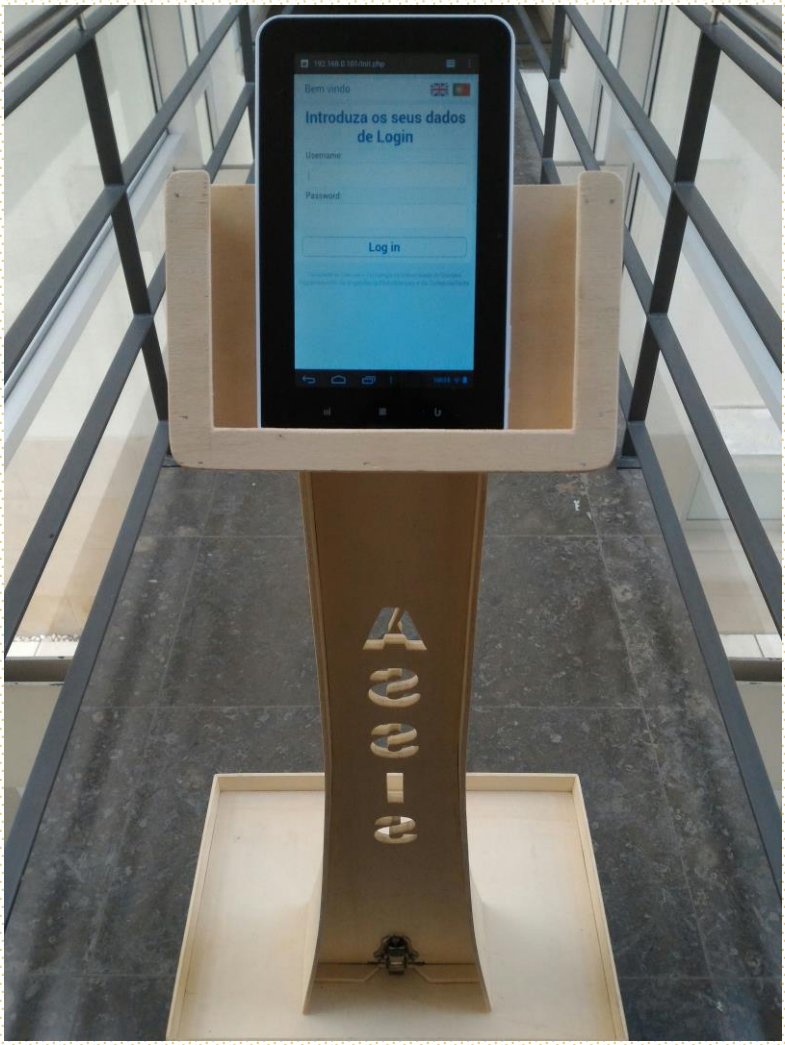

Manuel João Ventura Cruz

Robô Estafeta e Guia Autónomo

Dissertação de Mestrado

2017

· U C ·

Universidade de Coimbra

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Electrotécnica e de Computadores

Robô Estafeta e Guia Autónomo

Manuel João Ventura Cruz

Dissertação submetida para obtenção do grau de mestre em Engenharia Electrotécnica e de Computadores

Trabalho desenvolvido sob supervisão de:

Prof. Doutor António Paulo Mendes Breda Dias Coimbra
Prof. Doutor Mateus Daniel Almeida Mendes

Júri:

Prof. Doutor Jorge Manuel Moreira de Campos Pereira Batista (Presidente)
Prof. Doutor António Paulo Mendes Breda Dias Coimbra (Orientador)
Prof. Doutor Rui Alexandre de Matos Araújo (Vogal)

Trabalho desenvolvido no Instituto de Sistemas e Robótica da Universidade de Coimbra

INSTITUTO DE SISTEMAS E ROBÓTICA
UNIVERSIDADE DE COIMBRA

Agradecimentos

Quero agradecer aos meus orientadores, Prof. Doutor António Paulo Mendes Breda Dias Coimbra e Prof. Doutor Mateus Daniel Almeida Mendes, bem como ao Prof. Doutor Manuel Marques Crisóstomo por todo o apoio, incentivo e partilha de saber, imprescindíveis à realização desta dissertação.

Aos meus Pais, pelo esforço financeiro e por todo o apoio emocional nesta jornada.

Aos meus avós, por todo o apoio emocional e por tudo o que contribuíram para a pessoa que sou hoje.

Aos meus colegas e amigos, pela amizade, pelo convívio durante estes anos e por todo o apoio moral durante a realização deste trabalho.

Ao ISR, pelas condições e recursos proporcionados.

Resumo

A evolução tecnológica dos últimos anos, nomeadamente o aparecimento de processadores cada vez mais potentes e energeticamente mais eficientes e o crescimento das tecnologias móveis e de interacção homem-máquina, associados aos avanços na inteligência artificial e à crescente automatização, tornam a robótica uma das áreas em maior expansão na actualidade.

São notórios os avanços realizados na robótica cirúrgica e de manipulação bem como na robótica móvel, quer a nível de robôs de assistência como a nível de robótica de serviços e industrial. No entanto vários problemas continuam a ser alvo de investigação, nomeadamente a interacção dos robôs com o meio envolvente, ou seja, a capacidade de percepção. Assim vários investigadores têm-se debruçado na tentativa de dar aos robôs maior capacidade de percepção nomeadamente ao nível da "visão", bem como no desenvolvimento de algoritmos de navegação mais eficientes.

Nesta dissertação é feita uma renovação total da interface web de interacção com o robô, tendo sido criadas novas funcionalidades, tais como, uma barra de navegação dinâmica, dois modos de funcionamento e respectiva gestão, e uma opção de controlo manual. São ainda realizadas melhorias ao algoritmo de navegação, nomeadamente, correcção da condição de paragem e correcção na mudança de sequência de navegação. É ainda implementado o controlo das velocidades das rodas do robô, processamento das leituras dos sensores de proximidade de infravermelhos e auto localização.

Palavras-Chave: Robótica Móvel, Navegação Baseada em Visão, SDM, Interface web

Abstract

The technological evolution of recent years, namely the emergence of increasingly powerful and energy efficient processors and the growth of mobile technologies and human-machine interaction, combined with advances in artificial intelligence and increasing automation, make robotics one of the areas of greatest expansion.

The advances in chirurgical, manipulation and in mobile robotics, both at the level of robot assistance and at the level of service and industrial robotics, are remarkable. However, several problems continue to be investigated, namely the interaction of the robots with the environment, that is, the capacity of perception. Thus, several researchers have been trying to give robots greater ability to perceive, particularly at the level of "vision", as well as the development of more efficient navigation algorithms.

In this dissertation it was made a total renewal of the web interface of interaction with the robot, and were created new functionalities, such as a dynamic navigation bar, two modes of operation and respective management, and a manual control option. Improvements were made to the navigation algorithm, namely correction of the stopping condition and correction in the navigation sequence change. It were also implemented the control of the robot wheel speeds, processing of readings of the infrared proximity sensors and auto localization.

Keywords: Mobile Robotics, Vision Based Navigation, SDM, Web interface

Índice

Índice de Figuras

Índice de Tabelas

Abreviaturas

BMP	Bitmap Image File
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DSP	Digital Signal Processor
GPU	Graphics Processing Unit
HTML	HyperText Markup Language
IPO	Instituto Português de Oncologia
IR	Infra-red
LCD	Liquid Crystal Display
MCU	Multipoint Control Unit
MIPS	Millions of Instructions per Second
PHP	Hypertext Preprocessor
RAM	Random Access Memory
SDK	Software Development Kit
SDM	Sparse Distributed Memory
SRAM	Static Random Access Memory
WEB (www)	World Wide Web

"O começo de todas as ciências é o espanto de as coisas serem o que são".

(Aristóteles)

1.Introdução

Actualmente a robótica é uma das áreas tecnológicas em maior expansão. Desde robôs cicerones e assistentes pessoais a aspiradores, corta-relvas e outros veículos autónomos, cada vez mais os robôs de assistência e serviços são presença assídua nas vidas de milhões de pessoas.

A presente dissertação ocorre na sequência do desenvolvimento do robô ***Assis***. O Assis é um robô estafeta e cicerone que navega autonomamente através de sequências de imagens e de uma SDM. No final do seu desenvolvimento pretende-se que o Assis possa desempenhar funções em edifícios, museus e outros espaços públicos.

1.1 Objectivos

Esta dissertação ocorre no seguimento de outras duas dissertações tendo como objectivo implementar um algoritmo de contorno de obstáculos mais robusto que o existente. Contudo devido a problemas detectados tanto numa fase inicial da dissertação, como já numa fase intermédia, foi necessário alterar os objectivos previstos. Assim esta dissertação centrou-se na remodelação completa da interface gráfica existente, bem como na correcção de problemas do algoritmo de navegação, nomeadamente no reconhecimento dos destinos nos percursos realizados, na transição entre sequências e no processamento das leituras dos sensores IR.

1.2 Contribuições

Esta dissertação tem como contribuições principais:

- Remodelação total da interface gráfica;
- Resolução de problemas existentes no algoritmo de navegação:
    - Correcção da condição de paragem;
    - Correcção da transição de sequência de navegação;
    - Processamento das leituras dos sensores IR.
- Controlo da velocidade das rodas;
- Implementação de uma abordagem de auto localização;
- Implementação de um modo de controlo manual.

1.3 Estrutura da Dissertação

Esta dissertação é constituída por 7 capítulos:

No 1º capítulo é feito um enquadramento do trabalho realizado e são explicitados os objectivos da dissertação bem como as contribuições realizadas.

No 2º é descrita a arquitectura do sistema bem como as características técnicas dos vários elementos.

No 3º capítulo é realizado o levantamento do estado da arte relativo a algumas soluções de robôs de assistência, bem como algoritmos de contorno de obstáculos.

No 4º capítulo é apresentado todo o trabalho desenvolvido relativo à navegação e contorno de obstáculos.

No 5º capítulo é explicitado todo o trabalho realizado relativamente à interface com o utilizador.

No 6º capítulo são apresentados os resultados experimentais.

No 7º capítulo são apresentadas conclusões e proposto trabalho futuro.

2 . Estado da Arte

Neste capítulo é realizado o levantamento do estado da arte relativo a robôs de assistência com ênfase em soluções comerciais, bem como algoritmos de contorno de obstáculos sem recurso a mapeamento.

2.1 Robôs de Assistência

Pepper:

O Pepper [1] é um robô humanóide criado pela SoftBank Robotics que tem a capacidade de interpretar várias emoções, nomeadamente, alegria, tristeza, raiva ou surpresa. O robô consegue ainda distinguir sorrisos e linguagem não verbal como por exemplo movimentos da cabeça. Deste modo consegue adaptar o seu comportamento ao estado de espírito do seu interlocutor. O Pepper envia para a *Cloud* dados das suas interacções, por exemplo, regista se uma pessoa ri quando ele conta uma piada. A combinação de dados de vários robôs e interacções permite melhorar o comportamento do robô.

O Pepper apresenta as seguintes características principais:

- 1,20 metros de altura e uma massa de 28 kg;
- 4 microfones direccionais;
- 1 câmara 3D e duas câmaras HD;
- 3 rodas multi-direccionais;
- 2 sonares de ultra-sons, 6 lasers e 3 sensores de detecção de obstáculos;
- Bateria de iões de lítio com 12 horas de autonomia.

O robô está ainda equipado com um ecrã táctil e vários sensores tácteis nas mãos.

Figura 2.1 - Pepper, retirado de [1]

Gasparzinho:

O Gasparzinho [2][3] é um robô desenvolvido pelo ISR/IST e pela IdMind, englobado no projecto MOnarCH. O robô desempenha várias funções no serviço de pediatria do IPO de Lisboa, nomeadamente, assistente na sala de aula do IPO, brincar e realizar vários jogos com os utentes da pediatria e interacção com crianças e jovens nos corredores deste serviço.

O robô apresenta as seguintes características principais:

- 1,02 metros de altura e massa de 49 kg;
- 4 rodas mecanum;
- 4 baterias, permitindo uma autonomia superior a 3 horas;
- Ecrã táctil de 10.1';
- 2 x Laser Range Finder (LRF), frente e trás, com 5 metros de alcance;
- 1 sensor Kinect.

Figura 2.2 - Gasparzinho, retirado de [3]

***Care-O-bot*:**

O Care-O-bot [4] é um robô multifuncional criado pelo Fraunhofer Institute for Manufacturing Engineering and Automation. O robô é constituído por seis módulos independentes e configuráveis podendo deste modo adaptar-se às funções a realizar. Pode desempenhar diversas funções em ambiente doméstico, serviços ou indústria, nomeadamente servir bebidas, ajudar a cozinhar, assistência na doença, entregar pedidos em restaurantes, transportar materiais, etc.

Na configuração completa o robô apresenta as seguintes características principais:

- 1,52 metros de altura e uma massa de 140 kg;
- Bateria de iões de lítio de 48 V;
- 4-6 Intel NUC i5, 256 GB, 8 GB RAM;
- Ethernet gigabit em todos os módulos;
- Ecrã táctil de 15'.

Figura 2.3 - Care-O-bot, retirado de [4]

REEM:

O REEM [5] é um robô assistente desenvolvido pela PAL Robotics. Pode desempenhar várias tarefas, como recepção e acompanhamento de convidados, guia turístico, entertainer, telepresença, ponto de informação dinâmica, vigilância, etc.

O robô consegue interagir em nove línguas diferentes, podendo ser ensinados mais idiomas. Pode movimentar-se autonomamente ou ser controlado manualmente através de um tablet. É possível desenvolver aplicações personalizadas para interacção através de um ecrã de toque incorporado, bem como movimentos e discursos, consoante a aplicação desejada.

O robô apresenta as seguintes características:

- 1,70 metros de altura e 100 kg de massa, podendo carregar até 30 kg numa plataforma e 1 kg em cada mão;
- Baterias de iões de lítio de 48 volt, que permitem uma autonomia de 8 h;
- Ecrã táctil de 12.1 polegadas;
- 2 câmaras CMOS (uma câmara estéreo frontal e uma câmara normal traseira, ambas "global shutter").

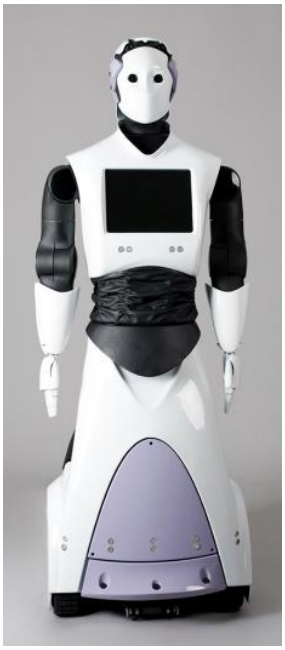

Figura 2.4 - REEM, retirado de [5]

2.2 Algoritmos de Contorno de Obstáculos

2.2.1 Campo de Forças Virtual

Em 1989 Borenstein [6] apresentou um algoritmo de navegação e reconhecimento baseado num campo de forças virtual. Este algoritmo baseia-se no facto de que um objectivo pode ser representado por uma força atractiva e um obstáculo por uma força repulsiva. Considera-se que são realizadas leituras de distância enquanto o robô navega e que estas são projectadas numa grelha (o "workspace" do robô é representado por uma grelha constituída por células de dimensão 10 x 10 cm), simultaneamente essa grelha é verificada utilizando uma janela de tamanho fixo, 33 x 33 células, sendo que o robô se encontra sempre no centro da janela. Considerando a soma vectorial das forças para uma dada janela R e a sua direcção δ (em graus), temos um comando de direcção da forma

$$\Omega = K_s(\theta - \delta) \tag{2.1}$$

onde θ é a orientação actual do robô (em graus) e K_S é uma constante proporcional (em s^{-1}).

2.2.2 Algoritmo de Navegação Adaptativo

Em 1997 Fujimori [7] desenvolveu um método de navegação baseado no cálculo adaptativo do ângulo de rotação do robô por forma a evitar obstáculos no seu percurso. Este método baseia-se no cálculo da velocidade angular da seguinte forma

$$\dot{\theta}(t) = -\eta(\theta(t) - \theta^*(t)) \tag{2.2}$$

onde θ é a orientação corrente do robô em radianos, θ^* é a orientação desejada para o robô em radianos e η é uma constante positiva. Durante a navegação em espaço aberto o ângulo $\theta^*(t)$ é dado pela seguinte expressão:

$$\theta^*(t) = \theta_t^* = \begin{cases} \emptyset(t) + \pi\,, & \emptyset(t) \le \theta(t) \\ \emptyset(t) - \pi\,, & \emptyset(t) > \theta(t) \end{cases} \tag{2.3}$$

e

$$\emptyset(t) \triangleq \tan^{-1}\frac{y(t)}{x(t)} \tag{2.4}$$

em que x(t) e y(t) são as coordenadas do robô, considerando o objetivo na origem do sistema de coordenadas.

No desvio de obstáculos o ângulo $\theta^*(t)$ é calculado de acordo com as leituras dos sensores de distância do robô da seguinte forma:

Consideram-se três sensores de ultra-sons, um alinhado com o eixo longitudinal do robô e os outros dois com desfasamentos de α e – α, respectivamente. Assume-se o seguinte:

- O máximo valor medido pelos sensores é $d_{máx}$;
- Quando não é detectado um obstáculo é devolvido pelo sensor o valor -1;
- Os obstáculos são polígonos convexos cujos ângulos internos são suficientemente superiores a α;
- A distância entre obstáculos é superior a $d_{máx}$.

Existem três situações possíveis: 1) os três sensores detectam obstáculos; 2) dois dos sensores detectam obstáculos; 3) apenas um sensor detecta um obstáculo.

Na primeira situação a orientação desejada é dada pela equação seguinte:

$$\theta_a^*(t) = \theta(t) + sign(d_l - d_r)\left(\frac{\pi}{2} - \varepsilon\right) \tag{2.5}$$

onde:

$$\varepsilon = \tan^{-1}((d\cos\alpha - d_c)/d\sin\alpha) \tag{2.6}$$

$$d \triangleq \max(d_r, d_l) \tag{2.7}$$

Na segunda situação existem dois casos possíveis: i) podem ser detectados obstáculos pelos sensores esquerdo e direito; ou ii) por um deles e pelo sensor central.

No caso i), uma vez que temos espaço livre na frente do robô, então a orientação desejada é igual à orientação do robô:

$$\theta_a^* = \theta(\mathrm{t}) \tag{2.8}$$

No caso ii) o ângulo de direcção desejado é dado pela equação:

$$\theta_a^* = \theta(t) + sign(d_l - d_r)\left(\frac{\pi}{2} - \varepsilon\right) + sign(d_l - d_r)\pi \tag{2.9}$$

onde ε é calculado pela equação (2.6)

No terceiro caso se o obstáculo for detectado pelo sensor esquerdo ou direito procedemos de acordo com a equação (2.8). Se for detectado pelo sensor central procedemos de acordo com a equação seguinte:

$$\theta_a^* = \theta(t) + \frac{\pi}{2} \tag{2.10}$$

2.2.3 Fuga Tangencial

Em 2008 A. Ferreira [8] apresenta um método de navegação baseado no ângulo de desvio que uma força de repulsão virtual provoca no robô. Este algoritmo tem como objectivo que o percurso de "fuga do robô" seja tangente ao obstáculo do qual o robô se pretende desviar, sendo a rotação calculada a partir de um sensor de posição. Considera-se que quando sujeito a uma força de repulsão o robô se encontra numa zona cuja impedância generalizada é dada por

$$Z(s) = B\ s + K \tag{2.11}$$

onde B e K representam constantes de amortecimento e elasticidade, respectivamente. Seja F_t, a componente da força de repulsão coincidente com o movimento do robô, o erro de impedância x_a, causado pela referida força é calculado como solução da equação

$$F_t = B\dot{x_a} + Kx_a \tag{2.12}$$

e o ângulo de rotação provocado pela força de repulsão é calculado da seguinte forma

$$\varphi = x_a sign(Fr) \tag{2.13}$$

onde Fr é a componente da força de repulsão perpendicular ao movimento. O algoritmo usa o mesmo controlo de posição que a arquitectura de controlo de impedância, Figura 2.5, mas o ângulo ϕ é calculado pela expressão

$$\varphi = sign(\beta)\frac{\pi}{2} - (\beta - \alpha) \tag{2.14}$$

onde β é o ângulo entre a orientação do robô e a menor distância ao obstáculo medida pelos sensores.

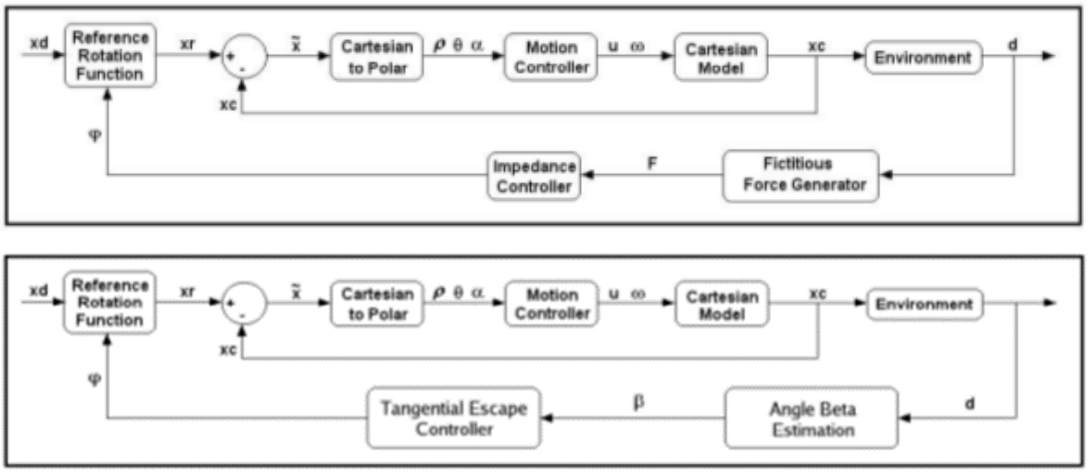

Figura 2.5 - Esquema de controlo por impedância e fuga tangencial, Ferreira 2008 [8]

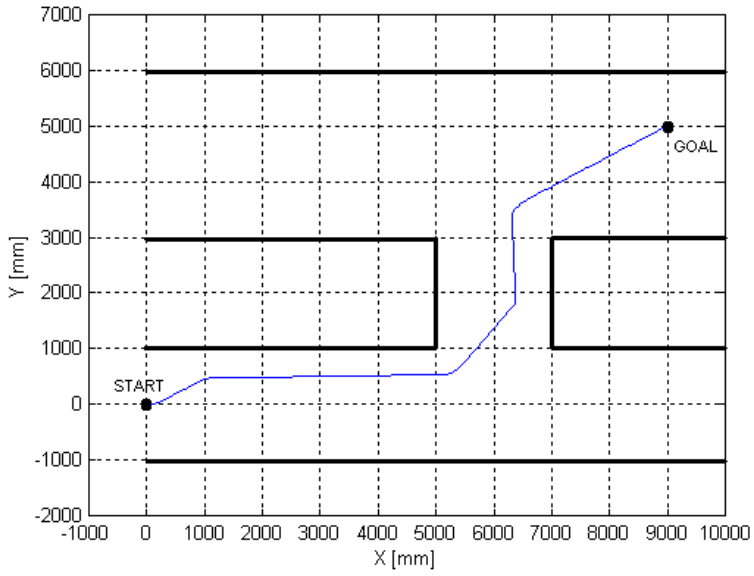

Figura 2.6 - Fuga tangencial, exemplo. Retirado de [8]

3. Arquitectura do Sistema

3.1 Descrição Geral

O sistema utilizado é constituído por um computador portátil "Asus KV55J" e uma plataforma robótica "Dr robot x80Pro" adaptada ao transporte de objectos. É ainda possível acoplar um suporte para um tablet de modo a facilitar a interacção com o utilizador [10].

O computador portátil realiza todo o controlo e processamento necessário à navegação do robot. Funciona ainda como servidor e base de dados através da aplicação "WampServer", que engloba um base de dados "MySQL" e um servidor "Apache". A base de dados permite armazenar locais, percursos, utilizadores e missões (tarefas) possíveis de realizar com o robô. O servidor permite o acesso remoto a uma interface gráfica de modo a interagir com a base de dados e enviar comandos ao robô. A interface pode ser acedida num computador remoto, no referido *tablet* ou num *smartphone* consoante as necessidades do utilizador. A comunicação entre a interface e o software de controlo é realizada através de um socket. O protocolo de comunicação é descrito no Anexo B.

A figura seguinte apresenta um diagrama da arquitectura do sistema.

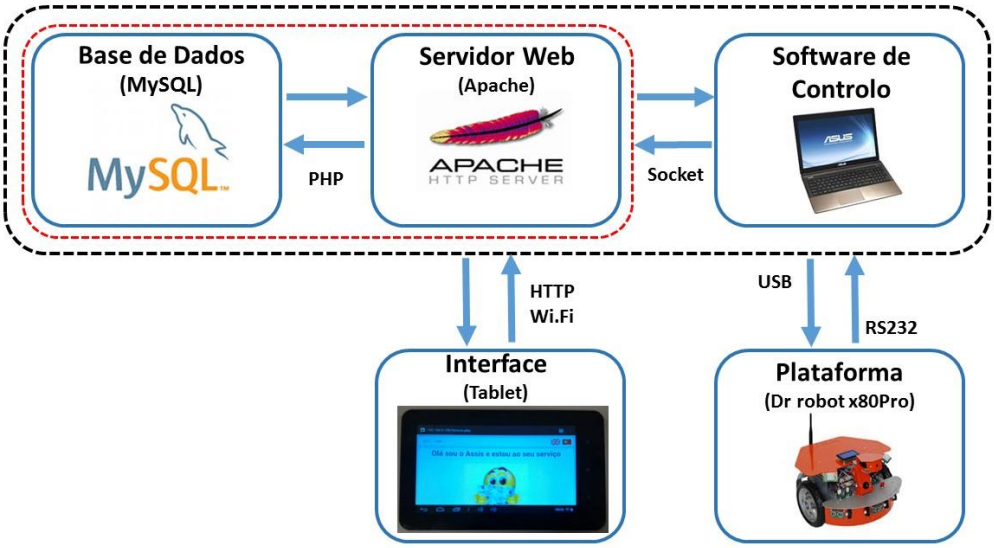

Figura 3.1 - Arquitectura do Sistema

3.2 Robô

O robô utilizado é uma plataforma x80 Pro da Dr Robot [11]. Esta plataforma consiste num robô de condução diferencial equipado com duas rodas de 18 cm de diâmetro actuadas por dois motores de 12 V com um binário de 40 kg.cm e uma roda *caster* na parte traseira, de modo a aumentar a estabilidade.

Está equipado com seis sensores de ultra-sons e sete sensores de infravermelhos, com alcance máximo de 2,55 m e 0,8 m, respectivamente, dois sensores de movimento piroeléctricos, um sensor de temperatura e um sensor de inclinação.

A plataforma está ainda equipada com e um LCD de 128 × 64 píxeis e uma câmara que permite captar um máximo de 15 imagens por segundo com resolução 176 × 144 em formato BMP.

Existem também um controlador de multimédia com um DSP de 120 MIPS de 16 bits com 256k × 16 bits de SRAM e um controlador para a parte sensorial e de movimento com um DSP/MCU de 40 MIPS de 16-bit com 2,5 k × 16 bits de SRAM.

O robô é alimentado por uma bateria de 7,2 V com capacidade de 4500 mAh, que permite uma autonomia de cerca de 4,5 horas. Tem uma massa de 3,5 kg e pode transportar cargas até 15 kg. Pode deslocar-se a uma velocidade máxima de 2,5 km/h.

Como referido na Secção 3.1, o robô foi adaptado para facilitar o transporte de objectos bem como a interacção com o utilizador. Assim foi acoplada ao robô uma estrutura modular, feita de contraplacado de choupo, com dois módulos. Um módulo é constituído por dois suportes em forma de U com cantos em forma de cunha, fixados à estrutura do robô por intermédio de uma base rectangular, e uma plataforma fixada aos topos desses suportes. Permite transportar objectos na plataforma bem como o computador portátil no espaço central dos suportes.

O segundo módulo é um suporte vertical com o nome do robô ao qual está acoplado um suporte para um tablet. Este módulo é fixado na plataforma por intermédio de dois bloqueios metálicos (vulgo bicos de pato).

3.3 Computador Portátil

O computador portátil utilizado é um Asus modelo KV55J [12] e tem as seguintes características: processador Intel Core I7-3630QM com frequência base de relógio de 2,40 GHz podendo atingir 3,4 GHz em modo turbo, cache de 6 MB, 4 Cores, 8 Threads e uma unidade de processamento gráfico Intel® HD Graphics 4000. O computador apresenta ainda 6 GB de RAM e uma GPU NVIDIA Gforce 635M com 2 GB de memória dedicada e 96 CUDA cores. Permite conexões via porta série, Wifi 802.11 b/g/n, ethernet e usb 2.0 e 3.0. O sistema operativo utilizado é o "Windows 7 Ultimate".

A Figura 3.2 mostra o robô e o computador no respectivo suporte.

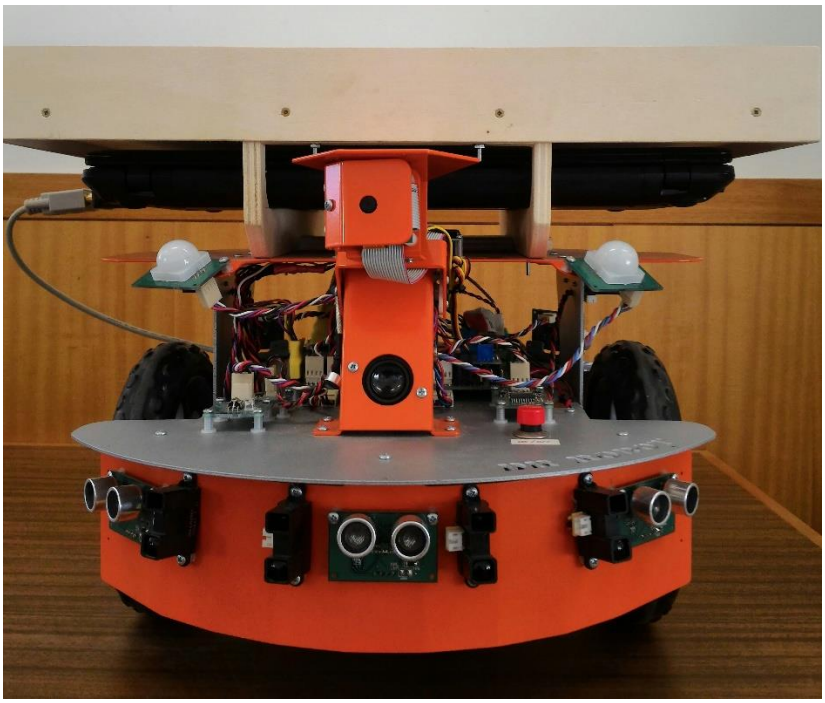

Figura 3.2 - Robô e computador

4. Navegação e Localização

4.1 Descrição Geral

Este capítulo foca-se nos vários aspectos relacionados com a navegação e localização do robô, nomeadamente, o algoritmo de navegação, o desvio de obstáculos e o processamento das leituras dos sensores usados nesse procedimento. São ainda descritos os novos algoritmos desenvolvidos para localização e navegação, bem como todas as correcções realizadas aos algoritmos implementados anteriormente e que apresentavam erros.

4.2 Trabalho Anterior

4.2.1 Navegação

Para navegar o robô faz uso de uma SDM implementada na GPU, de uma câmara, de 3 sonares e 6 sensores de proximidade de infravermelhos para detectar obstáculos. A cada inicialização do robô, a SDM é carregada com imagens e dados de navegação de sequências relativas a percursos previamente ensinados e guardados no disco duro. A navegação faz uso de um vector de dados associado a cada uma das imagens gravadas na SDM e de uma janela deslizante cujos detalhes são apresentados de seguida. As rotinas (métodos) envolvidos na navegação, bem como as variáveis mais importantes, são descritas no Anexo C.

Vector de dados:

Todas as imagens guardadas em disco têm associado um vector de dados, que corresponde ao nome da imagem. O vector de dados é constituído pelo número da sequência à qual pertence a imagem em questão, número de ordem da imagem na sequência, target da velocidade da roda direita em pulsos por segundo, target da velocidade da roda esquerda em pulsos por segundo e uma timestamp.

Janela deslizante:

O uso da janela deslizante tem como objectivo evitar que o robô se perca em percursos em que existem zonas muito semelhantes entre si. É útil também para evitar que o robô transite para sequências erradas, quando há duas ou mais sequências que tenham partes do percurso muito semelhantes entre si, e caso seja raptado (o problema do robô raptado ocorre quando o robô inadvertidamente deslocado de um ponto para outro). Assim, a cada nova predição é realizada uma pesquisa em toda a memória, sendo deste modo possível detectar raptos, caso ocorram. Após esta mesma pesquisa, caso a imagem obtida esteja fora da janela deslizante, é realizada uma nova pesquisa dentro da janela deslizante e registada a imagem mais próxima da imagem predita. Deste modo limita-se o intervalo de pesquisa e diminui-se a probabilidade de o robô se confundir com diferentes zonas muito semelhantes uma vez que é feita uma verificação apenas entre imagens próximas da imagem corrente.

A figura seguinte ilustra o funcionamento da janela deslizante.

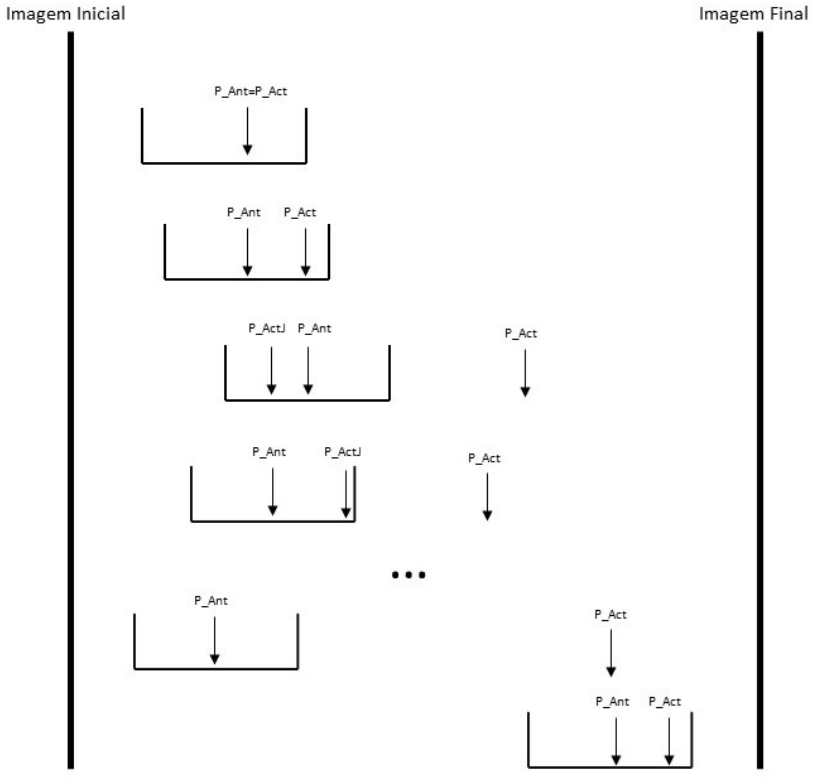

Figura 4.1 - Funcionamento da janela deslizante, adaptado de [12]

Descreve-se a seguir o princípio de funcionamento da navegação, implementada anteriormente [13].

Inicialmente o robô começa por captar uma imagem que é comparada com as imagens existentes na SDM e esta retorna um vector de dados da imagem mais próxima dessa imagem (imagem predita). Verifica se a sequência associada ao vector de dados corresponde à sequência desejada e seguidamente verifica se a imagem predita está dentro da janela deslizante. Se tal se verificar a janela é centrada nessa mesma imagem (a imagem corrente passa a ser a imagem predita). Obtém as velocidades de cada uma das rodas (esquerda e direita) a partir do vector de dados da imagem corrente, corrige-se o deslocamento horizontal e aplica-se os respectivos comandos de velocidade às rodas.

Se a imagem predita estiver fora da janela deslizante verifica-se dentro da janela qual a imagem mais perto da imagem predita. Centra-se a janela nessa imagem e assume-se a mesma como referência para a navegação (imagem corrente). Caso haja 80 predições consecutivas fora da janela deslizante, então assume-se que o robô está a navegar noutra zona do percurso e a janela é centrada na imagem predita (isto resolve o problema do robô raptado).

Se a sequência da imagem corrente for diferente da sequência da imagem predita procede-se da mesma forma que quando a imagem corrente se encontra fora da janela deslizante. Contudo o contador de predições fora da janela deslizante não é alterado. Caso haja 20 predições consecutivas e consistentes fora da sequência desejada, então assume-se que o robô está a navegar noutra sequência e a janela deslizante é centrada na imagem predita.

Caso a imagem corrente se encontre nos últimos 10 % de imagens da sequência gravadas na SDM então assume-se que estamos no final do percurso e termina a execução do algoritmo, caso contrário repetem-se os passos anteriores até ao final do percurso.

4.2.2 Desvio de Obstáculos

Para navegar de forma robusta é necessário que o robô consiga evitar quaisquer obstáculos existentes no seu percurso. Assim torna-se necessário detectar os obstáculos existentes na periferia do robô e saber a que distância estes se encontram. Para tal recorre-se a três sonares e quatro sensores IR existentes na parte frontal do robô.

São considerados obstáculos todos os objectos que se encontrem a menos de um metro do sonar frontal. Caso estes se encontrem a mais de 50 cm são considerados obstáculos temporários. Neste caso o robô apenas diminui a velocidade de modo a tentar evitar o obstáculo sem alterar a sua rota. Se os objectos estiverem a menos de 40 cm de um dos vários sensores de proximidade existentes na frente do robô, então o robô contorna o obstáculo desviando-se para o lado com mais espaço adoptando diferentes procedimentos, consoante esteja a navegar em linha recta ou a descrever uma curva[1].

Quando está a navegar em linha recta, são verificadas as leituras dos sete sensores existentes (quatro IR e três sonares), de modo a saber qual o sensor que está mais próximo do obstáculo. Após esta leitura o robô desvia-se para o lado contrário ao lado do sensor cuja distância medida é menor. Se este sensor for o sonar central então recorre-se aos dois sensores de infravermelhos mais próximos do centro para tomar a decisão. À medida que o robô se vai desviando os comandos enviados às rodas são armazenados numa pilha. Quando o desvio se encontra concluído, isto é, quando o obstáculo já não é detectado pelos sensores de proximidade, essa mesma pilha é esvaziada e a velocidade anteriormente imposta na roda direita é agora imposta à roda esquerda e vice-versa. Desta forma o robô recupera a orientação original e retoma a navegação baseada em visão.

No caso em que o robô se encontra a navegar numa curva o procedimento é similar, contudo a tolerância aumenta, passando de 40 para 30. Contrariamente à navegação em linha recta não se recorre a nenhuma pilha. Procede-se desta forma uma vez que no desvio o robô pode aproximar-se de uma parede da qual se vai tentar desviar e ao recuperar a orientação volta a deslocar-se em direcção à parede, ou seja, o robô vai ziguezaguear até deixar de detectar a parede. Após desviar-se do obstáculo o robô prossegue a navegação utilizando a informação visual.

[1] Para mais detalhes consultar [13].

4.3 Condição de Paragem e Mudança de Sequência

Verificou-se que nem sempre o robô reconhecia o destino de forma correcta, umas vezes parava demasiado cedo outras vezes não parava.

Inicialmente a condição de paragem implicava que a imagem corrente pertencesse às últimas 10 % de imagens da sequência actual e que a soma das velocidades das rodas fosse inferior a 500 pulsos/s, contudo a condição de paragem era mal calculada, uma vez que considerava o intervalo de imagens desde a primeira imagem gravada na SDM até à última imagem da sequência actual gravada na SDM. Assim, caso existissem múltiplas sequências gravadas na SDM e a sequência actual não fosse a primeira sequência do conjunto, era considerado para o cálculo da condição de paragem um intervalo de imagens muito superior ao desejado, o que fazia com que o robô parasse demasiado cedo.

Face ao problema anterior, o código foi alterado de modo a serem consideradas para efeitos de cálculo apenas imagens pertencentes à sequência actual. Foi também removida a restrição de velocidades, uma vez que se concluiu que esta restrição não acrescentava benefício algum.

Após esta alteração verificou-se que o problema persistia. Assim sendo, e considerando que o intervalo de 10 % corresponderia a um número elevado de imagens no caso de sequências com grande número de imagens, a condição de paragem foi novamente alterada. Deste modo foi considerado que o robô está no destino quando a imagem corrente corresponde a uma das últimas 30 imagens da sequência actual.

Esta última alteração implicou a implementação de duas funções, *Void Get_NumImg(int seq)* e *Void Get_Img_Act(char *dados)*, a primeira serve para verificar o número total de imagens da sequência actual, a segunda serve para saber o número de ordem da imagem corrente na sequência.

Para além da condição de paragem foi ainda detectada uma variável (*Terminou*, ver Figura C.1) que servia para sinalizar o final da navegação e não era reinicializada, o que fazia com que em navegações sucessivas os finais fossem detectados mais cedo que o previsto.

Relativamente à mudança de sequência era suposto que o robô mudasse de sequência após vinte predições sucessivas numa mesma sequência, fora da sequência actual. Contudo, o código existente fazia com que o contador de predições fora da sequência actual apenas fosse actualizado quando a sequência predita era igual à primeira sequência predita fora da sequência actual. Deste modo, se existissem predições correspondentes a sequências diferentes da primeira sequência predita fora da sequência actual o contador não era alterado. De igual modo, o contador só era reinicializado quando a sequência predita era a sequência actual, não era reinicializado quando ocorria uma alteração na sequência predita. Foi então corrigido o código de forma a contabilizar todas as predições sucessivas fora da sequência actual e sempre que ocorre uma mudança da sequência predicta o contador é reinicializado.

Exemplo: existem quatro sequências gravadas na SDM e o robô navega num percurso correspondente à sequência um. Ocorrem dez predições na sequência três, de seguida ocorrem dez predições na sequência dois e mais dez predições na sequência três. O código original faria com que o contador registasse um total de vinte predições na sequência três e como tal era alterada a sequência de navegação. Com as correcções realizadas são apenas registadas as dez últimas predições na sequência três, uma vez que o contador é reinicializado quando ocorre a mudança de sequência predita, da sequência dois para a sequência três.

Existia ainda um erro no código que fazia com que quando ocorria uma mudança de sequência o número de sequência actual era actualizado, mas a imagem de referência para a navegação só era actualizada na iteração seguinte. Embora não causasse problemas na navegação era problemático na verificação da condição de paragem, porque a sequência para a qual se verificava a referida condição não correspondia à sequência da imagem corrente. Uma vez detectado, este problema foi corrigido actualizando a imagem corrente e a respectiva sequência na mesma iteração.

Na figura seguinte está representado o fluxograma do algoritmo de navegação no estado actual.

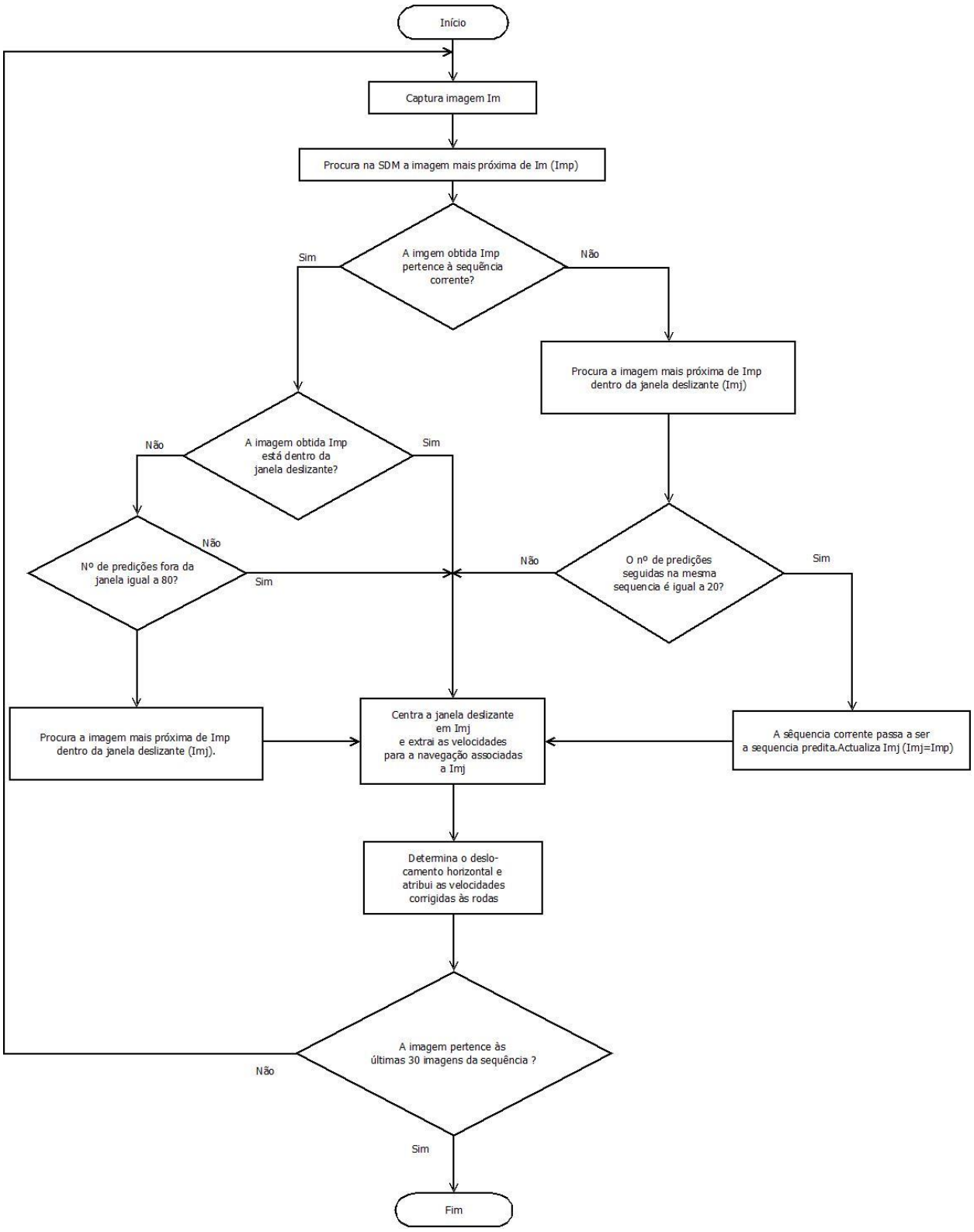

Figura 4.2 - Fluxograma do algoritmo de navegação

4.4 Auto Localização

No software desenvolvido anteriormente, o local onde o robô se encontra no inicio de uma missão tinha de ser indicado pelo utilizador directamente no código php e assumia-se que o local indicado correspondia à origem de um percurso. Contudo, o local onde o robô se encontra inicialmente pode não corresponder ao início de um percurso e existe sempre a possibilidade de o utilizador indicar uma localização errada. Por outro lado, não é viável, quer por questões de segurança, quer por questões de usabilidade, interagir com o código. Assim, foi necessário criar um mecanismo que permitisse ao robô localizar-se automaticamente.

Para esse efeito foi desenvolvida a função *Void Localizacao()*. Esta função é invocada após a SDM ser carregada com imagens de todos os percursos existentes na base de dados e devolve a imagem existente na SDM mais próxima da imagem captada pela câmara. Deste modo é retornada uma localização (nº de imagem e nº de sequência), que permite mostrar ao utilizador todos os destinos disponíveis do local onde o robô se encontra.

A auto localização é ainda usada para aferir se o robô reconhece correctamente o local onde se encontra, quer quando é colocado em modo de serviço, quer por opção de um utilizador com privilégios para tal. Em ambos os cenários são apresentadas ao utilizador a imagem captada pela câmara do robô e a imagem mais próxima obtida da SDM (ver secção 6.1). O utilizador indica se a localização obtida da SDM é correcta.

4.5 Sensores de Proximidade - Cálculo das Distâncias

Para se calcular correctamente a distância do robô aos obstáculos são usados os sensores de infravermelhos e os sonares.

No caso dos sensores de infravermelhos a relação entre o valor V da tensão de saída do sensor e o valor L, lido pelo ADC[2], é dada pela equação seguinte:

$$V = L \times \frac{3}{4095} \tag{4.1}$$

Sabendo a relação entre a distância medida e a tensão de saída, é possível determinar a relação entre as os valores retornados pelo ADC e a distância medida. Deste modo, utilizando a curva característica do sensor[3] e recorrendo a uma folha de cálculo, obteve-se a relação entre a distância medida e a tensão de saída, Figura 4.4 (curva a azul), bem como uma curva de interpolação (laranja traço interrompido) e a respectiva equação.

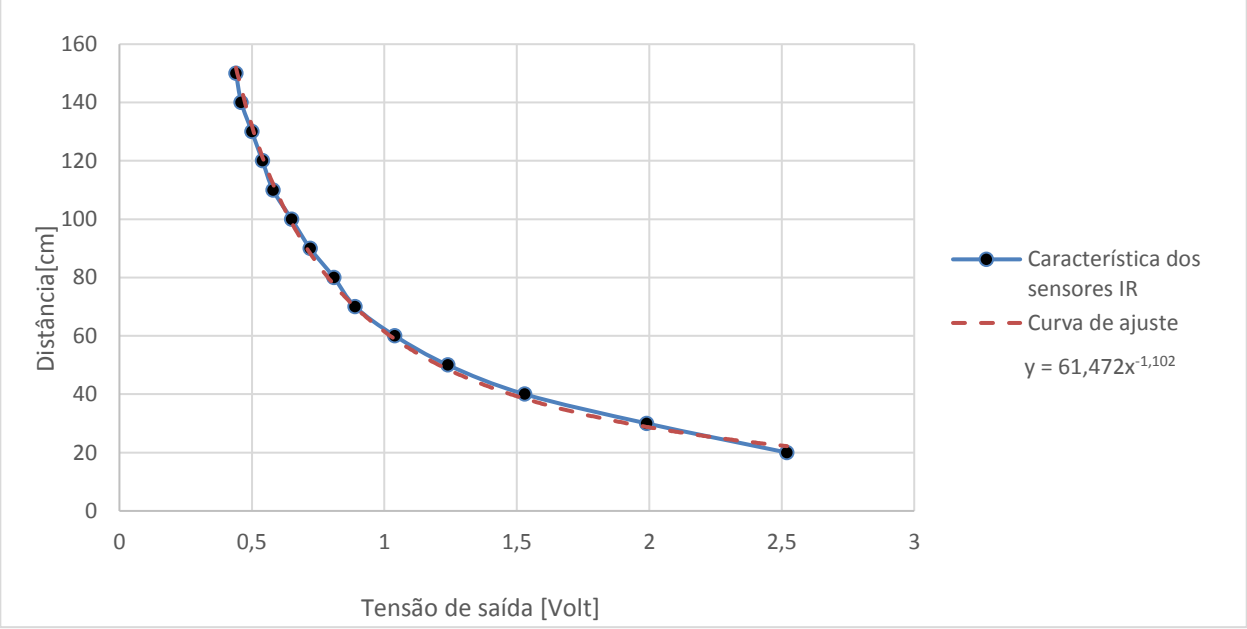

Figura 4.4 - Representação gráfica da relação entre distância medida e a tensão de saída, respectiva curva de ajuste e equação.

[2] Para mais detalhes consultar [15].
[3] Para mais detalhes consultar [14].

Substituindo x, na equação da curva de ajuste, pela equação (4.1) obtém-se a relação entre os valores dados pelo ADC e a distância medida (em centímetros):

$$d = 61{,}472 \times \left(L \times \frac{3}{4095}\right)^{-1{,}102} \tag{4.2}$$

Deste modo a leitura dos ADC's dos sensores de infravermelhos é realizada utilizando as funções *GetSensorIRRange(), GetCustomAD3(), GetCustomAD4(), GetCustomAD5(), GetCustomAD6(), GetCustomAD7()*. Posteriormente estas leituras são convertidas para distâncias usando a equação (4.2).

Relativamente à leitura dos sonares, é realizada directamente através das funções *GetSensorSonar1(), GetSensorSonar2() e GetSensorSonar3()*, existentes no *SDK* do robô, que correspondem respectivamente aos sonares esquerdo, frontal e direito. A figura seguinte representa a característica dos sonares.

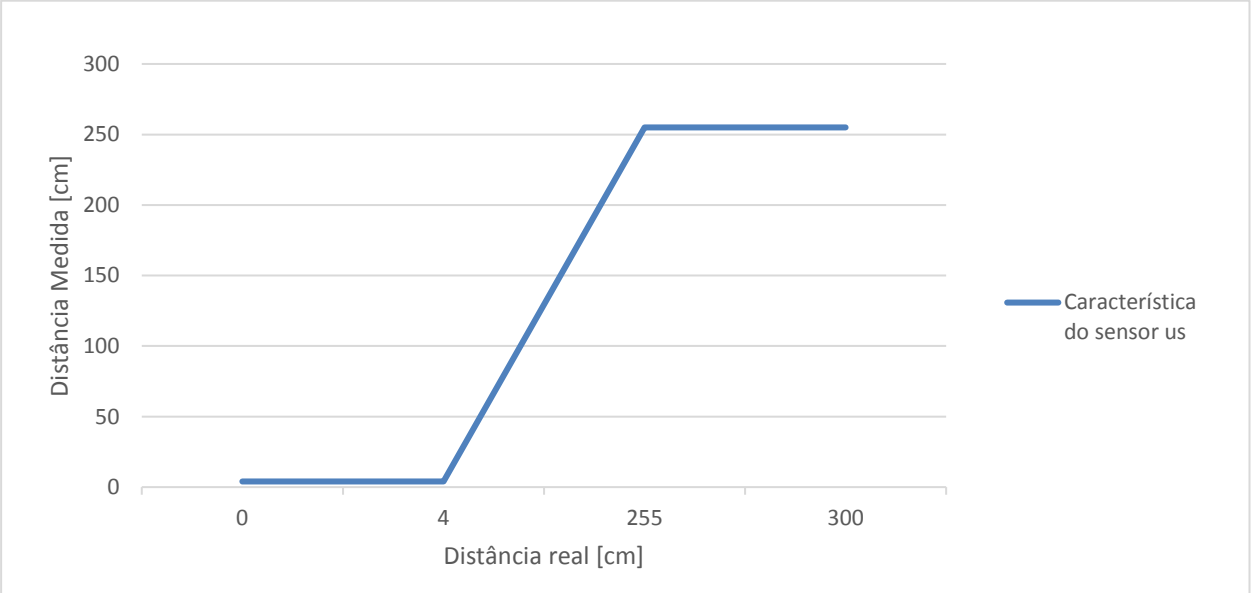

Figura 4.3 - Característica dos sonares

Após as leituras e processamentos já referidos, é aplicado um filtro de mediana às 3 últimas leituras de cada sensor (infravermelhos e sonares).

4.6 Controlo da Velocidade das Rodas

Durante a realização do trabalho verificou-se que existiam discrepâncias nos valores das velocidades das rodas quando aplicado o mesmo target (a mesma velocidade objectivo) a ambas as rodas, isto é, o robô não andava a direito em segmentos de percursos onde era suposto seguir a direito.

Inicialmente tentou-se resolver o problema compensando a diferença entre as velocidades das rodas esquerda e direita através de um offset fixo e posteriormente através de um offset dinâmico calculado através de leituras das referidas velocidades.

Uma vez que os resultados obtidos não eram satisfatórios foi contactado o fabricante, o qual enviou o documento [16], com um procedimento a adoptar. Testado o procedimento sugerido constatou-se que os resultados obtidos não eram os desejados. Assim foi desenvolvido um algoritmo de compensação que realiza a compensação das velocidades em relação aos targets e das diferenças entre a velocidade das rodas esquerda e direita, a partir da respectiva média. No entanto os resultados continuavam a não ser os esperados e foi necessário enveredar por outra estratégia.

Optou-se por manter a compensação das velocidades em relação ao target e realizar a compensação da diferença entre as velocidades a partir da diferença entre as leituras acumuladas de cada um dos encoders. De seguida descreve-se o funcionamento da compensação.

A compensação é usada quando o robô está em modo de controlo manual ou está a ser ensinado um percurso, e se pretende que ande em linha recta. Não é usada na navegação porque nesse caso os comandos recebidos pelo robô já se encontram compensados e está activo o algoritmo de compensação do desvio horizontal.

Na primeira iteração são lidos os valores actuais dos encoders, usando os métodos *GetEncodrPulse1()* e *GetEncoderPulse2()*, existentes no SDK fornecido pelo fabricante. Os valores obtidos são guardados como valores de referência. Este procedimento é realizado porque os valores das contagens dos encoders são guardados em registos internos do microcontrolador associado, aos quais só é possível fazer reset reinicializando o robô. Nas iterações seguintes as referências são actualizadas no final e correspondem sempre à última leitura relativa efectuada.

A cada iteração são lidos os encoders e é calculada a sua leitura relativa através das expressões:

$$Leitura_{Esq} = Ref_{esq} - Enc_{Esq} \tag{4.2}$$

$$Leitura_{Dir} = Enc_{Dir} - Ref_{Dir} \tag{4.3}$$

onde $Leitura_{Esq}$, $Leitura_{Dir}$, Ref_{Esq}, Ref_{Dir}, Enc_{Esq} e Enc_{Dir} são as leituras relativas, as referências e as leituras absolutas dos encoders esquerdo e direito, respectivamente.

Quando há reinicialização das contagens dos encoders as respectivas leituras relativas são calculadas da seguinte forma:

$$Leitura_{Esq} = \left(32767 - Enc_{Esq}\right) + Ref_{Esq} \tag{4.4}$$

$$Leitura_{Dir} = (32767 - Ref_{Dir}) + Enc_{Dir} \tag{4.5}$$

Cada uma das leituras relativas é somada ao respectivo acumulado.

De seguida cada uma das velocidades é lida sequencialmente três vezes, é calculada a respectiva mediana e é realizada a sua compensação em relação ao target de acordo com a seguinte expressão:

$$Comando = \ Comando + (\ Target - V_{Lida})/20 \tag{4.6}$$

Caso existam diferenças nas leituras acumuladas é realizada a seguinte compensação:

$$Comado_{Esq} = Comando_{Esq} + (Enc\_Dir_{Total} - Enc\_Esq_{Total})/20, \quad se\ Enc\_Esq_{Total} < \ Enc\_Dir_{Total} \tag{4.7}$$

$$Comado_{Dir} = Comando_{Dir} + (Enc\_Esq_{Total} - Enc\_Dir_{Total})/10, \quad se\ Enc_{Esq_{Total}} > \ Enc\_Dir_{Total} \tag{4.8}$$

onde $Comando_{Esq}$, $Comando_{Dir}$, Enq_Esq_{Total} e Enc_Dir_{Total} são os comandos de velocidade e as leituras acumuladas dos encoders, associados às rodas esquerda e direita respectivamente.

Sempre que existe alteração dos targets o comando é alterado na mesma proporção.

Os resultados obtidos são apresentados na Secção 6.2.

5.Interface Web

Este capítulo descreve a interface existente e o trabalho realizado sobre a mesma de acordo com o diagrama de blocos mostrado na figura 5.3.

5.1 Descrição Geral

O modo mais simples e versátil para controlar o robô é através de uma interface web desenvolvida em *PHP*, *HTML*, *JavaScript* e *CSS*, suportada por uma base de dados My*SQL*. Uma vez que a interface existente era minimalista e pouco funcional, esta foi totalmente remodelada. Assim foram implementadas as seguintes funcionalidades:

- Dois modos de funcionamento: serviço e manutenção
- Barra navegação
- Logout
- Shutdown
- Colocar em Serviço
- Auto Localização
- Controlo Manual

Foram ainda acrescentadas uma página de início (Menu) e uma página de boas vindas (Serviço), bem como reformuladas/remodeladas as seguintes páginas:

- Login
- Página Inicial
- Página de Gestão
- Tarefa Guia

As figuras seguintes correspondem aos diagramas das opções disponíveis na interface relativos a administradores e utilizadores genéricos respectivamente.

No anexo D são apresentados os correspondentes diagramas de casos de uso.

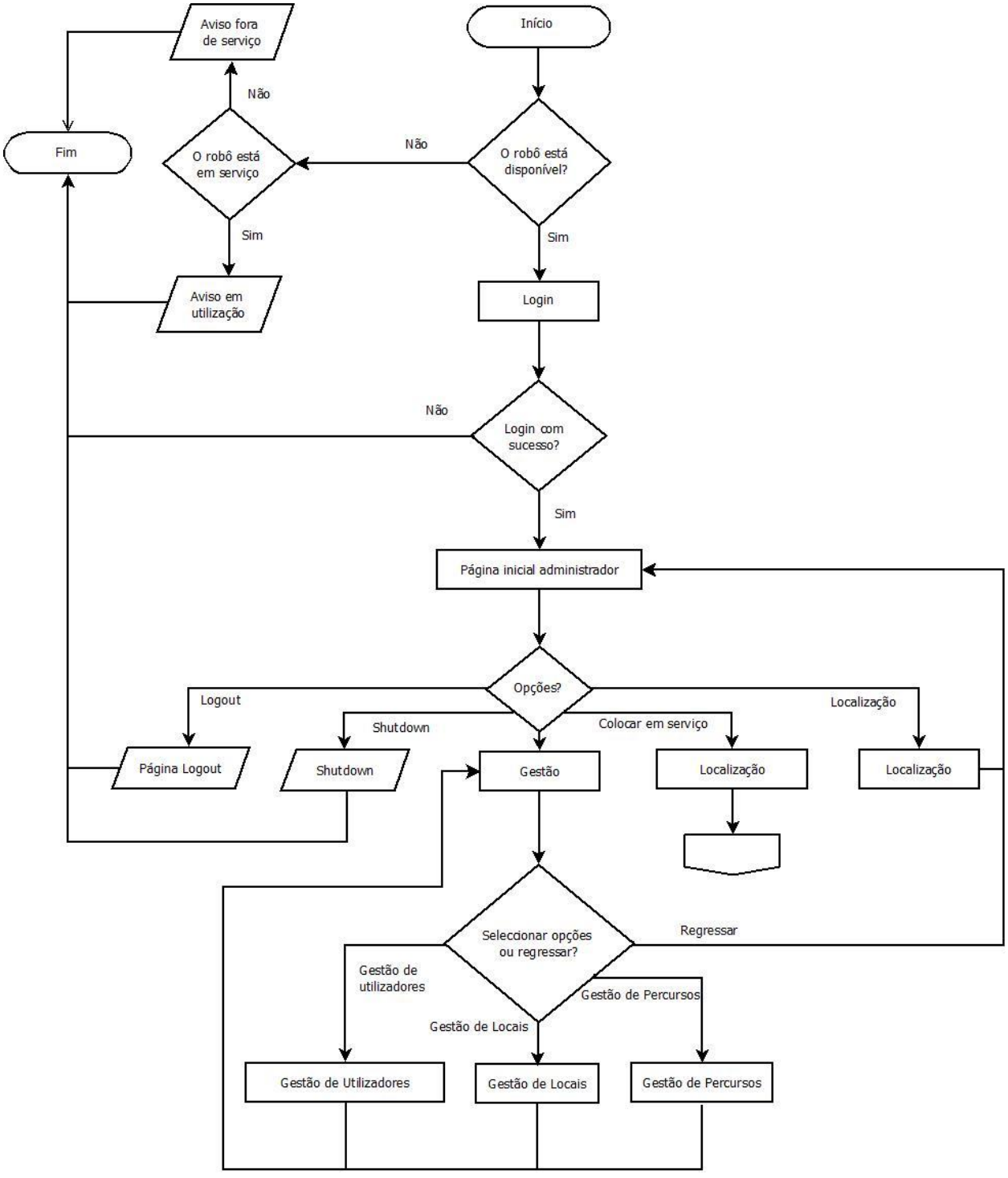

Figura 5.2 - Diagrama de opções disponíveis para um administrador

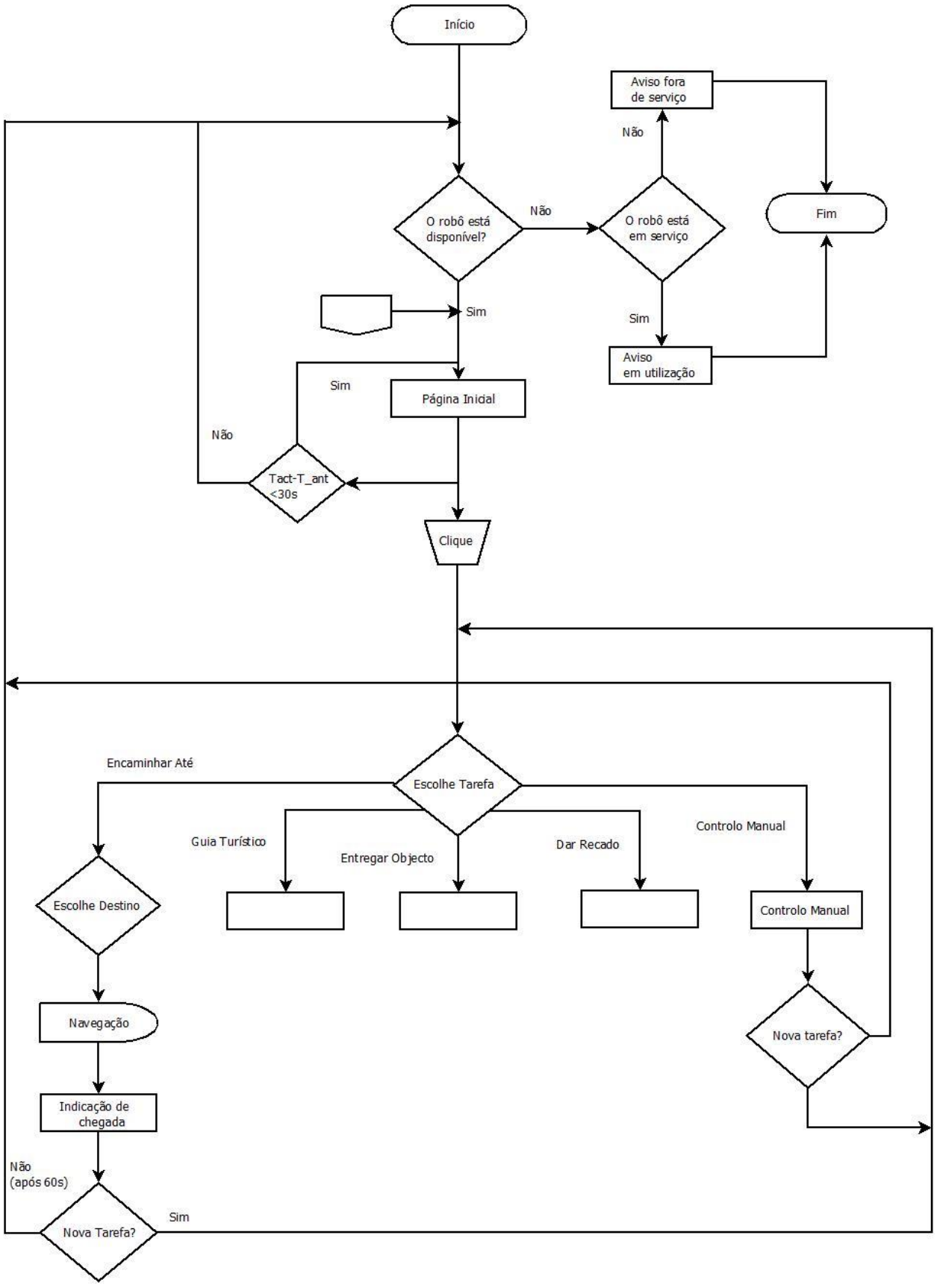

Figura 5.3 - Diagrama das opções disponíveis para um utilizador genérico

5.2 Funcionalidades Desenvolvidas

Modos serviço e manutenção

Uma vez que se pretende que qualquer pessoa possa utilizar o robô torna-se necessário restringir o acesso do público em geral a determinadas funcionalidades, bem como garantir que o robô não está acessível para terceiros durante a realização de missões. Assim foi acrescentada à base de dados a tabela estado que armazena o utilizador corrente, o modo de funcionamento e o id da tarefa em realização, (Figura 5.3). Deste modo quando o robô se encontra em manutenção ou em utilização (a realizar uma dada tarefa) o utilizador recebe os respectivos avisos.

Por omissão, quando um utilizador registado inicia sessão o robô é colocado em modo de manutenção. Consoante os privilégios atribuídos, este modo permite ao utilizador realizar diversas tarefas. Por exemplo, ensinar percursos e gerir locais e/ou utilizadores, sem que mais ninguém possa controlar o robô. O modo Serviço permite a qualquer pessoa (utilizador anónimo) utilizar o robô desde que este se encontre em repouso (Id Tarefa=0) quando iniciada a interacção.

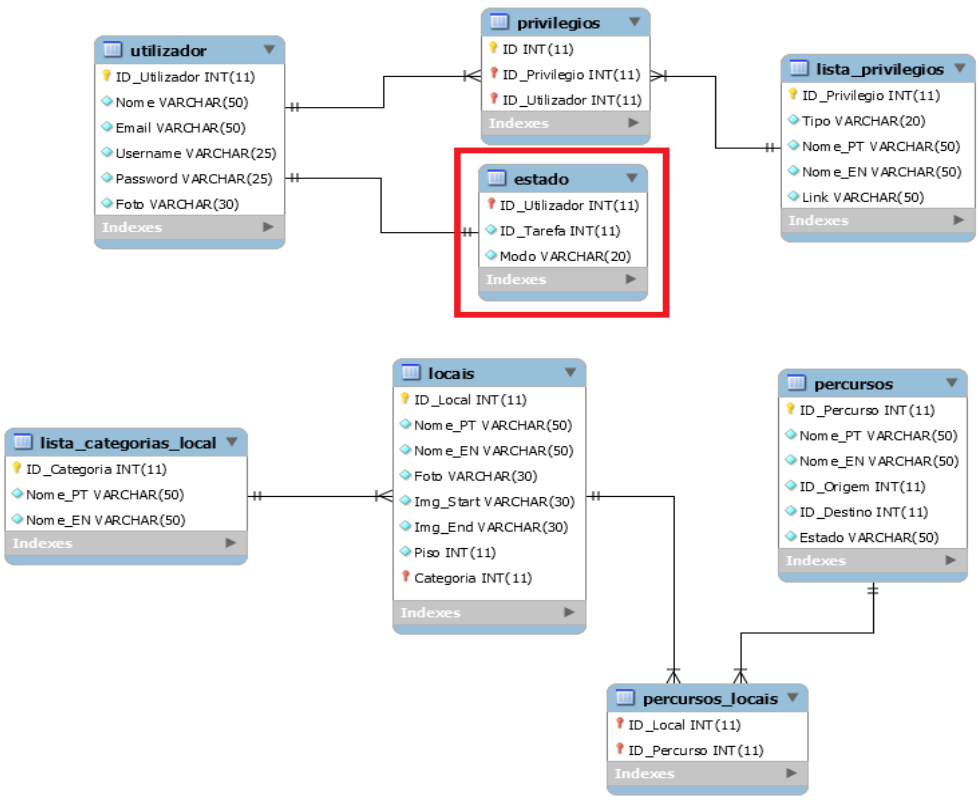

Figura 5.3 - Diagrama EER da base de dados

Barra de navegação

Para facilitar a navegação entre páginas e melhorar a usabilidade da interface foi implementada uma barra de navegação, como mostrado na Figura 5.4. Esta barra é construída dinamicamente de acordo com os privilégios associados ao utilizador que num dado momento está a controlar o robô. A barra é constituída por qualquer combinação das seguintes funcionalidades:

Início: permite ao utilizador retornar à página inicial.

Logout: permite ao utilizador fazer logout. Neste caso a funcionalidade já existia tendo sido apenas acrescentado uma caixa de confirmação.

Shutdown: pede ao utilizador para desligar o robô, verifica se este se encontra desligado e em seguida envia uma ordem (comando) ao software de controlo para este se desligar.

Colocar em serviço: altera o modo de funcionamento para modo serviço. Inicialmente o robô auto localiza-se, como descrito na Secção 4.6. Caso a localização esteja correcta é alterado o modo de funcionamento, ficando o robô em modo de Serviço e disponível para realizar missões.

Localização: permite a um utilizador com privilégios localizar o robô e aferir se o robô reconhece o local onde se encontra, ver secção 5.6.

Gestão: permite aceder à página de Gestão. As funcionalidades disponíveis nesta página estão dependentes dos privilégios do utilizador.

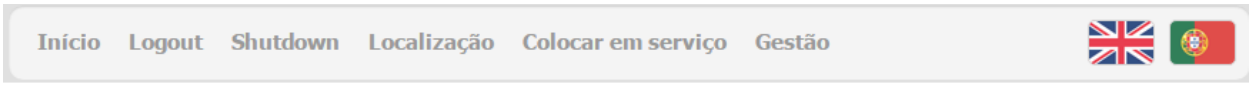

Figura 5.5 - Barra de navegação com todas as funcionalidades disponíveis

Controlo Manual

Aproveitando a interface existente para ensinar percursos foi desenvolvida uma tarefa de controlo manual. Esta funcionalidade permite ao utilizar controlar o robô em qualquer espaço da forma que lhe for mais conveniente. Pode também ser usado como complemento quando se ensina um percurso ao robô e se pretende testar esse mesmo percurso, não estando ensinado percurso de regresso. Permite ainda ao utilizador controlar o robô enquanto este o auxilia no transporte de pequenas cargas.

5.3 Páginas Criadas de Raiz

Página de Início (Menu)

A página de início, (Figura 5.5), apresenta os seguintes elementos: saudação de boas vindas, barra de navegação e apresenta ao centro a imagem captada pela câmara do robô em tempo real. É acedida após login por um utilizador registado.

Figura 5.5 - Menu inicial mostrado a qualquer utilizador registado

Página de Boas Vindas

A página de boas vindas, (Figura 5.6), denominada serviço foi criada com o intuito de apresentação quando o robô está em modo de serviço e acessível a qualquer pessoa. A página apresenta uma saudação e um emoji animado. Clicando nesse emoji o utilizador é encaminhado para a página de selecção de tarefas.

Figura 5.6 - Página de boas vindas, mostrada quando o robô se encontra em modo de serviço

5.4 Páginas remodeladas/reformuladas

Login

Esta página foi alterada por forma a remover a opção de registo, uma vez que esta foi incorporada na gestão de utilizadores, bem como um botão para recuperação da password cuja implementação não estava realizada. Assim qualquer utilizador passa a ser adicionado por um administrador com privilégios para tal. A Figura 5.7 mostra o aspecto actual da página.

Figura 5.7 - Página de login actual, mostrada quando se acede à interface e o robô não está em serviço nem controlado por um utilizador registado.

Página Inicial

Na versão da interface previamente existente, esta página servia como página de boas vindas, era acedida directamente após o utilizador iniciar sessão e apresentava lado a lado uma imagem do utilizador (ou outra escolhida por este aquando do registo) e as tarefas que o robô pode realizar. Após a reformulação esta página serve apenas como menu para seleccionar a tarefa a realizar. É acedida após clicar na página de boas vindas quando o robô se encontra em modo serviço e apenas apresenta a lista de tarefas disponíveis. Relativamente a esta lista, por forma a ser mais intuitivo o funcionamento das tarefas existentes, foram alteradas algumas nomenclaturas bem como descrições. As imagens seguintes ilustram as diferenças existentes.

Figura 5.8 - Página inicial antiga e actual

Página de Gestão

A página de gestão incorpora todas as opções de gestão de utilizadores, locais e percursos.

Inicialmente a página apresentava todas as opções existentes em dois subconjuntos: opções de administrador e opções de programador existindo para cada opção um *select menu* de modo a escolher o elemento (utilizador, local, percurso) que se pretendia gerir. Na versão actual as várias opções foram agrupadas em três subgrupos: utilizadores, locais percursos. Foi criada uma página referente a cada subgrupo.

Em cada nova página são apresentados os elementos referentes a esse subgrupo de forma independente, existindo para cada elemento botões com as opções disponíveis e um botão no cimo da respectiva página para adicionar novos elementos.

Relativamente à gestão de utilizadores existiam duas opções: eliminar utilizador e alterar privilégios, existindo ainda a possibilidade de registar um novo utilizador na página de login. Na versão actual, existe um botão para adicionar um novo utilizador e as opções eliminar utilizador e editar utilizador. Esta última opção permite alterar os privilégios atribuídos a um dado utilizador, bem como alterar os seus dados de registo.

Em relação à gestão de locais já existiam as opções de adicionar bem como eliminar locais da base de dados. Foi criada a opção editar local. Esta opção permite, por exemplo, alterar a imagem associada ao local bem como o seu nome ou categoria.

No que concerne aos percursos foram criadas as opções activar/desactivar e eliminar percurso. No caso desta última opção, já existia um botão com essa função, mas não existia código implementado. A opção activar/desactivar permite desactivar um dado percurso, por exemplo se parte do percurso estiver em manutenção ou existir uma qualquer situação que impeça o acesso ao local.

As figuras seguintes evidenciam as diferenças entre versão actual e a versão original.

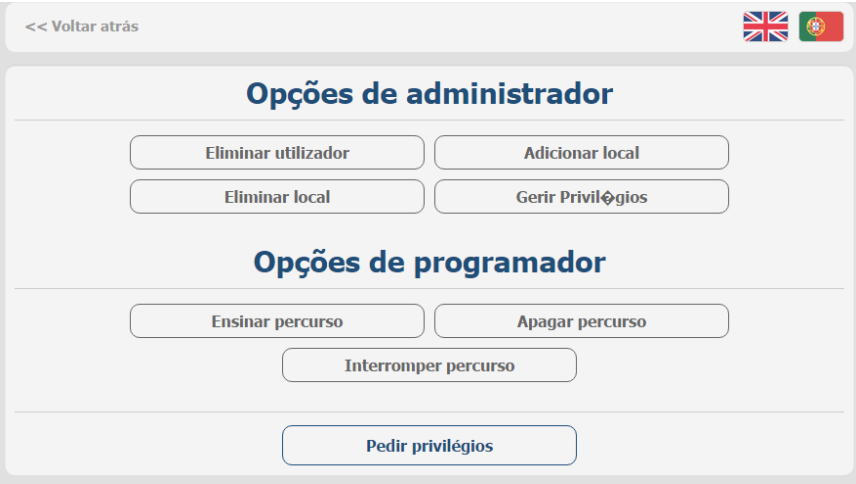

Figura 5.9 - Página de gestão (Definições), antiga

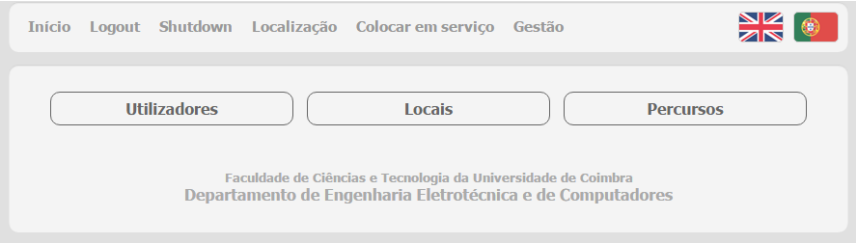

Figura 5.10 - Página de gestão actual

Tarefa - Encaminhar até

Na versão original (tarefa guia), o local de origem era indicado directamente no código *PHP*, uma vez que não existia um menu de selecção na interface nem um mecanismo de auto localização, este último estritamente necessário caso o robô se encontre algures num dado percurso. A partir da origem indicada no código eram apresentados ao utilizador os destinos possíveis, o utilizador escolhia o destino pretendido, a sequência correspondente era carregada na *SDM* e iniciava-se a navegação. No final era indicado ao utilizador que tinha chegado ao destino pretendido.

Após a selecção do destino era apresentado ao utilizador um aviso de preparação do percurso enquanto era carregada a sequência de imagens do percurso a realizar. De seguida, e enquanto decorria a navegação, eram mostradas uma indicação de que o robô ia iniciar o percurso, uma animação de uma barra de espera (quatro quadrados que iam acendendo à vez) e uma indicação para seguir o robô. No final era mostrado um aviso a informar o que tinha chegado ao final do percurso.

Na versão actual foi implementado um mecanismo de auto localização (ver secção 4.6) através do qual o robô indica ao utilizador os destinos possíveis a partir do local onde este se encontra. Assim, se esta localização corresponder a uma das primeiras quarenta imagens da respectiva sequência então assume-se que o robô está na origem do percurso a que corresponde essa sequência e são apresentados ao utilizador todos os destinos possíveis a partir desse local. Caso contrário, é apresentado ao utilizador o destino correspondente à sequência a que pertence a localização em questão.

Uma vez que não tinha sentido serem mantidas no ecrã durante todo o tempo do percurso, a indicação de início do percurso e a barra de espera animada foram removidas. Manteve-se indicação para seguir o robô e é apresentada a imagem captada pela câmara em tempo real.

No final é apresentada uma indicação de chegada ao local de destino, a imagem deste destino existente na base de dados e a imagem capturada pela câmara do robô. Existe ainda um botão caso o utilizador pretenda realizar uma nova tarefa. Foi implementado um temporizador que ao fim de um minuto faz a página mudar para a página de boas vindas.

6.Resultados

6.1 Auto Localização

Como referido na secção 4.4 a auto localização é usada em dois cenários. Na tarefa encaminhar até, para saber os destinos disponíveis, e para verificação da localização do robô quando é colocado em serviço ou por opção do utilizador.

As figuras seguintes ilustram o funcionamento da auto localização nos dois cenários.

Figura 6. 1 - Auto localização, indicação de destinos possíveis

Figura 6. 2 - Auto localização, colocação do robô em modo de serviço

6.2 Controlo das Velocidades das Rodas

Como referido na secção 4.6 foram realizadas várias tentativas para corrigir a disparidade entre as velocidades das rodas em percursos em linha recta. Nesta secção são apresentados os resultados obtidos utilizando o procedimento indicado pelo fabricante [16], e utilizando o algoritmo desenvolvido.

Para validação do controlo da velocidade das rodas utilizando o procedimento do fabricante foram realizados dois ensaios, um antes e outro após calibração, a cada uma das rodas. A seguir descreve-se o procedimento adoptado.

Com a bateria totalmente carregada o robô é colocado suspenso de forma a que as rodas rodem livremente sem contacto com o solo. São enviados dez targets de velocidade em pulsos/s para a roda. Para cada target enviado a roda gira durante um minuto e durante esse período são registadas as voltas completas efectuadas pela roda. Estes valores são obtidos através da visualização de uma marca colocada na roda e aproximando a posição final aos quadrantes, imaginado um referencial centrado no eixo de rotação. No final regista-se a contagem do respectivo encoder. Com os valores obtidos são calculados o número de pulsos por volta e a velocidade em pulsos por segundo. Todos os valores são truncados a duas casas decimais.

As tabelas seguintes apresentam os resultados obtidos. As Figuras 6.3 e 6.4 representam respectivamente as velocidades das rodas antes e após calibração.

Tabela 6.1 - Registo de dados pré-calibração

	Target (pulsos/min) [1]	Pulsos/min [2]	Voltas/min [3]	Pulsos/Volta [4]	Pulsos/s [5]	Erro relativo % [6] = ([5]-[1])/[1]
Roda Direita	100	3522	4,7	749,36	58,7	-41,3
	200	5899	7,88	749,08	98,32	-50,84
	300	8590	11,40	753,51	143,17	-52,28
	400	11382	15,13	752,53	189,70	-52,58
	500	13363	17,75	752,85	222,72	-55,46
	600	15877	21,13	751,57	264,62	-55,90
	700	18251	24,25	752,62	304,18	-56,55
	800	22897	30,45	751,95	381,62	-52,30
	900	27118	36,03	752,62	451,97	-49,78
	1000	30689	41,83	733,66	511,48	-48,85

Roda Esquerda	100	4740	6,30	752,38	79,00	-21,00
	200	7217	9,63	749,82	120,28	-39,86
	300	9899	13,20	749,92	164,98	-45,01
	400	12768	17,00	751,06	212,80	-46,80
	500	15402	20,50	751,32	256,70	-48,66
	600	18319	24,38	751,55	305,32	-49,11
	700	20399	26,13	780,82	339,98	-51,43
	800	24617	32,75	751,66	410,28	-48,71
	900	29195	38,70	754,39	486,58	-45,94
	1000	32512	43,25	751,72	541,87	-45,81

Tabela 6.2 - Registo de dados após calibração

	Target (pulsos/s) [1]	**Pulsos/min [2]**	**Voltas/min [3]**	**Pulsos/Volta [4]**	**Pulsos/s [5]**	**Erro relativo % [6] = ([6]-[1])/[1]**
Roda Direita	100	8077	10,75	751,35	134,62	34,62
	200	11929	15,88	751,43	198,82	-0,59
	300	14761	19,63	752,15	246,02	-17,99
	400	17107	22,75	751,96	285,12	-28,72
	500	27369	36,38	752,41	456,15	-8,77
	600	33629	44,75	751,49	560,48	-6,59
	700	36355	48,38	751,52	605,92	-13,44
	800	40811	54,30	751,58	680,18	-14,98
	900	45475	60,50	751,65	757,92	-15,79
	1000	48745	64,88	751,37	812,42	-18,76
Roda Esquerda	100	6681	8,88	752,79	111,35	11,35
	200	11654	15,50	751,87	194,23	-2,88
	300	15799	21,01	751,98	263,32	-12,23
	400	19791	26,63	743,32	329,85	-17,54
	500	28126	37,38	752,54	468,77	-6,25
	600	31967	42,56	751,06	532,78	-11,20
	700	35276	46,91	751,99	587,93	-16,01
	800	40178	53,38	752,75	669,63	-16,30
	900	44388	59,10	751,07	739,80	-17,80
	1000	47644	63,38	751,78	794,07	-20,59

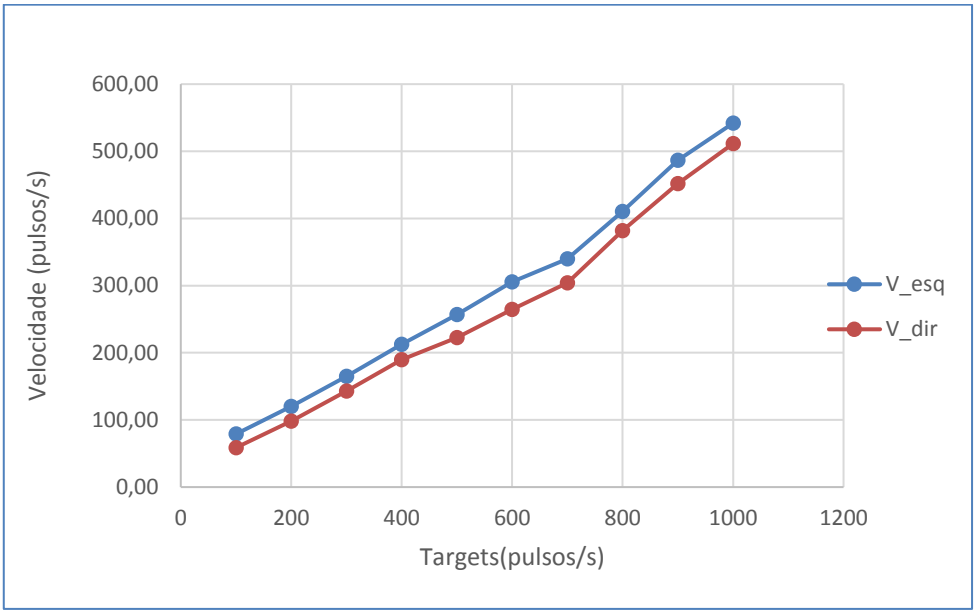

Figura 6.3 - Representação gráfica das velocidades das rodas pré-calibração, em função dos targets

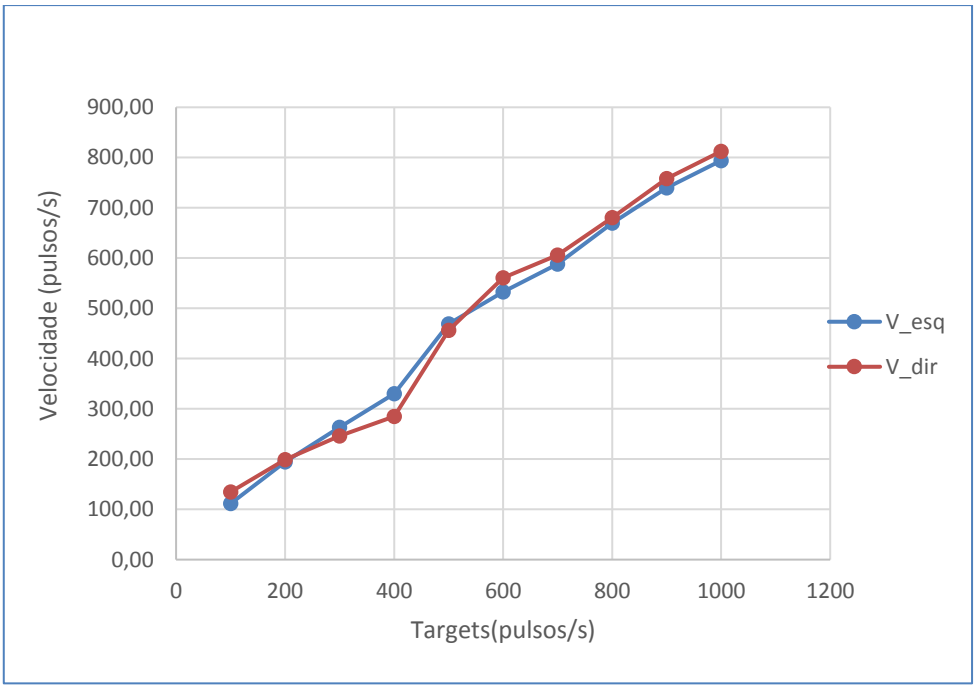

Figura 6.4 - Representação gráfica da velocidade das rodas após calibração

Relativamente ao algoritmo desenvolvido foram realizados dois ensaios em carga. Um ensaio sem qualquer compensação e um ensaio com o algoritmo em funcionamento. A título de comparação com os resultados anteriores foi ainda realizado um ensaio em vazio. Durante os ensaios foram registados num ficheiro de texto os valores das velocidades de cada uma das rodas e os valores acumulados das leituras dos encoders, as figuras seguintes apresentam os resultados obtidos.

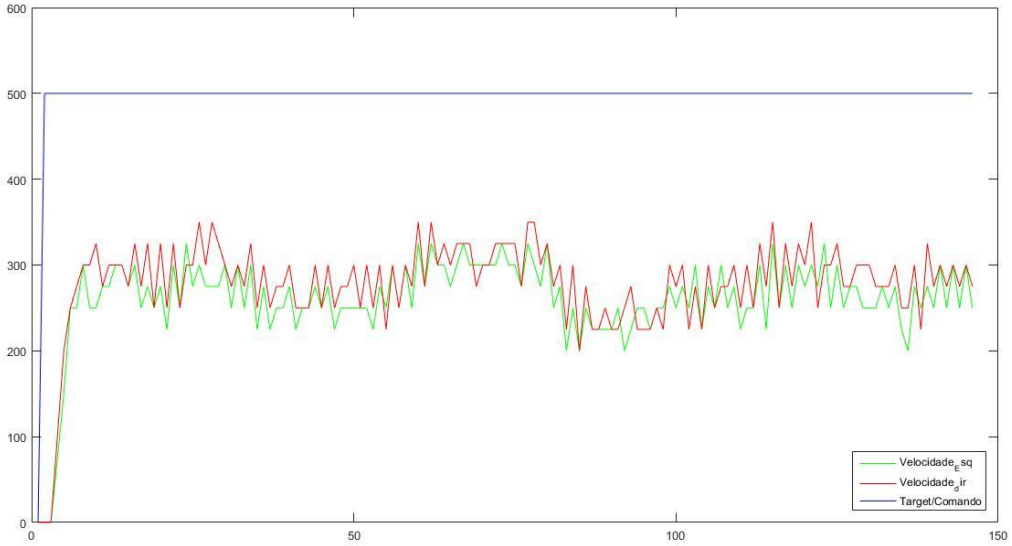

Figura 6.5 - Representação gráfica das velocidades das rodas e target (sem compensação)

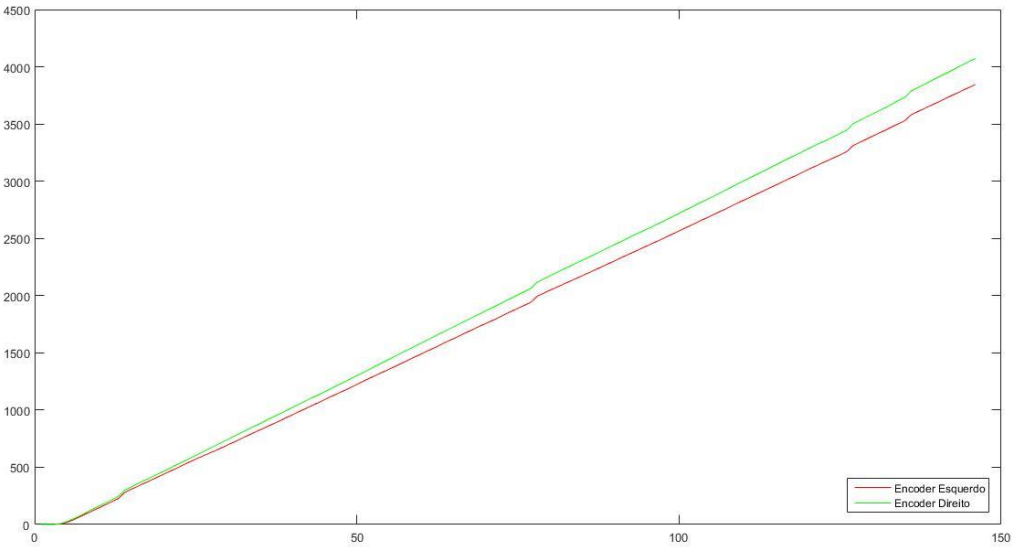

Figura 6.6 - Representação gráfica das leituras acumuladas dos encoders (sem compensação)

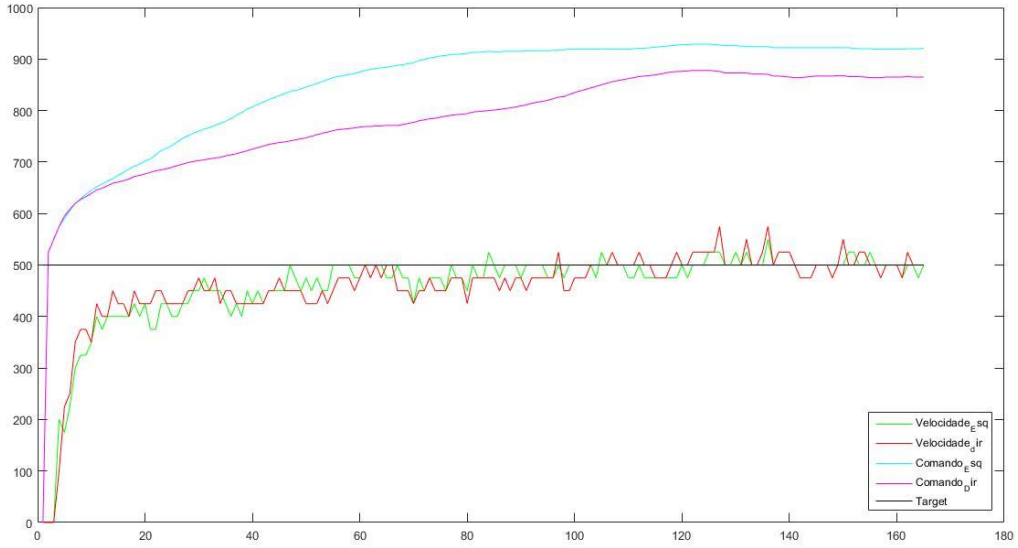

Figura 6.7 - Representação gráfica das velocidades, target e comandos (compensação activa)

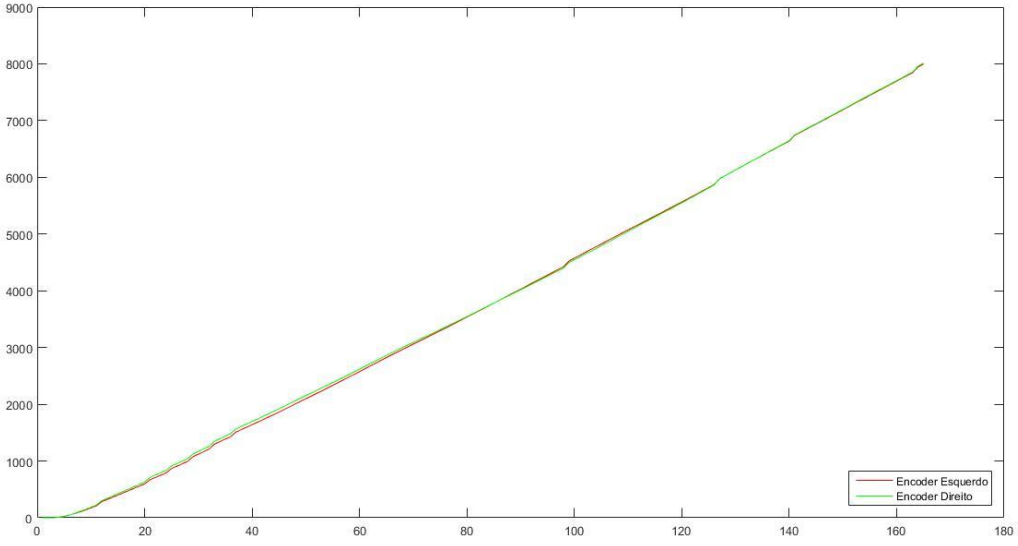

Figura 6.8 - Representação gráfica das leituras dos encoders (compensação activa)

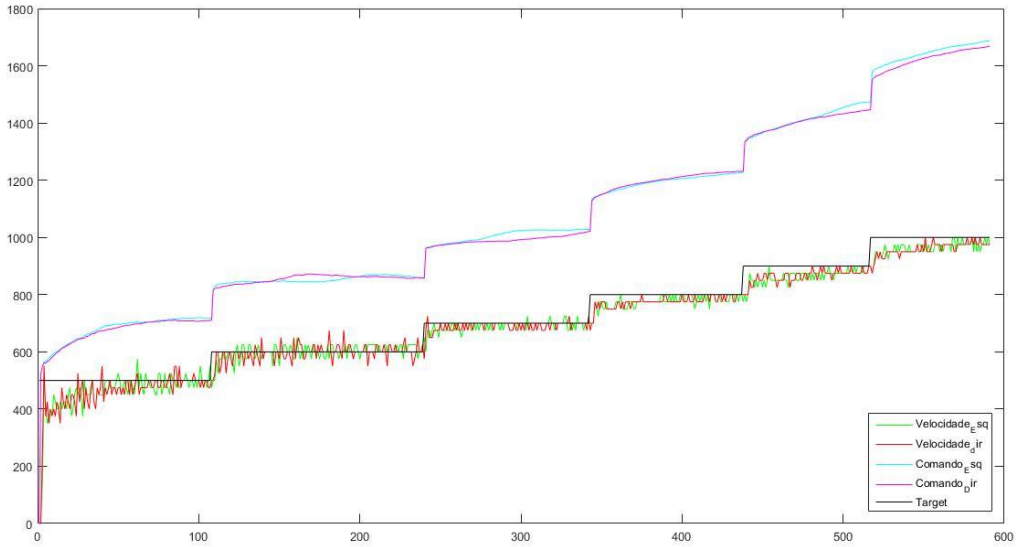

Figura 6. 9 - Representação gráfica das velocidades das rodas, comandos enviados e target (compensação activa, robô em vazio)

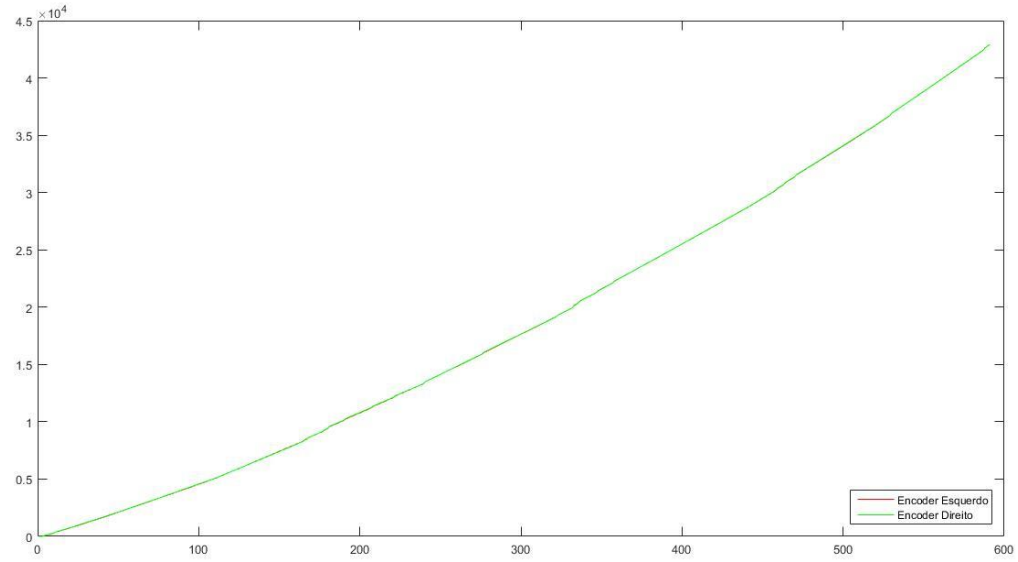

Figura 6. 10 - Representação gráfica das leituras acumuladas dos encoders (compensação activa, robô em vazio)

Como é visível nas figuras anteriores o algoritmo desenvolvido introduz grandes melhorias quer no seguimento do target quer na correcção das diferenças entre as velocidades das rodas.

6.3 Navegação

Para validação do algoritmo de navegação e respectivas alterações foram realizados três ensaios. Um ensaio com uma sequência de navegação gravada na memória e dois ensaios com duas sequências gravadas na memória, assumindo para mudança de sequência 20 e 100 predições consistentes respectivamente.

Os ensaios foram realizados entre o Laboratório CAD/CAE (R1.2), o gabinete 1.1 e as escadas da torre R.

As figuras seguintes ilustram os resultados obtidos para os 3 cenários descritos.

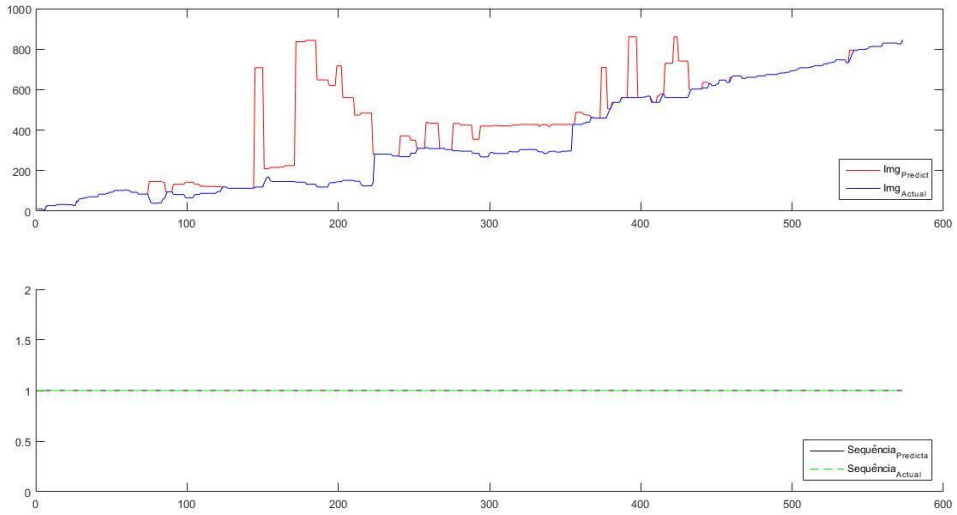

Figura 6.11 - Representação gráfica da evolução das imagens preditas, actuais, sequência predita e sequência actual para a navegação com apenas uma sequência

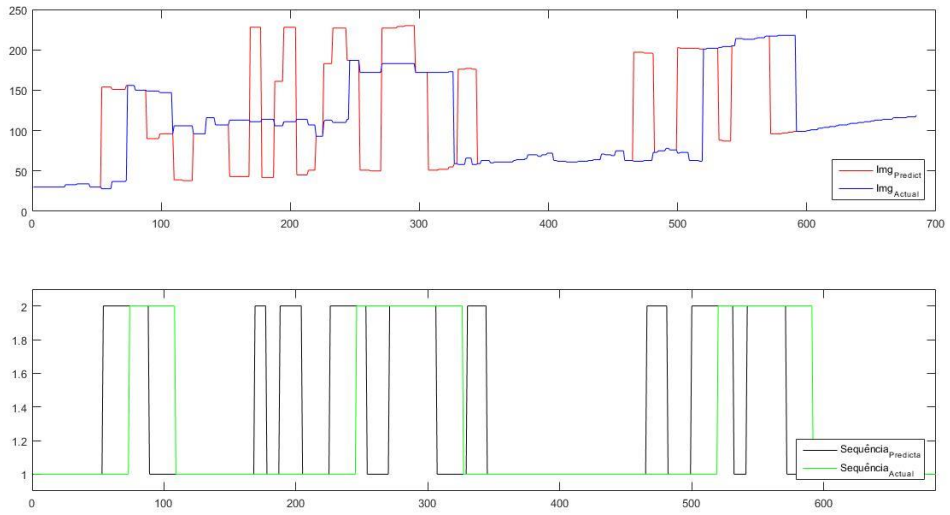

Figura 6.12 - Representação gráfica da evolução das imagens preditas, imagens correntes, sequências preditas e sequências actuais, mudança de sequência após 20 predições sucessivas e consistentes

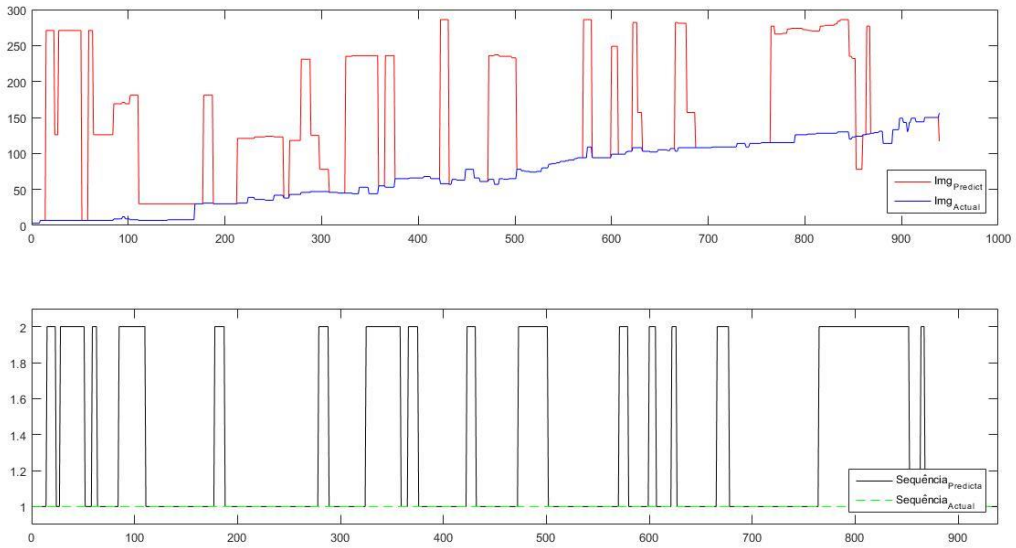

Figura 6.13 - Representação gráfica da evolução das imagens preditas, imagens correntes, sequências preditas e sequência actuais, mudança de sequência após 100 predições sucessivas e consistentes

O segundo e terceiro cenários apresentam uma taxa de predições com sucesso (na sequência desejada) de 70% e 69%, respectivamente, isto é, relativamente às predições com sucesso não existem alterações significativas. No entanto no último cenário a evolução das imagens preditas é muito mais linear e não apresenta mudanças de sequência, assim sendo podemos afirmar que no caso dos percursos analisados o algoritmo de navegação torna-se mais robusto quando o limiar para mudança de sequência é aumentado.

7.Conclusão e Trabalho Futuro

Neste capítulo são apresentadas as conclusões desta dissertação e é sugerido algum trabalho futuro.

7.1 Conclusão

Face aos constrangimentos detectados e à alteração dos objectivos daí resultante, é possível afirmar, após a realização deste trabalho, que todos os novos objectivos propostos foram cumpridos.

A interface Web renovada permite uma melhor interacção com o utilizador e uma utilização mais fácil e intuitiva do robô. Permite também a inclusão de novas funcionalidades com relativa facilidade. A implementação de dois modos de funcionamento permite uma melhor gestão e controlo da utilização do robô.

As correcções ao algoritmo de navegação apresentam bons resultados, o robô identifica os finais dos percursos e muda de sequência correctamente. As leituras dos sensores de proximidade de infravermelhos apresentam agora valores precisos e coerentes.

A auto localização funciona como previsto, no entanto apresenta constrangimentos devido à forma como foi implementada. O robô identifica o local onde se encontra de forma correcta, contudo, no caso do uso deste recurso na tarefa Encaminhar Até, caso o local seja comum a mais que um percurso e não seja origem desses mesmos percursos, apenas é indicado ao utilizador um destino possível de acordo com o número de sequência da imagem retornada pelo software.

O controlo da velocidade apresenta bons resultados e permite um controlo mais preciso do robô no processo de ensino, contudo o robô ainda apresenta uma ligeira tendência para fugir para a direita, possivelmente devido a pequenas diferenças no perímetro das rodas.

Tendo em conta o exposto e os resultados apresentados, conclui-se que as soluções desenvolvidas são boas e suficientemente robustas, ainda assim susceptíveis de melhoria.

7.2 Trabalho Futuro

No que concerne ao trabalho futuro, ficam em aberto as seguintes possibilidades:

Percursos:

No estado actual qualquer percurso ensinado ao robô é guardado em disco na sua totalidade. Isto implica, em algumas circunstâncias, o uso de espaço em disco desnecessário, além de aumentar o número de comparações a realizar aquando do carregamento de imagens na SDM. Deste modo torna-se útil um mecanismo que permita lidar com conjuntos de imagens comuns a vários percursos ensinados e permita descartar imagens com grande similaridade.

Pode também ser feita uma representação topológica dos percursos e construída uma matriz de adjacências de modo fundir percursos adjacentes.

De seguida apresentam-se alguns problemas relativos aos percursos com troços comuns, bem como uma possível abordagem para mitigar os problemas relatados.

Situações e problemas a resolver:

Percursos com troços comuns:

a) Saída dos troços comuns;
b) Destino pretendido quando o robô se localiza inicialmente dentro de um troço comum.

Possíveis soluções:

a) A saída de um troço comum levanta o problema de indicar ao robô qual a sequência a seguir. Se o número de sequência do troço comum é distinto do número de sequência de qualquer percurso a realizar basta indicar ao robô, que após um dado número de predições fora da sequência correspondente ao troço comum a sequência pretendida é a sequência correspondente ao percurso seleccionado pelo utilizador. Se o numero de sequência do troço comum é o número de sequência de um percurso gravado que contém esse troço, então, é possível uma abordagem semelhante à anterior, no entanto neste caso o número de predições fora da sequência correspondente ao troço comum, indicativas de um ponto de divergência, teria de ser muito menor que no caso anterior, pois se assim não fosse o robô seguiria o percurso ao qual corresponde o número de sequência do troço comum.

b) A escolha do destino quando o robô se localiza inicialmente num troço comum a vários percursos resolve-se facilmente caso os troços comuns tenham números de sequência distintos, assim basta que exista na base de dados uma tabela que associe a cada um desses números de sequência o número de sequência dos vários percursos (ID_Percurso) a que cada um dos segmentos é comum. Realizando uma pesquisa nesta tabela, usando o número de sequência devolvido pelo software de controlo aquando da localização do robô, é possível obter os ID_Percurso dos percursos a que o troço onde se encontra o robô é comum. Realizando uma pesquisa na tabela Percursos com esses mesmos ID's é possível mostrar ao utilizador os possíveis destinos, deixando ao seu critério a escolha do mesmo.

   No caso de o número de sequência do troço comum ser igual ao de um dado percurso, uma abordagem semelhante é válida, no entanto, quando o robô se localiza existe o problema de saber se o robô efectivamente se encontra num troço comum a vários percursos ou num qualquer ponto do percurso com o mesmo ID.

Missões:

Podem ser implementadas as seguintes missões:

- Guia Turístico
- Entregar Objectos
- Dar Recado
- Telepresença

No caso da missão "Guia Turístico" será útil, como já referido no contexto dos percursos, ter uma abordagem que permita fundir os percursos existentes.

A missão dar recado funcionaria tanto como complemento à missão entregar objecto, como de forma independente.

Navegação

Relativamente à navegação, um dos problemas verificados relaciona-se com mudanças de iluminação, isto é, o algoritmo existente é demasiado sensível a alterações nas condições de iluminação do percurso, no momento em que é realizado, quando comparadas com o momento em que o percurso é ensinado. Deste modo torna-se pertinente a implementação/adaptação de uma qualquer abordagem que permita lidar com este problema. Uma possível abordagem é apresentada em [17].

Poderá ainda ser implementado um método de navegação que permita ao robô navegar num percurso no sentido contrário àquele que é ensinado. Tal poderia ser feito por exemplo, mantendo a câmara com a mesma orientação e assumindo os valores de velocidade ensinados, mas com sinal negativo.

Referências:

[1] “Peeper.” [Online]. Available: https://www.ald.softbankrobotics.com/en/cool-robots/pepper. [Accessed: 10-Oct-2016].

[2] “Socrob-ISR-IST.” [Online]. Available: http://socrob.isr.tecnico.ulisboa.pt/dokuwiki/doku.php. [Accessed: 10-Oct-2016].

[3] “Página do Projecto MOnarCH.” [Online]. Available: http://monarch-fp7.eu/pt/. [Accessed: 10-Oct-2016].

[4] “Care-O-Bot-4.” [Online]. Available: http://www.care-o-bot-4.de/. [Accessed: 11-Oct-2016].

[5] “REEM.” [Online]. Available: http://pal-robotics.com/en/products/reem/. [Accessed: 11-Oct-2016].

[6] J. Borenstein and Y. Koren, “Real-time obstacle avoidance for fast mobile robots,” *IEEE Trans. Syst. Man Cybern.*, vol. 19, pp. 1179–1187, 1989.

[7] A. Fujimori, P. N. Nikiforuk, and M. M. Gupta, “Adaptive navigation of mobile robots with obstacle avoidance,” *IEEE Trans. Robot. Autom.*, vol. 13, no. 4, 1997.

[8] A. Ferreira, F. G. Pereira, R. F. Vassallo, T. F. Bastos Filho, and M. Sarcinelli Filho, “An approach to avoid obstacles in mobile robot navigation: the tangential escape,” *Sba Control. Automação Soc. Bras. Autom.*, vol. 19, no. 4, pp. 395–405, 2008.

[9] J. Borenstein and Y. Koren, “Obstacle avoidance with ultrasonic sensors,” *Robotics and Automation, IEEE Journl*, vol. 4. pp. 213–218, 1988.

[10] J. Caceiro, “Robô autónomo estafeta e Guia,” Dissertação de Mestrado, Departamento de Engenharia Electrotécnica e de Computadores, Universidade de Coimbra, 2014.

[11] “Dr.Robot X80Pro.” [Online]. Available: http://www.drrobot.com/products_item.asp?itemNumber=x80.

[12] “Asus K55VJ, página do fabricante.” [Online]. Available: https://www.asus.com/Notebooks/K55VJ/.

[13] A. Brandão, “Robô Autónomo Guia e para Vigilância Automática,” Dissertação de Mestrado, Departamento de Engenharia Electrotécnica e de Computadores, Universidade de Coimbra, 2013.

[14] “Sharp GP2Y0A02YK0F Datasheet.” [Online]. Available: http://www.sharpsma.com/webfm_send/1487.

[15] “WiRobot SDK Application Programming Interface (API) Reference Manual -(For MS Windows),” 2010.

[16] “Using Motor Calibration Board (P/N: MCB2005) for X80 series robots.”

[17] N. Winters, J. Gaspar, G. Lacey, and J. Santos-Victor, “Omni-directional Vision for Robot Navigation.”

[1]–[17]

Anexo A - Instruções para ligação do robô

1. Conectar o robô ao pc;
2. Ligar o robô e esperar que este arranque;
3. Verificar no gestor de dispositivos qual a porta série onde está ligado o adaptador usb-série de ligação ao robô (profilic-usb to serial);

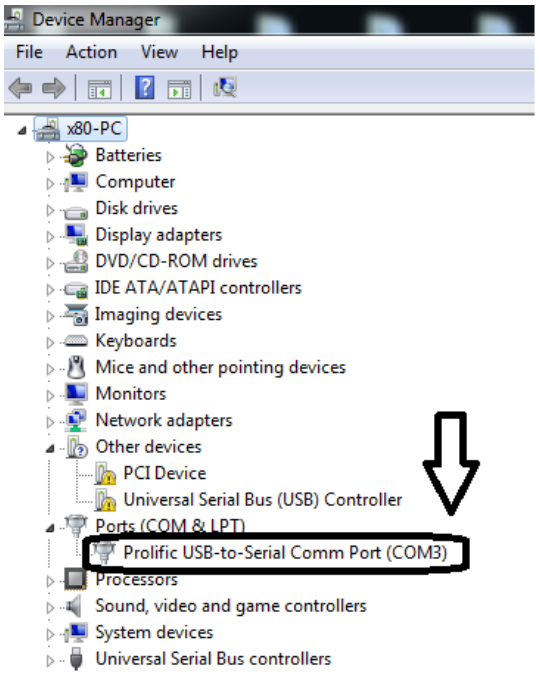

Figura A.1 - Gestor de dispositivos com o adaptador usb-serial identificado

4. Executar o WirobotGateway e configurar o mesmo com a porta série obtida no ponto anterior;

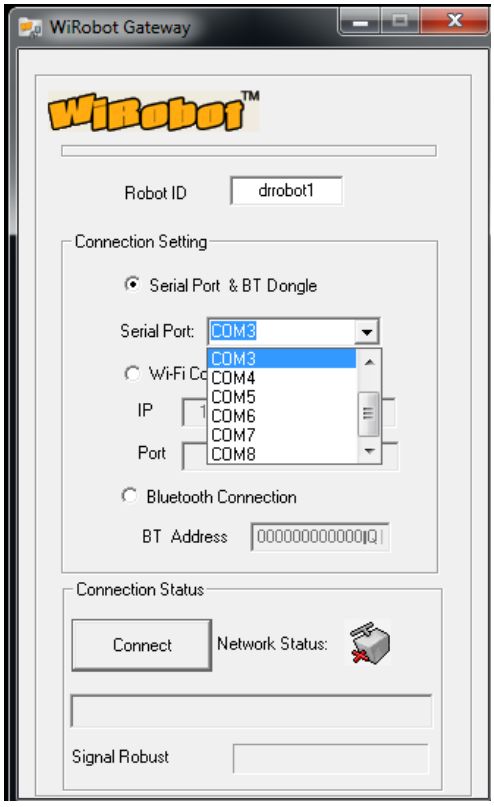

Figura A.2(a) - WirobotGateway, selecção de porta.

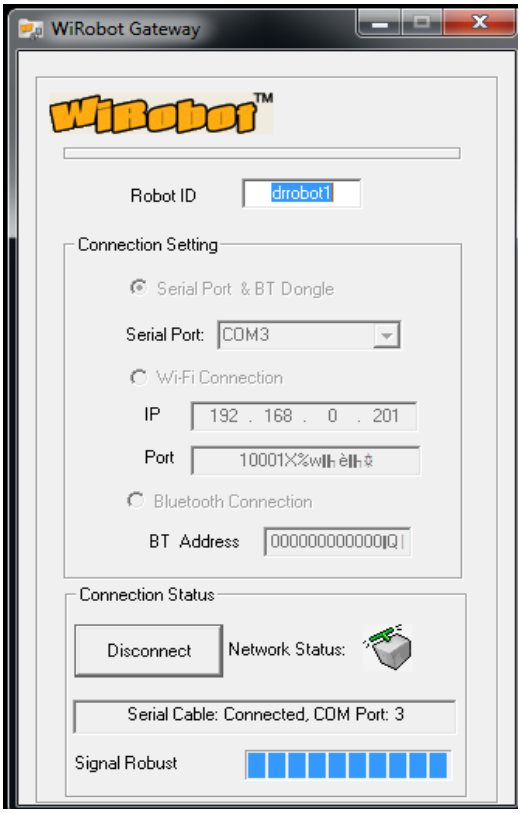

Figura A.2(b) - Wirobot conectado

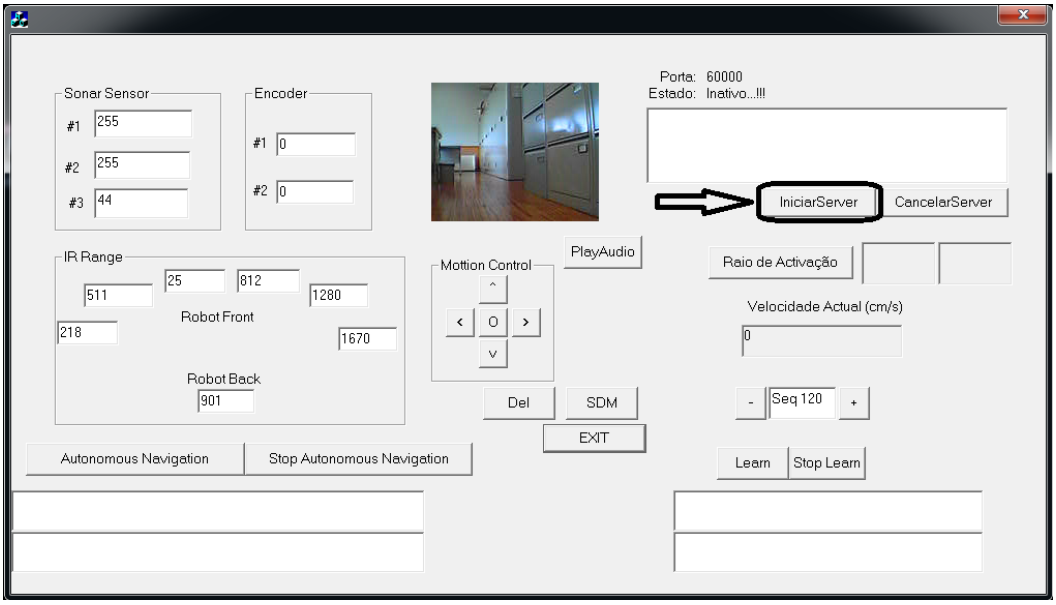

Figura A.3 - Vista do software de controlo.

5. Abrir e executar o software de controlo e se necessário iniciar o servidor do socket;
6. Iniciar Servidor, Wamp Server;

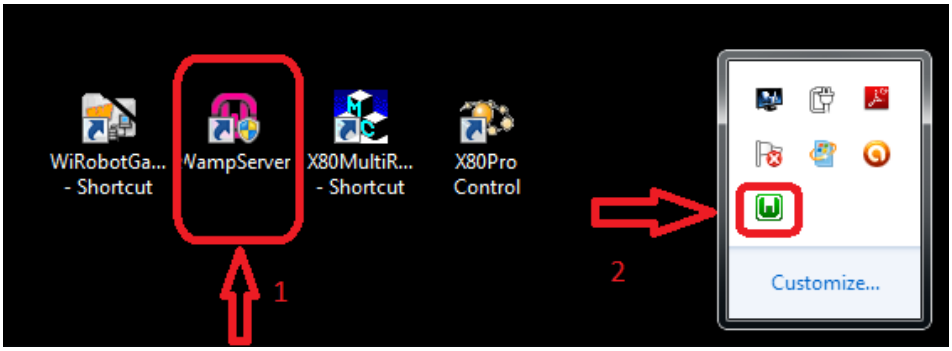

Figura A.4 - Passos para ligação do Wamp Server

7. Aceder à interface gráfica e realizar o login com os respectivos dados de utilizador.

Figura A.5 - Vista da interface com o utilizador.

Nota1: A tensão aos terminais da bateria não deve ser inferior a 7,2 Volts. Após o passo 2 caso o robô não apresente no visor lcd a tensão aos terminais da bateria active essa opção na interface da aplicação X80Pro Control. Se tal não for possível desligue o robô e verifique esse valor com o auxílio de um voltímetro.

Nota2: A aplicação WirobotGateway deve ser executada em modo de compatibilidade "Windows xp Service Pack 2" ou "Windows xp Service Pack 3".

Nota3: As instruções apresentadas neste documento reportam-se ao sistema operativo "Windows7"

Anexo B - Protocolo de comunicação entre o software de controlo e o servidor web

O protocolo de comunicação utilizado consiste num protocolo de arquitectura cliente-servidor implementado através de um socket TCP. Neste caso concreto o software de controlo funciona como servidor e o servidor web como cliente.

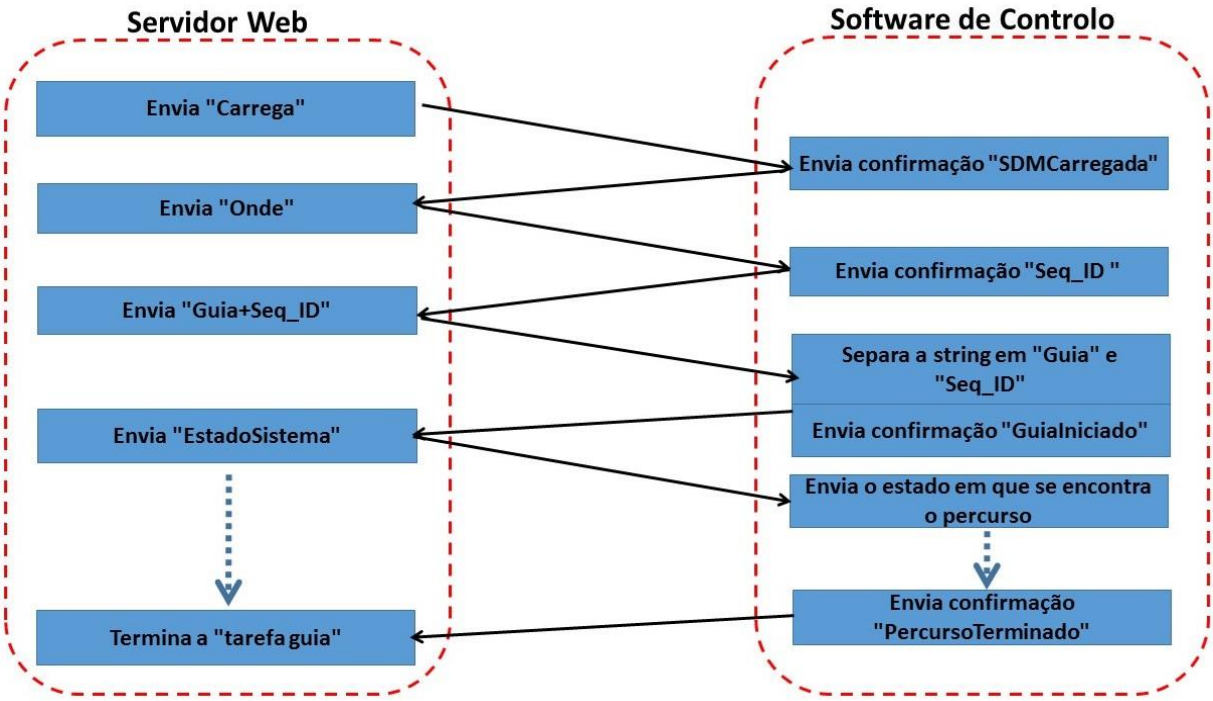

Figura B.1 - Exemplo de utilização do protocolo descrito: sequência de mensagens trocadas entre o servidor web e o software de controlo para realização da tarefa guia, adaptado de J.Caceiro[10].

O protocolo funciona da seguinte forma, o cliente envia ao servidor uma string constituída por um ou dois elementos separados pelo caracter "+" (comando e número do caminho a ensinar ou percorrer), o servidor recebe a string, separa os elementos e compara o a primeira substring com os comandos possíveis, apresenta-se à frente uma tabela com comandos existentes e respectivas respostas e descrições:

Tabela B.1- Comandos existentes, resposta gerada e respectiva descrição

Comando	Resposta	Descrição
Carrega	“SDMCarregada”	Carrega na SDM todos os percursos existentes na base de dados.
Centra	Velocidade da roda esquerda e velocidade da roda direita	Impõe a ambas as rodas uma velocidade de 500 unidades
Controlo	Velocidade da roda esquerda, velocidade da roda direita e leituras dos sonares	Inicia controlo manual.
Direita	Velocidade da roda esquerda, velocidade da roda direita e leituras dos sonares	Incrementa a velocidade de roda esquerda em 30 unidades e decrementa a velocidade da roda direita em 30 unidades
DireitaMais	Velocidade da roda esquerda, velocidade da roda direita e leituras dos sonares	Incrementa a velocidade de roda esquerda em 90 unidades e decrementa a velocidade da roda direita em 90 unidades
Ensinar	“PodeIniciarEnsino”	Inicia o ensino de um novo percurso.
Esquerda	Velocidade da roda esquerda, velocidade da roda direita e leituras dos sonares	Decrementa a velocidade de roda esquerda em 30 unidades e incrementa a velocidade da roda direita em 30 unidades
EsquerdaMais	Velocidade da roda esquerda, velocidade da roda direita e leituras dos sonares	Decrementa a velocidade de roda esquerda em 90 unidades e incrementa a velocidade da roda direita em 90 unidades
EstadoSistema	Qualquer uma das respostas listadas ou “TerminouPercurso” no caso de ter terminado a navegação autónoma	Actualiza o estado do sistema
Frente	Velocidade da roda esquerda,velocidade da roda direita e leituras dos sonares	Incrementa a velocidade de ambas as rodas em 30 unidades
FrenteMais	Velocidade da roda esquerda, velocidade da roda direita e leituras dos sonares	Incrementa a velocidade de ambas as rodas em 120 unidades
Guia	“IniciarGuia”	Inicia a realização de tarefa guia
Navegação	“aNavegar”	Inicia navegação autónoma.

Onde	Vector de dados da imagem mais próxima da imagem actual	Pede ao robô a sua localização actual
Para	Velocidade da roda esquerda, velocidade da roda direita e leituras dos sonares	Pára o movimento do robô.
ParарEnsinar	“ParadoEnsinar”	Termina o ensino de um dado percurso.
ParарControlo	“ParadoControlo”	Termina controlo manual
SDM	“carregadaSDM”	Carrega na SDM a sequência de imagens correspondente ao percurso indicado pelo comando guia.
Tras	Velocidade da roda esquerda, velocidade da roda direita e leituras dos sonares	Decrementa a velocidade de ambas as rodas em 30 unidades
TrasMais	Velocidade da roda esquerda e velocidade da roda direita	Decrementa a velocidade de cada uma das rodas em 90 unidades
Verifica	Leitura do sensor de IR traseiro sem pré-processamento.	Verifica de o robô está ligado

Nota: A velocidade é representada em pulsos por segundo.

Anexo C - Algoritmo de navegação - métodos e variáveis envolvidas

O algoritmo de navegação implementado é constituído pelas seguintes rotinas (métodos):

- OnMotorSensorEventDrrobotsdkcontrolctrl1() – método de serviço ao evento gerado pelo robô para leitura dos dados de velocidade dos motores das rodas esquerda e direita.
- OnAutonomousNavigation() - Método invocado no início da navegação autónoma de um percurso para inicialização dos ficheiros de log e variável de estado *Autonomo* para sinalizar que o robô se encontra no modo de navegação autónoma.
- NavegacaoAutonoma() - Método que executa o algoritmo de navegação autónoma descrito no capítulo 4.
- OnStopAN() - Método no fim do percurso percorrido autonomamente que fecha os ficheiros de log e altera a variável de estado *Autonomo* para sinalizar que o robô se encontra em espera.

A figura seguinte representa a o fluxograma das rotinas envolvidas no algoritmo de navegação, assim, sempre que o socket recebe um comando para iniciar a navegação autónoma, é chamada a rotina *OnAutonomousNavigation*, esta inicializa os logs da navegação e sinaliza o seu início colocando a 1 a variável *Autonomo.* Aquando da ocorrência do evento de serviço associado aos motores, considerando o estado da variável *Autonomo*, é invocada a função *NavegacaoAutonoma*. Quando esta termina é colocada a 1 a variável *terminou*, é invocada a função *OnStopAN,* os logs são fechados e a variável *Autonomo* é colocada a 0.

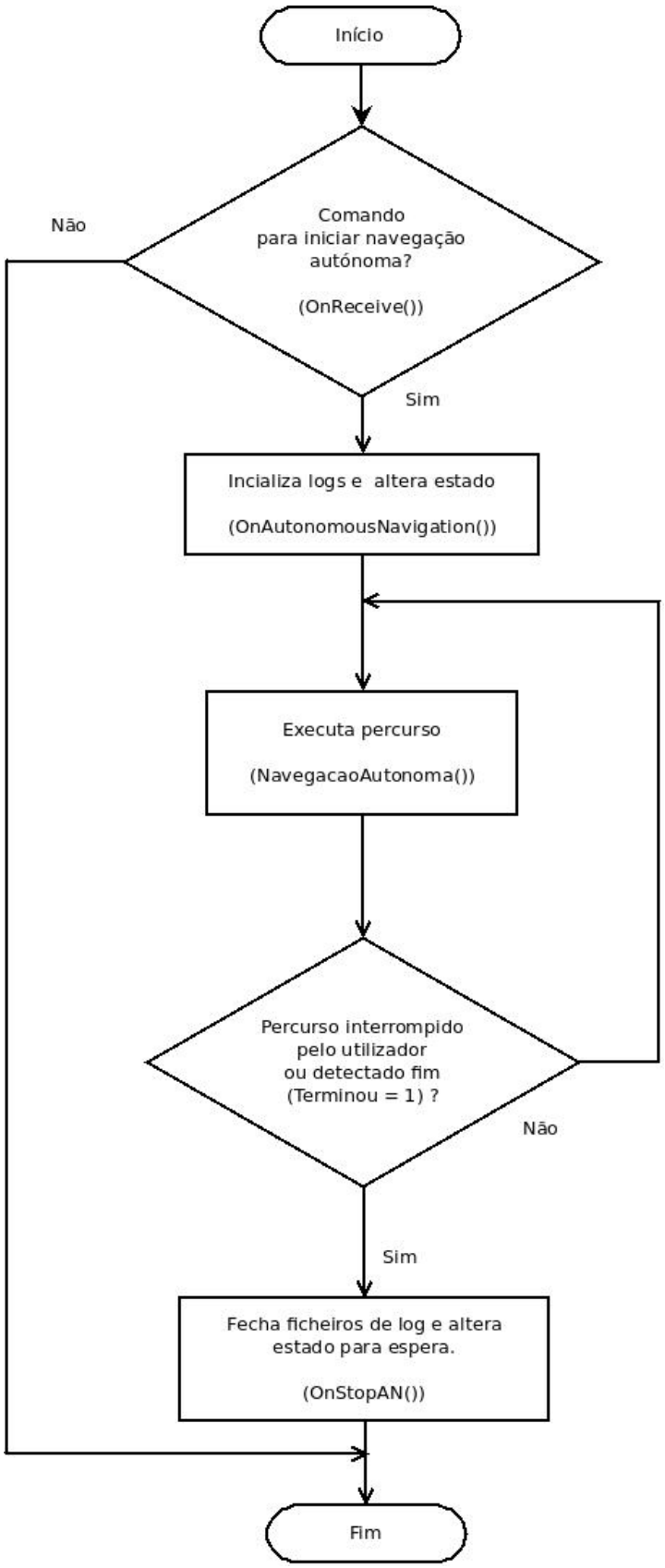

Figura C.1 - Fluxograma das rotinas envolvidas no algoritmo de navegação

A tabela seguinte indica as variáveis mais importantes ao funcionamento do algoritmo e a sua função.

Tabela C.1 - Principais variáveis envolvidas no algoritmo de navegação

Principais Variáveis			
Autonomo	Variável de estado que indica que robô se encontra em navegação autónoma.	CountPredict	Nº de predições
SeqAct	Indica a sequência devolvida pela predição actual	Terminou	Sinaliza o fim da navegação autónoma
Asec	Indica a sequência devolvida pela predição anterior	Vect_e	Vector que contém as distâncias de cada imagem à imagem actual (predita)
Seqs	Vector com os vários nºs de sequência gravados	Count4	Indica o nº de predições fora da janela deslizante
Nimgclue	Indica o nº de ordem da imagem devolvida pela predição	Count11	Indica o nº de predições fora da sequência actual
NImgclueAnterior	Indica o nº de imagem corrente	Extra	Log das imagens preditas
NimgclueJ	Indica o nº de ordem de imagem mais próxima da imagem devolvida pela predição, dentro da janela deslizante	Predictlog	Log do vector de dados associado às predições
Final	Vector que indica o nº de ordem da imagem final de cada um dos percursos.	FPredict	Log da imagem corrente
window	Dimensão da janela deslizante	Vesq	Velocidade da roda esquerda
CaminhoU	Indica o percurso actual	Vdir	Velocidade da roda direita
Dmin	Nº de ordem da imagem mais próxima dentro da janela deslizante	Seq_Count	Indica o total de sequências gravadas

Anexo D – Diagramas de Casos de Uso

Este anexo apresenta os diagramas de casos de uso referentes a utilizadores registados (administrador), bem como a utilizadores comuns (visitante). No caso dos utilizadores registados, com excepção do caso "Escolhe Língua", os casos de uso apresentados estão dependentes dos privilégios atribuídos ao utilizador.

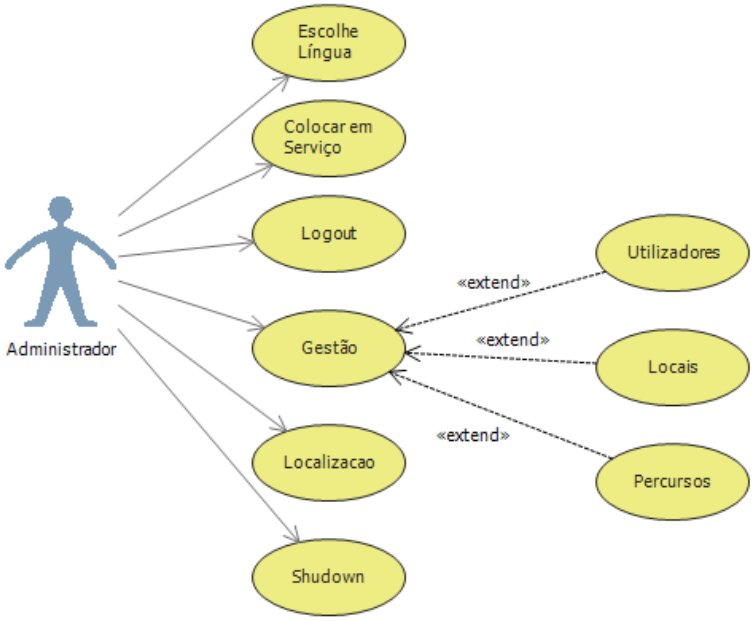

Figura D.1 - Diagrama de casos de uso referente a um utilizador registado

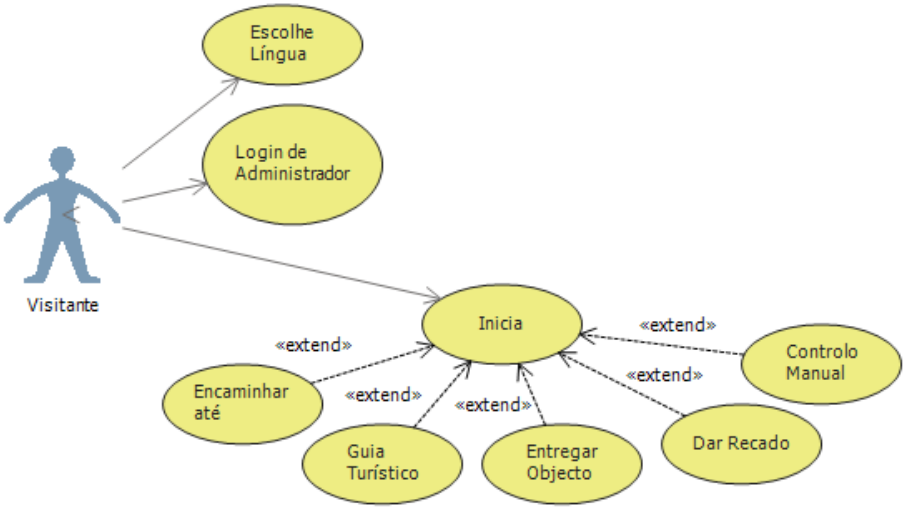

Figura D.2 - Diagrama de casos de uso referente a um utilizador comum