processo de identificação de objectos terá como entrada uma série de valores YUV provenientes da câmara (organizados num array), e terá como saída os objectos identificados.

Entre a entrada e a saída deste processo decorrem diversas operações que serão em seguida apresentadas sumariamente:

- **Classificação de cores** - Faz a correspondência entre os valores contidos no array e uma cor. Para isso é utilizada uma tabela de referência, denominada *look-up*, de valores YUV, que dado qualquer combinação de valores Y, U e V faz correspondência a uma cor (o número de cores que podem ser atribuídas é limitado, tendo sido utilizado neste projecto 11 cores diferentes reconhecidas pelo robô). Esta tabela é construída através de um processo denominado calibração⁹. O aspecto da informação que chega ao robô após esta fase pode ser vista na figura 7.



Figura 7: Mundo visto pelo ERS-7¹⁰

⁹ Todo o processo de construção da tabela de *look-up* pode ser consultado no Anexo D.

¹⁰ Imagem obtida com a ferramenta RoboCommander da equipa rUNSWift

- **Segmentação** - Um segmento é definido como “um grupo de píxeis que têm a mesma cor e que estão ligados” [rUNSWift_Rel, 2004]. São precisos pelo menos 2 píxeis para formar um segmento.
- **Formação de aglomerados** - Depois de toda a informação ter sido recolhida, todos os segmentos que se encontram conectados e que têm a mesma cor são considerados um só grupo, que denominaremos neste relatório de aglomerado.
- **Reconhecimento de objectos** - Um objecto pode ser formado por um único aglomerado, por vários aglomerados da mesma cor, ou mesmo por vários aglomerados de cores diferentes. Tendo toda a informação referente aos aglomerados detectados, estes devem passar por várias condições até que se possa dizer com certeza que um dado objecto foi reconhecido.
- **Controlo de coerência entre os objectos reconhecidos** - Tendo um objecto sido reconhecido, este é confrontado com outros objectos que também tenham sido, de modo a perceber se existem incoerências. Só após um objecto passar por uma série de condições e passar nos testes é que pode ser tipo como identificado e verdadeiro. A figura 8 apresenta o aspecto de um objecto identificado.

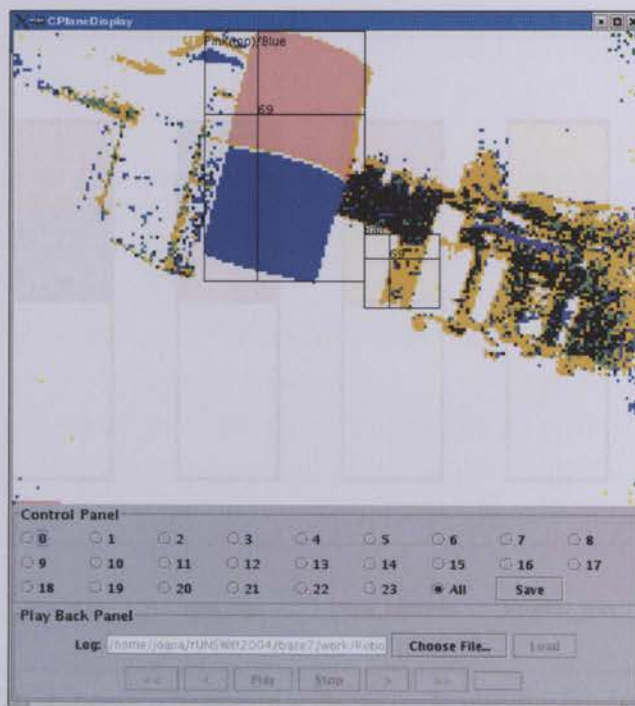


Figura 8: Objecto identificado pelo módulo de visão¹¹

¹¹ Imagem obtida com a ferramenta RoboCommander da equipa rUNSWift

4.3 Detecção e reconhecimento de postes

Os postes são constituídos por duas bandas de cor de cerca de 10cm cada, colocadas por cima de uma área branca com 20cm de altura, tal como se pode ver na figura 9.

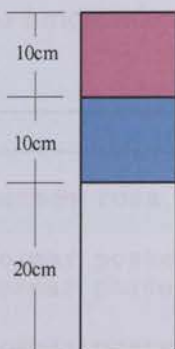


Figura 9: Esquema genérico dos postes utilizados

Para que seja claro quais os postes que foram utilizados e o nome utilizado para cada um, são agora apresentados todos os postes e respectivos nomes.

P o s t e s				
Nome	rosa_azul	azul_rosa	rosa_amarelo	amarelo_rosa

Figura 10: Aspecto e denominação de cada um dos postes

Para o reconhecimento de postes são necessários dois passos: primeiro encontrar os postes, segundo verificar se existem inconsistências entre eles.

4.3.1 Encontrar poste

Tal como foi referido anteriormente, os postes são constituídos por duas bandas de cor diferente, sendo que uma delas é sempre cor-de-rosa e a outra banda ou é azul ou é amarela. A função que permite a partir deste facto formar os potenciais postes é a função FindBeacons() cujo pseudo-código é mostrado na figura 11.

```
FindBeacons()  
1   Para primeiro aglomerado rosa até ao último fazer:  
2  
3       Se possível formar poste azul()  
4           então formar poste azul provisório(x=azul)  
5  
6       Se possível formar poste amarelo()  
7           então formar poste amarelo provisório(x=amarelo)  
8  
9       Se são formados dois postes do mesmo aglomerado  
10          então escolher só um deles (x=cor do escolhido)  
11  
12      Se banda rosa em cima  
13          então poste observado = a rosa_x  
14      Senão poste observado = x_rosa  
15  
16      Se poste já foi observado na mesma imagem  
17          Então ignorar poste  
18      Senão guardar poste
```

Figura 11: Pseudo-código da função FindBeacons()

A tentativa de formar poste a azul ou amarelo (ver figura 11) é feita por uma função à parte (denominada FormBeaconFromPink()) que irá tentar juntar cada um dos aglomerados azuis ou cada um dos aglomerados amarelos ao aglomerado rosa corrente, de forma a verificar se algum poste potencial pode ser formado. Para que os resultados produzidos por esta função sejam o mais consistentes possíveis algumas condições são testadas.

A primeira condição que a função FormBeaconFromPink irá testar é se o que o robô está a observar está abaixo da linha do horizonte. Esta linha indica o nível da altura da câmara. É calculada utilizando os valores dos 3 ângulos correspondentes aos 3 graus de liberdade da cabeça (denominados, *pan*, *tilt* e *crane*), e o comprimento entre a base do pescoço e o chão. Se o robô está a observar tudo abaixo da linha de horizonte é porque está a olhar para baixo e por isso não deveria estar a ver nenhum poste.

As outras condições da função FormBeaconFromPink são aplicadas na sequência:

- Condições referentes ao aglomerado rosa (ver tabela 1);

- Condições referentes ao aglomerado azul ou amarelo com o qual se tenta formar um poste (ver tabela 1);
- Condições que preservam a coerência da relação dos dois aglomerados rosa (ver tabela 2).

Aglomerado	Condição
rosa	1. O aglomerado está demasiado acima ou demasiado abaixo da linha de horizonte?
rosa ou azul ou amarelo	2. O rectângulo mais pequeno capaz de conter todo o aglomerado é demasiado pequeno? Se este rectângulo for demasiado achatado ou fino o mais provável é que se trate de ruído do fundo.
	3. A densidade do aglomerado é demasiado pequena? A densidade é aqui entendida como o valor resultante da divisão entre o valor da área do aglomerado e a área do rectângulo, referido na condição anterior. Se este valor for muito pequeno, significa que ou o aglomerado tem muitos buracos ou que tem uma orientação diagonal. Em qualquer dos caso pode-se ignorar este aglomerado para a criação de um poste.
	4. O resultado do quociente entre o tamanho do lado maior do rectângulo e o lado mais pequeno é maior que 1? Se for, significa que o aspecto do aglomerado será muito longe do de um quadrado, não fazendo por isso parte de um poste.
azul ou amarelo	5. O aglomerado já faz parte de outro poste?

Tabela 1: Condições aplicadas a um aglomerado individual

Aglomerado	Condição
rosa	6. O aglomerado está demasiado acima ou demasiado abaixo da linha de horizonte?
rosa ou azul ou amarelo	7. O rectângulo mais pequeno capaz de conter todo o aglomerado é demasiado pequeno? Se este rectângulo for demasiado achatado ou fino o mais provável é que se trate de ruído do fundo.
	8. A densidade do aglomerado é demasiado pequena? A densidade é aqui entendida como o valor resultante da divisão entre o valor da área do aglomerado e a área do rectângulo, referido na condição anterior. Se este valor for muito pequeno, significa que ou o aglomerado tem muitos buracos ou que tem uma orientação diagonal. Em qualquer dos caso pode-se ignorar este aglomerado para a criação de um poste.

	<p>9. O resultado do quociente entre o tamanho do lado maior do rectângulo e o lado mais pequeno é maior que 1?</p> <p>Se for, significa que o aspecto do aglomerado será muito longe do de um quadrado, não fazendo por isso parte de um poste.</p>
azul ou amarelo	10. O aglomerado já faz parte de outro poste?

Tabela 2: Condições aplicadas a dois aglomerados que potencialmente formam um poste

4.3.2 Verificações internas entre postes reconhecidos

Passados todos estes testes, um poste é considerado como tal. Contudo, há ainda que verificar a consistência dos resultados obtidos, se forem encontrados mais do que um poste, pois poderemos estar perante resultados enganadores.

Este teste é feito dentro de uma função denominada `BeaconsSanities()` cujo pseudo-código é apresentado na figura 12.

```

BeaconsSanities()
1   Se apenas identificou um poste
2   então voltar
3
4   Senão Se tiverem sido vistos três postes
5       então ignorar o que tiver cores diferentes dos outros
6
7       Senão
8           Se tiverem sido vistos quatro postes
9               então escolher o par a melhor distância
10
11      Senão Se tiverem sido vistos dois postes
12
13      Se (o poste devia estar do lado esquerdo ou do lado direito)
14      mas não está
15          então sim=1
16
17          Se um dos postes é muito mais alto que o outro
18              então sim=1
19
20          Se os seus tamanhos diferem muito
21              então sim=1
22
23          Se sim =1
24              Então escolher o poste com menor elevação

```

Figura 12: Pseudo-código da função `BeaconsSanities()`

Passados então todas estas condições o poste é então considerado válido e pode ser usado por outros módulos do software para daí tirar conclusões essenciais nas tomadas de decisão do ERS-7.

4.4 Detecção de intruso

Têm-se como um dos objectivos deste projecto encontrar formas de detectar pessoas. Neste ponto é descrito um dos métodos encontrados, que utiliza duas características de um rosto (em conjunto):

- Cor;
- Forma;

Este método tem, como qualquer outro, as suas fortalezas e as suas fraquezas. Pode-se dizer que as suas principais fortalezas são:

- Simplicidade;
- Rapidez;
- Eficácia.

A principal fraqueza é sem dúvida apenas permitir detectar rostos cuja cor de pele esteja contida numa delimitada gama de tons de pele. Ou seja, qualquer tom de pele mais escuro ou muito claro, não será detectado por este processo.

Este processo passa por duas fases:

- Identificação de todos os aglomerados cuja cor faz com que possam potencialmente pertencer a um rosto;
- Filtragem desses aglomerados, submetendo-os a teste da forma e tamanho.

A primeira fase deste processo está descrita no ponto deste Capítulo referente aos Princípios e Conceitos do Módulo de Visão, não sendo por isso abordado de novo este tópico. Interessa, antes, aqui dar ênfase as condições a que os aglomerados são submetidos (segunda fase deste processo), sendo estas apresentadas na tabela 3.

Condição
1. Tem a forma de uma cara?
A forma aqui considerada é a oval, sendo considerado que o aglomerado deve ser mais alto que a sua largura.
2. Ocupa suficientes píxeis?
Existe um tamanho limite para os aglomerados considerados.

Tabela 3: Condições aplicadas a aglomerados que potencialmente pertencem a um rosto

4.5 Detecção de movimento

O objectivo da implementação deste algoritmo é, tal como o método descrito no ponto anterior, o de detectar a presença de intruso, tendo como premissa que a responsabilidade de qualquer movimento detectado na imagem, é Humana.

Este método é utilizado quando o ERS-7 se encontra parado e consiste, basicamente, na comparação de duas imagens. Passa-se, agora, a analisar em mais detalhe alguns aspectos considerados como importantes para a implementação e compreensão do funcionamento deste método. O pseudo-código é apresentado na figura 13.

- **Frequência de amostragem** – 30 frames/s.

A frequência de amostragem utilizada por este método é a da cadência a que o módulo de visão recebe imagens da câmara.

- **Imagens comparadas** - imagem actual com a imagem obtida meio segundo antes.

Sendo objectivo deste algoritmo comparar duas imagens e descobrir se são diferentes, ou não, uma da outra, convém que o tempo que passou entre a captação de uma e da outra chegue para que, existindo movimento, haja diferenças notórias entre elas. Essa diferença deve poder, facilmente, ser atribuída à responsabilidade da presença de movimento, e não ser confundida com as pequenas diferenças que existem mesmo entre imagens iguais. Sendo assim, tendo em conta a cadência de imagens que chegam a este algoritmo (30 por segundo), se se fosse a comparar a imagem actual com a anterior, poucas ou nenhuma diferença encontraríamos entre as duas, por mais rápido que o movimento fosse feito. A velocidade dos movimentos Humanos não consegue competir com esta frequência de amostragem. A solução encontra-se, não em diminuir a frequência de amostragem mas em comparar a imagem actual com a imagem guardada de à meio segundo atrás.

- **Parâmetros comparados nas duas imagens para o mesmo píxel**

Cada píxel é caracterizado por três componentes: Y, Cr e Cb. Tentou-se encontrar uma maneira rápida e completa de comparar um píxel com outro, de modo a que alguma inconsistência entre os resultados obtidos e a realidade não se devesse a esta fase do processo.

A solução encontrada para comparar o píxel 1 com o píxel 2 e determinar se são iguais foi utilizar a expressão:

$$\text{abs}(Y1-Y2)+\text{abs}(Cr1-Cr2)+\text{abs}(Cb1-Cb2)$$

Teoricamente, se os píxeis são iguais, o resultado da expressão deveria ser zero, mas na realidade não é o que acontece. Assim têm de ser introduzidas umas certas margens de tolerância.

- **Margens de erro consideradas**

Para que dois píxeis sejam considerados iguais, a situação ideal seria que a componente Y de um fosse igual à componente Y do outro, e a mesma coisa com as componentes Cr e Cb. Contudo, numa situação prática existem pequenas variações nos valores de todas as componentes (não existindo uma que sofra menos deste mal que outra) mesmo que teoricamente os píxeis sejam iguais.

Esperava-se por isso que, para píxeis supostamente iguais, os seus valores não fossem exactamente os mesmos, mas que se encontrassem distanciados até um certo valor. Sendo assim, na comparação de dois píxeis, considerasse que os píxeis não são diferentes de distanciarem entre si de um valor inferior à margem imposta.

Ainda assim muitos píxeis são considerados como diferentes, ainda que na teoria deveriam ter o mesmo valor, ou aproximado, ou seja sido detectados como iguais na filtragem anterior. Foi por isso preciso determinar um valor máximo de píxeis diferentes detectados entre as duas imagens, acima do qual se considera que houve movimento.

```

1   Incrementar passagem
2   Tolerancia = 43000
3   Se imagem inválida
4       então mostra erro
5
6   Senão
7       Cria apontador para inicio da imagem
8       Cria 3 variáveis com acesso às camadas Y, Cr e Cb de cada píxel
9
10      Se passagem > 15
11          então para y=0 até y>320 fazer y++
12              para x=0 até x<416 fazer x++
13                  somar1 = Y + Cr + Cb do píxel[x][y] da
14                  da imagem recebida
15                  somar2 = Y + Cr + Cb do píxel[x][y] da
16                  imagem recebida à meio segundo atrás
17
18                  Se a primeira soma é diferente da segunda
19                      Se os valores diferem entre si
20                          mais de 20
21                              então incrementa diferente
22
23                  então guardar a imagem recebida
24      Senão
25          Se passagem <=15
26              então guardar a imagem recebida
27
28
29      Se diferente > Tolerancia
30          então movimento detectado
31

```

Figura 13: Pseudo-código do algoritmo de detecção de movimento

4.6 Conclusões

O módulo de visão é um dos módulos mais importantes no que toca à percepção do ambiente em que o ERS-7 se desloca. A partir do módulo de visão, o ERS-7 consegue saber se algum poste ou intruso está no seu campo de visão, e estando parado se movimento é detectado na imagem. Estas informações podem, e são, partilhadas com os outros módulos, nomeadamente com o módulo wireless e com o módulo responsável pela localização.

No caso do módulo de localização, que de acordo com as conclusões tiradas no último capítulo, seria uma muitas questões a que este projecto teria de dar resposta desenvolvendo um algoritmo para essa finalidade, o módulo de visão fornece as informações de *input*, das quais o ERS-7 retira dados tão preciosos como a distância ao poste, ângulo que faz com ele, etc. Como se pode antever, muitos são os erros a que o módulo de localização está sujeito pela sua acumulação na determinação das propriedades do poste observado, referidas anteriormente, e no próprio cálculo de coordenadas, como veremos a seguir. Sendo assim, será indispensável que não seja por falta de eficiência do módulo de visão que o ERS-7 perca a possibilidade de se localizar, e de o fazer convenientemente.

Capítulo 5

5. Localização

“Para que um robô móvel navegue de forma confiável, num ambiente interior, é necessário que ele saiba onde está” [Thrun et al., 1998].

5.1 Estado da Arte

A resposta à problemática da localização de um robô móvel já teve inúmeras respostas. Com base na referência [Santos et al.,2002], passa-se a descrever resumidamente os vários métodos, já usados com sucesso, e considerados como possíveis de implementar no ERS-7, para que seja mais fácil posteriormente justificar as opções tomadas.

A utilização de mapas ramifica-se basicamente em duas possibilidades: o mapa é conhecido à priori pelo robô, ou então não é conhecido. No primeiro caso, surge a possibilidade de pré-programar o robô para o trajecto a ser seguido, o que não impede que esteja alerta para situações fora do âmbito do mapa, como foi o caso do robô concorrente à edição de 1999 do concurso Micro-Rato [Lopes et al., 2000]. No segundo caso, o mapa terá de ser construído pelo robô, o que implica usualmente algoritmos de aprendizagem. Para qualquer das situações anteriores existem dois tipos de mapas: topológicos (simples mas com tendência a levar a caminhos não optimizados) e de grelha (de mais complexo planeamento mas de mais fácil compreensão para a percepção humana).

Uma segunda hipótese é a localização utilizando pontos de referência. Os pontos de referência podem ser de vários tipos, logo que sejam à priori conhecidos pelo robô, bem como a sua localização. Estes pontos de referência podem ser relativos, como um farol, ou absolutos, como o sol.

Por último, apresenta-se a hipótese da odometria, que consiste em avaliar qual a distância já percorrida pelo robô, utilizando para isso, por exemplo, codificadores rotativos (no caso do robô mover-se pela rotação de rodas).

Para além destas opções, outros métodos são descritos em [Santos et al.,2002], mas que não foram considerados realmente como hipótese para o algoritmo de localização a

desenvolver. Aqui ficam como referência: Localização relativa em grupos (percepção de localização feita por vários robôs que cooperam entre si); Localização absoluta com GPS; Localização utilizando bússolas electrónicas.

Os pontos seguintes dedicam-se a apresentar as opções tomadas, com base no conhecimento adquirido relativamente a hipóteses já testadas e confirmadas com sucesso, e seus detalhes.

5.2 Visão Geral

O Módulo de Localização, foi encarado neste projecto como uma importante fonte de dados para o Sistema de Navegação, tendo sido com este objectivo que ele foi desenvolvido. Nesta perspectiva, criar uma técnica que permitisse localizar o robô globalmente a partir do nada e que conseguisse recuperar das eventuais falhas de localização que possa ter, foi a motivação para o desenvolvimento do algoritmo de localização. Esta motivação, levou a que a localização baseada em mapas à priori conhecidos não pudesse ser utilizada, e a baseada na odometria apenas utilizada em algumas situações, uma vez que com apenas este tipo de localização não é possível recuperar nunca dos erros acumulados. Assim, o método principal de localização será a baseada em pontos de referência. Isto porque o ERS-7 tem uma longa tradição de se localizar a partir da visualização de postes (veja-se o caso do futebol robótico) e porque o método restante da construção dinâmica do mapa partiria também da utilização dos postes como referência e seria mais um complemento ao algoritmo de localização.

De um modo geral, o módulo de localização tem como *input* os dados enviados pelo sensor de imagem processados no módulo de visão (detecção e identificação de postes) ou a actualização dos valores dos motores, e pretende-se que a informação de *output* seja:

- **actualizada** - este módulo deve processar os dados enviados pelo módulo de visão em tempo real.
- **fiável** - deve-se tentar evitar a acumulação de erros ao longo do tempo.

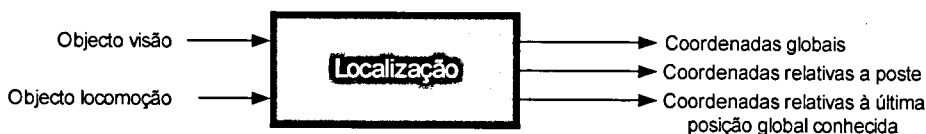


Figura 14 Inputs e Outputs do módulo de localização

Esta informação de *output* variará conforme a localização a ser feita seja relativa ou global. Estas duas formas de localização serão analisadas em mais pormenor nas secções seguintes.

5.3 Localização Relativa

A Localização Relativa, consiste em saber em que posição é que o robô se encontra relativamente a.

Num dos casos em que se usa a localização relativa, pretende-se saber onde está o ERS-7 relativamente a um dado poste.

Para a obtenção desta informação, o ERS-7 precisa apenas de analisar a informação enviada pelo módulo de visão, relativa à identificação de um poste, obtendo as coordenadas (ver figura 15): distância ao poste (d) e o ângulo do corpo do ERS-7 em relação ao poste (θ).



Figura 15 Localização Relativa a um poste

Este método é usado quando o ERS-7 vê o poste para onde se está a dirigir e quer saber se já o atingiu.

Outros dos casos em que se usa a localização relativa é quando o ERS-7 quer saber quanto virou em relação à última posição conhecida. Esta última posição referida é composta por três variáveis (x , y , θ) e diz respeito à localização global que em seguida será analisada. Neste caso o módulo de localização não pode recorrer às imagens processadas pelo módulo de visão, porque este pode não ter enviado nada, mas antes ao processo

anteriormente descrito de odometria. Como foi referido anteriormente, este método está sujeito à acumulação de erros, mas é a única possibilidade nesta situação - ir lendo os valores enviados pelo módulo de locomoção (valores dos motores), processar a informação e calcular quanto o ERS-7 virou.

Como se pode intuir, este processo de localização é extremamente simples face à localização global, como poderá ser visto no capítulo seguinte. A sua principal vantagem vai de encontro aos requisitos pretendidos para os dados de *output* deste módulo: a partir dos dados de *input* o ERS-7 consegue localizar-se, convenientemente, rapidamente.

5.4 Localização Global

Como se pode ver na figura 16, pretendeu-se desenvolver um sistema de localização global, em que as variáveis a determinar são (x, y, θ) . Há que referir que o θ , é o ângulo formado entre a direcção do corpo do ERS-7 e a linha de 0° , indicada na figura.

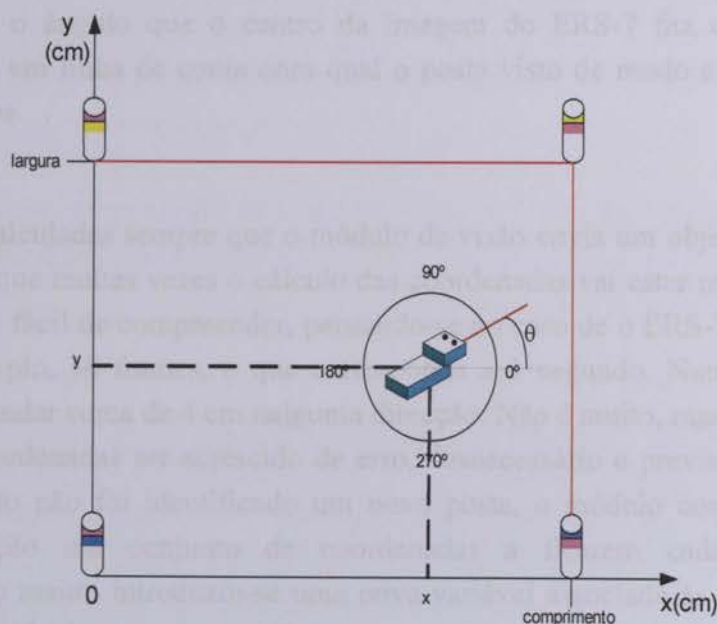


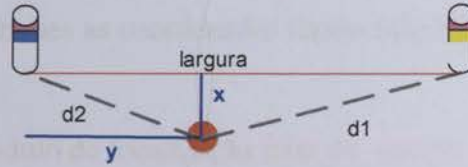
Figura 16 Localização Global

A questão que se coloca é: como são determinadas estas coordenadas? Cruzando a informação do poste visto, com a do último poste visto antes deste. Um dos dois pontos onde a informação se cruza, são as coordenadas x e y do ERS-7 (ver figura 17). Como um dos pontos será sempre fora da área, não será difícil distinguir as verdadeiras coordenadas. Como se pode observar na figura 17, sendo os comprimentos d_1 , d_2 e largura conhecidas, facilmente se determinam os valores de x e y , sendo tudo uma questão de aritmética básica.

No caso de pontos próximos, verifica-se que a distância entre os pontos é muito pequena em relação ao comprimento do caminho dos pontos, logo a distância entre os pontos é muito pequena. Esta situação é um exemplo de um caminho muito longo e a distância entre os pontos é muito pequena.

Um exemplo típico de um caminho muito longo e a distância entre os pontos é muito pequena.

Seja θ o ângulo que a direcção do corpo do ERS-7 faz com a cabeça, e o ângulo que o centro da imagem do ERS-7 faz com o poste identificado, entrando em linha de conta com qual o poste visto de modo a saber o que fazer com estes ângulos.



5.5 Conclusões

Figura 17 Determinação de coordenadas x e y

Quanto à variável θ , é determinada utilizando o ângulo que a direcção do corpo do ERS-7 faz com a cabeça, e o ângulo que o centro da imagem do ERS-7 faz com o poste identificado, entrando em linha de conta com qual o poste visto de modo a saber o que fazer com estes ângulos.

As coordenadas são calculadas sempre que o módulo de visão envia um objecto de visão (poste). Isto significa que muitas vezes o cálculo das coordenadas vai estar redondamente errado. Esta situação é fácil de compreender, pensando-se no caso de o ERS-7 não avistar um poste à, por exemplo, 30 frames, o que corresponde a 1 segundo. Num segundo o ERS-7 tem tempo de andar cerca de 4 cm nalguma direcção. Não é muito, mas o suficiente para o cálculo das coordenadas ser acrescido de erro, desnecessário e previsível. Já para não falar que enquanto não foi identificado um novo poste, o módulo comportamento tinha à sua disposição um conjunto de coordenadas a ficarem cada vez mais desactualizadas. Sendo assim, introduziu-se uma nova variável associada às coordenadas que dá a sua probabilidade de estarem correctas.

Localização (de modo passivo e de modo activo)

Este conceito de incerteza face à localização, tem sido explorado em inúmeros sistemas, como é o caso de [Bugard et al., 1996] que propõe um sistema baseado numa grelha de probabilidade representando o modelo do mundo e que permite estimar a posição absoluta do robô e de [Kaelbling et al., 1996] que considera o problema de actuar sobre incerteza. Um método muito conhecido, e usado, é a localização de *Markov* que mantém a densidade de probabilidade sobre todo o espaço onde o robô possa estar. [Fox et al., 1999] descreve em pormenor este método.

No caso do presente projecto, nenhum dos métodos anteriores foi usado, mas da ideia transmitida do conjunto dos vários surgiu a forma final da aplicação do conceito de incerteza. Este conceito é em exclusivo associado às coordenadas globais, e o seu valor é calculado tendo em conta dois aspectos:

- Diferença temporal entre os postes utilizados para cruzar a informação;
- Há quantas frames as coordenadas foram calculadas.

Sendo assim, o módulo de localização trata de, sempre que um poste é observado, calcular coordenadas e atribuir-lhe uma percentagem de certeza. Competirá assim ao módulo comportamento analisar as informações enviadas pela localização e tomar decisões.

5.5 Conclusões

Os métodos de localização, já documentados, são inúmeros e cabe ao projectista de uma aplicação saber escolher o que melhor se adequa às finalidades do seu projecto e às características do robô. No caso presente, a utilização do método baseado em pontos de referências, pareceu o mais adequado e permitiu que o módulo tomasse caminhos bastantes adequados às características do ERS-7 e ao projecto presente. A nível de características do ERS-7 seria impensável implementar um método de localização que não fosse fortemente apoiado no sensor de visão, já que este é uma das mais valia quanto toca a retirar informação fidedigna do meio ambiente.

Por outro lado, se o módulo de visão é a fonte de informação do módulo de localização, a eficácia e fiabilidade do módulo de localização será a fortaleza do sistema de navegação, no módulo comportamento. Tendo, por isso, em atenção as diversas situações a que o módulo comportamento terá de atender, dois módulos de localização foram desenvolvidos, nomeadamente a localização relativa e a absoluta. No próximo capítulo referente ao comportamento poderá ser visto em detalhe a utilização de cada um deste tipos de localização e o modo como o módulo comportamento utiliza as informações da localização (de modo passivo e de modo activo).

Capítulo 6

6. Comportamento

“In preparing for battle I have always found plans useless, but planning is indispensable”

Dwight Eisenhower

6.1 O Conceito

Quando se pensa no comportamento de um robô é impossível escudá-lo das influências vindas do exemplo animal (inclusive do humano), não só a nível da concepção como dos resultados esperados. Nesta perspectiva, o princípio básico do comportamento do ERS-7 tem muito de orgânico – responder com acções aos estímulos recebidos (ver figura 18).

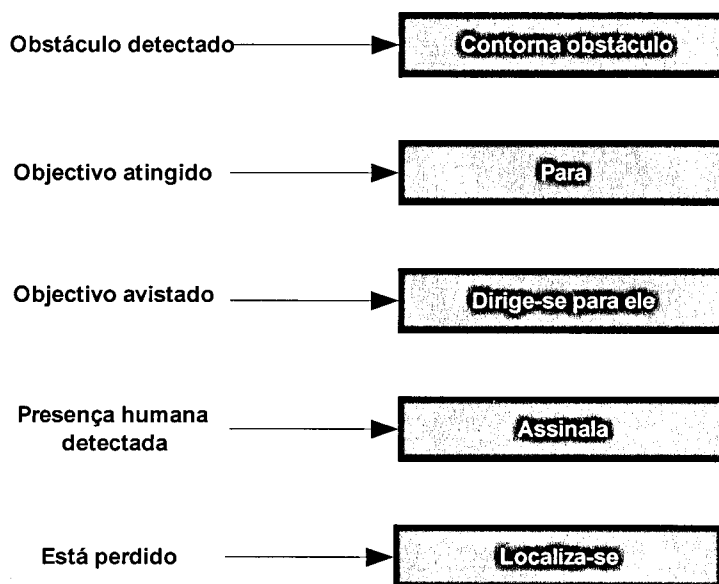


Figura 18 Estímulos versus respostas

Para além desta ideia simples de ver o comportamento como uma série de respostas despoletadas por estímulos, é importante também considerar este módulo na sua vertente

temporal [Arkin, 1998] (o que é efectuado nas figuras 19 e 20). Como se pode observar, o comportamento é responsável:

- Por assinalar a detecção de um intruso ou movimento;
- Pela navegação de objectivo em objectivo;
- Pelo desvio e contorno de obstáculos;
- Pela paragem sempre que um objectivo é atingido para detecção de movimento;
- Pela localização activa.

O comportamento implementado permite que o ERS-7 seja capaz de se ir adaptando ao ambiente que o rodeia não tendo por isso à priori um plano, mas ter a possibilidade de ir planeando.

O ciclo representado na figura 19 é repetido infinitamente até o programa em execução no ERS-7 ser parado, e mostra as decisões que o módulo comportamento tem de tomar no início de cada ciclo. Este ciclo é executado 30 vezes por segundo, sendo esta velocidade ditada pelo sensor de imagem (recebe 30 frames/s), que para além de ser uma importante fonte de informação, é o que mais rápido debita informação. Assim, cada vez que o módulo de visão recebe uma imagem e processa a informação nela contida, o módulo de localização actualiza as informações ou os sensores de distância apresentam novos valores, o comportamento está preparado para lidar com eles.

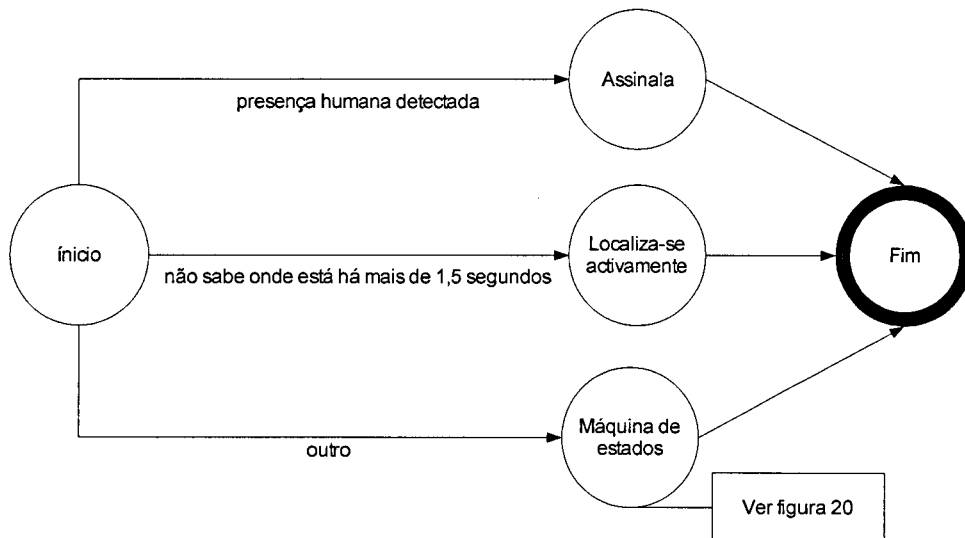


Figura 19 Comportamento do ERS-7

A lógica desta estrutura para o comportamento reside em agrupar sob a forma de uma máquina de estados (ver figura 20) uma série de acções que são executadas dentro de uma lógica sequencial e em considerar que tanto a necessidade da localização activa como de

assinalar a detecção de presença humana são situações de excepção que podem ocorrer para qualquer fase em que a máquina de estados se encontre.

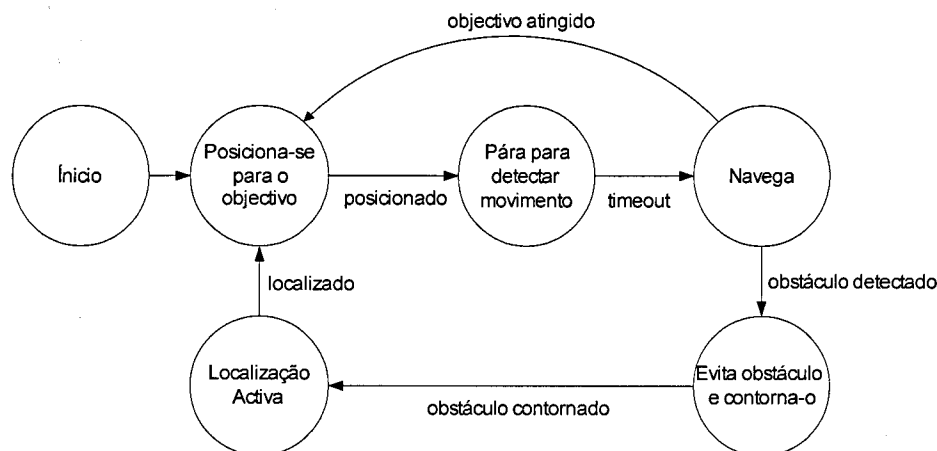


Figura 20 Máquina de estados

Os próximos pontos deste capítulo pretendem ver em mais detalhe o comportamento, nomeadamente o sistema de navegação e a capacidade de detecção e desvio de obstáculos.

6.2 Localização Activa

Uma das situações de excepção do comportamento do ERS-7, mencionada anteriormente, é a localização activa. Esta consiste em fornecer activamente dados aos módulo de localização para que o ERS-7 possa encontrar as suas coordenadas (válidas). Para isso o ERS-7 roda sobre si mesmo sem sair do mesmo sítio.

O ERS-7 considera que está perdido, e localizasse activamente, se estas condições se verificarem simultaneamente:

- Encontra-se no estado de navegação;
- Não vê o objectivo há mais de 1,5 segundos;
- O parâmetro probabilidade retornado pelo módulo de visão é inferior a 80%.

O pseudo-código deste algoritmo, para quando ele se está a dirigir para um determinado objectivo, é apresentado na figura 21.

```

1   Probabilidade_minima=80
2
3   Se objectivo é poste amarelo_rosa
4       Se viu poste amarelo_rosa
5           então continua a navegar
6
7       Senão
8           Se viu qualquer outro poste
  
```

```
9           e coordenadas são válidas
10          e probabilidade a elas associada >= Probabilidade_minima
11          então o ERS-7 posicionasse para o objectivo
12
13          Senão continua a rodar
```

Figura 21: Pseudo-código da Localização Activa

6.3 Sistema de Navegação

O objectivo do algoritmo de navegação é permitir que o ERS-7 seja capaz de se deslocar do ponto A para o ponto B. Parece simples?

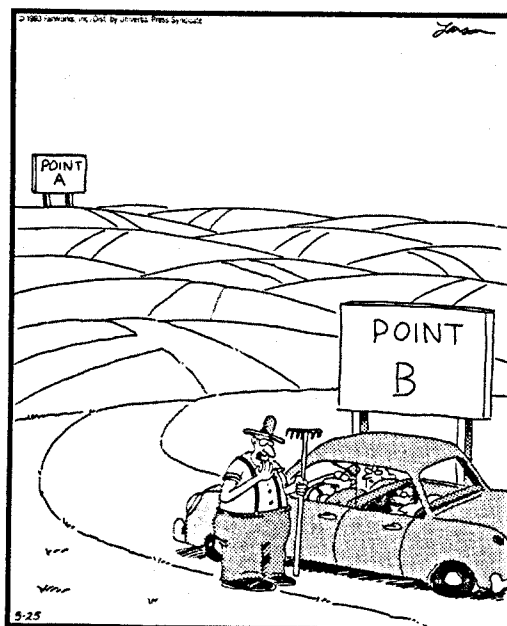


Figura 22 The Far Side de Gary Larson

Para a resolução deste problema várias outras questões tiveram de ser respondida, tais como:

- Como é que o ERS-7 sabe onde está o objectivo relativamente a ele se o estiver a ver?
- E se não o estiver ver?
- Como é que o ERS-7 sabe se já atingiu o objectivo?
- Como é que o ERS-7 pode compensar eventuais desvios de rota?
- E se se deparar com um obstáculo no caminho para o objectivo?

Exceptuando a última pergunta (cuja resposta será tratada com detalhe no próximo ponto), as restantes perguntas fazem ver o quanto o sistema de navegação está dependente da informação recebida do módulo de localização. Assim sendo, a navegação é totalmente baseada nos postes, sendo eles que, através do módulo de localização (relativa ou global) ou directamente do através do módulo de visão, permitem que o sistema de navegação seja capaz dar resposta ás perguntas anteriores.

O percurso que o ERS-7 deve percorrer é apresentado na figura 23. Se o ERS-7 não encontrar obstáculos no decorrer do caminho, a configuração da trajectória deverá aproximar-se à apresentada na figura. Contudo, o mais provável é que surjam obstáculos e, por isso, o que o sistema de navegação define são pontos finais, pontos finais e sequência no percurso total entre os vários pontos. Todo o caminho intermédio será da responsabilidade do comportamento do ERS-7, que poderá ter de definir estratégias para contornar obstáculos da melhor forma possível.

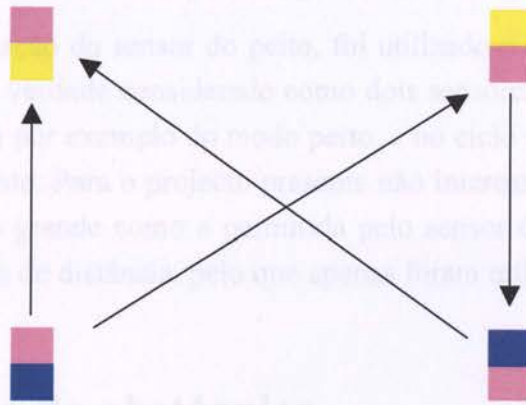


Figura 23 Sequência de pontos objectivos no percurso do ERS-7

6.4 Detecção e Desvio de Obstáculos

A detecção e desvio de obstáculos, ainda que sendo uma parte importante da navegação do ERS-7, pode ser vista como um sistema independente que é iniciado sempre que um obstáculo está no caminho do robô e como uma interrupção ao normal funcionamento do sistema de navegação.

6.4.1 Detecção de obstáculos

A detecção é exclusivamente realizada através dos três sensores de distância que o ERS-7 possui na cabeça e no peito. A tabela 4 apresenta as características dos sensores utilizados.

Sensor	Limite inferior (cm)	Limite superior (cm)
Peito	10	90
Cabeça (perto)	5	50
Cabeça (longe)	20	150

Tabela 4: Características dos sensores de distância do ERS-7

Estas características são retiradas das informações que a Sony disponibiliza à cerca deste modelo (ver [Model_Information, 2004]). Contudo, a posição do ERS-7 considerada pela Sony difere da utilizada neste projecto, em que o ERS-7 se desloca, não apoiado nas extremidades das patas, mas sobre a parte inferior destas, como se andasse sobre os “cotovelos”. Dada esta situação, o limites superior do sensor do peito apresentado não pode ser o acima considerado, uma vez que este se encontra a apontar para o chão. Assim, o limite superior real, para a posição adoptada pelo ERS-7, teve de ser determinado, o que foi feito após algumas experiências. O valor passível de usar como limite máximo do sensor do peito é de 16cm. Acima deste valor a percentagem de erro tornava-se demasiado elevada, tendo-se neste caso uma percentagem de erro de aproximadamente 5%.

Para além da utilização do sensor do peito, foi utilizado o sensor da cabeça. O sensor da cabeça pode ser na verdade considerado como dois sensores em um, sendo que num ciclo são lidos os valores por exemplo do modo perto, e no ciclo seguinte os valores do longe, e assim sucessivamente. Para o projecto presente não interessava detectar obstáculos a uma distância assim tão grande como a permitida pelo sensor da cabeça longe, mas antes na gama dos 5 a 30 cm de distância, pelo que apenas foram utilizados os valores do sensor da cabeça de perto.

6.4.2 Desvio de obstáculos

As principais preocupações no desenvolvimento do sistema de desvio de obstáculos, uma vez detectados, foram:

- Perceber quais as competências a atribuir ao sistema de desvio de obstáculo;
- Estudar a melhor forma de o ERS-7 contornar o obstáculo;
- Estudar como o ERS-7 poderia continuar a deslocar-se para o objectivo após o contorno do obstáculo ter provocado um considerável desvio de rota;

A figura 24 apresenta o algoritmo implementado sob a forma de um diagrama. Seguidamente passa-se a apresentar as soluções encontradas para as preocupações mencionadas anteriormente.

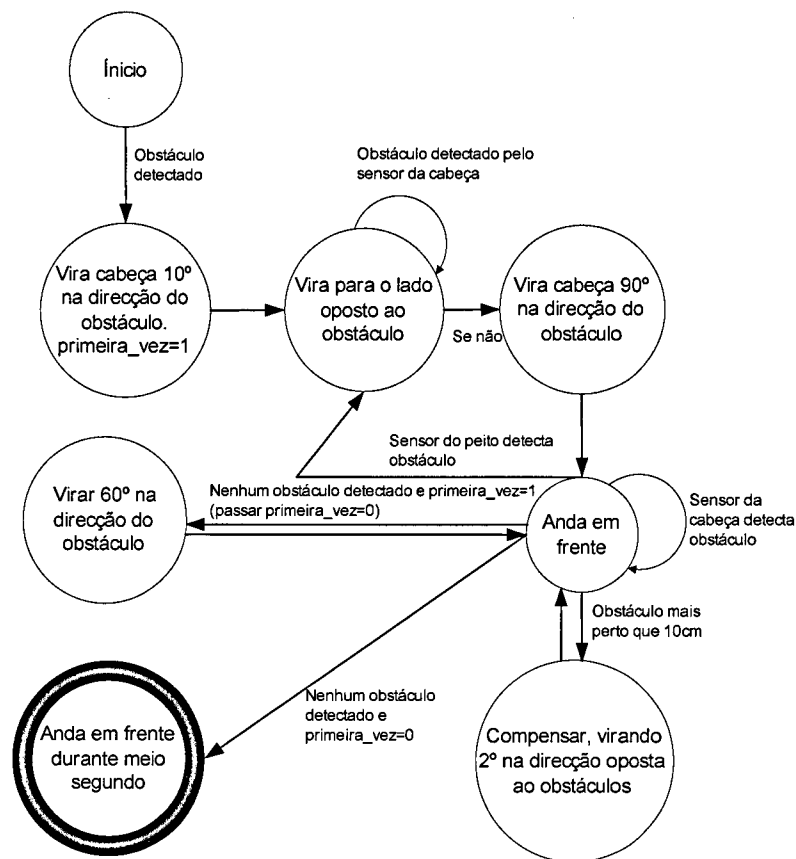


Figura 24 Algoritmo de desvio de obstáculo

As competências do Sistema de Desvio de Obstáculo são então:

- Evitar o obstáculo de forma lógica;
- Permitir contornar um obstáculo qualquer que seja a sua forma;
- Contornar o obstáculo de forma a ficar numa zona livre desse obstáculo no caminho para o objectivo;
- Durante o processo de contorno do obstáculo estar alerta para o surgimento de novos obstáculos.

Um método muito próprio foi desenvolvido para que o ERS-7 vá de encontro às competências acima definidas. A figura 25 apresenta de forma esquemática o que acontece quando o ERS-7 executa o algoritmo de desvio de obstáculos (ver figura 24).

O modo como o ERS-7 contorna um obstáculo, tido como a melhor maneira encontrada, está intimamente ligado à última preocupação apresentada referente a como é que o ERS-7 consegue continuar a deslocar-se para o objectivo depois de ter sofrido este desvio de rota. Tal como foi referido nas competências tidas para o sistema de desvio de obstáculo, quer-se que o obstáculo seja contornado de tal forma que o ERS-7, no final desse processo, esteja numa zona em que o caminho para o objectivo esteja desimpedido. Contudo, não foi

definida como competência deste sistema, que o ERS-7, também no final deste algoritmo, esteja orientado para o objectivo. Essa função é deixada ao módulo comportamento: se o objectivo não estiver em linha de vista no fim do desvio do obstáculo é necessário que o ERS-7 proceda a localizar-se activamente.

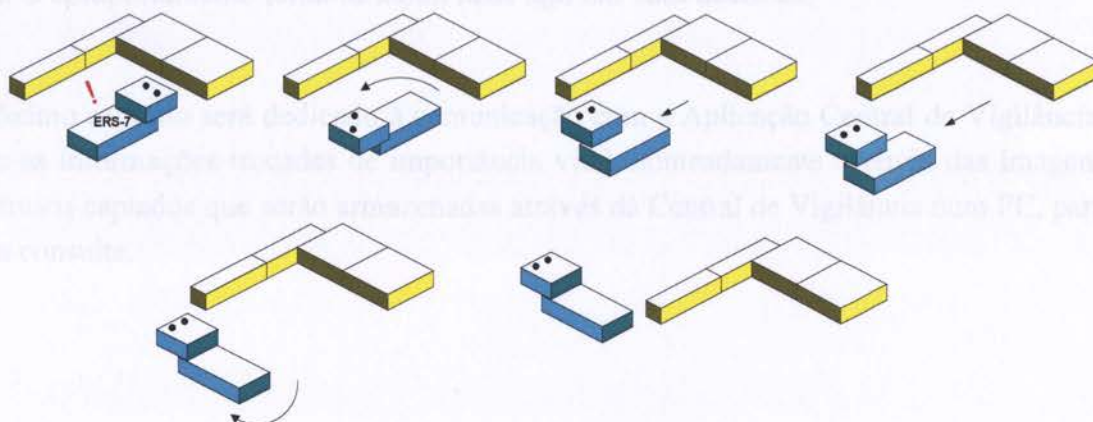


Figura 25 ERS-7 contorna obstáculo

6.5 Conclusões

O comportamento do ERS-7 é baseado em reacções. Reacções à chegada de estímulos do mundo exterior, ou à falta deles. Apesar desta faceta dinâmica do comportamento, de acção quase instintiva face ao mundo exterior, a base do comportamento é uma máquina de estados, onde a missão nuclear do ERS-7 está definida - executar um dado percurso pela ordem definida, enquanto procura por situações de excepção.

Nesta fase do projecto, é possível fazer já um paralelo entre o sistema implementado e os sistemas de segurança tradicionais, referidos no segundo capítulo. Já existe uma mobilidade associada à autonomia de análise das imagens captadas (que o módulo de visão permite), tendo sido esta uma das motivações para o desenvolvimento deste sistema. Esta mobilidade proporciona também autonomia ao ERS-7, sendo responsáveis por isso os algoritmos de: navegação, detecção e contorno de obstáculo e localização activa;

Concretizou-se neste capítulo, a importância das informações enviadas pelo módulo localização para o módulo comportamento. Se as coordenadas (relativas ou globais) calculadas no módulo de localização, repetidamente, tiverem uma probabilidade de estarem correctas acima da realidade, o módulo comportamento perderá muito mais tempo na tarefa de localizar-se activamente, sendo tempo que não está a procurar intrusos, que é a tarefa primária deste sistema.

Para além da importância da localização, verificou-se que o método de localização utilizando pontos de referência apresenta-se extremamente adequado ao sistema de navegação implementado (e comportamento em geral), permitindo utilizar formas de localização extremamente simples, como a relativa, e mais complexas quando a situação o exige. O comportamento torna-se assim mais ágil nas suas decisões.

O próximo capítulo será dedicado à comunicação com a Aplicação Central de Vigilância, sendo as informações trocadas de importância vital, nomeadamente o envio das imagens de intrusos captados que serão armazenadas através da Central de Vigilância num PC, para futura consulta.

Capítulo 7

7. Aplicação Central de Vigilância

7.1 Visão Geral

A capacidade do ERS-7 comunicar, através da rede sem fios, com uma aplicação exterior a si é uma importante faceta deste sistema. Permite ter duas aplicações completamente diferentes em estreita colaboração, tirando-se daí enorme vantagem.

Exemplo disso é a memória disponível para armazenamento no ERS-7 que é extremamente limitada, reduzida ao espaço existente no *memory stick* (que já está ocupado pelo código que está a ser executado). Por isso, ser capaz de transferir as imagens para um PC, onde espaço para armazenamento não é um problema, é uma grande vantagem. Outro aspecto que deve ser tido em consideração, apesar da ideia principal do sistema desenvolvido é de o ERS-7 ser o mais autónomo possível, é o facto de muitas vezes poder ser vantajoso comandar o ERS-7 à distância para iniciar ou parar (situação de emergência por exemplo).

Este capítulo descreverá, então, a comunicação entre o ERS-7 e a Aplicação Central de Vigilância, focando-se em primeiro lugar na transferência de dados e em seguida descrevendo em pormenor a Aplicação Central de Vigilância.

7.2 Conceito

A Aplicação Central de Vigilância tem duas principais funcionalidades:

- ➔ Permitir controlar o ERS-7 remotamente;
- ➔ Armazenar as imagens de intrusos detectados pelo ERS-7.

Para além destas funcionalidades, outras foram consideradas como secundárias, mas que poderiam trazer algum interesse extra a esta aplicação. Tratam-se das possibilidades de mostrar em tempo real o que o ERS-7 está a ver e da visualização do intruso logo que seja detectado pelo ERS-7.

Os dados que irão ser transferidos entre o ERS-7 e a Aplicação Central de Vigilância, são:

- ➔ Comandos iniciar e parar;
- ➔ Imagens de intrusos detectados.

Como pode ser visto na figura 26 o protocolo utilizado é o TCP, pelos que as mensagens serão as típicas utilizadas nesse protocolo (ver [TCP, 2000]).

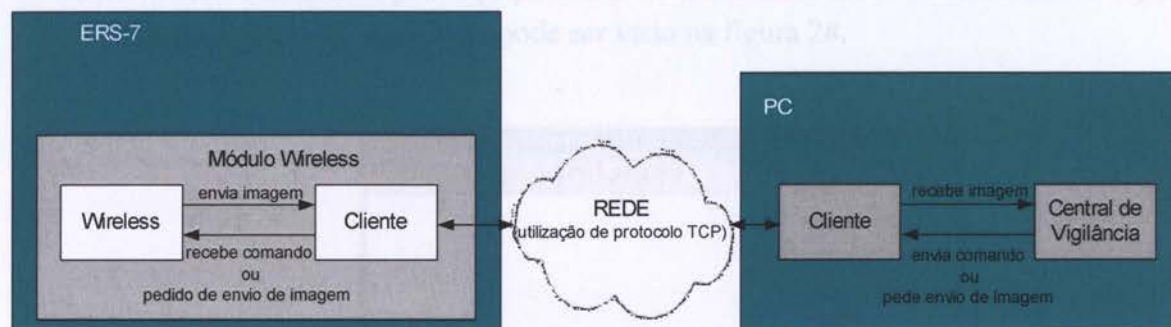


Figura 26 Transferência de dados

As funcionalidades e dados transmitidos são agora mostrados sob a forma de uma série de casos de uso (apresentados na figura 27).

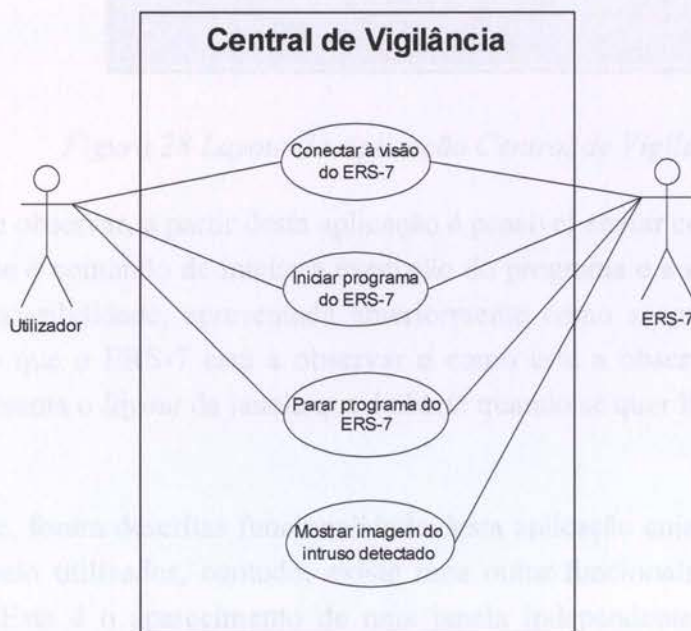


Figura 27 Casos de uso da Aplicação Central de Vigilância

7.3 Interface

Para além do desenvolvimento de uma aplicação que estivesse de acordo como o diagrama de casos de uso acima apresentado, teve-se como objectivo que a Aplicação Central de Vigilância fosse uma interface amigável entre o ERS-7 e o Humano que eventualmente possa estar a supervisionar o sistema. A interface resultante, surgiu da adaptação de uma já existente, desenvolvida pela equipa rUNSWift, chamada RoboCommander. O *layout* da Aplicação Central de Vigilância pode ser visto na figura 28.

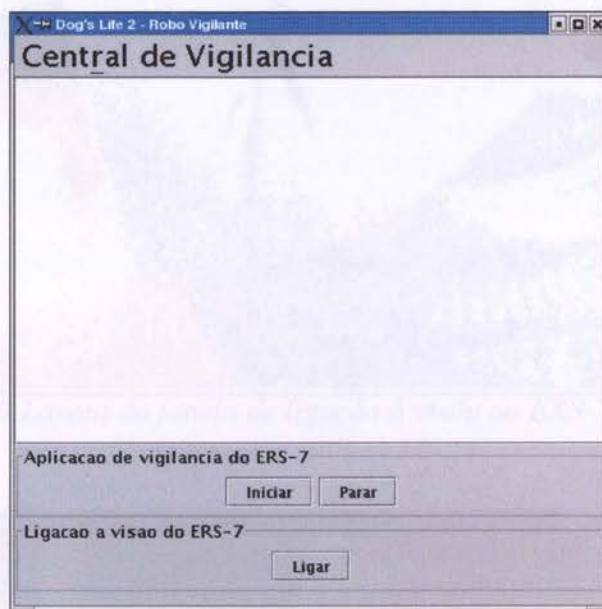


Figura 28 Layout da Aplicação Central de Vigilância

Como se pode observar, a partir desta aplicação é possível enviar comandos para o ERS-7, nomeadamente o comando de iniciar a execução do programa e a sua paragem. Para além disso, outra possibilidade, apresentada anteriormente como secundária, é a de ver, em tempo real, o que o ERS-7 está a observar e como está a observar (botão “Ligar”). A figura 29 apresenta o *layout* da janela que é aberta quando se quer ligar à visão do ERS-7.

Anteriormente, foram descritas funcionalidade desta aplicação cuja execução, ou não, são controladas pelo utilizador, contudo, existe uma outra funcionalidade que é controlada pelo ERS-7. Esta é o aparecimento de uma janela independente sempre que o ERS-7 detecta um intruso ou movimento (o *layout* dessa janela é apresentado na figura 30).

A imagem mostrada na janela será gravada pela aplicação com o nome “hora_minuto_segundo”, conforme o valor destes no relógio do computador na altura em que a imagem é gravada. A diferença entre a detecção do intruso por parte do ERS-7 e a

gravação da imagem no disco do computador não excede 1 segundo, por isso esta denominação é válida. A figura 31 mostra onde cada imagem é gravada conforme a data em que chegou ao computador.



Figura 29 Layout da janela de ligação à visão do ERS-7



Figura 30 Layout da janela controlada pelo ERS-7

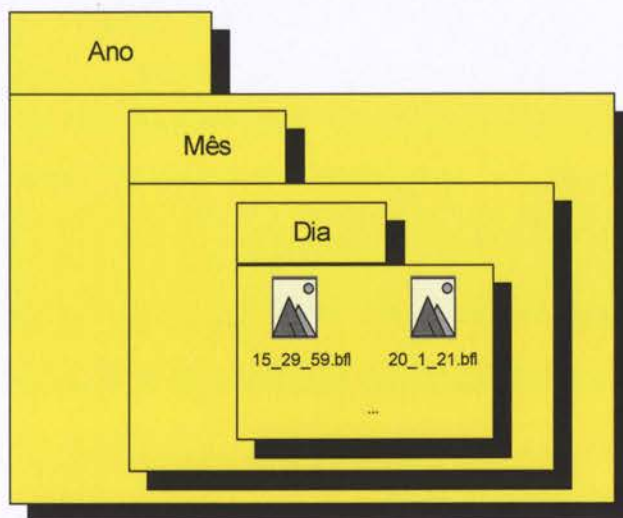


Figura 31 Armazenamento do ficheiro que contém a imagem enviada pelo ERS-7

7.4 Conclusões

Se um dos objectivos deste projecto é que o ERS-7 desempenhe as suas tarefas de forma autónoma, não é de descurar a necessidade de uma aplicação que vise situações em que se pretende ter algum controlo sobre o robô ou monitorizá-lo. Se este projecto visa implementar um sistema de segurança, também ele deve considerar situações em que a segurança de propriedade alheia ou de pessoas pode ser posta em causa pelo ERS-7. Assim, os botões parar e iniciar, foram colocados nesta aplicação como forma de numa situação de emergência ter maneira de parar o ERS-7 remotamente, e voltar a iniciar o seu programa posteriormente.

Mas se a Aplicação Central de Vigilância é uma forma de controlar o ERS-7, permite também a última forma de autonomia deste sistema: armazenar de forma adequada e lógica os ficheiros que contêm as imagens de intrusos detectados pelo ERS-7.

Estando o sistema descrito nos seus conceitos e implementação, passa-se à descrição dos testes efectuados, os resultados obtidos e análises feitas a partir deles.

Capítulo 8

8. Testes e Análise de Resultados

Os testes efectuados ao sistema, têm por objectivo determinar a eficácia dos algoritmos implementados e poder a partir dos resultados definir estratégias tendo em vista trabalho futuro. Tentou-se que os teste fossem concretos, resultando deles medidas numéricas, de modo a que a análise de resultados seja o menos subjectiva possível.

8.1 Detecção de intruso

8.1.1 Calibração de cor

A visão do ERS-7 tem à sua disposição 11 cores para classificar as imagens que recebe. O número reduzido de cores, faces às milhares existentes no mundo real, faz com que o robô tenha que usar um método de aproximação. Para que esta aproximação seja o mais correcta possível o processo de calibração (ver Anexo D) deve ser feito com extremo cuidado. Neste teste, interessa apenas pensar na cor laranja, que é a cor com que o ERS-7 irá ver as tonalidades do rosto.

Uma das grandes preocupações, ao pensar neste algoritmo como forma de detectar pessoas quando o ERS-7 se encontra em movimento, foi de que as tonalidades de cor de pele abrangidas por este método poderiam ser extremamente restritas. Para além desta restrição, outra questão a ponderar foi a má interpretação como cor pertencente a um rosto, de um objecto que não o fosse, o que levaria a resultados enganosos.

O teste efectuado consistiu na utilização de 9 formas ovais (como se de amostras de pele se tratassem), com aproximadamente as dimensões de um rosto, que foram colocadas diante da câmara do ERS-7 a várias distâncias . A figura 32 mostra as tonalidades utilizadas, e a figura 33 o aspecto de uma amostra vista pelo ERS-7.

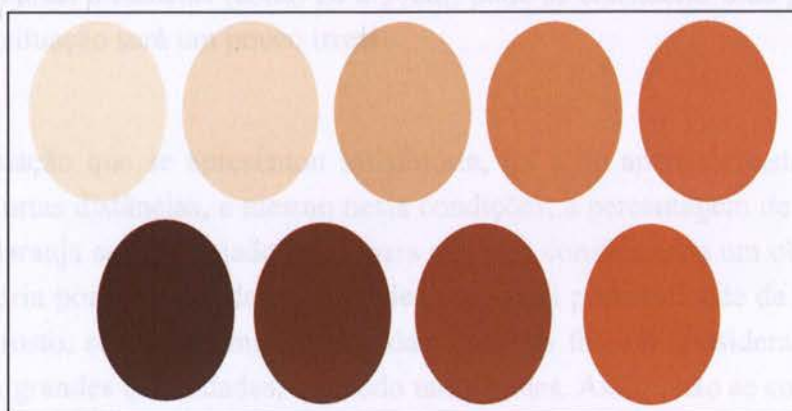


Figura 32 Forma e tonalidades utilizadas no teste

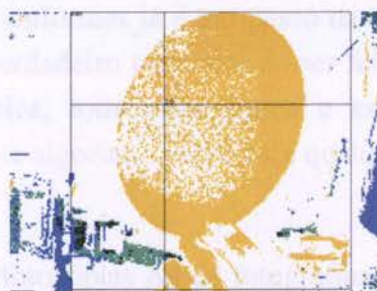


Figura 33 Amostra vista pelo ERS-7

Os resultados obtidos são mostrados na tabela 5. É de referir que os resultados obtidos não são válidos para quaisquer condições: o espaço de teste foi iluminado uniformemente com luz directa.

AMOSTRA Nº	Percentagem de píxeis identificados para a distância (cm):					
	20	30	50	100	150	200
1	10%	2%	<1%			
2	50%	30%	5%	<1%		
3	95%	80%	60%	20%	10%	5%
4	98%	98%	98%	98%	100%	100%
5	100%	100%	100%	100%	100%	100%
6	100%	100%	100%	100%	100%	100%
7	98%	98%	90%	80%	90%	100%
8	60%	50%	40%	10%	5%	<1%
9	10%	5%	<1%			

Tabela 5: Resultados do teste gamas de tonalidade

Numa primeira análise dos resultados, pode-se dizer que a gama de cores considerada pelo ERS-7 é bastante abrangente, contra as expectativas, incluindo desde os tons mais claros de pele aos mais escuros. Dentro das amostras utilizadas, pode-se dizer que uma boa aproximação à gama de tonalidades que o ERS-7 utilizará como pertencentes a um rosto humano, no contexto de execução da aplicação, será da amostra 3 à amostra 7. Note-se

que para curtas distâncias (como 20 a 30cm) pode-se considerar uma gama mais alargada, mas essa situação será um pouco irreal.

Outra situação que se apresentou satisfatória, foi a de apenas considerar as cores mais claras a curtas distâncias, e mesmo nesta condições, a percentagem de píxeis identificados com cor laranja ser demasiado baixa para que seja considerados um objecto. Esta situação é satisfatória porque, uma das tonalidades com mais probabilidade de ser confundida com a de um rosto, se os tons mais claros das amostras fossem considerados, seria o branco, que é, em grandes quantidades, utilizado nas paredes. Assim, não se corre esse risco.

8.1.2 Rostos

Se a detecção de formas uniformes já é um passo na direcção do sucesso do algoritmo de detecção de intrusos, o verdadeiro teste terá de ser feito com rostos, que contêm cabelos, barba, sobrancelhas, lábios, tons mais claros e escuros de pele misturados, etc. É importante perceber se este algoritmo serve para qualquer tipo de rosto.

Neste teste utilizaram-se fotocópias A4 de fotografias de rostos, tentando-se que fossem o mais variados possíveis em tonalidades (utilizando-se as determinadas como viáveis no teste anterior) e homogeneidade da pele. Todos os testes foram realizado à distância inicial de 150cm do ERS-7. Não tendo sido, o rosto, identificado como tal a esta distância, a imagem é aproximada, apontando-se a distância a que passa a ser reconhecida.

Os primeiros resultados apresentados, são referentes a rostos que têm, pelo menos, uma grande área homogénea (por exemplo na testa ou na parte lateral do rosto) o que facilita a tarefa do módulo de visão. Por outro lado, tenta-se diversificar na tonalidade (ver Tabela 6).




Fotografias			
Foi identificado pelo ERS-7 como um rosto?	Sim	Sim	Não
A que distância? (cm)	120	80	---

Tabela 6: Resultados dos testes a primeiro conjunto de rostos

O segundo conjunto de resultados, apresentados na tabela 7, consideram rostos mais difíceis de identificar, também para três tonalidades diferentes de pele, devido à falta de grandes áreas homogêneas.




Fotografias			
Foi identificado pelo ERS-7 como um rosto?	Sim	Sim	Não
A que distância? (cm)	150	150	---

Tabela 7: Resultados dos testes a segundo conjunto de rostos

Várias conclusões, se podem tirar destes resultados. A primeira é que nem por o rosto ser mais homogêneo será mais facilmente identificado. Na verdade o que dita a fácil identificação, ou não, do rosto é a sua luminosidade. Se um rosto tiver grande áreas luminosas, como é o caso da primeira e da segunda fotografia do primeiro conjunto, estas serão interpretadas como zonas brancas o que só dificulta o processo de identificação. Os resultados obtidos levam a crer que, apesar do teste à calibração da cor, feito no ponto anterior, se ter apresentado como satisfatório, na prática, deixa de fora algumas tonalidades que se esperavam que o ERS-7 considerasse (como é o caso dos tons mais escuros e mais claros). O problema agora posto, é que o alargamento da gama de tonalidades, a que o ERS-7 irá denominar laranja e tentar formar delas um objecto, fará com que muitos mais objectos sejam erroneamente identificados como rostos. Nesta perspectiva de salvaguardar a percentagem de erro deste algoritmo, apesar da gama de tonalidades não ser tão abrangente como o esperado, manteve-se a calibração da cor tal como está.

8.2 Navegação

8.2.1 Sem obstáculos

Tem-se por objectivo, neste teste, analisar o desempenho do ERS-7, conforme a área a vigiar estiver rodeada por paredes, ou no caso de não estar, determinando a taxa de sucesso do algoritmo de navegação em cada um dos casos. Por sucesso entende-se: o ERS-7 fazer o percurso sem necessitar da intervenção humana em caso algum.

O ERS-7 executa o seu programa normalmente, dentro de uma área com $4m^2$, com a única diferença de se saber à priori que o robô não irá encontrar nenhum obstáculo no seu caminho.

Os resultados obtidos são apresentados na tabela 8, para o primeiro caso (área não rodeada por parede) e para o segundo (área rodeada por parede). Pode-se observar que a taxa de sucesso no primeiro caso é de cerca de 50%, enquanto que no segundo é de 100% em termos da autonomia que o algoritmo de navegação confere ao ERS-7.

Repetição Nº	Área não rodeada por parede			Área rodeada por parede	
	Tempo (s)	(1)	(2)	Tempo (s)	(1)
1	151	sim	Sim	304	não
2	157	sim	Sim	240	não
3	102	não	Não	321	não
4	219	Não	Sim	416	não
5	120	não	Não	345	não
Média	149,8			325,2	
(1)	foi necessária intervenção humana?				
(2)	o robô saiu da área a vigiar?				

Tabela 8: Resultados dos testes ao algoritmo de desvio de obstáculos

Perante as taxas de sucesso obtidas, os resultados são contundentes: o ERS-7 deve deslocar-se dentro de uma área resguardada por paredes para que o algoritmo de navegação seja eficaz. Contudo, como se pode observar na tabela de resultados, o tempo de execução do percurso aumenta substancialmente, uma vez que o ERS-7 irá ter de lidar com as paredes, o que para ele são sinónimo de obstáculos. Este tempo extra que o ERS-7 leva a fazer o circuito pelos quatro postes, no segundo caso, não é nada crítico uma vez que enquanto deambula pela a área a vigiar pode também estar a desempenhar a sua tarefa principal - procurar intrusos e capturar a sua imagem.

8.2.2 Desvio de obstáculos

Uma das questões colocadas no capítulo referente ao módulo comportamento (Capítulo 6) dizia respeito a como o ERS-7 contorna um obstáculo. Se no ponto referente ao contorno de um obstáculo (6.4.2), ficou esclarecido qual o método utilizado, resta contabilizar a sua eficácia temporal e ver a facilidade com que o ERS-7 volta a estar no caminho para o objectivo depois do desvio de rota provocado pelo obstáculo.

Neste teste utilizaram-se três obstáculos diferentes classificados como fácil, médio e difícil de contornar (figura 34) e um cronómetro.



Figura 34 Obstáculos utilizados, da esquerda para a direita: fácil, médio e difícil de contornar

O teste processou-se da seguinte forma: estava o ERS-7 a navegar para o poste objectivo, quando a 40cm de si, surge um obstáculo (ver figura 35).

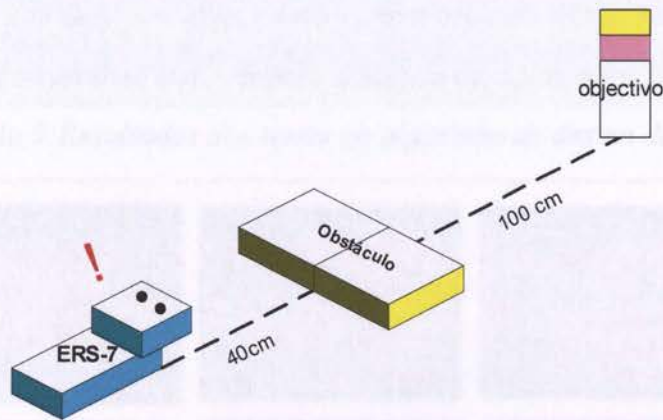


Figura 35 Esquema das condições dos testes

A tabela 9 apresenta os resultados obtidos, e as figuras 36 a 38 exemplos de como o ERS-7 se desviou dos obstáculos.

Repetição nº	TIPO DE OBSTÁCULO								
	fácil			médio			difícil		
	(1) segundos	(2) segundos	(3)	(1) segundos	(2) segundos	(3)	(1) segundos	(2) segundos	(3)
1	11,70	1,13	0	24,22	1,48	0	22,29	2,85	0
2	10,21	0,41	0	17,93	2,43	0	15,55	2,05	0

3	12,65	0,29	0	16,19	2,05	0	12,99	3,99	0
4	10,61	2,01	0	12,32	0,80	0	17,56	3,26	0
5	12,23	1,27	0	15,50	2,30	0	19,31	2,81	0
6	16,37	0,64	0	15,65	1,86	0	17,56	3,73	0
7	13,52	1,39	0	16,02	1,94	1	18,44	2,56	0
8	15,52	1,30	0	15,68	1,25	0	15,84	2,70	0
9	12,31	0,27	0	19,04	1,39	0	19,29	1,75	0
10	10,91	1,19	0	17,15	1,82	0	17,90	3,61	0
Média	12,60	0,99		16,97	1,73		17,67	2,97	

(1) tempo que demora a contornar o obstáculo

(2) tempo que demora a voltar a estar no caminho para o objectivo

(3) volta a esbarrar-se com o mesmo obstáculo depois de o contornar? (sim=1;não=0)

Tabela 9: Resultados dos testes ao algoritmo de desvio de obstáculos



Figura 36 ERS-7 desvia-se de um obstáculo do tipo fácil



Figura 37 ERS-7 desvia-se de um obstáculo do tipo médio

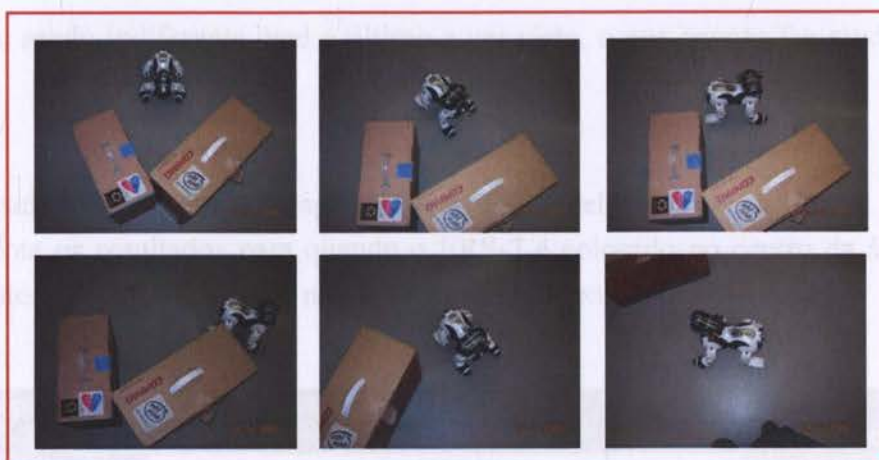


Figura 38 ERS-7 desvia-se de um obstáculo do tipo difícil

Mais do que os tempos que o ERS-7, leva a contornar os obstáculos ou a voltar a dirigir-se para o objectivo, é de referir a sua capacidade de lidar com várias configurações de obstáculos, como foi visto neste teste. Já os resultados numéricos apontam para tempos de contorno de obstáculo razoáveis, que sem dúvida podiam ser melhorados, mas que nem por isso deixam de por em causa o sucesso do algoritmo. Mas a surpresa deste algoritmo é a rapidez com que o ERS-7 consegue voltar a estar na direcção do objectivo: a média para o obstáculo mais difícil, que provoca um maior desvio de rota, não ultrapassa os 3 segundos, o que é extremamente satisfatório. Outra consideração que deve ser feita é que apenas 1 vez em 30 o ERS-7 voltou a esbarrar-se como o mesmo obstáculo imediatamente após a o ter contornado o que corresponde a uma taxa de sucesso do algoritmo de 96,7%.

8.2.3 Determinação de coordenadas

Passa-se agora a fazer o estudo da precisão do módulo de localização, no que diz respeito às coordenadas (x, y, θ) que calcula. A única maneira de confirmar se as coordenadas calculadas correspondem à realidade, é utilizando uma fita métrica, o que por si só irá introduzir alguns erros.

Para além das coordenadas, interessa também saber se a probabilidade associada às coordenadas dá de facto o seu grau de certeza, de acordo com o que é constatado na realidade.

O teste é feito com o ERS-7 parado, forçando a sua visualização de postes, rodando-o, no mesmo sítio, dentro de uma área quadrada com $2,25 \text{ m}^2$. Tentam-se obter coordenadas válidas, mas também inválidas, de modo a estudar o comportamento da probabilidade face às coordenadas obtidas. As coordenadas têm um valor esperado, tendo-se testado o algoritmo de localização para três pontos diferentes, e sempre utilizando os mesmo dois

postes, sendo indiferente qual o último a ver visto, o que apenas faz mudar a coordenada θ .

Os resultados obtidos são apresentados nas tabelas 10, 11 e 12. A primeira tabela apresenta os resultados para quando o ERS-7 é colocado no centro da área, e as tabelas seguintes quando o robô está numa situação mais extrema.

Repetição n°	ultimo poste visto	x	y	θ obtido	θ esperado	P (%)
1	rosa_azul	74,7	80,3	190	225	90
2	rosa_azul	87	87,7	189	225	95
3	rosa_azul	81,5	82,19	198	225	94
4	azul_rosa	86,7	87,4	330	315	83
5	azul_rosa	81,7	92	325	315	84
6	azul_rosa	82,2	91,7	319	315	92
7	azul_rosa	76,8	103,6	330	315	27
8	rosa_azul	60,5	70,3	211	225	31
9	rosa_azul	75,4	80,38	220	225	63
10	azul_rosa	71,9	69,5	330	315	40

Tabela 10: Resultados do teste de coordenadas para o ponto (85, 85)

Repetição n°	ultimo poste visto	x	y	θ obtido	θ esperado	P (%)
1	rosa_azul	41,2	81,3	202	225	96
2	rosa_azul	44,5	81,3	202	225	96
3	rosa_azul	42,0	77,4	199	225	85
4	rosa_azul	44,1	80,7	204	225	88
5	rosa_azul	45,2	87,3	207	225	90
6	rosa_azul	43,9	90,1	210	225	91
7	rosa_azul	24,3	99,4	208	225	50
8	azul_rosa	17,0	87,8	320	315	40
9	rosa_azul	38,0	72,9	201	225	56
10	rosa_azul	45,2	82,5	210	225	96

Tabela 11: Resultados do teste de coordenadas para o ponto (40, 85)

Repetição n°	ultimo poste visto	x	y	θ obtido	θ esperado	P (%)
1	azul_rosa	87,3	43,1	320	315	95
2	azul_rosa	87,2	41,5	319	315	96
3	azul_rosa	70,1	59,2	330	315	57
4	rosa_azul	69,4	31,6	324	225	31
5	rosa_azul	95,8	43,9	321	225	90
6	azul_rosa	82,5	41,7	332	315	88
7	azul_rosa	80,2	48,3	317	315	85
8	rosa_azul	79,3	15,9	340	225	42
9	azul_rosa	56,3	41,8	325	315	64
10	azul_rosa	87,4	45,2	320	315	95

Tabela 12: Resultados do teste de coordenadas para o ponto (85, 40)

A primeira conclusão a que é possível chegar é que o parâmetro probabilidade distingue bem entre as coordenadas que, dentro de uma certa margem de erro, podem ser consideradas válidas e as completamente despropositadas. Esta informação será valiosa para o módulo comportamento. Agora, dentro das coordenadas válidas a probabilidade não consegue classificar com muita precisão as que melhor se aproximam da realidade, podendo-se considerar que para probabilidades acima dos 80% de certeza que as coordenadas são válidas.

Outro aspecto importante é que o módulo de localização não piora os seus resultados pelo facto do robô se encontrar numa extremidade do espaço a considerar, como é demonstrado nos resultados.

8.2.4 Localização Activa

Já que o tempo dispendido na localização activa, é tempo em que o ERS-7 não está a desempenhar a sua tarefa primária, pretende-se estimar o tempo que leva o ERS-7 a localizar-se activamente conforme o número de postes que estão à vista e verificar se no final da localização activa o ERS-7 fica ou não bem posicionado.

O ERS-7 é colocado no centro da área a vigiar, com $4m^2$, orientado para o poste diametralmente oposto ao poste objectivo. Espera-se que no final da localização activa o ERS-7 se encontre de frente para o poste objectivo, passando a navegar para ele. Este teste foi executado estando três poste visíveis (postes que não são o objectivo) e cronometrou-se o tempo que o ERS-7 levou até se localizar e posicionar em direcção ao objectivo. O mesmo se passou tendo 2 postes visíveis, e por último testou-se, estando visível apenas o poste objectivo.

Os tempos cronometrados e a estimativas da direcção do ERS-7 no final do algoritmo, em relação ao poste objectivo são apresentados na tabela seguinte.

Repetição Nº	Tempo médio de localização, se estiverem visíveis:					
	3 postes (s)	(1)	2 postes (s)	(1)	apenas o poste objectivo (s)	(1)
1	1,84	10	3,06	20	2,62	40
2	2,30	90	2,99	60	2,91	40
3	2,94	40	3,42	90	2,54	30
4	3,79	30	2,61	20	3,04	10
5	3,37	40	3,41	50	3,23	50
6	3,44	30	2,75	40	3,22	10
7	2,40	10	3,51	90	2,95	50
8	3,64	60	3,60	30	3,21	10

9	2,96	90	3,24	10	3,11	10
10	3,85	70	3,25	10	3,25	40
Média	3,05	47	3,18	42	2,70	29
(1)	graus aproximados a que fica da direcção do objectivo					

Tabela 13: Resultados dos testes ao algoritmo de localização activa

Como se pode ver na tabela 13, a diferenças entre a situação que utiliza 3 postes e a que utiliza 2, quando observada a média do tempo que o ERS-7 demora a localizar-se, é praticamente nenhuma. Isto, porque o ERS-7 consegue localizar-se na primeira volta, logo que vê o segundo poste. Por isso, a situação de 3 postes disponíveis para o ERS-7 localizar-se acaba por corresponder à situação de ter só dois: o terceiro nunca é usado. Isto significa que o ERS-7 não perde a oportunidade de se localizar, não necessitando de voltas extras para esse fim. O mesmo acontece para a posição final do robô quando acaba de se localizar - acaba por ser idêntica nos dois primeiros casos. A responsabilidade da grande diferença entre a direcção do ERS-7 no final de se localizar e a direcção do objectivo, ocorrida em algumas repetições, é atribuída ao acumular de erros da odometria, sendo através desse processo que o ERS-7 se posiciona neste caso. Quanto à terceira situação, em que o robô apenas pode utilizar o poste objectivo para se localizar (o que implica a utilização de localização relativa) o tempo diminui, não de maneira muito expressiva mas evidente. A grande alteração, é sobretudo na posição final do robô, que melhora significativamente uma vez que, os erros acumulados pela odometria no processo de posicionamento em relação ao objectivo, neste caso não existem. Ainda assim, poderia espera-se que o ângulo final fosse 0 em todas as repetições do teste, contudo o ERS-7 considera-se posicionado logo que detecte o poste na sua imagem, quer esteja ainda um pouco desfasado da direcção certa. Mas esta diferença entre a direcção do ERS-7 e o poste, não impediu o robô de chegar ao objectivo, sem necessitar de se localizar novamente.

Capítulo 9

9. Conclusões e Perspectivas de Desenvolvimento

9.1 Conclusões

Este projecto permitiu demonstrar que o uso de um robô móvel como base de um sistema de segurança é viável, e que pode trazer vantagens face ao conceito tradicional de “cão de guarda”. As características extremamente completas do ERS-7 são aliadas a uma arquitectura potenciada pela descentralização e execução paralela e cooperativa de tarefas, tornando-se assim o sistema mais ágil face às respostas que lhe são exigidas.

Os algoritmos de visão são a base do sistema implementado, e permitem desempenhar duas importantes tarefas: detectar postes e detectar presença humana. O algoritmo de baixo nível de detecção de postes, pelo seu bom funcionamento, permitiu verificar que o ERS-7 não perde uma oportunidade de desempenhar as acções desencadeadas pelo estímulo “viu poste”, por não o ter identificado numa imagem. Exemplos dessa acções são a localização global e relativa que podem sempre confiar na informação recebida do módulo de visão.

Se uma parte deste sucesso deve-se aos algoritmos, outra deve-se à calibração da tabela look-up. Tendo o ERS-7 apenas 11 cores para atribuir a todas as percepções através da sua câmara, a calibração é um aspecto extremamente sensível. Sendo assim, é preferível que a detecção de objectos se baseie mais na sua forma e relações entre aglomerados do que quase totalmente na cor (como é o caso da detecção de intruso).

A localização é a espinha dorsal do comportamento do ERS-7, o que fez com que os métodos implementados de localização tivessem de ir ao encontro das necessidades desse módulo. As necessidades do módulo de navegação, em especial do seu algoritmo de

navegação, foram satisfeitas desenvolvendo-se a localização relativa, que se apresenta um método rápido e eficaz para situações mais simples, e a localização global, mais complexa, que retorna resultados fiáveis. A utilização do conceito de incerteza verificou-se uma mais valia para o ERS-7, uma vez que faz com ele não se tenha de preocupar com situações em que não sabe se as coordenadas estão bem ou não: sabe à priori qual o grau de incerteza, e pode decidir se está disposto ou não a arcar com as consequências da utilização de valores menos correctos.

Esta questão de arcar com as consequências diz respeito ao módulo de comportamento, que é o módulo de mais alto nível. O algoritmo de navegação desempenha a tarefa de controlar os pontos para onde o ERS-7 se deve dirigir e conta com duas das fortalezas deste módulo que são os algoritmos que tratam de situações de excepção, como é o caso do algoritmo de detecção e desvio de obstáculos, capaz de enfrentar qualquer configuração de obstáculo e rapidamente voltar a dirigir-se para o objectivo e o algoritmo de localização activa que fornece dados activamente ao módulo de localização, permitindo que mesmo que o ERS-7 se encontre na pior situação consiga recuperar rapidamente e voltar ao percurso pré-estabelecido.

A Aplicação Central de Vigilância desempenha um papel fundamental neste sistema, ainda que afastada do centro de acção. Armazena as informações enviadas pelo ERS-7, trazendo mais autonomia a este sistema, uma vez que nem a tarefa de armazenamento lógico das imagens captadas pelo ERS-7, em sítio para referência futura, precisa de intervenção Humana. Está também contemplada nesta aplicação a intervenção em situações de emergência, no caso do ERS-7 se revelar momentaneamente um perigo para alguém ou para alguma coisa, permitindo que o ERS-7 seja parado remotamente.

De uma forma geral, este sistema de vigilância toca um pouco de cada umas das áreas principais da robótica móvel, como é o caso da navegação em ambientes dinâmicos, localização, comportamento reactivo, e conseguiu estender-se por todos os problemas apresentados e dar-lhes resposta.

9.2 Perspectivas de Desenvolvimento

O trabalho realizado possui ainda algumas vertentes por explorar, como é o caso de:

- Utilização de base de dados para reconhecimento de faces, permitindo distinção entre intruso e pessoal autorizado;

- Introdução de algoritmos de aprendizagem, nomeadamente construção dinâmica do mapa do mundo;
- Sistema distribuído de vigilância constituído por vários agentes capazes de cooperar entre si para o mesmo fim;

Referências Bibliográficas

[AIBO, 2005] *Sony AIBO Europe - Official Website - User guides* [Em Linha]. Disponível em http://www.eu.aibo.com/1_2_library.asp [Consultado em 12/3/2005]

[Aibo_League, 2005] *WebHome - Website - Four-Legged League* [Em Linha]. Disponível em <http://www.tzi.de/4legged/bin/view/Website/WebHome> [Consultado em 20/4/2005]

[Arkin, 1998] Arkin, R., *Behavior-Based Robotics*, MIT Press, pp. 65-120, 1998.

[Bugard et al., 1996] Bugard, W., Fox, D., Hennig, D., Schmidt, T., *Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids*, In the fourteenth National Conference on Artificial Intelligence (AAAI-96), 1996.

[Fox et al., 1999] Fox, D., Bugard, W., Thrun, S., *Markov Localization for Reliable Robot Navigation and People Detection*, Dagstuhl Seminar on Modelling and Planning for Sensor-Based Intelligent Robot Systems, 1999.

[Kaelbling et al., 1996] Kaelbling, L., Kurien, J., Cassandra, A., *Acting under Uncertainty: Discrete Bayesian Models for Mobile-Robot Navigation*, em Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, 1996.

[Kortenkamp et al., 1998] Kortenkamp, D. Bonasso, R., Murphy, R., *Artificial Intelligence and Mobile Robots*, In AAAI Press/The MIT Press, 1998.

[Lopes et al., 2000] Lopes, L, Lau, N. e Reis, L. P., *Control and Decision-Making demonstrated on a Simple Compass-Guided Robot*, SMC2000 - Proc. of the 2000 IEEE International Conference on Systems, Man and Cybernetics. Nashville, Tennessee, 2000.

[Model_Information, 2004] *OPENR SDK Model Information for ERS-7* [Em Linha]. Disponível em http://www.cs.lth.se/DAT125/docs/ModelInformation_7_E.pdf [Consultado em 29/2/2005]

[Neves, 1999] Neves, Maria da Conceição, *Arquitetura Híbrida, Distribuída e Adaptável para o Controlo de um Robô Móvel: Metodologia e Implementação*, pp. 105, 1999.

[Nourbakhsh, 1998] Nourbakhsh, I., *Dervish: An Office-Navigating Robot*, em *Artificial Intelligence and Mobile Robots*, The AAAI Press/The MIT Press, pp. 73-90, 1998.

[OPENR] *[AIBO SDE] official web site* [Em Linha]. Disponível em <http://openr.aibo.com/> [Consultado em 28/2/2005]

- [Reis,2003] L. P. Reis, *Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico*, Tese de Doutoramento, Faculdade de Engenharia da Universidade do Porto, pp. 1-10, Porto, 2003
- [Remagnino et al., 1998] Remagnino, P., Maybank, S., Fraile, R., Baker, K., Morris, R., Chapter *Automatic Visual Surveillance of Vehicles and People in Advanced Video-based Surveillance Systems*, Edited by C.S.Regazzoni, G.Fabri and G.Vernazza, Hingham, MA., USA, pp. 97-107, 1998.
- [Remagnino et al.,2004] Remagnino, P., Shihab, A., Jones, G., *Distributed Intelligence for Multi-Camera Visual Surveillance*. Em *Pattern Recognition* 37, No 4, pp. 675-689, 2004.
- [RoboCup] *RoboCup Official Site* [Em Linha]. Disponível em <http://www.robocup.org/> [Consultado em 2/3/2005]
- [rUNSWift, 2004] *rUNSWift 2005* [Em Linha]. Disponível em <http://www.cse.unsw.edu.au/~robocup/> [Consultado em 2/3/2005]
- [rUNSWift_Rel, 2004] Wong, T., *The Sonny Legged Robot League COMP4911 Thesis B Report*. Wales, United Kingdom, 2004.
- [Rybsky et al., 2000] Rybski, P., Stoeter, A., Erickson, M., Gini, M., Hougen, D., Papanikolopoulos, N., *A Team of Robotic Agents for Surveillance*, Proceedings of the Fourth International Conference on Autonomous Agents, pp. 9-16, Barcelona, Spain, June 2000
- [Santos et al., 2002] Santos, F., Silva, V., Almeida, L., *Auto-localização em pequenos robôs móveis e autónomos: O caso do robô Bulldozer IV*, Actas do Encontro Científico do ROBÓTICA 2002-Festival Nacional de Robótica, publicadas na revista *Electrónica e Telecomunicações* 3(6), DETUA, Abril, 2002.
- [Sony, 2005] *From engineering dream to robotic pet: the AIBO story* [Em Linha]. Disponível em http://www.aibo-europe.com/downloads/AIBO_Story.pdf. [Consultado em 20/5/2005]
- [TCP, 2000] Comer, D., *Internetworking with TCP/IP. Principles, Protocols, and Architectures*, Prentice Hall, pp. 209-290, 2000.
- [Tekkotsu] *Tekkotsu: Homepage*[Em Linha]. Disponível em <http://www.cs.cmu.edu/~tekkotsu> [Consultado em 4/5/2005]
- [Thrun et al., 1999] Fox, D., Bugard, W., Thrun, S., *Active Markov Localization for Mobile Robots*, *Robotics and Autonomous Systems (RAS)*, 25, pp. 195-207, 1998.

[Toshtada, 2001] *Toshtada Doi - AIBO creator* [Em Linha]. Disponível em http://inventors.about.com/library/inventors/bl_Toshtada_Doi.htm, 2001 [Consultado em 20/6/2005]

Anexo A

Informações técnicas gerais

Este anexo contém dados técnicos que foram necessários no desenvolvimento deste projecto. Contém a lista do hardware que foi utilizado, todo o procedimento para a instalação num PC das ferramentas que foram necessárias para que fosse possível desenvolver o software de vigilância. Para além disso é apresentada uma cópia do ficheiro wlandt.txt que é colocado no memory stick e que contém as definições necessárias para que o robô consiga comunicar com um PC.

A.1 Lista de Hardware utilizado

O hardware utilizado neste projecto foi:

- Um PC com o Sistema Operativo Linux ligado à rede com fios
- Um leitor/escritor de cartões de memória
- Access Point ligado à rede com fios do PC
- Robô ERS-7 da Sony
- Memory Stick de 16Mb

A.2 Instalação de Software

Há que ter dois factores em conta: que o sistema operativo em que o projecto foi desenvolvido foi Linux e que o ambiente de desenvolvimento utilizado foi o OPEN-R. Esta descrição só é portanto válida para a instalação do OPEN-R, e outras ferramentas que foram necessárias, em Linux.

A.2.1 OPEN-R SDK

Os **ficheiros necessários** estão disponíveis a partir da página <http://openr.aibo.com/> (sendo necessário estar registado no site).

Os ficheiros necessários são:

- OPEN_R_SDK-1.1.5-r3.tar.gz
- gcc-3.3.2.tar.gz
- binutils-2.14.tar.gz
- newlib-1.10.0.tar.gz
- build-devtools-3.3.2-r1.sh

Modo de instalação

1. Colocar os ficheiros descarregados em /usr/local
2. Para desempacotar os ficheiros gcc-3.3.2.tar.gz, binutils-2.14.tar.gz e newlib-1.10.0.tar.gz executar os comandos na consola:

```
cd /usr/local
```

```
./ build-devtools-3.3.2-r1.sh
```

3. Para finalmente desempacotar o OPEN-R SDK executar os comandos na consola (partindo do principio que se está em /usr/local)

```
tar zxvf OPEN_R_SDK-1.1.5-r3.tar.gz
```

No final destes passos o OPEN-R SDK está instalado.

A.2.2 Java 2 Platform, Standard Edition

Basta a instalação do J2SE 1.4.2_08 SDK que está disponível em <http://java.sun.com/j2se/1.4.2/download.html>. As instruções de instalação podem ser encontradas na página: <http://java.sun.com/j2se/1.4.2/install.html>.

A.3 Configuração do acesso à rede sem fios

Para que o ERS-7 consiga comunicar com um PC através de uma rede sem fios, necessita de um ficheiro que configura o acesso a essa rede. Esse ficheiro é denominado wlanflt.txt or wlanconf.txt e é colocado no memory stick em /OPEN-R/SYSTEM/CONF. Em seguida é apresentado o wlanconf.txt utilizado para configurar a comunicação do ERS-7 pela rede sem fios.

wlanconf.txt

```
# WLAN
#
HOSTNAME=AIBO1
ESSID=AIBONET
WEPENABLE=1
WEPKEY=SUPER
APMODE=1
#
# IP network
#
#USE_DHCP=1
#
# If DHCP is not used (USE_DHCP=0), you need to specify IP
# network configuration.
#
USE_DHCP=0
#IP atribuído ao ERS-7
ETHER_IP=192.168.102.232
ETHER_NETMASK=255.255.255.0
#IP do Access Point com que o ERS-7 irá comunicar
IP_GATEWAY=192.168.102.28
DNS_SERVER_1=193.136.28.138
#
# SSDP
#
SSDP_ENABLE=1
```

Anexo B

Especificações externas do ERS-7

Ao longo do projecto tiveram de ser tidas em conta as especificações e limitações externas do ERS-7, que são agora a apresentadas. Todos os esquemas utilizados neste anexo foram retirados do documento, produzido pela Sony, [Model_Information, 2004].

B.1 Medidas externas

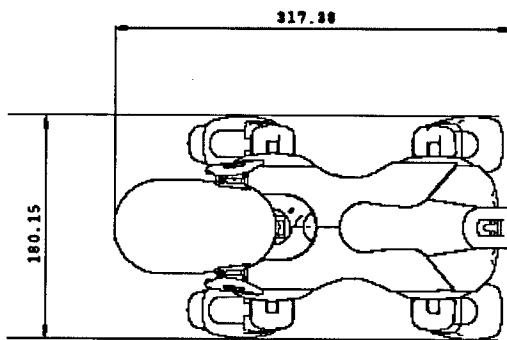


Figura 39 Vista de cima do ERS-7

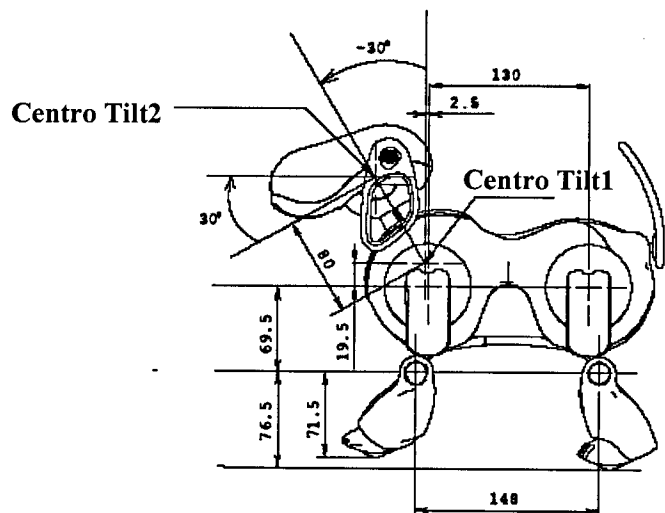


Figura 40 Vista de lado do ERS-7

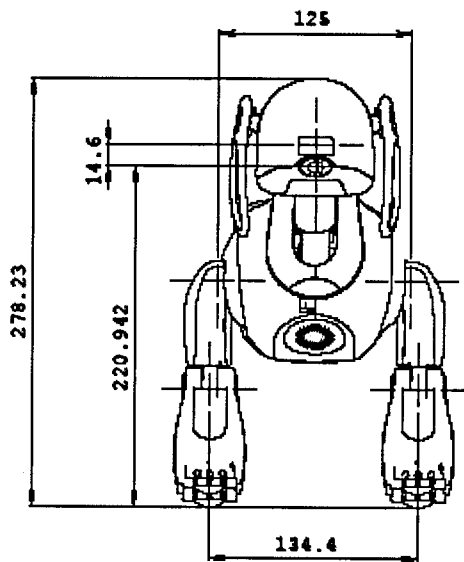


Figura 42 Vista de frente do ERS-7

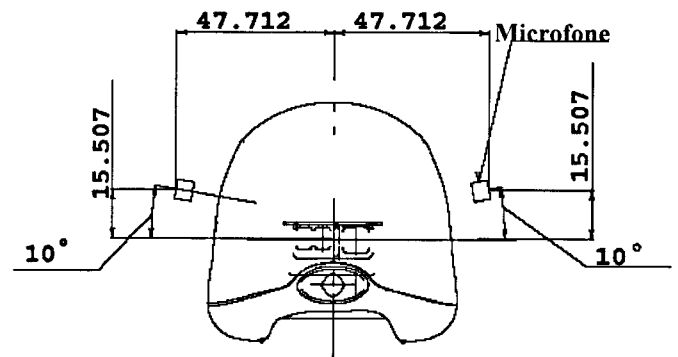


Figura 41 Vista de frente da cabeça do ERS-7

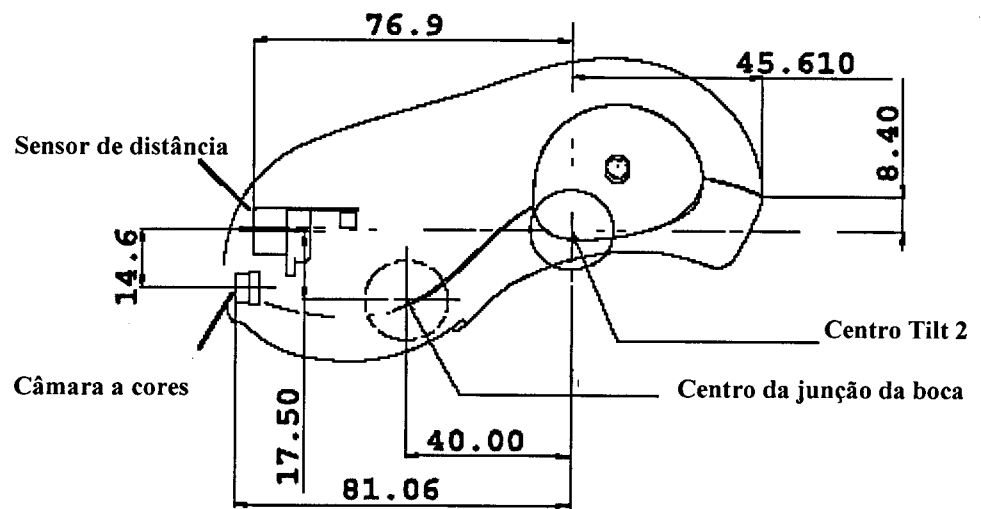


Figura 43 Vista de lado da cabeça do ERS-7

Unidade: milímetros

B.2 Limitações dos movimentos das juntas a nível de Hardware

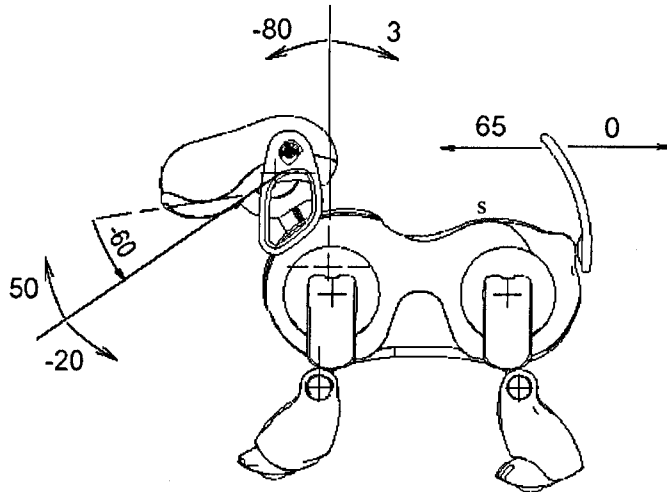


Figura 44 Limitações a nível do movimento da boca, Tilt 1, Tilt 2 e cauda

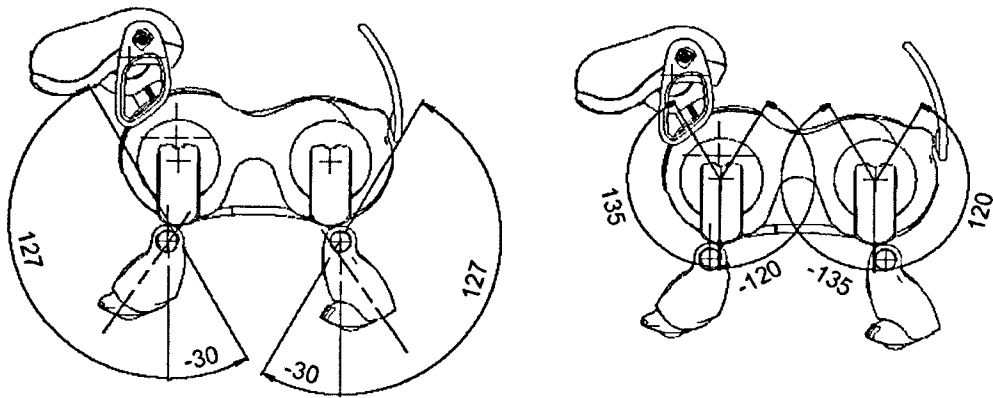


Figura 45 Limitações a nível do movimento das patas

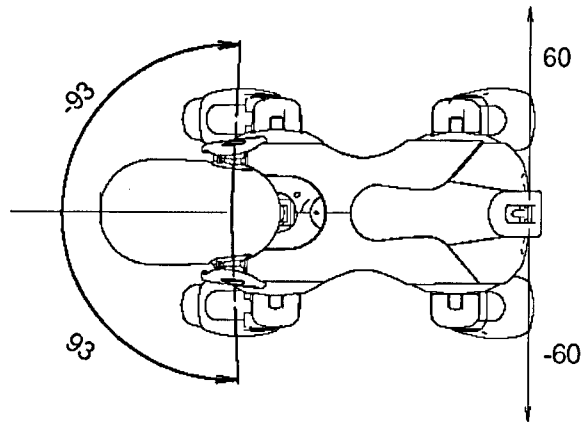


Figura 46 Limitações a nível do movimento da cabeça em relação ao corpo e cauda

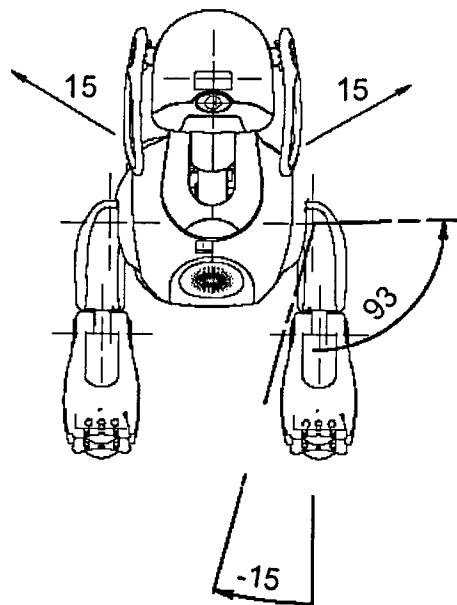


Figura 47 Limitações a nível do movimento das orelhas e da parte inferior das patas

Unidade: graus

B.3 Limitações dos movimentos das juntas a nível de Software

A nível anatómico há que ter em contas as limitações das juntas que são agora apresentadas, sob a forma de tabelas e que são diferentes das limitações reais apresentadas no ponto anteriores.

Parte	Graus
Pescoço tilt1	(-75, 0)
Pescoço pan	(-88, 88)
Pescoço tilt2	(-15, 45)
Boca	(-55, -3)
Perna esquerda J1	(-115, 130)
Perna esquerda J2	(-10, 88)
Perna esquerda J3	(-25, 122)
Perna direita J1	(-130, 115)
Perna direita J2	(-10, 88)
Perna direita J3	(-25, 122)
Cauda tilt	(5, 60)
Cauda pan	(-45, 45)

Tabela 14: Limitações das Junções únicas

Perna da frente		Perna de trás	
J1 (graus)	J2 (graus)	J1 (graus)	J2 (graus)
(-115, -76)	(-1, 88)	(115, 71)	(0, 88)
(-75, -54)	(-2, 88)	(70, 61)	(-1, 88)
(-55, -46)	(-3, 88)	(60, 51)	(-2, 88)
(-45, -36)	(-4, 88)	(50, 46)	(-3, 88)

(-35, -31)	(-5, 88)	(45, 41)	(-4, 88)
(-30, -26)	(-6, 88)	(40, 36)	(-5, 88)
(-25, -21)	(-7, 88)	(35, 31)	(-6, 88)
(-20, -16)	(-8, 88)	(30, 26)	(-7, 88)
(-15, -11)	(-9, 88)	(25, 21)	(-8, 88)
(-10, 20)	(-10, 88)	(20, 16)	(-9, 88)
(21, 25)	(-9, 88)	(15, -10)	(-10, 88)
(26, 30)	(-8, 88)	(-11, -15)	(-9, 88)
(29, 40)	(-7, 88)	(-16, -20)	(-8, 88)
(41, 45)	(-6, 88)	(-21, -25)	(-7, 88)
(46, 50)	(-5, 88)	(-26, -30)	(-6, 88)
(51, 55)	(-4, 88)	(-31, -35)	(-5, 88)
(56, 65)	(-3, 88)	(-36, -40)	(-4, 88)
(64, 130)	(-2, 88)	(-41, -45)	(-3, 88)

Tabela 15: Relação entre as duas junções (J1 e J2) das pernas da frente e de trás

Boca	trh1	trh2
(-30, -3)	(-75, -39)	(5, 45)
(-30, -3)	(-40, 0)	(-5, 45)

Tabela 16: Relação entre o movimento da boca e das junções do pescoço

Outros protótipos foram então desenvolvidos, cada vez mais se aproximando da forma do primeiro AIBO que foi lançado.

É de salientar que o software de todos os protótipos e modelos foram desenvolvidos para a OPEN-R (que será descrita com mais detalhe numa próxima sessão deste capítulo).

Anexo C

Aibo, nota histórica

As primeiras raízes do AIBO remontam ao início dos anos 90, em que enormes avanços eram feitos em termos de processadores, inteligência artificial, reconhecimento de voz e tecnologias visuais. Foi neste contexto que o Dr. Toshitada Doi, responsável pelo Laboratório de Criaturas Digitais da Sony, teve a ideia de juntar todas estas tecnologias em expansão de forma a criar um robô autónomo. A Sony proponha-se a criar não só um robô funcional mas um verdadeiro companheiro dos humanos.

Figura 49 Protótipo

Durante as primeiras fases de desenvolvimento deste projecto em 1992, os engenheiros responsáveis depararam-se então com desafios nunca antes enfrentados em robótica, nomeadamente permitir que o robô se deslocasse sobre patas, conter um programa de Inteligência Artificial que lhe possibilitasse a interacção com vários sensores (de toque, de visão e de som) e que não fosse necessário reprogramar para cada tarefa que fosse realizar.

Em 1997 surgiu o primeiro protótipo, fruto de extensa pesquisa e trabalho de desenvolvimento, conseguindo-se atingir o objectivo de locomoção sobre patas, ainda que nesta altura sobre 6 patas.



Figura 48 Primeiro protótipo

Outros protótipos foram então desenvolvidos, cada vez mais se aproximando da forma do primeiro AIBO que foi lançado.

É de referir que o software de todos os protótipos e modelos foram desenvolvidos com a interface OPEN-R (que será descrita com mais detalhe num ponto posterior deste capítulo).



Figura 49 Protótipo

Em Maio de 1999 foi lançado o primeiro modelo, o ERS-110, no Japão e nos Estados Unidos da América. Dos 5,000 exemplares postos à venda, os 3,000 no Japão esgotaram em 20 minutos e os 2,000 dos Estados Unidos da América em 4 dias.



Figura 50 Modelo ERS-110



Figura 51 Modelo ERS-210



Figura 52 Modelos ERS-311 e ERS-312 (da esquerda para a direita)



Figura 53 Modelo ERS-220

A este modelo seguiram-se os apresentados nas figuras 51, 52 e 53, que de modelo para modelo foram apresentando sempre os novos avanços dos Laboratórios da Sony.

No ERS-210, lançado em Outubro de 2000, a câmara CCD do ERS-110 foi substituída por um sensor de imagem CMOS, verificaram-se melhorias na expressão das emoções em reacção ao ambiente e na mobilidade. Foi introduzida a capacidade de reconhecimento de voz.

Os ERS-311 e ERS-312 começaram a ser comercializados em Setembro de 2001. Concebidos sobretudo para ser adoráveis, são capazes de emitir sons amistosos e realizar movimentos para expressar as suas emoções.

Em Novembro de 2001 seguiu-se o ERS-220 que com o seu ar futurista, incorpora uma série de sensores novos em relação aos seus antecessores permitindo uma série de novas acções de alto nível.

O ERS-7 foi lançado em 2003 e deixou para trás de si quatro modelos base e dois protótipos. Trouxe melhorias nas áreas de reconhecimento de voz (sendo-lhe possível reconhecer até 180 comandos de voz diferentes) e de reconhecimento visual de padrões, permitindo-lhe reconhecer diferentes comandos que lhe são apresentados sob a forma de um padrão visual impresso numa carta e identificar a sua estação de carregamento, dirigir-se para ela e colocar-se sozinho em posição de carregamento.



Figura 54 Robô ERS-7 da Sony

log4j.jar	https://log4j.dev.java.net/	Os ficheiros jar devem ser copiados para o path do java/bin
java.jar	http://math.nist.gov/javanumerics/java/	
C4.5	http://www.nuqueest.com/Personal/	Ter ficheiro ReadMe que vem dentro do arquivo.jar para detalhes de instalação.

Tabela 17: Arquivos necessários para poder iniciar a calibração de cores

Anexo D

Calibração da cor

Para que o módulo de visão consiga identificar convenientemente os objectos é necessária que a tabela look-up que faz a correspondência entre os valores YUV e uma dada cor. O processo de calibração permite obter essa tabela look-up, que é tão importante para que o módulo de visão funcione correctamente. Neste anexo é descrito todo o processo de obtenção dessa tabela, desde a preparação do computador, instalando os ficheiros necessários, passando pelos passos da calibração em si e terminando na calibração manual de cores.

D.1 Preparação

Para a calibração de cores será necessário preparar o computador para tal, sendo necessários ter instalados alguns ficheiros. A tabela 17 especifica quais os arquivos que são necessários instalar, onde podem ser obtidos e como devem ser instalados.

Nome do arquivo	Onde está disponível	Instalação
jogl.jar	https://jogl.dev.java.net/	Os ficheiros .jar devem ser copiados para o path do java/bin
jama.jar	http://math.nist.gov/javanumerics/jama/	
C4.5	http://www.rulequest.com/Personal/	Ler ficheiro ReadMe que vem dentro do arquivo .tar para detalhes de instalação.

Tabela 17: Arquivos necessários para poder iniciar a calibração de cores

D.2 Como calibrar

A calibração de cores passa por diversos passos que são agora enunciados.

Recolha de fotografias

É necessário obter fotografias do que o ERS-7 está realmente a ver, se modo a sobre elas classificar as cores. Este processo de recolha é feito quando o software está a ser executado pelo ERS-7, tendo este de estar configurado para comunicar via rede sem fios, com o computador que vai recolher as fotografias.

O primeiro passo para a recolha de fotografias é executar a ferramenta RoboCommander, disponibilizada pelo código rUNSWift, e esperando até que todas as mensagens relativas ao RoboCommander na consola parem executar o cliente noutra consola. Estando a aplicação RoboCommander a comunicar com o ERS-7, carregar no botão YUVPLANE, tirando assim uma fotografia ao que o ERS-7 está a ver. Gravar a fotografia com a extensão .BFL. Criar uma pasta onde são colocadas todas as imagens tiradas, tendo-se o cuidado de verificar as permissões dessa pasta.

Etiquetar fotografias

A etiquetagem de fotografias vai ser feita na ferramenta IC, localizada em /rUNSWift/base7/colour/ic.

Estando o IC a correr ir abrindo as imagens consecutivamente. Para cada uma das imagens:

- notar se há muito ruído, seleccionando uma cor e vendo se na imagem essa cor está bem atribuída. Se houver muito ruído carregar no botão CLEAR da aplicação para limpar a imagem.
- Para cada cor possível (explícitas na parte inferior da aplicação), com a ferramenta BRUSH, ir pintando da imagem aquilo que for da cor seleccionada. É de notar que apenas se deve pintar aquilo que se tem a certeza e nunca os bordos dos objectos.
- Faz-se isto para todas as cores e para todas as imagens, indo sempre gravando as imagens.

No final da classificação de todas as imagens, copiar a pasta que contém as imagens para /base7/colour/classify.

Geração da tabela de look-up

Esta parte do processo de calibração de cores pressupõe que o algoritmo de aprendizagem C4.5 esteja instalado correctamente.

No local onde foi colocada a cópia da pasta com as imagens classificadas, correr o `./dtcalibrate.sh <nome da pasta com as imagens>`.

Isto criará uma pasta com diversos ficheiros, com o nome da pasta que contém as imagens numa pasta chamada files.

D.3 Classificação Manual

Muitas vezes a calibração, nesta altura do processo ainda não está perfeita, pelo que precisa passar por um processo manual, mais perfeccionista, mas também mais moroso que o processo anterior que é o processo manual de classificação.

- Abrir o mc.java à semelhança do ic e do RoboWirelessBase.
- Abrir o ficheiro de calibração e cada uma das imagens
- Neste método de os ajustes são feitos pixel a pixel. É preciso ter cuidado para não estragar a calibração anterior, modificando apenas os pixels mais importantes.

D.4 Ficheiro obtido

Na pasta `/base7/colour/classify/files/<Nome da pasta com as Imagens>`, estará um ficheiro com a extensão `.cal`, que deve ser renomeado para `nmca.cal` e copiado para a pasta `/robot7/cfg` para que seja sempre copiado para a raiz do memory stick sempre que se fizer `make prep` em `/robot7`.

O ficheiro `nmca.cal` será a tabela de *look-up* que queríamos obter.

Anexo E

Vigilante.cc

```
#include <math.h>
#include "kim.h"
#include "Behaviours.h"
//#include "gps.h"

#ifdef COMPILE_ALL_CPP
#include "UNSW2004.h"
#endif //COMPILE_ALL_CPP

#include "../share/Common.h"
#include "../share/WalkLearningParams.h"

#ifndef OFFLINE
#include <OPENR/OPENRAPI.h>
#include "../vision/Vision.h"
#endif //OFFLINE

#include <iostream>
#include <sstream>
#include <iomanip>
#include "../vision/InfoManager.h"
#include "../vision/VisualCortex.h"

#define OUT(x) if (Behaviours::bWRITE_TO_CONSOLE) cout << x
const bool bDebug = false;
const bool bDebugCommand = false;

using namespace Behaviours;
using namespace std;

//////////////////////////////////VARIABLES//////////////////////////////////
double cabeca= 0.0; //graus que a cabe?a faz com o corpo

double isto[4]; //array que guarda aas cooredenadas do ERS-7
/*****
VARIÁVEIS DA MAQUINA DE ESTADOS
*****/
int estado=0; //ESTADO INICIAL - NAVEGA
int estado_anterior=54;
int contador_localizacao=0; //conta ? quanto tempo o ERS-7 est? perdido
int objectivo=2; //inicialmente o objectivo ? chegar ao poste 2 (amarelo_rosa)
int viu_poste=0;
```

```

int faz_localizacao=0;
double graus_virar=0; //total de graus que o robo ter? de virar no estado 99

//variaveis p o estado 99 ser capaz de colocar o robo em posi?ao
double inicio_virar=0;
double actualizacao_virar=0;

int contador_100=0;
int contador_12=0;
int primeira_vez_12=1; //1-> ? a 1? vez / 0->n ? a 1? vez
double distancia=0; //distancia do robo ao poste
int primeira_vez_50=1; //1-> ? a 1? vez / 0->n ? a 1? vez
int contador_60=0;
int contador_51=0;
int primeira_vez_61=1;
int obstaculo=0; //se =0 se n esta a contornar nenhum obstaculo

int cabeca_obstaculo=0;
int obstaculo_var=0; /esquerda
////////////////////////////////////

ostream & operator<<(ostream &out, const Vector &v) {
    out << "[" << v.x << "," << v.y << "]";
    return out;
}

namespace Kim {
    int actionCounter = 0;

    void Vai_Virar();

    //////////////////////////////////
    //ESTADOS////////////////////////////////
    //////////////////////////////////
    bool MaquinaEstados();
    bool Estado_Localizacao_Global();

    bool Estado_Posicionamento(); //robo posiciona-se em rela??o ao poste p o
qual se vai dirigir -99
    bool Estado_Detectar_Movimento(); //robo p?ra para procurar activamente
por intruso - 100

    bool Estado_Navegar();

    //estados referentes ao contornar obstaculo
    bool Estado_60(); //vira na direc??o contr?ria ao obst?culo
    bool Estado_61(); // anda em frente at? ultrapassar o obst?culo tendo em
aten??o outros obstaculos

    //////////////////////////////////

    void moveHeadAround(bool reset,int direction,int higherTily,int lowerTily,int
leftPanx,int rightPanx);

    void moveHeadTo(int Tily,int Panx);

    void initKimForward() {
        cout << "Initializing ERS-7 Surveillance System..." << endl;
    }

    static const int ON = 2;
    static const int OFF = 1;
    static const int HETERESIS = 25;

```

```

static const int ALPHA = 60;
static const int BETA = 20;
static const int CLOCKWISE = 1;
static const int COUNTERCW = -1;
static const int OUTMOST = 40;

static const int MAX_SCANNING_INTRUDER = 90; //# of vision frames a
30frames/s
static const int MAX_TIME_WALK = 20; //# of vision frames a 30frames/s
static const int MAX_TIME_LOOK = 30; //# of vision frames a 30frames/s
double MAX_TIME_KEEP_TRACK = distancia/4; //# of vision frames a 30frames/s
double MAX_TIME_LOST = 45; //# 1.5 segundos p ?rea de 2m por 2m

void doKimForward()
{
    int chestIR = sensors->sensorVal[ssCHEST_INFRARED];
    int headnearIR = sensors->sensorVal[ssINFRARED_NEAR];
    actionCounter ++;
    static int count = 0;

    if(estado==0 && contador_localizacao>=MAX_TIME_LOST)
    {
        estado=10; //estado q faz uso da localizacao global
        contador_localizacao=0;
    }

    gps->ESTADO=estado;
    cout<<"ESTADO ACTUAL COMPORTAMENTO:"<<estado<<"\n"<<endl;

    MaquinaEstados();

    //////////////////////////////////////CORDENADAS////////////////////////////////////
        double angulo=0;

        isto[0]=gps->GPSCoordenada_X();
        isto[1]=gps->GPSCoordenada_Y();
        isto[2]=gps->GPSCoordenada_Z();
        isto[3]=gps->GPSProbabilidade();

    //////////////////////////////////////
}

////////////////////////////////////ESTADOS ESTADOS ESTADOS ESTADOS ESTADOS ESTADOS////////////////////////////////////

bool MaquinaEstados() {
    static bool reset = false;
    int counter=0;
    if (reset) {
        reset = false;
        return true;
    }
    else {
        setDefaultParams();
    }
}

```



```

        setWalkParams();

        if (Estado_Localizacao_Global())
        {
            return true;
        }

        if (Estado_Posicionamento())
        {
            return true;
        }

        if (Estado_Detectar_Movimento())
        {
            return true;
        }
        if (Estado_Navegar())
        {
            return true;
        }

        //obstaculos
        if (Estado_60())
        {
            return true;
        }
        if (Estado_61())
        {
            return true;
        }
        reset = true;
        return false;
    }

}

//faz uso da localiza?ao global feita pelo modulo de localiza?ao
bool Estado_Localizacao_Global()
{
    if(estado!=10)
    {
        return false;
    }else{

        contador_localizacao=0;
        //POSTE AMARELO_rOSA
        if(objectivo==2)
        {
            //se vir o objetivo
            if(gps->canSee(vobYellowLeftBeacon))
            {

                cout <<"viu o poste objetivo\n"<<endl;

                estado=0;          //continua a navegar ou para p
                foto

                gps->ESTADO=estado;
                return false;
            }
        }
    }
}

```

```

//se vir qualquer um dos outros postes
if(gps->canSee(vobYellowRightBeacon)
|| gps->canSee(vobBlueRightBeacon)
|| gps->canSee(vobBlueLeftBeacon))
{

    if(gps->canSee(vobYellowRightBeacon)){viu_poste=5;}
    if(gps->canSee(vobBlueRightBeacon)){viu_poste=4;}
    if(gps->canSee(vobBlueLeftBeacon)){viu_poste=3;}

    //se a probabilidade ? aceit?vel e as coordenada

v?lidas

    //tentar ver se d? para baixar a probabilidade
    if(isto[3]>=80 &&
    isto[0]>=0 && isto[0]<FIELD_WIDTH &&
    isto[1]>=0 && isto[1]<FIELD_LENGTH)
    {
        Vai_Virar();
        faz_localizacao=1;
        estado=99;
        gps->ESTADO=estado;
        inicio_virar=gps->GPSVirou();
        return false;
    }

    walkType = NormalWalkWT;
    forward = 0;
    //fazer com q o lado para o qual vira seja mais

inteligente

    turnCCW = 40;
    moveHeadTo(20,0); //estava em 30
    return true;
}

}

//POSTE AZUL_rOSA
if(objectivo==3)
{
    //se vir o objetivo
    if(gps->canSee(vobBlueLeftBeacon))
    {

        cout <<"viu o poste objetivo\n"<<endl;
        estado=0; //continua a navegar ou para p

foto

        gps->ESTADO=estado;
        return false;
    }
    //se vir qualquer um dos outros postes
    if(gps->canSee(vobYellowRightBeacon)
|| gps->canSee(vobBlueRightBeacon)
|| gps->canSee(vobYellowLeftBeacon))
    {
        if(gps->canSee(vobYellowRightBeacon)){viu_poste=5;}
        if(gps->canSee(vobBlueRightBeacon)){viu_poste=4;}
        if(gps->canSee(vobYellowLeftBeacon)){viu_poste=2;}
    }
}

```

```

v?lidas                                     //se a probabilidade ? aceit?vel e as coordenada

//tentar ver se d? para baixar a probabilidade
if(isto[3]>=80 &&
isto[0]>=0 && isto[0]<FIELD_WIDTH &&
isto[1]>=0 && isto[1]<FIELD_LENGTH)
{
    Vai_Virar();
    faz_localizacao=1;
    estado=99;
    gps->ESTADO=estado;
    inicio_virar=gps->GPSVirou();
    return false;
}

walkType = NormalWalkWT;
forward = 0;
//fazer com q o lado para o qual vira seja mais

inteligente                                turnCCW = 40;
                                             moveHeadTo(20,0);
                                             return true;
}
}

//POSTE rOSA_AZUL
if(objectivo==4)
{
    //se vir o objetivo
    if(gps->canSee(vobBlueRightBeacon))
    {
        cout <<"viu o poste objetivo\n"<<endl;
        estado=0;          //continua a navegar ou para p

foto                                         gps->ESTADO=estado;
                                             return false;
    }
    //se vir qualquer um dos outros postes
    if(gps->canSee(vobYellowRightBeacon)
|| gps->canSee(vobBlueLeftBeacon)
|| gps->canSee(vobYellowLeftBeacon))
    {
        if(gps->canSee(vobYellowRightBeacon)) {viu_poste=5;}
        if(gps->canSee(vobBlueLeftBeacon)) {viu_poste=3;}
        if(gps->canSee(vobYellowLeftBeacon)) {viu_poste=2;}

v?lidas                                     //se a probabilidade ? aceit?vel e as coordenada

//tentar ver se d? para baixar a probabilidade
if(isto[3]>=80 &&
isto[0]>=0 && isto[0]<FIELD_WIDTH &&
isto[1]>=0 && isto[1]<FIELD_LENGTH)
{
    Vai_Virar();
    faz_localizacao=1;
    estado=99;
    gps->ESTADO=estado;
    inicio_virar=gps->GPSVirou();
    return false;
}

```

```

        walkType = NormalWalkWT;
        forward = 0;
        //fazer com q o lado para o qual vira seja mais
inteligente
        turnCCW = 40;
        moveHeadTo(20,0);
        return true;
    }

}

//POSTE ROSA_AMARELO
if(objectivo==5)
{
        //se vir o objetivo
        if(gps->canSee(vobYellowRightBeacon))
        {
                cout <<"viu o poste objetivo\n"<<endl;
                estado=0;          //continua a navegar ou para p
foto
                gps->ESTADO=estado;
                return false;
        }
        //se vir qualquer um dos outros postes
        if(gps->canSee(vobYellowRightBeacon)
        || gps->canSee(vobBlueLeftBeacon)
        || gps->canSee(vobYellowLeftBeacon))
        {
                if(gps->canSee(vobBlueRightBeacon)){viu_poste=4;}
                if(gps->canSee(vobBlueLeftBeacon)){viu_poste=3;}
                if(gps->canSee(vobYellowLeftBeacon)){viu_poste=2;}
v?lidas
                //se a probabilidade ? aceit?vel e as coordenada

                //tentar ver se d? para baixar a probabilidade
                if(isto[3]>=80 &&
                isto[0]>=0 && isto[0]<FIELD_WIDTH &&
                isto[1]>=0 && isto[1]<FIELD_LENGTH)
                {
                        Vai_Virar();
                        faz_localizacao=1;
                        estado=99;
                        gps->ESTADO=estado;
                        inicio_virar=gps->GPSVirou();
                        return false;
                }

                walkType = NormalWalkWT;
                forward = 0;
                //fazer com q o lado para o qual vira seja mais
inteligente
                turnCCW = 40;
                moveHeadTo(20,0);
                return true;
        }
}

```

```

    }

    walkType = NormalWalkWT;
    forward = 0;
    //fazer com q o lado para o qual vira seja mais
inteligente
    turnCCW = 40;
    moveHeadTo(20,0);
    return true;
}

}

//VIRAR PARA A POSICAO EM QUE TER? DE ANDAR
bool Estado_Posicionamento()
{

    if(estado!=99)
    {
        return false;
    }else{

        //cout<<"ESTADO 99\n"<<endl;

        actualizacao_virar=gps->GPSVirou();
        cout<<"Inicio VIRAR:"<<inicio_virar<<"\n"<<endl;
        cout<<"ACTUALIZACAO VIRAR:"<<actualizacao_virar<<"\n"<<endl;

        //esta no centro da area a posicionar-se para o proximo
objectivo
        if (estado_anterior==54 && obstaculo==0 && faz_localizacao==0 )
        {
            if((actualizacao_virar-inicio_virar) >= graus_virar ||
(gps->canSee(vobYellowLeftBeacon) && abs(isto[2]=gps->GPSCoordenada_Z())<=5))
            {
                //cout<<"ANGULO DO POSTE EM RELACAO ? CABE?A
_2"<<isto[2]<<"\n"<<endl;

                graus_virar=0;
                estado=100; /*estado correspondente ? procura
activa de intrusos*/

                gps->ESTADO=estado;
                inicio_virar=0;
                actualizacao_virar=0;
                return false;
            }
        }

        //esta a contornar um obstaculo
        if(obstaculo==1)
        {

            if (cabeca>=0 && (actualizacao_virar-inicio_virar) >= 60)
            {
                //turnCCW = 40;

```

```

        estado=61;
        gps->ESTADO=estado;
        atualizacao_virar=0;
        inicio_virar=0;
        return false;
    }

-60) else if (cabeca<0 && (atualizacao_virar-inicio_virar) <=
    {
        //turnCCW = -40;
        estado=61;
        gps->ESTADO=estado;
        atualizacao_virar=0;
        inicio_virar=0;
        return false;
    }

    if (cabeca>=0)
    {
        turnCCW = 40;
    }else if (cabeca<0)
    {
        turnCCW = -40;
    }
    walkType = NormalWalkWT;
    forward = 0;
    return true;
}

if(faz_localizacao==1)
{
    //cout<<"Graus
virar:"<<abs(graus_virar)<<graus_virar<<"\n"<<endl;

    if(graus_virar<=0)
    {

        cout<<"A VIRAR PARA LADO DIREITO\n"<<endl;

        walkType = NormalWalkWT;
        forward = 0;
        moveHeadTo(20,0);
        turnCCW = -40;

    }else{

        cout<<"A VIRAR PARA LADO ESQUERDO\n"<<endl;
        walkType = NormalWalkWT;
        forward = 0;
        moveHeadTo(20,0);
        turnCCW = 40;
    }

    if(atualizacao_virar<0 && inicio_virar>0)
    {

        if(inicio_virar-
abs(graus_virar)<=atualizacao_virar)

```

```

        {
            graus_virar=0;
            estado=0;
            gps->ESTADO=estado;
            inicio_virar=0;
            atualizacao_virar=0;
            faz_localizacao=0;
            return false;
        }

    }else if(atualizacao_virar>0 && inicio_virar<0)
    {

        if(inicio_virar+atualizacao_virar>=graus_virar)
        {
            graus_virar=0;
            estado=0;
            gps->ESTADO=estado;
            inicio_virar=0;
            atualizacao_virar=0;
            faz_localizacao=0;
            return false;
        }

    }else if((atualizacao_virar-inicio_virar) >=
abs(graus_virar)
            && graus_virar>=0)
    {
        graus_virar=0;
        estado=0;
        gps->ESTADO=estado;
        inicio_virar=0;
        atualizacao_virar=0;
        faz_localizacao=0;
        return false;
    }

    }else if((inicio_virar-atualizacao_virar) >=
abs(graus_virar)
            && graus_virar<0)
    {
        graus_virar=0;
        estado=0;
        gps->ESTADO=estado;
        inicio_virar=0;
        atualizacao_virar=0;
        faz_localizacao=0;
        return false;
    }

    }
    return true;
}

turnCCW = 40;
walkType = NormalWalkWT;
forward = 0;
moveHeadTo(20,0);
return true;
}
}

```

```

//PARA PARA DETECTAR ACTIVAMENTE INTRUSOS
bool Estado_Detectar_Movimento()
{
    if(estado!=100)
    {
        return false;
    }else{

        //cout<<"ESTADO 100\n"<<endl;

        contador_100 += 1;

        if (contador_100 > MAX_SCANNING_INTRUDER)
        {
            //cout << "Acaba a detec??o activa de intrusos" << endl;
            contador_100 = 0;
            estado=0;
            gps->ESTADO=estado;
            return false;
        }else{

            walkType = NormalWalkWT;
            forward = 0;
            turnCCW = 0;
            moveHeadTo(30,0);
            return true;

        }

    }

}

//ANDAR AT? AO POSTE//
bool Estado_Navegar()
{

    int chestIR = sensors->sensorVal[ssCHEST_INFRARED];
    int headnearIR = sensors->sensorVal[ssINFRARED_NEAR];

    if(estado!=0)
    {
        return false;
    }else{

        if(primeira_vez_12==1)
        {
            primeira_vez_12=0;
        }

        //cout<<"ESTADO 12\n"<<endl;

        //se dectar obstaculo enquanto anda
        if ( chestIR <= 160000 || (headnearIR > 50000 && headnearIR <=
290000))
        {

```



```

        if (cabeca>=0)
        {
            cabeca_obstaculo=1; //obstaculo ? esquerda
            moveHeadAround(false, CLOCKWISE, 20, 20, 45, 45);
            //moveHeadTo(20,45);
        }

        if (cabeca<0)
        {
            cabeca_obstaculo=-1; //obstaculo ? direita
            moveHeadAround(false, CLOCKWISE, 20, 20, -45, -45);
            //moveHeadTo(20,-45);
        }

        primeira_vez_12=1;
        estado=60; //estado inicio do contorno do obstaculo
detectado

        gps->ESTADO=estado;
        return false;
    }

    //poste em linha de vista (amarelo_rosa)
    if (gps->canSee(vobYellowLeftBeacon) && objetivo==2 )
    {
        contador_12=0;
        contador_localizacao=0;
        distancia=gps->GPSDistancia();
        //cout<<"DISTANCIA:"<<distancia<<"\n"<<endl;
        //cout<<"DISTANCIA/4:"<<(distancia/4)<<"\n"<<endl;

        //se ja estiver dentro da ?rea do ponto de controle
        if(distancia<=50)
        {
            primeira_vez_12=1;
            estado_anterior=42;

            objetivo=3;

            estado=10;
            gps->ESTADO=estado;
            return false;
        }
        //se n?o continua a andar
        walkType = NormalWalkWT;
        forward = 4;
        turnCCW = 0;//turnCCW = compensar_rota;
        left=0;
        moveHeadAround(false, CLOCKWISE, ALPHA, BETA, OUTMOST, -
OUTMOST);
        return true;

    }else if(!gps->canSee(vobYellowLeftBeacon) && objetivo==2
){contador_localizacao++;}

    //poste em linha de vista (azul_rosa)
    if (gps->canSee(vobBlueLeftBeacon) && objetivo==3 )
    {
        contador_12=0;

```

```

        contador_localizacao=0;
        distancia=gps->GPSDistancia();
        //cout<<"DISTANCIA:"<<distancia<<"\n"<<endl;
        //cout<<"DISTANCIA/4:"<<(distancia/4)<<"\n"<<endl;

        //se ja estiver dentro da ?rea do ponto de controle
        if(distancia<=50)
        {
            primeira_vez_12=1;
            estado_anterior=23;

            objectivo=5;

            estado=10;
            gps->ESTADO=estado;
            return false;
        }
        //se n?o continua a andar
        walkType = NormalWalkWT;
        forward = 4;
        turnCCW = 0;//turnCCW = compensar_rota;
        left=0;
        moveHeadAround(false, CLOCKWISE, ALPHA, BETA, OUTMOST, -
OUTMOST);

        return true;

    }else if(!gps->canSee(vobBlueLeftBeacon) && objectivo==3
){contador_localizacao++;}

//poste em linha de vista (rosa azul)
if (gps->canSee(vobBlueRightBeacon) && objectivo==4 )
{
    contador_12=0;
    contador_localizacao=0;
    distancia=gps->GPSDistancia();
    //cout<<"DISTANCIA:"<<distancia<<"\n"<<endl;
    //cout<<"DISTANCIA/4:"<<(distancia/4)<<"\n"<<endl;

    //se ja estiver dentro da ?rea do ponto de controle
    if(distancia<=50)
    {
        primeira_vez_12=1;
        estado_anterior=54;

        objectivo=2;

        estado=10;
        gps->ESTADO=estado;
        return false;
    }
    //se n?o continua a andar
    walkType = NormalWalkWT;
    forward = 4;
    turnCCW = 0;//turnCCW = compensar_rota;
    left=0;
    moveHeadAround(false, CLOCKWISE, ALPHA, BETA, OUTMOST, -
OUTMOST);

    return true;
}

```

```

    }else if(!gps->canSee(vobBlueRightBeacon) && objetivo==4
)(contador_localizacao++;)

        //poste em linha de vista (rosa_azul)
    if (gps->canSee(vobYellowRightBeacon) && objetivo==5 )
    {
        contador_i2=0;
        contador_localizacao=0;
        distancia=gps->GPSDistancia();
        //cout<<"DISTANCIA:"<<distancia<<"\n"<<endl;
        //cout<<"DISTANCIA/4:"<<(distancia/4)<<"\n"<<endl;

        //se ja estiver dentro da ?rea do ponto de controle
        if(distancia<=50)
        {
            primeira_vez_i2=1;
            estado_anterior=35;

            objetivo=4;

            estado=10;
            gps->ESTADO=estado;
            return false;
        }
        //se n?o continua a andar
        walkType = NormalWalkWT;
        forward = 4;
        turnCCW = 0;//turnCCW = compensar_rota;
        left=0;
        MoveHeadAround(false, CLOCKWISE, ALPHA, BETA, OUTMOOT, -
OUTMOOT);
        return true;

    }else if(!gps->canSee(vobYellowRightBeacon) && objetivo==5
)(contador_localizacao++;)

        contador_i2++;

        walkType = NormalWalkWT;
        forward = 4;
        turnCCW = 0;
        left=0;
        MoveHeadAround(false, CLOCKWISE, ALPHA, BETA, OUTMOOT, -OUTMOOT);
        return true;
    }

//ESTADOS QUE PERMITEM CONTORNAR UM OBSTACULO
bool Estado_60() //vira at? ter o obstaculo do seu lado
{
    int chestIR = sensors->sensorVal[ssCHEST_IRFAPED];
    int headnearIR = sensors->sensorVal[ssHEADNEAR_IRFAPED];

    if(estado!=60)
    {
        return false;
    }else{

```

```

        contador_60++;

        if(primeira_vez_50==1)
        {
            primeira_vez_50=0;
        }

        //vira at? que detecte o caminho livre em frente
        if (chestIR <= 160000 || ( headnearIR > 50000 && headnearIR <=
290000))
        {

            walkType = NormalWalkWT;
            forward = 0;
            left = 0;

            //cout<<"LADO DA CABE?A:"<<cabeca<<"\n"<<endl;

            if (cabeca>=0)
            {
                moveHeadAround(false, CLOCKWISE, 20, 20, 20, 20);
                //moveHeadTo(20,45);
                turnCCW = -40;
            }
            else if (cabeca<0)
            {
                moveHeadAround(false, CLOCKWISE, 20, 20, -20, -20);
                //moveHeadTo(20,-45);
                turnCCW = 40;
            }

            return true;
        }else{

            if(contador_60<=10) //15//n?o verificado o novo valor
            {
                contador_60=0;
                primeira_vez_50=1;
                estado=0;
                gps->ESTADO=estado;
                return false;
            }

            primeira_vez_50=1;
            contador_60=0;
            estado=61;
            gps->ESTADO=estado;
            return false;
        }

    }

}

bool Estado_61()
{

```

```

int chestIR = sensors->sensorVal[ssCHEST_INFRARED];
int headnearIR = sensors->sensorVal[ssINFRARED_NEAR];

turnCCW = 0;

if(estado!=61)
{
    return false;
}
else{
    turnCCW = 0;
    contador_51++;

    if(contador_51<=MAX_TIME_LOOK)
    {

        if(chestIR <= 160000)
        {

            estado=60;
            gps->ESTADO=estado;
            contador_51=0;
            moveHeadTo(20,0);
            return false;
        }

        //vira a cabe?a para o lado do obstaculo
        if(cabeca>=0)
        {
            //cout<<"cabeca maior que zero\n"<<endl;
            moveHeadAround(true, CLOCKWISE, 20, 20, 90, 90);
            if ( headnearIR > 50000 && headnearIR <= 300000)
            {
                if(headnearIR < 150000)
                {
                    turnCCW = -20;
                }
            }
            //cout<<"contador zero\n"<<endl;
            contador_51=0;
        }
        }else if(cabeca<0)
        {
            //cout<<"cabeca menor que zero\n"<<endl;
            moveHeadAround(true, CLOCKWISE, 20, 20, -90, -90);
            if ( headnearIR > 50000 && headnearIR <= 300000)
            {
                if(headnearIR < 150000)
                {
                    turnCCW = 20;
                }
            }
            contador_51=0;
        }
    }

    walkType = NormalWalkWT;
    forward = 3;
    left=0;
    return true;
}
else{

    contador_51=0;

    if(primeira_vez_61==1)

```

```

        {
            obstaculo=1;
            inicio_virar=gps->GPSVirou();
            cout<<"inicio da viragem:"<<inicio_virar<<"\n"<<endl;
            estado=99;
            gps->ESTADO=estado;
            primeira_vez_61=0;
            return false;
        }
        obstaculo=0;
        cout<<"vai para o estado de localiza?ao\n"<<endl;
        estado=10;
        gps->ESTADO=estado;
        primeira_vez_61=1;
        return false;
    }
}

}

//////////FIM FIM FIM FIM ESTADOS FIM FIM FIM //////////

//////////FUNÇÕES AUXILIARES////////////////////////////////////

void Vai_Virar()
{
    double x=0, y=0, angulo=0;
    double a1=0, a2=0, a3=0, a4=0;

    x=isto[0];
    y=isto[1];
    angulo=isto[2];

    cout<<"X"<<x<<"\n"<<endl;
    cout<<"Y"<<y<<"\n"<<endl;

    a1=RAD2DEG(atan((FIELD_LENGTH-y)/(FIELD_WIDTH-x))); //angulo ao poste 2
    a2=RAD2DEG(atan((y)/(FIELD_WIDTH-x))); //angulo ao poste 3
    a3=RAD2DEG(atan((x)/(y))); //angulo ao poste 4
    a4=RAD2DEG(atan((FIELD_LENGTH-y)/(x))); //angulo ao poste 5

    cout<<"Angulo 1"<<a1<<endl;
    cout<<"Angulo 2"<<a2<<endl;
    cout<<"Angulo 3"<<a3<<endl;
    cout<<"Angulo 4"<<a4<<endl;
    cout<<"\n"<<endl;

    if(viu_poste==2)
    {
        cout<<"viu poste 2\n"<<endl;

        if(objectivo==3){
            //graus_virar=-(a1+a2);
            graus_virar=(90-a1)+90+90+(90-a2);

            //if(graus_virar>360){graus_virar=graus_virar*(int(graus_virar) % 360);}
            if(angulo<0){graus_virar=graus_virar+angulo;}
            if(angulo>0){graus_virar=graus_virar-angulo;}
        }
    }
}

```

```
        } //vira pela esqeerda (direita)

    if(objectivo==4){

        graus_virar=(90-a1)+90+(90-a3);

//if(graus_virar>360){graus_virar=graus_virar*(int(graus_virar) % 360);}
        if(angulo<0){graus_virar=graus_virar-angulo;}
        if(angulo>0){graus_virar=graus_virar+angulo;}

        } //vira pela esquerda
    if(objectivo==5){

        graus_virar=(90-a1)+(90-a4);

//if(graus_virar>360){graus_virar=graus_virar*(int(graus_virar) % 360);}
        if(angulo<0){graus_virar=graus_virar-angulo;}
        if(angulo>0){graus_virar=graus_virar+angulo;}

        } //vira pela esquerda
    }

if(viu_poste==3)
{
    cout<<"viu poste 3\n"<<endl;

    if(objectivo==2){

        graus_virar=a2+a1;

//if(graus_virar>360){graus_virar=graus_virar*(int(graus_virar) % 360);}
        if(angulo<0){graus_virar=graus_virar-angulo;}
        if(angulo>0){graus_virar=graus_virar+angulo;}

        } //vira pela esquerda
    if(objectivo==4){

        //graus_virar=-((90-a2)+a3);
        graus_virar=a2+90+90+(90-a3);

//if(graus_virar>360){graus_virar=graus_virar*(int(graus_virar) % 360);}
        if(angulo<0){graus_virar=graus_virar+angulo;}
        if(angulo>0){graus_virar=graus_virar-angulo;}

        } //vira pela esquerda (direita)

    if(objectivo==5){

        graus_virar=a2+90+(90-a4);

//if(graus_virar>360){graus_virar=graus_virar*(int(graus_virar) % 360);}
        if(angulo<0){graus_virar=graus_virar+angulo;}
        if(angulo>0){graus_virar=graus_virar-angulo;}

        } //vira pela esquerda
    }

if(viu_poste==4)
{

    cout<<"viu poste 4\n"<<endl;
```

```
if(objectivo==2){
    graus_virar=a3+90+a1;
//if(graus_virar>360){graus_virar=graus_virar*(int(graus_virar) % 360);}
    if(angulo<0){graus_virar=graus_virar-angulo;}
    if(angulo>0){graus_virar=graus_virar+angulo;}
} //vira pela esquerda

if(objectivo==3){
    graus_virar=a3+(90-a2);
//if(graus_virar>360){graus_virar=graus_virar*(int(graus_virar) % 360);}
    if(angulo<0){graus_virar=graus_virar-angulo;}
    if(angulo>0){graus_virar=graus_virar+angulo;}
} //vira pela esquerda

if(objectivo==5){
    //graus_virar--((90-a3)+a4);
    graus_virar=a3+90+90+(90-a4);
//if(graus_virar>360){graus_virar=graus_virar*(int(graus_virar) % 360);}
    if(angulo<0){graus_virar=graus_virar+angulo;}
    if(angulo>0){graus_virar=graus_virar-angulo;}
} //vira pela esquerda (direita)
}

if(viu_poste==5)
{
    cout<<"viu poste 5\n"<<endl;

    if(objectivo==2){
        //graus_virar--((90-a4)+(90-a1));
        graus_virar=a4+90+90+a1;
//if(graus_virar>360){graus_virar=graus_virar*(int(graus_virar) % 360);}
        if(angulo<0){graus_virar=graus_virar+angulo;}
        if(angulo>0){graus_virar=graus_virar-angulo;}
    } //vira pela esquerda (direita)

    if(objectivo==3){
        graus_virar=a4+90+(90-a2);
//if(graus_virar>360){graus_virar=graus_virar*(int(graus_virar) % 360);}
        if(angulo<0){graus_virar=graus_virar-angulo;}
        if(angulo>0){graus_virar=graus_virar+angulo;}
    } //vira pela esquerda

    if(objectivo==4){
        graus_virar=a4+(90-a3);
//if(graus_virar>360){graus_virar=graus_virar*(int(graus_virar) % 360);}
        if(angulo<0){graus_virar=graus_virar-angulo;}
        if(angulo>0){graus_virar=graus_virar+angulo;}
    } //vira pela esquerda
}
```



```

cout<<"GRAUS A VIRAR"<<graus_virar<<"\n"<<endl;
cout<<"ANGULO"<<angulo<<"\n"<<endl;
}

static const int UP = 0;
static const int LEFT = 1;
static const int DOWN = 2;
static const int RIGHT = 3;
static const int HEAD_SPEED = 7;

static const int countercwNext[4][2] = {
    {1,0}, {0, -1}, { - 1,0}, {0,1}
}; //4 direction corresponding to 4 sides of the rectangle
static const int clockwiseNext[4][2] = {
    { - 1,0}, {0,1}, {1,0}, {0, -1}
};

//moving around a box bounded by ( higherTily, lowerTily, leftPanx, rightPanx
) !!! leftPanx > 0 > rightPanx
void moveHeadAround(bool reset,
                    int direction,
                    int higherTily,
                    int lowerTily,
                    int leftPanx,
                    int rightPanx)
{
    static int curPanx, curTily;
    static int side = 0;
    double HeadPan = MICRO2DEG(sensors->sensorVal[ssHEAD_PAN]);

    if (reset) {
        curPanx = 0;
        curTily = higherTily;
        side = UP;
    }

    if (curPanx > leftPanx) {
        side = LEFT;
        curPanx = leftPanx;
    }
    else if (curPanx < rightPanx) {
        side = RIGHT;
        curPanx = rightPanx;
    }
    else if (curTily < lowerTily) {
        side = DOWN;
        curTily = lowerTily;
    }
    else if (curTily > higherTily) {
        side = UP;
        curTily = higherTily;
    }

    int offPanx, offTily;
    if (direction == CLOCKWISE) {
        offPanx = clockwiseNext[side][0] * HEAD_SPEED;
        offTily = clockwiseNext[side][1] * HEAD_SPEED;
    }
    else {
        offPanx = countercwNext[side][0] * HEAD_SPEED;

```

```
        offTily = countercwNext[side][1] * HEAD_SPEED;
    }

    curPanx = curPanx + offPanx;
    curTily = curTily + offTily;

    headtype = ABS_PT;
    panx = curPanx;
    cabeca= panx;
    tilty = curTily;
}

void moveHeadTo(int Tily,int Panx)
{
    headtype = ABS_PT;
    panx = Panx;
    cabeca= panx;
    tilty = Tily;
}
}
```





FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

BIBLIOTECA



0000105170