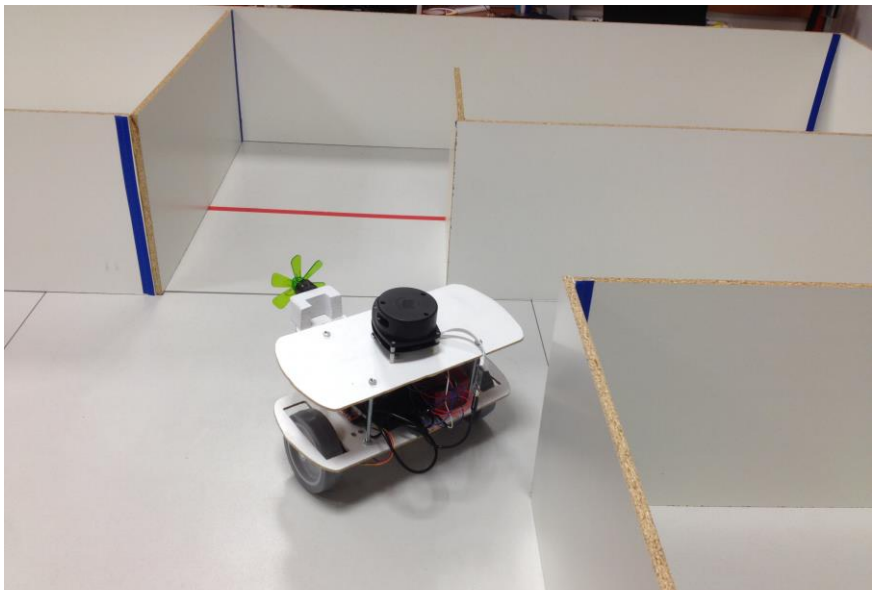
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia Mecânica

ISEL



Desenvolvimento de um Robô para Combate a Fogos num Cenário de Simulação

MIGUEL ÂNGELO GALEGO MARQUES
(Licenciado em Engenharia Mecânica)

Trabalho Final de Mestrado para obtenção do grau de Mestre em Engenharia Mecânica

Orientador (es): Doutor João Manuel Ferreira Calado
Mestre Fernando Paulo Neves da Fonseca Cardoso Carreira
Doutor Francisco Mateus Marnoto de Oliveira Campos

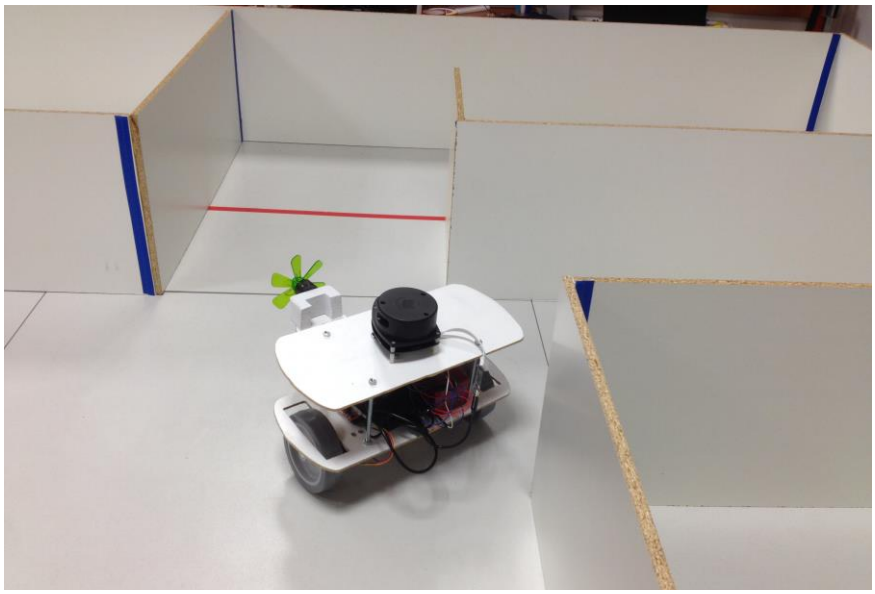
Júri:
Presidente: Doutora Maria Teresa Moura e Silva
Vogais:
Doutor Carlos Baptista Cardeira
Doutor Francisco Mateus Marnoto de Oliveira Campos

Janeiro de 2017

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia Mecânica

ISEL



Desenvolvimento de um Robô para Combate a Fogos num Cenário de Simulação

MIGUEL ÂNGELO GALEGO MARQUES
(Licenciado em Engenharia Mecânica)

Trabalho Final de Mestrado para obtenção do grau de Mestre em Engenharia Mecânica

Orientador (es): Doutor João Manuel Ferreira Calado
Mestre Fernando Paulo Neves da Fonseca Cardoso Carreira
Doutor Francisco Mateus Marnoto de Oliveira Campos

Júri:
Presidente: Doutora Maria Teresa Moura e Silva
Vogais:
Doutor Carlos Baptista Cardeira
Doutor Francisco Mateus Marnoto de Oliveira Campos

Janeiro de 2017

Dedicatória

…Aos meus Pais, pelo apoio incondicional

Agradecimentos

A presente dissertação resulta do meu esforço pessoal, em que o auxílio que me foi prestado teve um papel bastante importante, por isso, gostaria de aqui expressar o meu reconhecimento e sinceros agradecimentos a todos os que me ajudaram a realizar este projeto.

A nível profissional, começo por agradecer aos meus orientadores, Prof. João Calado, Prof. Francisco Campos e Prof. Fernando Carreira pelo auxílio prestado ao longo do ano. Aos meus colegas de grupo da Unidade Curricular de Robótica Industrial ao qual pertenci e que me auxiliaram no projeto relacionado com a deteção da chama, aos meus colegas de curso Alexandre Lobo e Ricardo Graça, pela disponibilidade demonstrada.

Por fim, e não menos importante, quero agradecer à minha família e namorada pelo apoio incondicional dado ao longo destes cinco anos de estudos.

Resumo

Na presente dissertação aborda-se o desenvolvimento de um robô móvel demonstrativo das funcionalidades necessárias ao combate a incêndios. Em resumo, o robô deve ter a capacidade de explorar um ambiente desconhecido, com vista a detetar a zona de incêndio e extinguir o foco de incêndio por meios mecânicos.

Para tal, recorreu-se a um sistema de navegação autónomo sustentado na fusão sensorial entre um sensor *rangefinder* e a odometria. Pretende-se que este sistema conduza um robô móvel através de um labirinto (que simula uma habitação) explorando-o na totalidade, ao mesmo tempo que evita embater nas paredes e noutros obstáculos.

Para além da navegação, existem outros sistemas importantes no robô, como o sistema de deteção da chama e o sistema para a sua extinção. O primeiro utiliza como sensor uma câmara termográfica que permite detetar a presença da chama. No sistema de extinção, é utilizada uma ventoinha que será acionada com base na informação recebida por parte da câmara termográfica.

Palavras-chave:

Robótica Móvel

Combate a Incêndios

Diagrama de Voronoi

Exploração baseada em Fronteiras

Rangefinder

Câmara termográfica

Abstract

The present dissertation addresses the development of a mobile robot that is demonstrative of the features needed for firefighting. In short, the robot should be able to explore an unfamiliar environment, with the goal of detecting the zone of fire and extinguishing the fire outbreak by mechanical means.

To this end, we developed an autonomous navigation system based on the sensory fusion of a rangefinder sensor and odometry. This system is intended to drive a mobile robot through a maze (which simulates a housing) to fully explore it, while avoiding the walls and other obstacles.

Besides navigation, there are other important systems in the robot, namely the flame detection and the extinction systems. The first uses a thermal camera as the sensor to detect the existence of a flame. The extinguishing system comprises, a fan that is triggered by the information of the thermal camera.

Keywords:

Mobile Robotics

Firefighting

Voronoi Diagram

Frontier-based exploration

Rangefinder

Thermal camera

Lista de Símbolos

Símbolos relativos ao Sistema de Locomoção

A_i	Área referente ao elemento i
x_i	Centróide referente ao elemento i
x_c	Coordenada x do centroide

Símbolos relativos ao Algoritmo *Split and Merge*

d_i	Distância do ponto à reta que melhor se ajusta ao conjunto de pontos
E_{fit}	Erro associado ao ajuste da reta ao conjunto de pontos
S_{x^2}, S_{y^2}, S_{xy}	Parâmetros referentes ao método dos mínimos quadrados
$\bar{x}$	Média ponderada dos valores de x
$\bar{y}$	Média ponderada dos valores de y
θ_i	Ângulo entre o vetor da distância da origem O ao ponto i e o referencial x
ρ	Distância normal entre origem O e reta que melhor se aproxima ao conjunto de pontos
ϕ	Ângulo entre ρ e o referencial x

Símbolos relativos à Cinemática do Robô Móvel

e	Erro de postura
K	Matriz de ganho
L	Distância entre rodas motoras do robô

P	Posição genérica do robô
r	Raio da roda motora
$R(\theta)$	Matriz de rotação ortogonal
s_k	Deslocamento realizado pelo ponto central do robô
t	Tempo
T	Tempo de discretização
v	Velocidade linear do robô
X_I, Y_I	Eixos de inércia globais
X_R, Y_R	Eixos de referência locais do robô, centrados em P
$x(t)$	Deslocamento efetuado na direção x
$\dot{x}(t)$	Velocidade linear na componente x
$y(t)$	Deslocamento efetuado na direção y
$\dot{y}(t)$	Velocidade linear na componente y
α	Ângulo entre eixo local do robô X_R e um eixo de inércia global X_I
β	Ângulo entre vetor ρ e o eixo de inércia global X_I
γ	Ângulo de viragem do robô
θ	Ângulo de orientação do robô com um eixo de inércia global
$\boldsymbol{\xi}$	Vetor de postura do robô (posição e ângulo)

ρ	Distância euclidiana entre a posição P do robô e a posição desejada
φ	Ângulo de rotação da roda motora
ω	Velocidade angular do robô

Símbolos relativos ao Diagrama de Voronoi Generalizado

$\mathfrak{F}$	Região de Voronoi
$\mathcal{O}_i$	Obstáculo
q	Ponto de coordenadas genéricas
$\mathcal{Q}_i$	Espaço dos obstáculos
$\mathcal{Q}_{free}$	Espaço livre
$R(q)$	Posição do robô
S_{ij}	*Two-equidistant surface*
$\mathcal{W}$	Espaço de trabalho (*workspace*)

Símbolos relativos ao Algoritmo ICP (*Iterative Closest Point*)

d_{max}	Valor de *threshold* máximo para coincidir pontos
T	Matriz de transformação final
T_0	Matriz de transformação inicial
$\bar{t}$	Translação relativa entre duas nuvens de pontos
R	Rotação relativa entre duas nuvens de pontos

U, V	Matrizes unitárias resultantes da operação de decomposição em valores singulares
ξ	Estimação da postura relativa entre duas nuvens de pontos

Símbolos relativos ao Algoritmo de Dijkstra

$PERM$	*Array* dos vértices que são considerados permanentes
$PRED$	*Array* da sequência de registo dos vértices permanentes
$w(i,j)$	Peso associado entre os vértices i e j
$\lambda(i)$	Estimativa do peso associado entre os vértices i e 1

Símbolos relativos ao Algoritmo de Exploração e Mapeamento

A_{ij}	Matriz de coeficientes da transformação afim
b	Vetor de informação
C_k	Conjunto dos *scans* com os quais se realiza a fusão do *scan* k
d_{ij}	Diferença entre p_i e p_j
e_{ij}	Erro entre uma medição de postura do robô e a diferença entre estados estimados
$F_x \quad F_v$	Jacobianos da cinemática direta do robô relativos às variáveis de estado e de atuação respetivamente
H	Matriz de informação
J_{ij}	Jacobiano de e_{ij}
$P_i \quad P_j$	Vetores posição do robô nas iterações i e j
P_{ij}	Matriz de covariância

S^0	Conjunto de pontos obtidos numa leitura do *rangefinder* no referencial do mundo
S_k^0	Novo *scan* obtido em cada iteração k do algoritmo de exploração
S^r	Conjunto de pontos obtidos numa leitura do *rangefinder* no referencial do robô
$SE(2)$	Grupo das transformações no espaço euclidiano de dimensão 2
T_{ik}	Matriz de transformação que alinha o *scan* i com o *scan* k pelo método ICP
th_f	Incerteza máxima admissível numa direção
x	Vetor de estados
$\hat{x}$	Variável multidimensional dos estados estimados
Δx	Variação aplicada ao vetor de estado estimado
λ_{i1} λ_{i2}	Valores próprios da matriz de covariância
$\sum$	Matriz de covariância global
Ω_{ij}	Matriz de informação relativa à medição

Nomenclatura

AGV	*Automated Guided Vehicles*
BJT	*Bipolar Junction Transistor*
CAD	*Computer-Aided Design*
CPU	*Central Processing Unit*
CSI	*Camera Serial Interface*
DSI	*Display Serial Interface*
FINE	*First INtelligent Extinguisher*
FL	FLIR LEPTON ®
GPIO	*General Purpose Input/Output*
GVD	*Generalized Voronoi Diagram*
HDMI	*High-Definition Multimedia Interface*
I^2C	*Inter-Integrated Circuit*
ICP	*Iterative Closest Point*
IPG	Instituto Politécnico da Guarda
NCARAI	*Navy Center for Applied Research in Artificial Intelligence*
OLE	*Off-road Loescheinheit*
PC	*Personal Computer*
RPI	Raspberry Pi®
SAFFiR	*Shipboard Autonomous Firefighting Robot*
SLAM	*Simultaneous Localization And Mapping*
SO	Sistema Operativo
SPI	*Serial Peripheral Interface*
TM	*Trajectory Map*

UART	*Universal Asynchronous Receiver/Transmitter*
USB	*Universal Serial Bus*
VM	*Voronoi based Map*
ZOH	*Zero Order Holder*

Índice

Índice de Figuras

Índice de Tabelas

1 INTRODUÇÃO

Os incêndios são atualmente um problema transversal a todos os países e Portugal, infelizmente, não é exceção. Todos os anos Portugal é muito afetado por incêndios florestais de grandes dimensões. Segundo PORDATA, Base de Dados de Portugal Contemporâneo (2016) entre 2000 e 2014 ocorreram 348 447 incêndios em Portugal Continental que corresponderam a uma área ardida de 1 995 049ha.

Um incêndio bastante conhecido em Portugal foi o ocorrido nos armazéns do Chiado, a 25 de Agosto de 1988 e que resultou em duas mortes: um morador e um bombeiro.

Além disso, levou à destruição total ou parcial de vários edifícios, perdas de património, entre outros, resultando num dano direto de prejuízos estimado em 80 milhões de euros. A reconstrução e normalização da área afetada demorou cerca de doze anos (Revista Segurança, 2015).

Para além das consequências ambientais e económicas dos incêndios, o seu combate é um trabalho árduo e perigoso que coloca a vida dos bombeiros em risco. Em 2014, só nos E.U.A. faleceram em serviço 64 bombeiros e 63 350 ficaram feridos (Haynes & Molis, 2015). Estes dados tornam evidente a necessidade de criar alternativas de modo a reduzir o número de perdas humanas. Uma das soluções possíveis, e aquela que é focada nesta dissertação, passa pela substituição de homens por robôs nas tarefas perigosas.

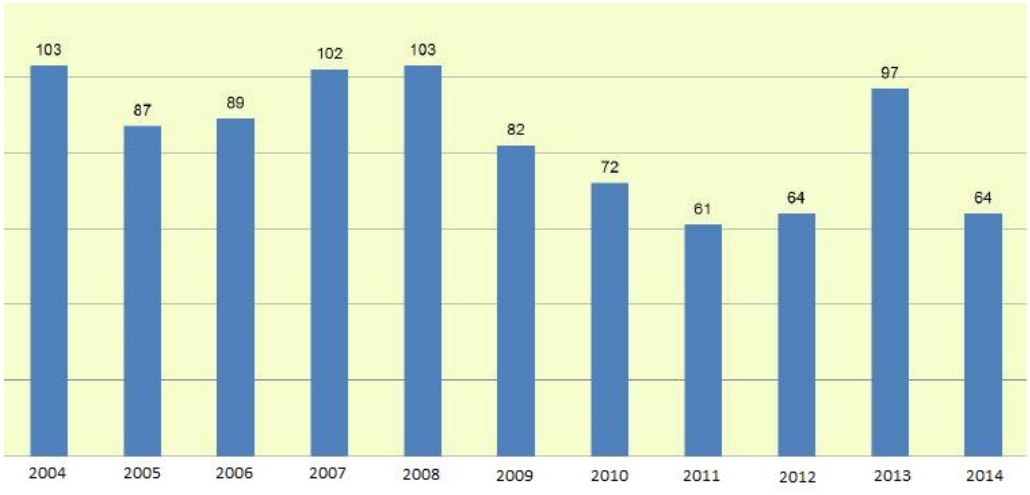


Figura 1.1 - Mortes de bombeiros em serviço nos E.U.A. Fonte: (Haynes & Molis, 2015)

A utilização de robôs móveis é justificada, entre outras, em tarefas arriscadas para o trabalhador humano. Um exemplo disso é o combate a incêndios, onde um robô móvel pode desenvolver a sua função, evitando a exposição desnecessária dos bombeiros a riscos.

1.1 Objetivos

A seguinte dissertação tem como objetivo projetar e desenvolver um protótipo de um robô autónomo para combate a incêndios em ambientes estáticos. Para tal, robô tem de ter a capacidade de explorar um ambiente desconhecido, com vista a detetar a zona de incêndio, e extinguir o foco de incêndio por meios mecânicos.

O desenvolvimento do protótipo passa pelas seguintes etapas:

1. Projeto do *chassis* do robô;
2. Programação dos sensores e atuadores necessários para os seguintes sistemas:
   i. Navegação e localização;
   ii. Deteção de chama;
   iii. Extinção de incêndio;

Um outro objetivo consiste em preparar o robô desenvolvido para uma futura participação na competição nacional do "Robô Bombeiro", organizada pelo Instituto Politécnico da Guarda (IPG). Para o robô ser bem-sucedido na competição terá de ser capaz de navegar de forma autónoma pelo labirinto, encontrar a chama e extingui-la no menor tempo possível.

1.2 Robótica

O termo robô foi utilizado pela primeira vez em 1921 pelo checoslovaco Karel Čapek, sendo uma transformação da palavra checa "*robota*", que tem como significado escravo ou criado. A primeira patente para o que hoje se considera um robô foi apresentada em 1954 por George C. Devol e emitida em 1961. O dispositivo composto por um braço mecânico com uma pinça, foi montado em trilhos e a sequência dos movimentos era codificada como padrões magnéticos armazenados sobre um tambor rotativo. A primeira empresa de robótica, a Unimation, foi fundada por Devol e Joseph Engelberger em 1956 e o seu primeiro robô industrial (Figura 1.2) foi instalado em 1961 (Corke, 2011).

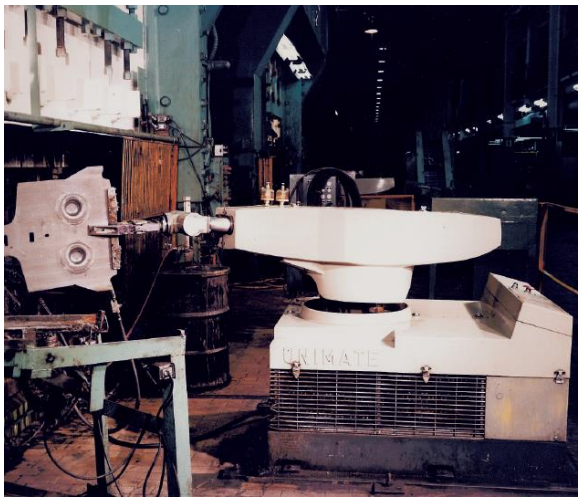


Figura 1.2 – O primeiro robô da empresa Unimation a trabalhar na fábrica da General Motors. Fonte: (Corke, 2011)

Nos anos sessenta ao introduzir-se na indústria os robôs manipuladores como um elemento no processo produtivo, levou a um aumento do interesse dos pesquisadores para conseguir manipuladores mais rápidos, precisos e fáceis de programar devido à ampla gama de possibilidades que oferecia. A consequência direta desse avanço originou um novo passo na automação industrial, que tornou mais flexível a produção com o nascimento da noção de célula de fabrico robotizada (Secchi, 2012).

O trabalho realizado pelos robôs manipuladores consiste frequentemente em tarefas repetitivas, como a alimentação das diferentes máquinas da célula de fabrico robotizada. Para tal, é necessário situar as máquinas no interior do volume de trabalho do manipulador, o que se torna impossível quando as células sofrem ampliações drásticas.

Uma solução para esse problema foi o desenvolvimento de um veículo móvel sobre trilhos para proporcionar um transporte eficaz dos materiais entre as diferentes zonas da cadeia de produção. Desta forma, surgem os primeiros Veículos Guiados Automaticamente (*Automated Guided Vehicles* - AGV's).

Figura 1.3 - AGV criado pela empresa SWISSLOG usado para transportar blocos de motor de um posto de montagem para outro, sendo guiado por um fio elétrico instalado no chão. Fonte: (Siegwart, Nourbakhsh, & Scaramuzza, 2011)

A possibilidade de estruturar o ambiente industrial permite a navegação de veículos com capacidades sensoriais e de raciocínio mínimas. No entanto, diante de mudanças inesperadas na área de trabalho, que afetem o desenvolvimento normal da navegação, estes sistemas encontram-se impossibilitados de executar ações alternativas que permitam retomar a sua atividade. Isso leva a que, em aplicações fora do âmbito industrial, onde é caro ou impossível estruturar o ambiente, dotem-se veículos de um maior grau de inteligência e perceção. Desta forma obtêm-se veículos de propósito geral, aptos para operar em qualquer classe de ambiente, e que se convencionou designar como os *robôs móveis*.

Figura 1.4 – Exemplo de um robô móvel: *Rover* em Marte. Fonte: (Corke, 2011)

1.3 Robótica Móvel

Segundo Pieri (2002), a robótica móvel é o termo que agrega o estudo e desenvolvimento de aplicações para sistemas robotizados e móveis. Estes, por sua vez, são dispositivos mecânicos montados sobre uma base não fixa, que agem sob o controlo de um sistema computacional, equipado com sensores e atuadores que permitem a interação com o ambiente (Bekey, 2005).

Têm aplicações muito variadas que estão normalmente relacionadas com tarefas arriscadas ou nocivas para a saúde humana, na agricultura, no transporte de cargas perigosas ou em tarefas de exploração solitárias ou cooperativas junto a outros veículos não tripulados. Exemplos clássicos são as tarefas de manutenção de reatores nucleares (Herndon & Haley, 2001), a manipulação de materiais explosivos, a exploração subterrânea e subaquática (Fauske, Gustafsson, & Hegrenæs, 2007), entre outros.

Os robôs móveis que operam em ambientes não explorados (sem mapa) depararam-se com incertezas significativas na posição e na identificação de objetos. A incerteza é tal que, mover-se de um ponto A até um ponto B é uma atividade arriscada para um robô móvel, comparando com um manipulador industrial, para o qual se trata de uma atividade trivial. Em compensação, por ter de lidar com tantas incertezas do ambiente em redor, não se espera que o robô móvel siga trajetórias ou alcance o seu destino final com o mesmo nível de precisão que se espera de um manipulador industrial (Secchi, 2012).

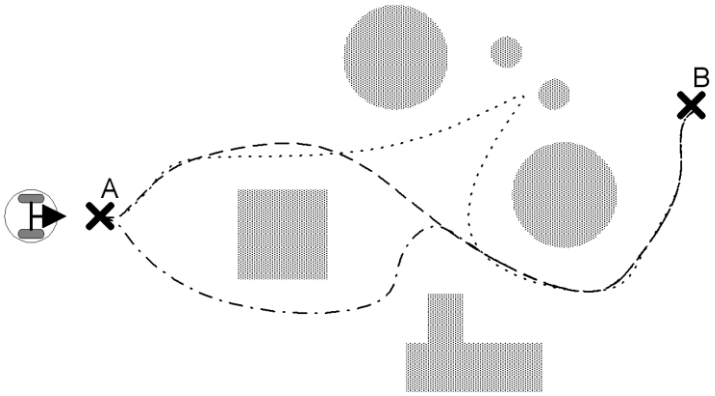


Figura 1.5 - Possíveis trajetórias que o robô móvel poderia seguir do ponto A para o ponto B. Fonte: (Secchi, 2012)

Os principais problemas associados a um robô móvel prendem-se com as operações de (i) localização, (ii) geração de trajetórias e de (iii) controlo dos seus movimentos, com base na informação proveniente do sistema de sensores externos (ultra som, *laser*, visão) e internos (encoders). Estas operações devem garantir o movimento entre pontos do ambiente de trabalho de maneira segura, sem risco de colisão.

O robô móvel autónomo caracteriza-se por uma conexão inteligente entre as operações de perceção e ação, que definem o seu comportamento e permitem-lhe chegar à execução dos objetivos programados sobre o ambiente com alguma incerteza.

O grau da sua autonomia depende em grande medida da capacidade para abstrair o ambiente em redor e converter a informação obtida em ordens, de tal modo que, aplicadas sobre os atuadores do sistema de locomoção, garanta a realização eficaz de sua tarefa. Desse modo, as duas grandes características que o distinguem de qualquer tipo de veículo são:

- **Perceção**: O robô móvel deve ser capaz de determinar a relação com o seu ambiente de trabalho através do sistema sensorial a bordo. A capacidade de perceção do robô móvel traduz-se na síntese de toda a informação oferecida pelos sensores, com o objetivo de gerar mapas globais e locais do ambiente de acordo com os diversos níveis de controlo.
- **Raciocínio**: O robô móvel deve ser capaz de decidir que ações são solicitadas em cada momento, segundo o estado do robô e o do seu ambiente a redor, para alcançar os seus objetivos. A capacidade de raciocínio do robô móvel traduz-se no planeamento de trajetórias globais seguras e na habilidade para modificá-las no caso de surgirem obstáculos inesperados (controlo local de trajetória) para permitir ao robô, a execução dos objetivos solicitados.

O modo como o robô interage com o ambiente está intimamente ligado com a forma como o seu sistema computacional utiliza a informação sobre o ambiente. Neste contexto, o tipo de arquitetura do sistema computacional tem um papel crucial (Bekey, 2005). Segundo Arkin (1998) "uma arquitetura para robôs está mais relacionada com uma arquitetura de *software*, e não tanto com a componente de *hardware* do sistema de controlo".

Entre as diversas características de implementação de uma arquitetura de robôs móveis, o parâmetro que mais se diferencia é a forma de raciocínio.

O raciocínio determina a forma como a o sistema age mediante um estímulo, seja ele uma entrada de dados especificando um objetivo ou uma leitura dos sensores. O raciocínio dentro da robótica móvel pode ser classificado como: Reativo, Deliberativo ou Híbrido.

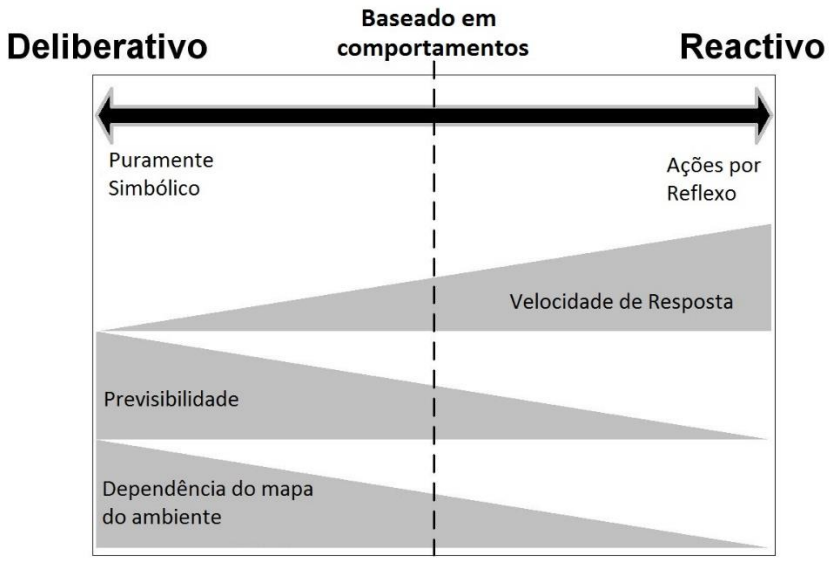


Figura 1.6 Esquema de arquiteturas – Deliberativa e Reativa (adaptado). Fonte: (Secchi, 2012)

1.3.1 Arquitetura Deliberativa

A Arquitetura Deliberativa baseia-se numa estratégia puramente simbólica, dependendo, em larga medida, da semelhança entre o ambiente e o modelo contruído pelo robô para o representar. Por exemplo, um robô móvel preparado para navegar num ambiente estático não estará capacitado para navegar num corredor pelo qual circulam pessoas; por forma a lidar com ambientes mais complexos, como dinâmicos, é necessário encontrar arquiteturas deliberativas mais complexas. Tal conduz a sistemas de controlo e de processamento de informação mais complexos e de menor velocidade de resposta (Secchi, 2012).

Por outro lado, as estratégias deliberativas incluem uma análise de estabilidade que permite garantir, *a priori*, sob que condições de ambiente o robô móvel cumprirá com seus objetivos.

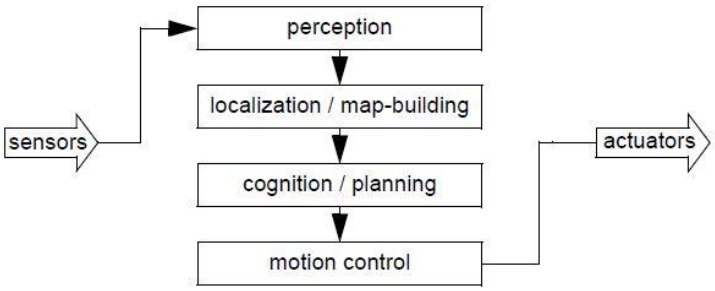


Figura 1.7 – Esquema da Arquitetura Deliberativa. Fonte: (Siegwart, Nourbakhsh, & Scaramuzza, 2011)

A Arquitetura Deliberativa tem como vantagem a utilização do mapa permite que a estimação da posição por parte do robô esteja disponível de forma transparente para os operadores humanos; a existência do mapa representa um meio de comunicação entre o ser humano e robô: o humano pode simplesmente dar ao robô um novo mapa, caso o robô se encontro num novo ambiente; o mapa, se criado pelo robô, pode também ser utilizado por seres humanos (Siegwart, Nourbakhsh, & Scaramuzza, 2011).

Esta arquitetura exige um maior esforço inicial no desenvolvimento para criar um robô móvel de navegação. Com o objetivo, que o esforço inicial no desenvolvimento resulte numa arquitetura que permita mapear e navegar numa variedade de ambientes, amortizando assim o custo do projeto inicial ao longo do tempo.

A grande característica da abordagem baseada em mapas é a construção de um mapa do ambiente. De modo a desenvolver-se um sistema de navegação robusto, é necessário que o ambiente contenha informação suficiente para que o robô realize a localização com precisão.

1.3.2 Arquitetura Reativa

A Arquitetura Reativa baseia-se num esquema de ações por reflexo, ou seja, o ambiente em redor é percebido como um estímulo (distância dos objetos, nível de luz, temperatura), que origina uma ação de controlo em função da sua intensidade (Secchi, 2012).

O principal objetivo da arquitetura reativa é a resposta rápida a uma variedade de ocorrências ou situações no ambiente permitindo que os robôs operem em ambientes altamente dinâmicos. Esta rapidez deve-se à simplicidade do tratamento da informação sensorial e à forma direta pela qual a perceção, ou estímulo, está associado com uma ação, ou resposta (Junior, 2006).

Segundo Rodrigues (2010) as principais vantagens da arquitetura reativa em relação à arquitetura deliberativa estão relacionadas com o facto de em muitos casos o robô não possuir conhecimento antecipado sobre o ambiente onde irá atuar; os modelos do ambiente geralmente não são precisos; podem ocorrer situações inesperadas ou mudanças rápidas no ambiente.

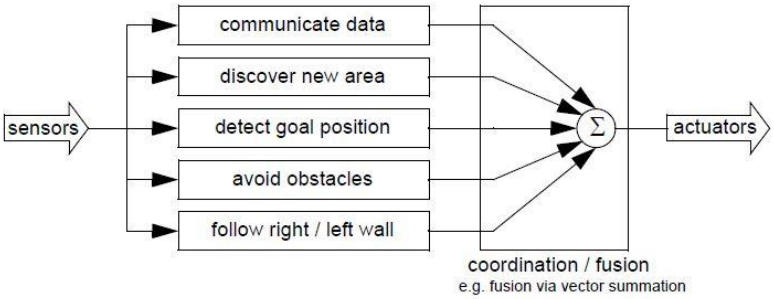


Figura 1.8 - Esquema da Arquitetura Reativa. Fonte: (Siegwart, Nourbakhsh, & Scaramuzza, 2011)

Esta arquitetura tem como principais desvantagens: o método não é expandido diretamente para outros ambientes ou para ambientes maiores, muitas vezes, o código de navegação é específico para um local, sendo necessário outro código para um novo ambiente; os procedimentos subjacentes (seguir a parede da esquerda), devem ser cuidadosamente projetados para produzir o comportamento desejado, esta tarefa pode ser demorada e depende bastante do *hardware* específico do robô e das características do ambiente; a adição de cada novo comportamento, obriga o projetista do robô a agrupar todos os comportamentos existentes, de modo a garantir que as interações ação de controlo resultante são estáveis (Siegwart, Nourbakhsh, & Scaramuzza, 2011).

1.3.3 Arquitetura Hibrida

A Arquitetura Hibrida baseia-se em resgatar as vantagens da arquitetura deliberativa e da arquitetura reativa. De modo a garantir a estabilidade de múltiplos controladores simples, operando em paralelo, junto com técnicas de aprendizagem para melhorar o desempenho do robô para conseguir torná-lo independente do modelo do ambiente (Secchi, 2012).

O objetivo é conseguir algoritmos de controlo confiáveis (característica dos algoritmos da arquitetura deliberativa) e que tenham uma velocidade de resposta de acordo com a velocidade do robô móvel (característica dos algoritmos da arquitetura reativa).

1.4 Classificação dos robôs móveis

Existem diversas taxonomias para a classificação dos robôs móveis, que podem agrupar-se segundo os critérios de locomoção, controlo, ambiente ou funcionalidade (Pieri, 2002).

1.4.1 Locomoção

Um robô móvel necessita de mecanismos de locomoção para se deslocar através do seu ambiente. Para esse efeito existe uma grande variedade de mecanismos possíveis sendo a sua seleção um aspeto importante no projeto de um robô móvel. A maioria destes mecanismos de locomoção foram inspirados pelos seus equivalentes biológicos - Figura 1.9 (Siegwart, Nourbakhsh, & Scaramuzza, 2011).

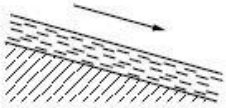
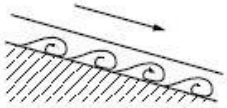
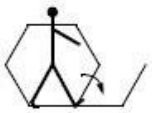
Type of motion	Resistance to motion	Basic kinematics of motion
Flow in a Channel 	Hydrodynamic forces	Eddies 
Crawl 	Friction forces	Longitudinal vibration 
Sliding 	Friction forces	Transverse vibration 
Running 	Loss of kinetic energy	Periodic bouncing on a spring 
Walking 	Loss of kinetic energy	Rolling of a polygon 

Figura 1.9 – Tabela síntese de mecanismos de locomoção utilizados em sistemas biológicos. Fonte: (Siegwart, Nourbakhsh, & Scaramuzza, 2011)

Existe no entanto, uma exceção: a roda. É uma invenção humana que atinge uma eficiência extremamente alta em terreno plano. Este mecanismo não é completamente estranho aos sistemas biológicos, pois o nosso sistema de andar bípede pode ser aproximado a um polígono a rolar, com lados iguais de comprimento d (Figura 1.10). À medida que o tamanho do passo diminui, o polígono aproxima-se de um círculo.

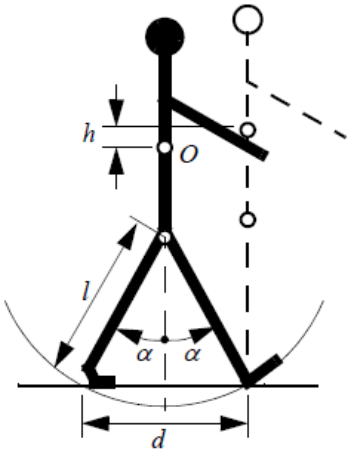


Figura 1.10 - Esquema do mecanismo de locomoção bípede e a aproximação a um círculo. Fonte: (Siegwart, Nourbakhsh, & Scaramuzza, 2011)

Segundo Siegwart, Nourbakhsh, & Scaramuzza (2011) as questões-chave da locomoção são: a estabilidade, que está relacionada com o número e geometria dos pontos de contacto, o centro de gravidade, a estabilidade estática e dinâmica e a inclinação do terreno - características do contacto que estão relacionada com o ângulo de contacto e a fricção - e o meio ambiente.

1.4.1.1 Locomoção com Pernas

A locomoção com pernas é caracterizada por uma série de pontos de contacto entre o robô e o terreno. As suas principais vantagens são a adaptabilidade e manobrabilidade em terrenos acidentados. Uma vez que apenas é necessário um conjunto de pontos de contacto, a qualidade do solo entre esses pontos não terá importância, desde que o robô possa manter uma distância do solo adequada. Além disso, um robô com pernas é capaz de atravessar um furo ou fenda, desde que o seu alcance consiga exceder a largura do orifício. Uma outra vantagem da locomoção com pernas é o potencial para manipular objetos no ambiente com grande habilidade (Siegwart, Nourbakhsh, & Scaramuzza, 2011).

As principais desvantagens da locomoção com pernas estão relacionadas com a potência e a complexidade mecânica. A perna, que pode ter vários graus de liberdade, deve ser capaz de suportar parte do peso total do robô, e, em muitos robôs deve ser capaz de elevar e baixar o robô. Além disso, a alta manobrabilidade só poderá ser alcançada se as pernas tiverem um número suficiente de graus de liberdade de modo a transmitir forças em diferentes direções.

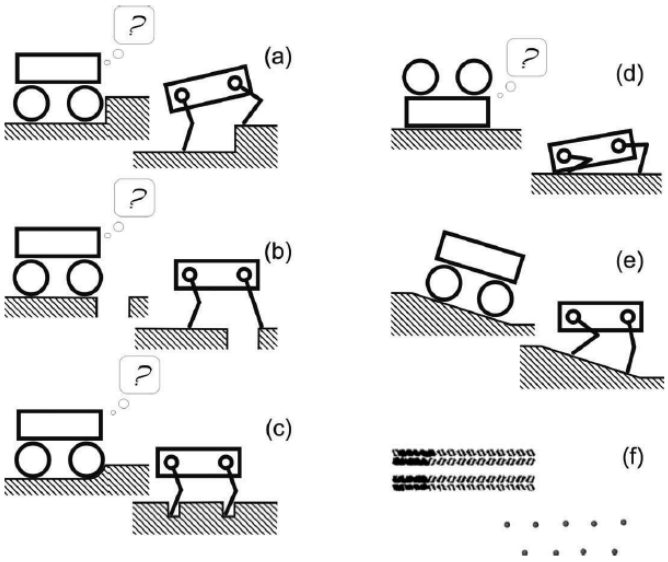


Figura 1.11 – Vantagens do robô com pernas em relação ao robô com rodas. Fonte: (Siegwart, Nourbakhsh, & Scaramuzza, 2011)

Como referido, os robôs com pernas são particularmente adequados para terrenos acidentados, pois são capazes de atravessar obstáculos como degraus (*a*), fendas (*b*), ou areais (*c*) que são intransitáveis para os robôs de sistemas de locomoção com rodas. Além disso, o elevado número de graus de liberdade permite ao robô levantar-se quando cai (*d*) e manter a sua carga nivelada (*e*). Os sistemas de pernas não requerem um caminho contínuo de apoio, sendo apenas necessário alguns pontos de apoio selecionados, o que também reduz o impacto ambiental (*f*).

Como os robôs com pernas são inspirados na natureza, é instrutivo examinar sistemas biológicos de locomoção com pernas bem-sucedidos. Animais de grande porte, tais como mamíferos e répteis, têm quatro pernas, enquanto os insetos têm seis ou mais pernas. Em alguns mamíferos, a capacidade de andar com apenas duas pernas tem sido aperfeiçoada. Esta capacidade de manobra excecional tem um preço: controlo necessário para manter o equilíbrio é muito mais complexo.

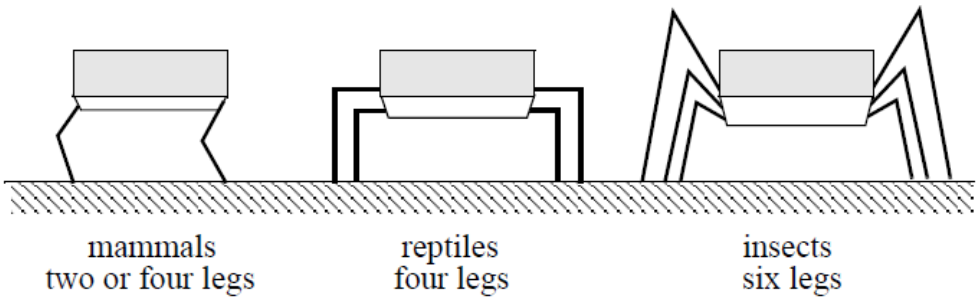


Figura 1.12 - Síntese das possibilidades dos robôs com pernas. Fonte: (Siegwart, Nourbakhsh, & Scaramuzza, 2011)

Um exemplo de robôs com pernas são os da série humanoide da Honda que estão a ser projetados não para fins de entretenimento, mas como auxiliares humanos para a sociedade.

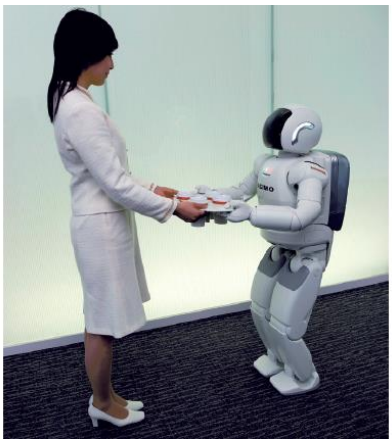


Figura 1.13 - Robô Humanoide Asimo da Honda. Fonte: (Corke, 2011)

Uma característica importante dos robôs bípedes é a sua forma antropomórfica. Eles podem ser construídos para ter dimensões aproximadas às dos seres humanos, e isso torna-os excelentes veículos para a pesquisa em interação homem-robô.

1.4.1.2 Locomoção com Lagartas

Consiste numa forma alternativa de direção, denominado de escorregamento / derrapagem que pode ser utilizada para reorientar o robô girando as rodas que estão a enfrentar a mesma direção a velocidades diferentes ou em direções opostas. Um exemplo conhecido da locomoção com lagartas são os tanques militares (Siegwart, Nourbakhsh, & Scaramuzza, 2011).

Figura 1.14 – Robô Pioneer com locomoção com lagartas, com a função de explorar o sarcófago em Chernobil. Fonte: (Herndon & Haley, 2001)

Estes tipos de robôs têm um contato com o solo muito maior, e isso permite melhorar significativamente a sua capacidade de manobra em terrenos soltos em comparação com as rodas convencionais. No entanto, devido a esta grande área de contacto com o solo, alterar a orientação do robô geralmente requer uma curva em derrapagem, em que uma grande porção da faixa deve deslizar contra o terreno.

A desvantagem de tais configurações é acoplada à direção do escorregamento / derrapagem. Devido à grande quantidade de derrapagem durante uma curva, é difícil de prever exatamente o centro de rotação do robô e a alteração da posição e orientação, que também está sujeita a variações, dependendo do atrito com o solo. Portanto, estimação em tais robôs é bastante imprecisa. Este é o *trade-off* que é feito em troca de boa manobrabilidade e tração em terrenos acidentados e solto. Além disso, uma abordagem de escorregamento / derrapagem sobre uma superfície de elevado atrito pode fazer com que ultrapasse as capacidades de binário dos motores utilizados. Em termos de eficiência energética, esta abordagem é razoavelmente eficiente em terreno solto, mas extremamente ineficiente em casos opostos.

1.4.1.3 Locomoção com Rodas

A locomoção através de rodas tem sido, de longe, o mecanismo de locomoção mais popular na robótica móvel e em veículos feitos pelo homem. Permite alcançar muito bons ganhos de eficiência, através de uma implementação mecânica relativamente simples (Siegwart, Nourbakhsh, & Scaramuzza, 2011).

O equilíbrio não costuma ser um problema em projetos de robôs com rodas, pois são quase sempre concebidos de modo que todas as rodas estejam em contato com o solo em todos os momentos. Assim, três rodas são suficientes para garantir o equilíbrio estável, embora o número mínimo de rodas para que o robô seja estável sejam duas. Quando são utilizados mais de três rodas, é necessário um sistema de suspensão para permitir que todas as rodas mantenham contato com o solo quando o robô encontra terreno irregular. As principais preocupações com os robôs com rodas tendem a concentrar-se nos problemas de tração e estabilidade, no sentido de as rodas do robô serem capazes de fornecer tração e estabilidade suficiente para que o robô consiga mover-se em todo o espaço de trabalho e manobrabilidade e controlo.

Em relação às rodas, existem diferentes tipos com as suas respetivas vantagens e desvantagens, que por sua vez, combinadas permitem configurações de rodas que proporcionam formas particulares de locomoção para um robô móvel. As rodas mais utilizadas podem ser classificadas

em quatro classes, como se mostra na Figura 1.15. Diferem amplamente na cinemática, e, portanto, a escolha do tipo roda tem um grande efeito sobre a cinemática global do robô móvel.

As principais classes de rodas são: (*a*) roda orientável centrada com dois graus de liberdade; rotação em torno do eixo da roda (motorizada) e o ponto de contacto (*b*) roda orientável não centrada (roda livre) com dois graus de liberdade; rotação em torno de uma articulação ligeiramente afastada (*c*) roda sueca ou omnidirecional com três graus de liberdade; rotação em torno do eixo da roda (motorizada), em torno dos rolos, e em torno do ponto de contacto (*d*) roda esférica.

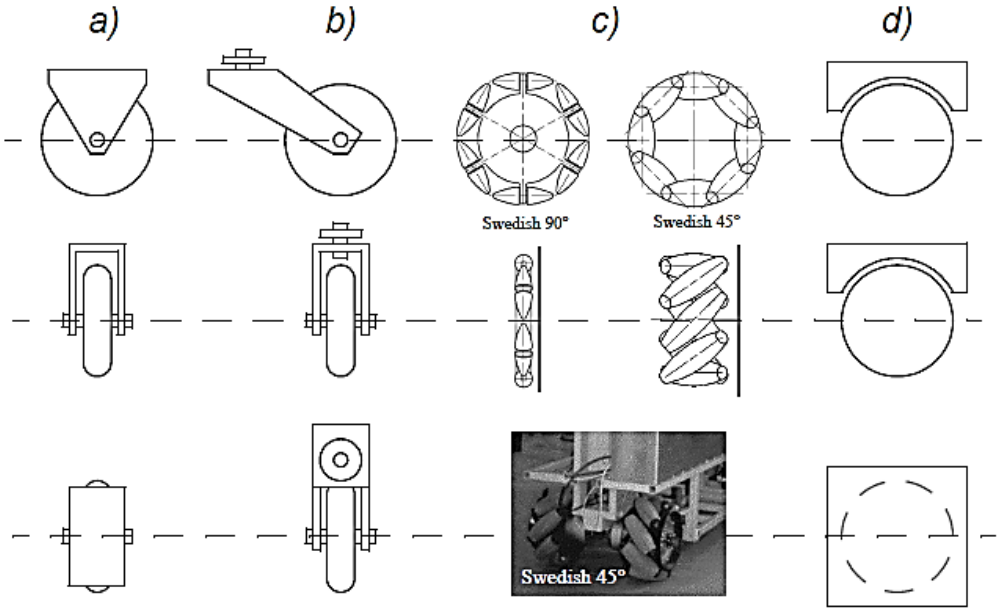


Figura 1.15 - Principais classes de rodas. Fonte: (Siegwart, Nourbakhsh, & Scaramuzza, 2011)

As rodas orientável centrada e não centrada têm um eixo principal de rotação e são, portanto, bastante direcionais. Para mover-se numa direção diferente, a roda deve ser guiada primeiramente ao longo de um eixo vertical. A principal diferença entre estas duas rodas é que a roda orientável centrada pode realizar este movimento de direção sem efeitos secundários, pois o centro de rotação passa através da área de contacto com o solo, enquanto que a roda orientável não centrada roda em torno de um eixo desfasado, provocando uma força a ser transmitida para o chassis do robô durante a condução.

Por vezes deseja-se uma capacidade de rolar lateralmente, mas as rodas *standard* fornecem um benefício significativo ao curvar - atrito lateral entre as rodas e a estrada, de forma gratuita, a aceleração centrípeta que de outro modo exigiria um atuador extra para fornecer essa força (Corke, 2011).

A roda sueca é semelhante a uma roda normal, mas tem um certo número de rolos passivos em torno da sua circunferência e os seus eixos de rotação encontram-se no plano da roda. É conduzida como uma roda comum, mas tem um coeficiente de atrito muito baixo na direção lateral.

A roda esférica é semelhante à esfera rolante de um cursor dos computadores antigos, mas acionado por dois atuadores de modo a que possa conseguir atingir uma determinada velocidade em qualquer direção.

A escolha dos tipos de rodas para um robô móvel está fortemente ligada à escolha do arranjo das rodas, ou a sua geometria. É necessário considerar estas duas questões em simultâneo ao projetar o mecanismo de locomoção de um robô com rodas, para tal é necessário ter em conta as três características fundamentais de um robô: estabilidade, manobrabilidade, e controlabilidade (Siegwart, Nourbakhsh, & Scaramuzza, 2011).

A escolha da configuração das rodas não é simples, pois não existe nenhuma configuração "ideal" que maximize simultaneamente a estabilidade, manobrabilidade e controlabilidade. Cada aplicação robô móvel coloca restrições exclusivas sobre o problema do projeto do robô, e a tarefa do projetista é escolher a configuração mais adequada possível.

1.4.2 Controlo

Quando classificados segundo o tipo de controlo, os robôs podem ser separados em três categorias:

Tele-operados: o operador realiza todos os movimentos que o robô deve fazer como por exemplo transporte de material perigoso, escavações de minas, a limpeza industrial, a inspeção de projetos nucleares, trabalhos de vigilância, inspeção entre outros (Secchi, 2012). Dentro da área da tele-operação existe outro tipo de controlo que se encontra em pesquisa. Tem como objetivo de reduzir a fadiga do operador, este controlo denomina-se por telepresença. A telepresença a realidade virtual, onde o operador tem *feedback* completo dos sensores e sente-se como se fosse o próprio robô (Murphy, 2000).

Semi-autónomos: caraterizam-se pela tarefa de controlo ser compartilhada ou alternada com o operador humano. Exemplo disso são os *rovers* Spirit e Opportunity que navegam autonomamente na superfície de Marte, apesar dos operadores humanos fornecerem as metas principais. Ou seja, os operadores dizem ao robô onde ir e o robô determina os detalhes da rota (Corke, 2011).

Autónomos: O robô realiza a tarefa por si só, tomando as suas próprias decisões baseando-se nos dados obtidos do ambiente.

1.4.3 Ambiente

Existem três distinções entre o ambiente de um robô móvel, estando estas associadas ao meio envolvente, à sua área e trabalho e aos objetos a seu redor. O meio ambiente pode ser caracterizado das seguintes formas: ambiente aéreo, aquático e terrestre. O meio ambiente é de extrema importância para um robô pois, influencia o modo como o robô é projetado. Consoante o meio, os sensores, atuadores e até mesmo a estrutura irão variar.

O tipo de estrutura do ambiente de trabalho é a característica que mais impõe restrições sobre um robô, estas agrupam-se segundo a área de trabalho e segundo os objetos presentes no ambiente em redor (Secchi, 2012).

Segundo a área de trabalho, o ambiente pode ser interior (*indoor*) ou exterior (*outdoor*). Considera-se interior quando a área de trabalho está claramente definida por paredes e teto, onde a iluminação é principalmente artificial. Por outro lado, o ambiente é exterior quando a área de trabalho não está claramente delimitada e a iluminação é principalmente natural.

Segundo os objetos presentes no ambiente em redor do robô, este pode ser classificado como dinâmico ou estático. Considera-se estático quando os objetos presentes no ambiente não mudam de forma nem de posição e possuem características físicas particulares (forma, cor etc.) que permitem associá-los a formas geométricas conhecidas ou distingui-los de outros objetos (portas abertas, mesas de trabalho etc.). Por outro lado, o ambiente é dinâmico quando ocorrem alterações com o decorrer do tempo e tais mudanças são imprevisíveis, ou quando a associação entre os objetos do ambiente em redor e determinadas características físicas não é viável.

1.4.4 Funcionalidade

Segundo Pieri (2002), através da funcionalidade do robô móvel, é possível criar os seguintes grupos: robôs industriais, de serviço, de campo e pessoais. Contudo, existe uma sobreposição, entre os três primeiros devido à diferença dos ambientes onde atuam e a necessidade de maior autonomia - Figura 1.16.

Segundo Secchi (2012) “a própria tarefa determina, numa primeira fase, a estrutura particular de um robô móvel”.

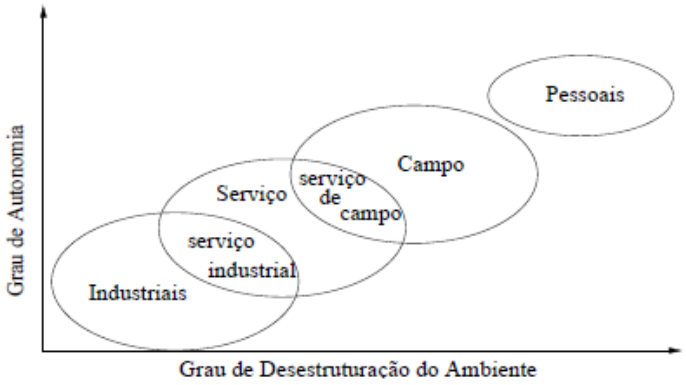


Figura 1.16 – Posição dos respetivos grupos, consoante a variação do grau de autonomia e da desestruturação do ambiente. Fonte: (Pieri, 2002).

Robôs Industriais: robôs móveis utilizados em linhas de produção. Estes robôs recebem tarefas determinadas *a priori* na forma de uma sequência explícita de ações e executam esse programa automaticamente. O ambiente é completamente estruturado e ajustado para a execução da tarefa. Geralmente, os robôs móveis industriais são plataformas móveis programadas para seguir linhas desenhadas no chão e utilizadas para tarefas pesadas, como transporte de materiais em sistemas de produção – AGV's. Um exemplo é o AGV proposto por Sankari & Imtiaz (2016).

Robôs de Serviço: robôs móveis utilizados para serviços gerais. O ambiente é estruturado e o robô possui um modelo deste ambiente, que é conhecido previamente, porém possui certa autonomia, pois processa informação sensorial, para atuar em situações imprevistas, como desviar-se de uma pessoa ou objeto (Carreira, Calado, & Cardeira, 2011) e (Figueiredo, 2015). Estes robôs recebem macro comandos da tarefa que devem realizar e são utilizados para tarefas de limpeza em geral (pisos, condutas de ar, etc.), em sistemas de vigilância e no transporte de materiais leves (correspondências internas, material hospitalar, etc.). Um exemplo é o robô móvel proposto por Carreira, Canas, Silva, & Cardeira (2006) para distribuição de refeições em hospitais.

Robôs de Serviço Industrial: são veículos que navegam de forma autónoma, em ambiente industrial, para realizam tarefas específicas. A navegação pode ser realizada com recurso à estruturação do ambiente, como extração de linhas, pontos ou detetando *beacons* artificiais Bernardino (2016). Numa abordagem oposta, Carreira, et al. (2012) e Carreira F. , Calado, Cardeira, & Oliveira (2015) propõem um sistema de localização de robôs móveis, que não requer a estruturação do ambiente, nem a utilização de nenhum tipo de *beacons* artificiais.

Robôs de Campo: robôs utilizados em ambientes não estruturados, pouco conhecidos e geralmente perigosos. As principais atividades destes robôs são: exploração espacial, de cavernas, vulcões e limpeza de acidentes nucleares (Herndon & Haley, 2001), exemplo disso são os *rovers* utilizados em Marte.

Robôs de Serviço de Campo: robôs de serviço que atuam em ambientes externos que podem ser previamente modelados ou não. Geralmente, caso exista um modelo, este é precário e exige a necessidade do processamento sensorial para complementar o modelo existente. Estes robôs são utilizados na realização de tarefas agrícolas (Hiremath, van der Heijden, van Evert, Stein, & ter Braak, 2014) e para navegação em autoestradas.

Robôs Pessoais: robôs comerciais, que não realizam tarefas específicas, mas interagem com os humanos e aprendem a localizar-se no ambiente, como por exemplo o "robô cão" AIBO (Moon, 2002).

1.5 Robôs Bombeiros

Esta secção está dividida em duas partes: Estado da Arte referente aos robôs utilizados na competição e Estado da Arte referente aos robôs de combate a incêndios. O primeiro começa por explicar as regras principais da competição e de seguida aborda dois robôs utilizados, explicando o seu funcionamento, arquiteturas e estratégias utilizadas.

O segundo consiste em mostrar alguns robôs de combate a incêndios e explicar o seu modo de operação, classificá-los consoante os parâmetros referidos anteriormente (locomoção, controlo, ambiente e funcionalidade). Sendo este ultimo parâmetro (funcionalidade) comum para todos os robôs aqui referidos, à exceção do último que tem como funcionalidade resgatar vítimas em escombros, uma tarefa também desempenhada por bombeiros.

Os robôs aqui referidos têm uma grande variedade no que diz respeito à sua classificação dado que se procurou apresentar robôs que usam abordagens diversas para um objetivo em comum. Algumas destas abordagens encontram-se implementadas atualmente enquanto outras são ainda projetos para o futuro.

1.5.1 Robôs utilizados na Competição

A competição "Robô Bombeiro" tem como objetivo a utilização da robótica como um meio educacional. De modo a que um robô vença a competição é necessário que seja capaz de

navegar com sucesso pelo labirinto, detetar a chama (simulada por uma vela), extingui-la e regressar ao local de início no menor tempo possível.

Para poder participar na competição o robô necessita de cumprir alguns pré-requisitos. Entre eles, destaca-se o dimensionamento do robô, que tem como dimensões máximas 31x31x31 cm. Para a extinção da chama é possível utilizar os seguintes métodos: ar, gás inerte (CO_2), *spray* de água ou meios mecânicos (por exemplo uma esponja). Não sendo permitido a utilização de qualquer tipo de pós

Apesar do principal fator ser o tempo despendido, existem outros que permitem aumentar ou diminuir a sua pontuação. O fator de embater em obstáculos (paredes e obstáculo "cão"), o modo que o participante escolhe, estes fatores permitem atenuar o tempo demorado. Entre eles destacam-se o modo de começo arbitrário, a não utilização de extintor com ar, modo com mobília e ativação por sinal sonoro.

Como primeiro exemplo de um robô participante, apresenta-se o robô criado por um aluno do IPG. Consiste num robô com um sistema de locomoção de lagartas, um sistema de extinção da chama da vela composto por modo de água e ar. Para a navegação é baseado em sonares e um módulo de deteção de *landmarks*, para a deteção da chama utiliza um sensor denominado por UV TRON.

Figura 1.17 - Exemplo de um robô participante do concurso. Fonte: Retirado do cartaz exposto na Competição de 2016.

De uma forma resumida, o seu modo de funcionamento consiste em navegar no labirinto orientando-se pela parede à sua direita, para esse efeito recorre aos sonares. Esta navegação

ocorre até o robô se deparar com a linha branca (*landmark*) que está situada na entrada dos quartos. O robô sabendo que está dentro de um quarto irá procurar a chama através do sensor UV TRON. Caso o robô não encontre a vela, volta ao modo de navegação pela parede à sua direita, caso encontre entra no modo/estado de centrar os atuadores com a posição da vela. De seguida recorre ao sistema de extinção por modo de água para extinguir a chama, em caso de falha, o robô recorre ao modo de ar, de modo a garantir a sua extinção. Após a extinção da vela o robô volta ao seu ponto de partida, terminando a sua participação.

Um outro robô participante contém uma abordagem semelhante ao apresentado anteriormente, apenas com diferença da sua navegação ser orientada pela parede à esquerda - Figura 1.18.

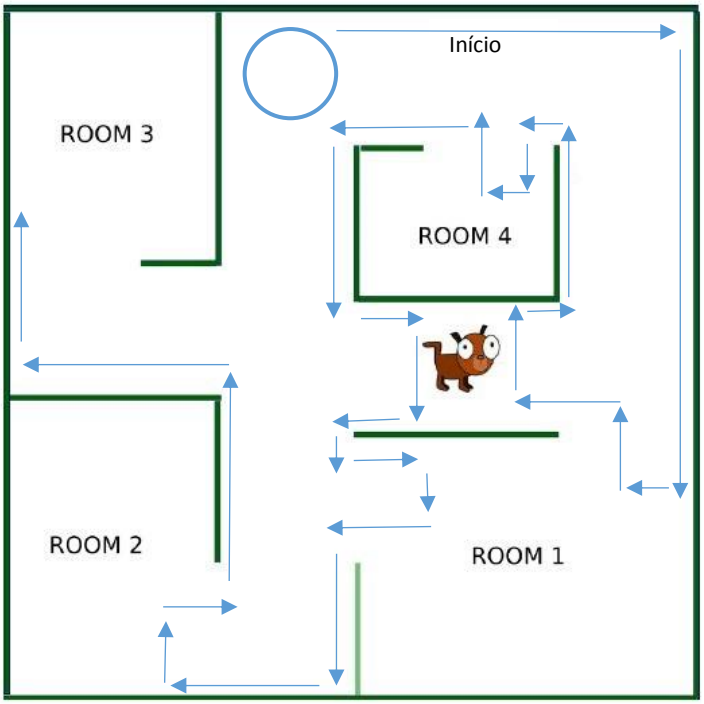


Figura 1.18– Esquema de exploração realizada pelo robô participante do concurso. Fonte: Retirado do cartaz exposto no Competição de 2016.

Nesta abordagem, o robô inicia a sua missão no círculo, seguindo depois navegando pela parede à sua esquerda. Sempre que se localiza na entrada de um quarto (ao ultrapassar ligeiramente a linha branca) verifica se existe a chama nesse quarto. Caso não se verifique a existência da chama, o robô recua ligeiramente e roda 180°, continuando a sua navegação pela parede à sua esquerda. Caso encontre o obstáculo "cão", roda 180° e volta a navegar pela parede à sua esquerda, neste caso entrando no quarto "ilha".

Apos análise de alguns robôs participantes no concurso, é possível concluir que utilizam uma arquitetura reativa. Baseiam a sua navegação pelo seguimento de paredes e atuam com base na informação sensorial, não tendo por isso um processo de planeamento de trajetória.

1.5.2 Robôs de combate a incêndios

1.5.2.1 LUF 60

LUF 60 é um robô de suporte ao combate a incêndios controlada por controlo remoto *wireless* com um alcance até 300m. Contém um ventilador de pressão positiva de alta capacidade e um canhão de água/espuma, esta combinação permite a dissipação de fumos, calor, gases tóxicos e redução da intensidade do fogo, permitindo que os bombeiros e equipas de resgate prossigam com segurança (Tan, et al., 2013).

Tem como características a mobilidade elevada bem como a flexibilidade, garantida por um sistema de lagartas, que lhe permite subir e descer escadas com uma inclinação de 30° e capacidade para mover um carro fora das vias de acesso.

Figura 1.19 - LUF 60 em funcionamento. Fonte: (LUF Fire - Fighter, s.d.)

Este robô está projetado para suportar as condições severas de funcionamento e operar em espaços confinados, podendo ser utilizado em incêndios industriais, em túneis ferroviários, hangares de aviões, centrais elétricas, armazéns e edifícios comerciais, entre outros. As suas especificações técnicas estão apresentadas na Tabela 1.1.

Medidas	
Comprimento x Largura	2.330m x 1.350m
Altura (Variação da posição do canhão)	2m a 2.5m
Peso	2 200kg

Informações de Desempenho	
Velocidade	0 - 6 km/h
Bomba de água	2 modos, 2 400 ltr./min a 10 bar ou 1 000 ltr./min a 15 bar, Capacidade 100 HP
Canhão de água	360 bocais, 15-20 bar
Potência do Ventilador	35 kW, caudal volúmico 90 000 m³/h, 165 km/h

Tabela 1.1 - Especificações técnicas da LUF 60. Fonte: (Tan, et al., 2013)

1.5.2.2 Portable Fire Evacuation Guide Robot System

Este sistema consiste num robô portátil que tem como objetivo servir de guia para evacuação de incêndios (Figura 1.20). Este pequeno e leve robô móvel de 2kg pode ser facilmente transportado e controlado remotamente por uma consola do tamanho de um computador portátil (Kim, Kim, Lee, Kang, & An, 2009).

O robô, ilustrado na Figura 1.21, é constituído pelos seguintes componentes: uma câmara para observar o local do incêndio, sensores para reunir dados de temperatura, monóxido de carbono (CO), e concentrações de oxigénio (O_2) e um microfone com altifalante para comunicações de voz de emergência entre bombeiros e vítimas.

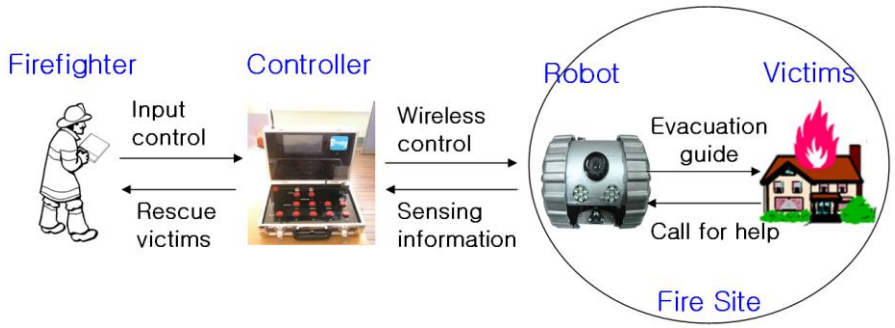


Figura 1.20 – Esquema de funcionamento do Portable Fire Evacuation Guide Robot System. Fonte: (Kim, Kim, Lee, Kang, & An, 2009)

O robô foi projetado com o composto de alumínio para resistência térmica com impermeabilização e um quadro de distribuição de impacto para a resistência ao impacto. Além disso, oferece vários serviços de redes multimédia, como vídeo, voz e transmissão de dados entre os bombeiros e as vítimas, com base em canais *wireless* separados.

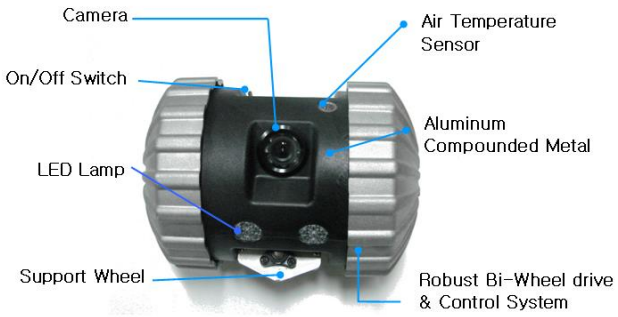


Figura 1.21 - Portable Fire Evacuation Guide Robot System. Fonte: (Kim, Kim, Lee, Kang, & An, 2009)

1.5.2.3 First INtelligent Extinguisher (FINE) Robot

O FINE é um extintor de incêndio inteligente projetado para lutar eficazmente contra incêndios domésticos (Figura 1.22) (Davoult & Lanne, s.d.). O sistema está projetado para alertar os bombeiros caso ocorra um incêndio e ninguém esteja em casa e, de seguida, de forma autónoma, move-se em direção ao fogo tentado apagá-lo através da pulverização do pó até à chegada dos bombeiros. Uma particularidade adicional deste robô é a de poder ser utilizado como um extintor de incêndio tradicional, como ilustrado na Figura 1.23.

Figura 1.22 - Protótipo do robô FINE. Fonte: (Davoult & Lanne, s.d.)

Os componentes deste robô incluem um termómetro infravermelho para detetar chamas, uma base de rolamento, um recipiente com pó, sensores para evitar colisões com obstáculos e um detetor de fumo. Outros detalhes específicos acerca do robô são desconhecidos, uma vez que este está em fase de desenvolvimento para uso comercial.

Figura 1.23 - Utilizações do robô FINE. a) Extintor de incêndio tradicional. b) Tentativa de extinção de forma autónoma do incêndio. Fonte: (Davoult & Lanne, s.d.)

1.5.2.4 OLE (Off-road Loescheinheit)

O OLE (aparelho de extinção "*off-road*") é um robô de combate a incêndios autónomo com a forma de um inseto com seis pernas desenvolvido na Universidade de Magdeburg-Stendal na Alemanha (Dumiak, 2008). O seu *design* foi inspirado no bicho-de-conta, um inseto da família da centopeia que tem a capacidade de se enrolar (ver Figura 1.24).

Este robô inseto tem como objetivo patrulhar uma área de floresta específica e procurar por incêndios. No caso de encontra uma potencial ameaça irá notificar os bombeiros, bem como tentar apagar o fogo por si só.

Para esse efeito, o robô foi equipado com tanques para a água e agentes de extinção em pó, biossensores, sensores de infravermelhos e de calor, sendo guiado por GPS.

Figura 1.24 - Robô OLE. Fonte: (Dobrow, s.d.)

O robô tem a capacidade de se enrolar e de retrair as suas pernas quando pressente o perigo, de modo a que o seu escudo à base de fibra de cerâmica resistente ao fogo o proteja.

Os *designers* têm sugerido duas maneiras diferentes para que OLE realize o seu trabalho. Uma ideia é a de colocar os robôs em potenciais *hotspots* perto de cidades e acampamentos, onde permaneceriam fechados, à espera que os seus sensores detetem um incêndio dentro de um raio de meia milha. Outra ideia é que o robô patrulhe as florestas, à procura de chamas, embora a vida das baterias e obstáculos na floresta limitem o seu alcance.

1.5.2.5 SAFFiR (Shipboard Autonomous Firefighting Robot)

O SAFFiR (Figura 1.25) é um protótipo de um robô humanoide autónomo de combate a incêndios com o objetivo de localizar e suprimir incêndios dentro de navios e estruturas (Naval Research Laboratory (NRL), 2014).

Esta investigação está dividida em três partes: desenvolvimento da plataforma robótica inovadora e de materiais resistentes ao fogo que está ao cargo do Instituto Politécnico da Virginia *(Virginia Tech)*; algoritmos de perceção e navegação autónoma, da responsabilidade da Universidade da Pensilvânia; tecnologia de interação homem-robô, e modelos computacionais cognitivos que irão permitir que o bombeiro robótico possa trabalhar e interagir naturalmente com os bombeiros navais, temas que são estudados no NCARAI (*Navy Center for Applied Research in Artificial Intelligence*).

O robô é composto com tecnologia de sensor multimodal desenvolvido para navegação avançada e um conjunto de sensores que inclui uma câmara, sensor de gás e câmaras infravermelho e ultravioleta que lhe permitem captar imagem através do fumo e detetar fontes de calor excessivo.

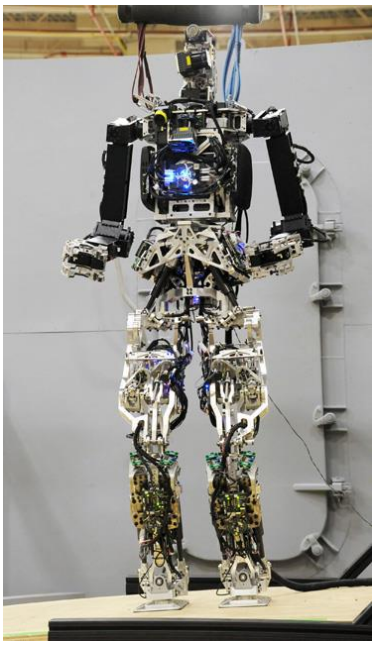


Figura 1.25 - Protótipo do robô SAFFiR. Fonte: (Naval Research Laboratory (NRL), 2014)

O robô está a ser projetado para se movimentar de forma autónoma ao longo de um navio de modo a interagir com as pessoas, construir um modelo do ambiente, patrulhar em busca de anomalias estruturais e desempenhar tarefas de combate a incêndios perigosas para o seres humanos.

1.5.2.6 Raposa

Ao contrário dos outros robôs anteriormente referidos, o robô Raposa foi projetado para de salvamento. É um robô semiautónomo português, desenvolvido pelo Instituto de Sistemas e Robótica / Instituto Superior Técnico de Lisboa em parceria com a IdMind-Engenharia de Sistemas, Lda.

O robô Raposa foi projetado e construído para operar em ambientes externos e hostis para a presença do homem, tais como escombros resultantes de colapsos de estruturas. A estrutura mecânica do robô é composta por um corpo principal e um corpo à frente e a sua locomoção é assegurada por um sistema de lagartas, que permite o seu movimento mesmo estando invertido (Marques, et al., 2006).

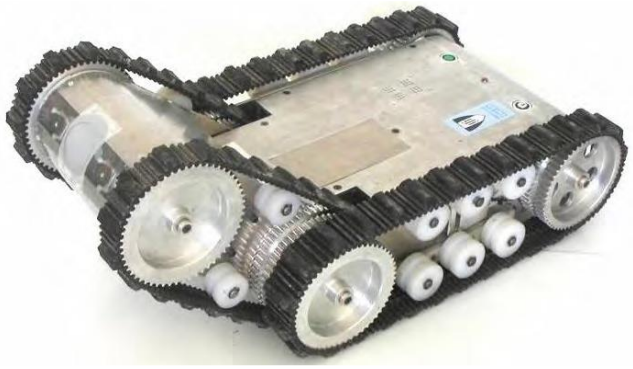


Figura 1.26 – Robô Raposa. Fonte: (Marques, et al., 2006)

O corpo da frente contém uma inclinação variável, que permite ultrapassar obstáculos mais altos que o corpo principal do robô (por exemplo, subida de escadas). O robô está equipado com uma câmara termográfica e duas *webcams* e sensores de gás, temperatura e humidade. O robô utiliza comunicação *wireless* com a opção de utilizar cabo. A comunicação por cabo permite também a alimentar o robô, que é efetuada pelo ponto de acesso situado na traseira do robô. A ligação por cabo é conseguida através do operador remoto e com o auxílio de uma câmara localizada no interior do robô.

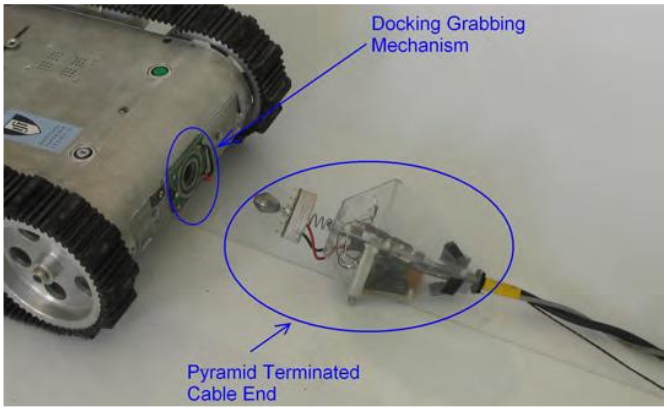


Figura 1.27 - Detalhe da ligação por cabo. Fonte: (Marques, et al., 2006)

1.6 Organização da Dissertação

Esta dissertação está organizada da seguinte forma:

No segundo capítulo é apresentado o conceito do robô desenvolvido, partindo do seu objetivo (combate a incêndios) e com isso selecionar os sensores e atuadores a utilizar e as suas respetivas ligações e interações entre si.

O terceiro capítulo é dedicado a estudos e ensaios preliminares realizados com os sensores e atuadores implementados. Tem como objetivo testar as comunicações dos sensores e atuadores e analisar a sua aplicação.

No quarto capítulo é abordado com maior enfase o sistema de navegação do robô, dando destaque à cinemática do robô móvel e metodologias utilizadas no mapeamento e planeamento.

O quinto capítulo é apresentado os resultados obtidos nos vários cenários de simulação e as respetivas conclusões de cada ensaio.

Por fim no sexto capítulo é dedicado para avaliar o trabalho realizado e apresentar sugestões para o trabalho futuro.

2 PROTÓTIPO

Um robô é um sistema mecatrónico, como tal, é compostos por três áreas principais: a estrutura mecânica (*chassis*), os sistemas eletrónicos (*hardware*) e a programação para perceção, navegação e controlo (*software*). Sendo nestas três áreas que se irá debruçar o presente capítulo.

2.1 Conceito

O robô projetado é um robô móvel autónomo de pequenas dimensões, que tem como principal objetivo apagar uma chama de vela. Para tal, o robô é dotado com a capacidade de navegar por corredores e quartos no cenário de simulação, evitando obstáculos, ser capaz de detetar chama e por fim extingui-la.

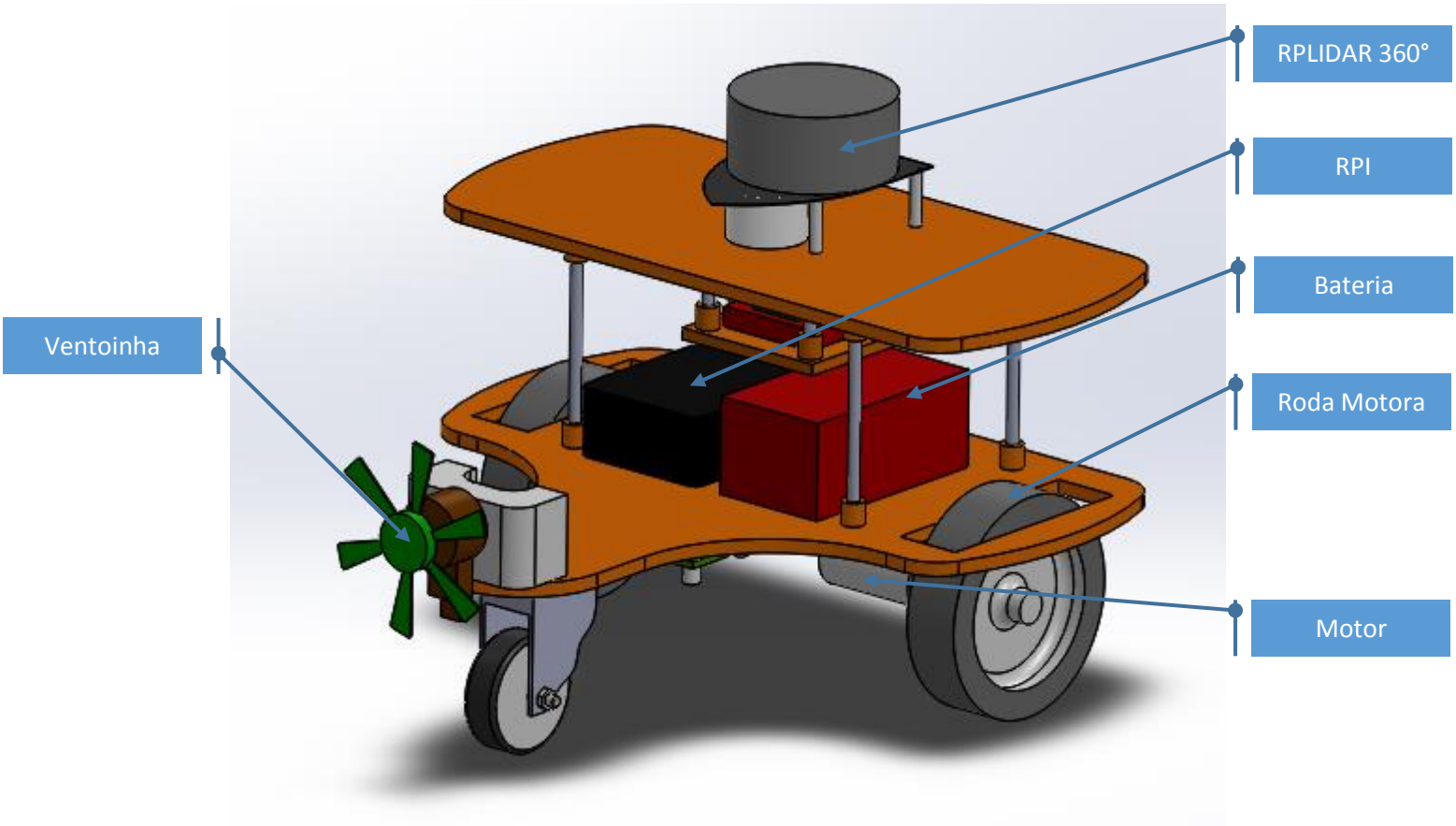


Figura 2.1 - Modelação 3D do robô bombeiro realizada em Solidworks®.

Para atingir o seu objetivo (extinguir a chama) o robô necessita de um conjunto de sensores e atuadores, de modo a que o robô seja capaz de "sentir" e "atuar" no ambiente em que se encontra. Para esse efeito, é necessário que exista algum tipo de comunicação entre a

informação recebida dos sensores e a informação transmitida aos atuadores. Ou seja, é necessário um componente intermediário que permita processar a informação de maneira correta. A este componente dá-se o nome de Unidade de Processamento. Nesta dissertação para a função de Unidade de Processamento foi escolhido um Raspberry Pi® (RPI).

Este robô contém dois sistemas distintos, o de navegação e o de deteção e extinção de chama. Estes estão representados na Figura 2.3 sendo que as ligações a vermelho representam o sistema de navegação e a verde o sistema de deteção e extinção da chama.

O sistema de navegação do robô é constituído por um *rangefinder* RPLIDAR 360°® para realizar mapeamento que fusionado com odometria permite realizar a navegação do robô através de SLAM (*Simultaneous Localization And Mapping*).

O sistema de deteção e extinção de chama é constituído por uma câmara termográfica FLIR LEPTON ® (FL) e uma ventoinha, responsável pela extinção da chama.

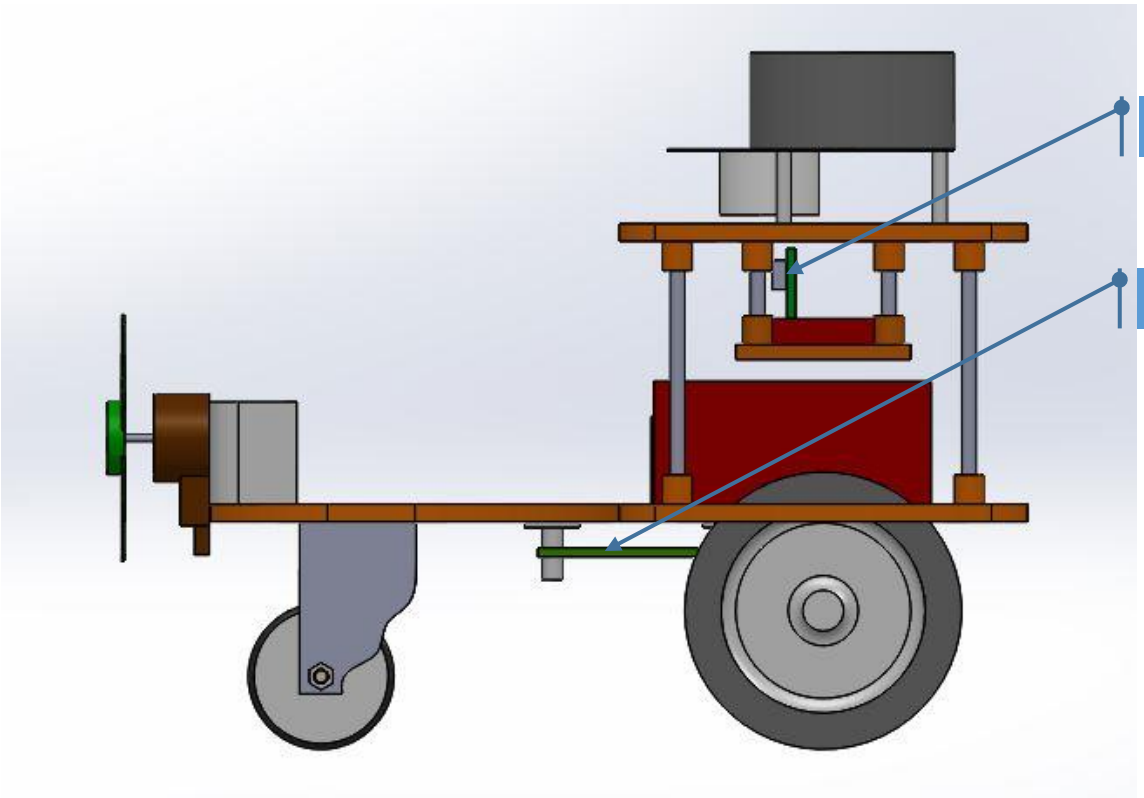


Figura 2.2 - Modelação 3D do robô bombeiro realizada em Solidworks® - vista lateral.

No sistema de navegação, inicialmente o RPI recebe informação do RPLIDAR 360° ®, obtendo uma primeira informação do ambiente em redor. Com base nessa informação, o RPI comanda os motores de modo a que o robô se movimente, esse movimento é medido através de odometria. O que permite dar novas informações ao robô acerca da sua posição e assim sucessivamente. O sistema de navegação será apresentado em detalhe no capítulo 4.

O sistema de deteção e extinção da chama só é acionado quando o robô entra num quarto. Quando este sistema é ativado a câmara termográfica FL recolhe informação sobre a existência da chama. Quando a chama for detetada, o RPI comanda a ventoinha realizar a sua extinção.

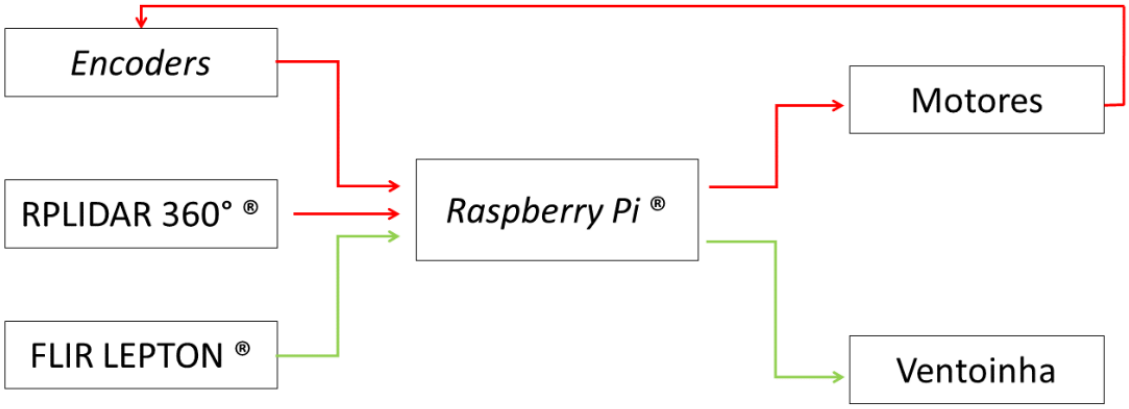


Figura 2.3 – Diagrama de fluxo de informação.

O modo de funcionamento do robô pode ser explicado através de um fluxograma que, de uma forma expedita, permite perceber as ações que o robô toma ao longo da sua missão (Figura 2.4 e Figura 2.5). Numa primeira fase o robô realiza um *scan* com o sensor RPILIDAR 360° para obter um mapa do ambiente em redor e de seguida move-se consoante a melhor opção. Após o movimento o robô recebe dados referentes à sua posição. Com base na sua posição e o mapa local através do *scan* efetuado, o robô depara-se com a seguinte questão: "Estou num quarto?", com base na resposta a essa questão o robô irá realizar a sua ação. Caso a resposta seja positiva (condição lógica verdadeira) o robô começa a tentar detetar a chama através da câmara termográfica FL, caso seja negativa (condição lógica falsa), o robô repete a sequência feita anteriormente (ciclo *repeat until*). Caso exista a chama dentro do quarto o robô ir-se-á aproximar da chama e extingui-la, caso contrário, o robô repete a instruções anteriores até encontrar o quarto onde se encontra a chama.

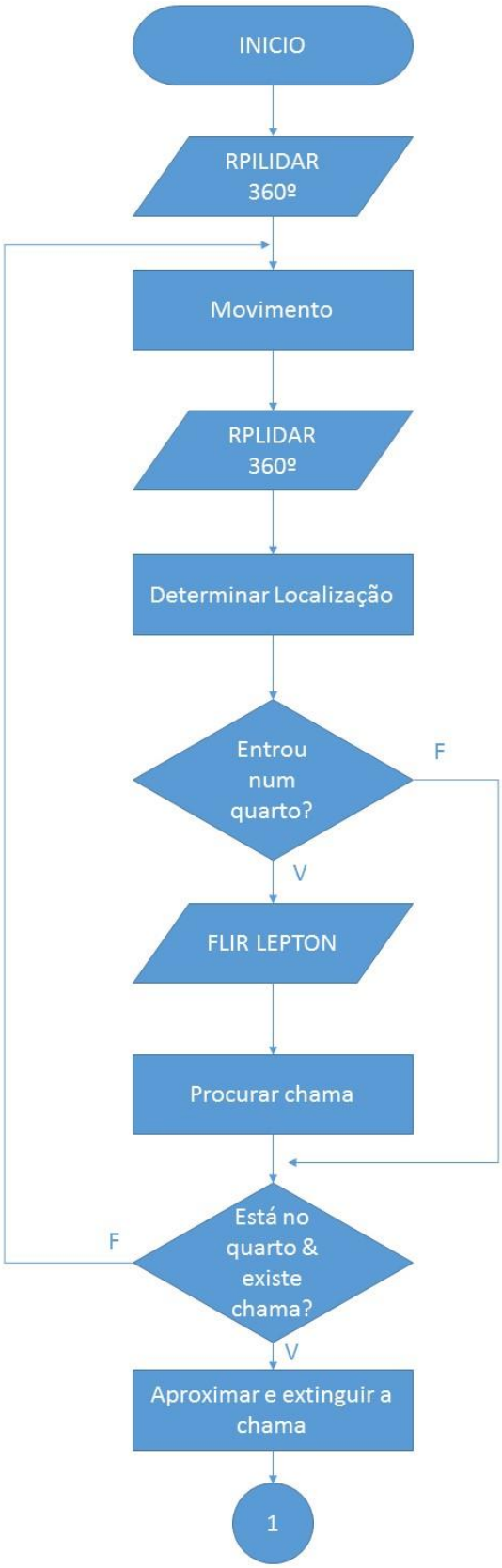


Figura 2.4 - Fluxograma geral do robô (parte 1).

Pelas regras do concurso para o qual se está a preparar o robô para participar, após a extinção da chama o robô tem de voltar ao ponto de partida (Figura 2.5).

De modo a que o robô consiga planear o caminho de volta, tem de realizar *scans* e leituras de odometria de modo a determinar a sua posição sucessivas vezes até chegar ao ponto de partida, realizando assim um novo ciclo *repeat until*.

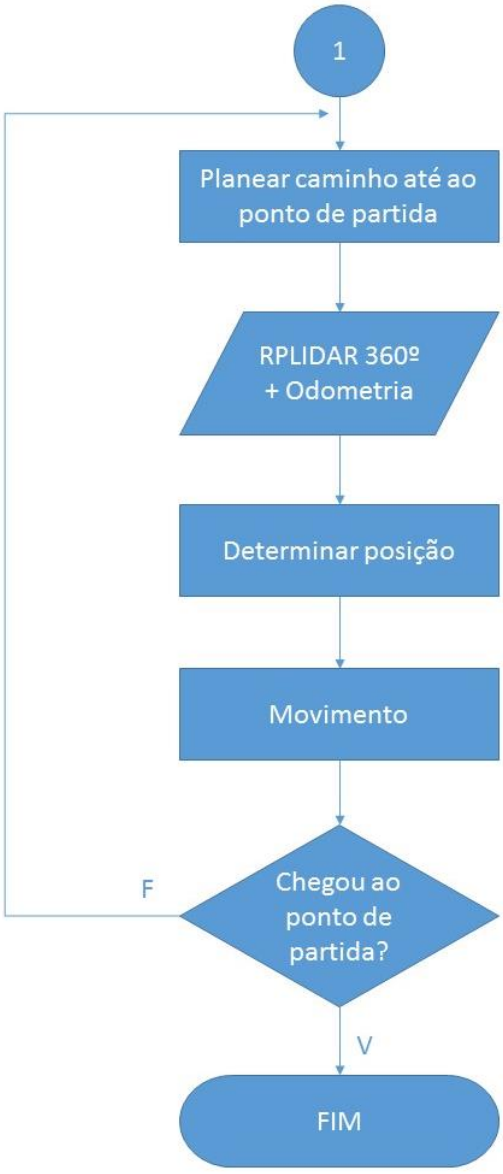


Figura 2.5 – Fluxograma geral do robô (parte 2).

2.2 Chassis

O *chassis* é a estrutura base onde se realiza a montagem do *hardware*, de modo a otimizar o espaço/volume ocupado e que se consiga uma colocação dos sensores em locais estratégicos para obter as melhores leituras possíveis.

Normalmente os *chassis* dos robôs móveis têm a forma circular ou quadrangular/retangular. Considerando as duas formas, com o diâmetro da forma circular igual ao comprimento da forma quadrangular. Através da Figura 2.6, verifica-se que um robô com chassis quadrangular tem um maior risco de ficar preso por um obstáculo ou de não conseguir encontrar o caminho pelo caminho estreito. Por outro lado, o robô com chassis circular consegue encontrar o caminho pelo caminho estreito, pois é capaz de rodar à frente do obstáculo sem ficar preso. Enquanto que o robô com *chassis* quadrangular necessita de recuar e só depois rodar, e mesmo assim não é garantido que o robô não fique preso.

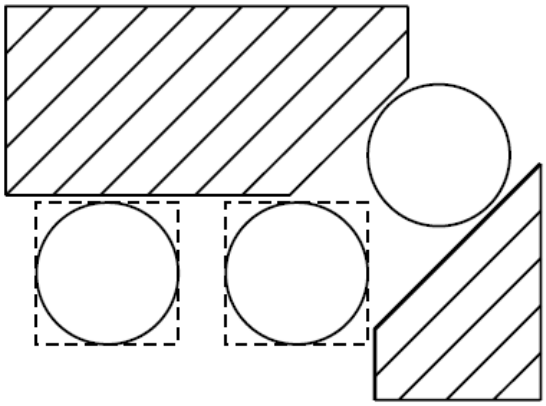


Figura 2.6 – Esquema de navegação do robô com *chassis* circular vs *chassis* retangular. Fonte: (Goris, 2005)

Numa primeira fase, tendo em conta ambas as possibilidades, foi escolhida a forma circular por ser aquela que oferece ao robô uma maior manobrabilidade – Figura 2.6.

A nível de restrições o *chassis* do robô é restringido a nível dimensional devido às regras do Concurso "Robô Bombeiro", referidas anteriormente. Tendo essa informação partiu-se para a primeira forma do chassis:

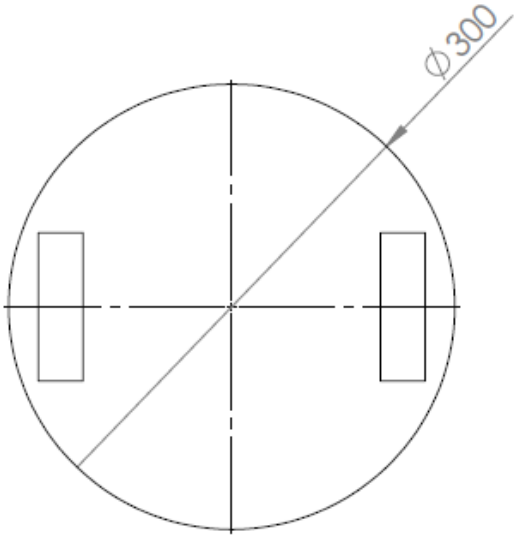


Figura 2.7 – Esboço da primeira forma do *chassis*.

Considerando esta forma banal, decidiu-se criar uma forma diferente, original, estética e funcional, acrescentando também a ideia associada à poupança na área ocupada (representada pela área cortada na Figura 2.8). Tendo esses conceitos em mente, chegou-se a uma nova forma:

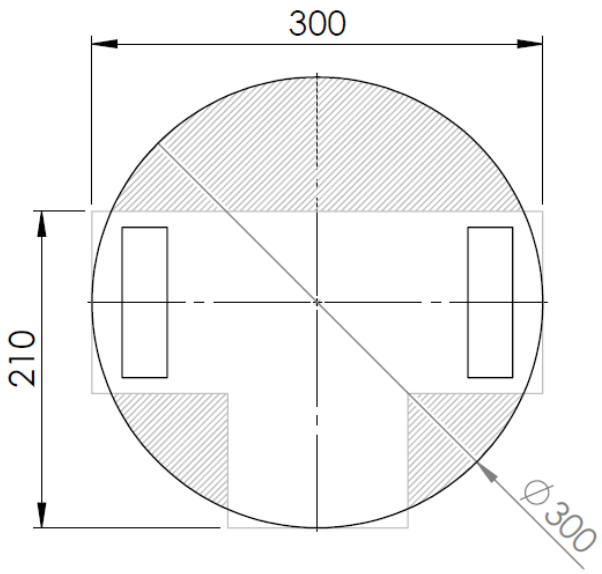


Figura 2.8 – Esboço da segunda forma do *chassis*.

Tal como referido anteriormente, as formas circulares são as mais adequadas, pois evitam que o robô fique preso, o que levou ao arredondamento da estrutura. Nota-se na Figura 2.8 que o

chassis tem um comprimento de 210mm, o que permite alongar um pouco mais a estrutura. Com base nessas informações foi possível chegar à versão final do chassis (Figura 2.9):

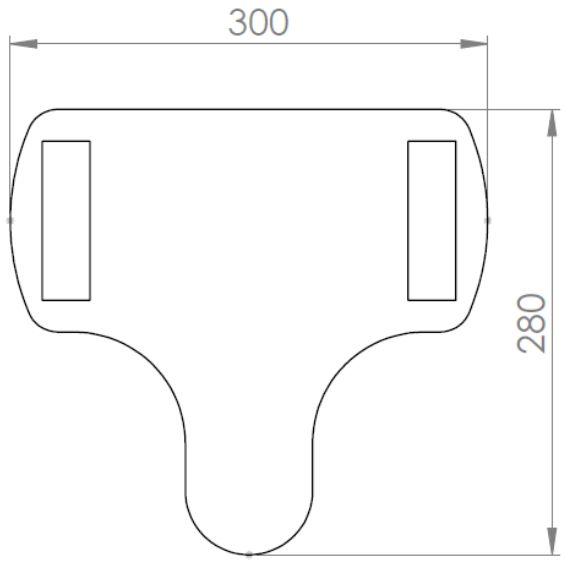


Figura 2.9 – Esboço do *chassis* otimizado tendo em conta as restrições.

Esta forma permite otimizar a área ocupada pelo *chassis*, sem descuidar do posicionamento do *hardware* acoplado, e oferece uma proteção às rodas contra embate a paredes e/ou obstáculos devido a estares localizadas no interior do *chassis*. Os desenhos técnicos com detalhes da estrutura do robô situam-se no Anexo 1.

2.2.1 Protótipo Virtual

Tendo o conceito do *chassis* definido, decidiu-se modelar o *chassis* do robô recorrendo ao *software* SolidWorks® de modo a obter-se um protótipo virtual - Figura 2.10. Este surgiu com a necessidade de averiguar o espaço/volume ocupado pelo *hardware*, de modo a respeitar os constrangimentos dimensionais impostos pela competição.

Apesar da base do *chassis* estar otimizada, esta não permite colocar todo o *hardware* de modo eficaz para a correta funcionalidade do robô. Exemplo disso é o *rangefinder* que não pode estar obstruído, de modo a garantir uma leitura correta. O que levou a acrescentar duas estruturas (andares), uma dedicada para a câmara FL (intermédia) e outra para o RPLIDAR 360° (superior) – ver Figura 2.10, Figura 2.11 e Figura 2.12.

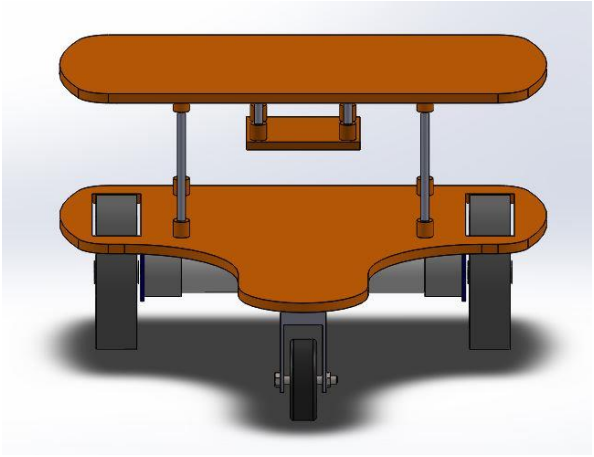


Figura 2.10 – Protótipo virtual do *chassis* do robô em SolidWorks® - vista de frente.

Com base no modelo 3D foi possível obter alguns desenhos técnicos:

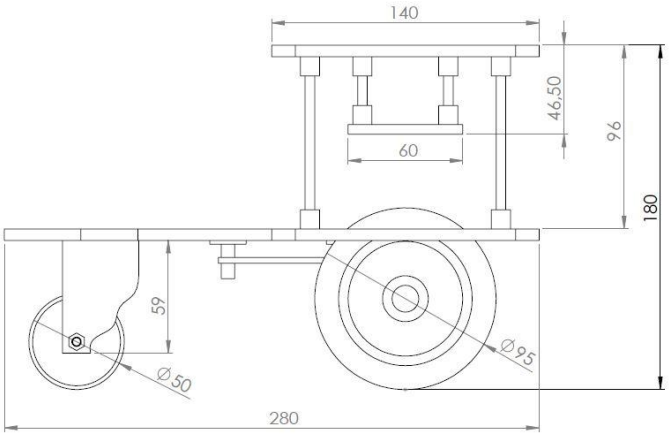


Figura 2.11 – Desenho do robô em Solidworks® - vista lateral.

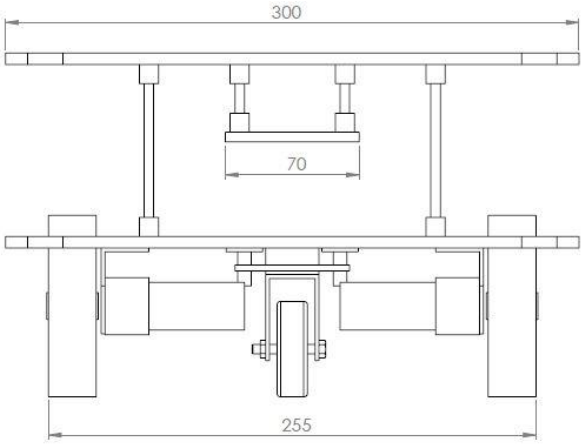


Figura 2.12 - Desenho do robô em Solidworks® - vista traseira.

2.2.2 Protótipo Físico

A construção do protótipo físico tem como finalidade ter um protótipo mais próximo possível do idealizado para realizar ensaios e confirmar se a posição dos componentes de *hardware* se adequam à função. Tendo o *chassis* modelado em Solidworks®, produziu-se o protótipo físico funcional. Para tal, recorreu-se a placas de madeira contraplacada - Figura 2.13.

Figura 2.13 - Placas de madeira contraplacada utilizadas na construção do protótipo.

Após a construção do *chassis* para o robô juntou-se o *hardware*, obtendo assim a forma anteriormente descrita - Figura 2.14.

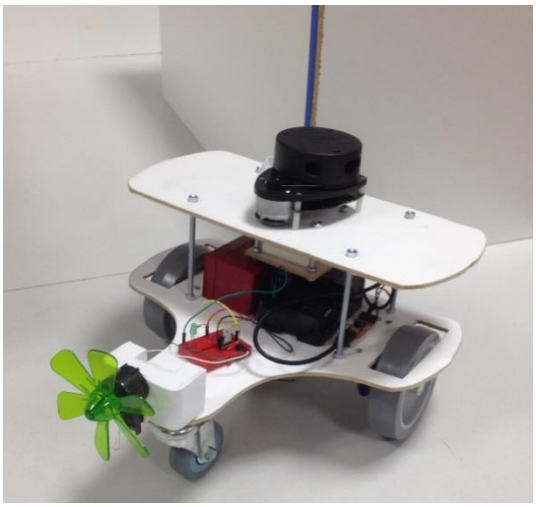


Figura 2.14 - Protótipo físico com *hardware* acoplado.

Após a acoplação do *hardware*, verificou-se que o *chassis* produzido tem espaço suficiente para o hardware a ser utilizado e que permite o funcionamento dos sensores sem causar obstruções. É importante realçar o facto de o *chassis* ainda ter ainda algum espaço livre, ficando esse destinado para uma eventual adição de *hardware*.

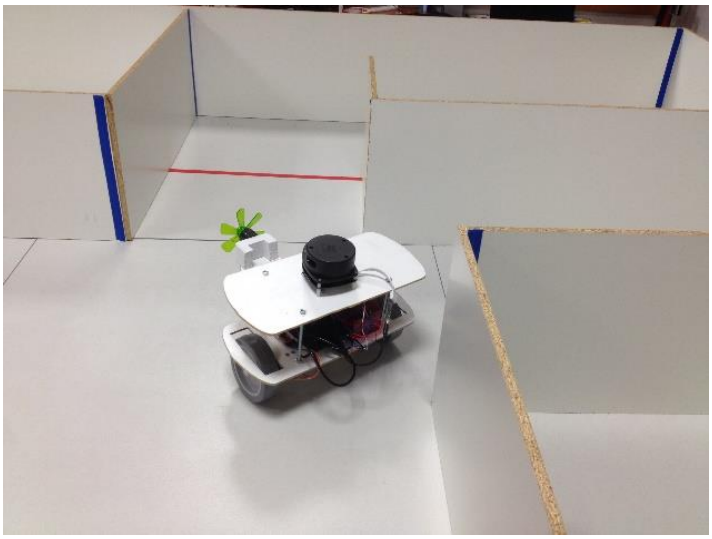


Figura 2.15 - Protótipo físico no cenário de simulação.

2.3 Hardware

2.3.1 Unidade de Processamento – Raspberry Pi

O RPI é um microcomputador desenvolvido no Reino Unido pela *Raspberry Pi Foundation*. Este equipamento surgiu em 2012 e tem sido melhorado desde então e disponibilizado em novas versões. Tendo como objetivo disponibilizar um computador simples (*user friendly*) e de baixo custo (Raspberry Pi Foundation, s.d.).

Neste projeto é utilizado o RPI 2 Modelo B que tem as seguintes características:

- Processador QUAD Core ARM Cortex-A7 @ 900 MHz;
- 1GB de RAM @ 400MHz;
- 4 Portas USB (Universal Serial Bus);
- 40 Pinos GPIO (General Purpose Input/Output);
- Porta HDMI (High-Definition Multimedia Interface);
- Porta de ligação *Ethernet*;
- *Jack* de áudio e de vídeo composto;
- Interface de câmara, *Camera Serial Interface* (CSI);
- Interface de display, *Display Serial Interface* (DSI);
- Entrada de cartão *microSD*;

Figura 2.16 - Raspberry Pi 2 Modelo B.

O RPI utiliza um sistema operativo (SO) livre (*opensource*) baseado em Linux®, denominado de *Raspbian*. Este SO consiste numa versão não oficial do *Debian* com configurações de compilação ajustadas para produzir código otimizado que é executado no RPI (Raspbian, s.d.). O que proporciona um desempenho significativamente mais rápido para aplicativos que fazem uso intenso de operações aritméticas em ponto flutuante. Todas as outras aplicações também melhoram o seu desempenho através do uso de instruções avançadas da unidade central de processamento (CPU) do RPI

No RPI qualquer linguagem que possa ser compilada sob uma arquitetura ARMv6 (RPI 1) ou ARMv7 (RPI 2) pode ser usada para o desenvolvimento de *software*. Sendo que a linguagem *Python* é a mais utilizada. (Raspberry Pi Foundation, s.d.).

Uma das grandes características do RPI é a existência de portas GPIO que estão localizadas no canto superior esquerdo da placa (Figura 2.16). As ligações GPIO são constituídas por 40 pinos, duas fileiras de 20, tendo cada pino a sua finalidade, com vários pinos a trabalharem em conjunto permite formar circuitos particulares. A disposição e os respetivos sinais das portas GPIO podem ser vistas na Figura 2.17.

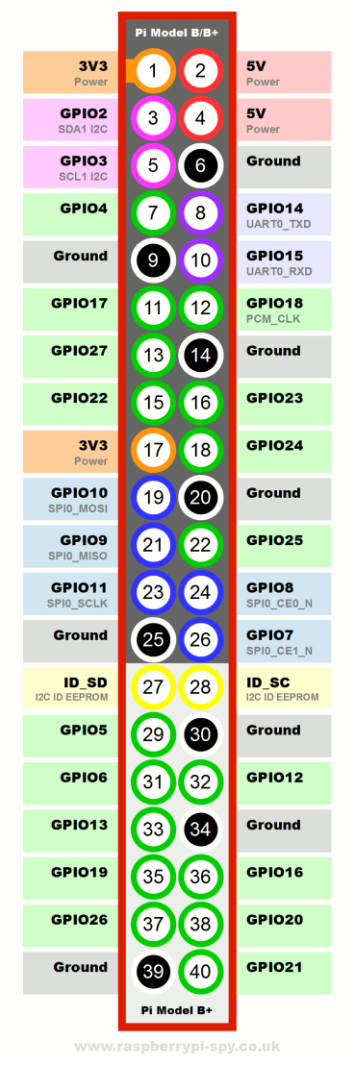


Figura 2.17 - Diagrama dos pinos GPIO para o modelo RPI 2 B. Fonte: (Hawkins, 2012)

Para além destes pinos de uso geral, as portas GPIO têm pinos dedicados a alguns barramentos: (Upton & Halfacree, 2012)

- Série UART (*Universal Asynchronous Receiver/Transmitter*), um módulo que contém, ao mesmo tempo, os circuitos de transmissão e receção necessários para as comunicações série assíncronas;
- SPI (*Serial Peripheral Interface*), um protocolo de dados série síncronos utilizado para comunicação entre controlador e um ou mais periféricos que também pode ser utilizado entre dois controladores;
- I²C (*Inter-Integrated Circuit*), um barramento de comunicação criada pela *Philips*, para *chips* para se comunicarem uns com os outros;

O uso de qualquer um dos pinos exige cuidado, pois possuem um nível lógico de 3.3V e não são tolerantes a níveis de 5V, já que são ligados diretamente ao *chip Broadcom* o que significa que não existem proteções contra picos de corrente.

2.3.2 Sistema de perceção

2.3.2.1 RPLIDAR 360°

RPLIDAR 360° é um *laser scan* omnidirecional 2D de baixo custo com a capacidade de realizar leituras nos 360°, em torno do robô, com um alcance de deteção superior a 6 metros. Consiste num sistema de *laser* de medição por triangulação que pode trabalhar em todos os tipos de ambientes internos e externos. O *output* do sensor é uma nuvem de pontos que pode ser utilizada para mapeamento, localização ou ambas em simultâneo (SLAM) e modelação do ambiente (RoboPeak, 2014).

Figura 2.18 - *Rangefinder* RPLIDAR 360°. Fonte: (RoboPeak, 2014)

2.3.2.2 FLIR LEPTON

A FL, em que FLIR significa *Forward Looking Infrared*, é uma câmara infravermelha de banda larga concebida para interagir facilmente com dispositivos móveis e outros aparelhos eletrónicos. Esta câmara recebe a radiação infravermelha, emitida pelo ambiento em redor, transformando-a por sua vez, em valores que estão relacionados com a temperatura (FLIR, 2014).

Posteriormente, como esses valores são relativos a cada *pixel* (80x60) captado pela FL e são armazenados sob a forma de *array*, através de processamento de imagem é possível criar uma imagem térmica uniforme.

Figura 2.19 – Câmara termográfica FL com instrumentação de ligação. Fonte: (FLIR, 2014)

2.3.2.3 Encoders

Os encoders são sensores internos, que permitem medir o número de revoluções realizadas cada roda, que por sua vez permite estimar a localização do robô - odometria.

Apesar da sua simplicidade, a odometria está sujeita a erros, pois a integração de informação sobre movimentos incrementais leva à acumulação de erros. Razão pela qual, este sensor raramente ser o sensor principal num sistema de navegação. Estes erros podem ser sistemáticos ou não sistemáticos (Ribeiro, 1999).

Figura 2.20 - Erro acumulativo de localização de um robô decorrente de pequenos erros de odometria causados pelo escorregamento das rodas. Fonte: (Thrun, 2002)

Os erros sistemáticos são causados por características do robô e/ou dos sensores, como por exemplo: diâmetro desigual ou desalinhamento das rodas, diâmetro das rodas diferente do valor nominal, incerteza sobre o ponto de contacto da roda.

Erros não sistemáticos são causados pela relação do robô com o ambiente: movimento sobre solos não uniformes, movimento sobre obstáculos inesperados no solo, escorregamento das rodas (solo escorregadio, grandes acelerações do veículo, rotações rápidas, etc.).

Nesta dissertação, os encoders utilizados estão acoplados aos motores, fazendo parte de um *kit* de locomoção (designação comercial: RD02). Que é constituído por duas rodas motoras com encoder (EMG30) e uma placa controladora (MD25).

2.3.3 Atuadores

2.3.3.1 Motores

Os motores são os atuadores que definem as velocidades dadas às rodas, consoante o cenário a que o robô está sujeito (informação proveniente dos sensores). A sua comunicação é efetuada através da placa controladora MD25, que recebe informação proveniente do RPI que por sua vez transmite aos motores.

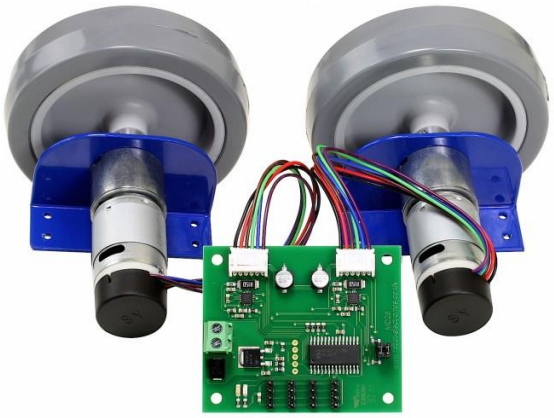


Figura 2.21 - *Kit* de locomoção utilizado (RD02). Fonte: (Robot Electronics, 2016)

2.3.3.2 Ventoinha

A ventoinha é o atuador responsável pela extinção da chama. Este atuador entra em funcionamento após ter-se conhecimento da localização da chama e o robô estar a uma distância que permita a sua extinção.

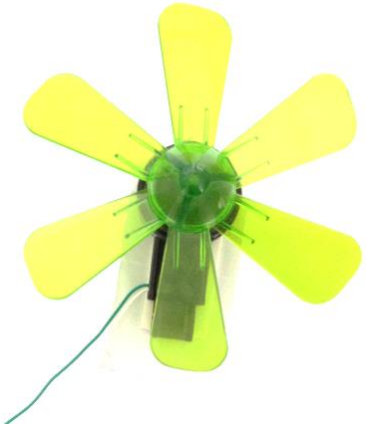


Figura 2.22 - Ventoinha utilizada como agente de extinção.

2.3.4 Arquitetura de Hardware

Sabendo os componentes de *hardware* que fazem parte do robô é possível criar um esquema intuitivo que permite explicar a interação entre os sensores, atuadores e a unidade de processamento – Arquitetura de *Hardware*.

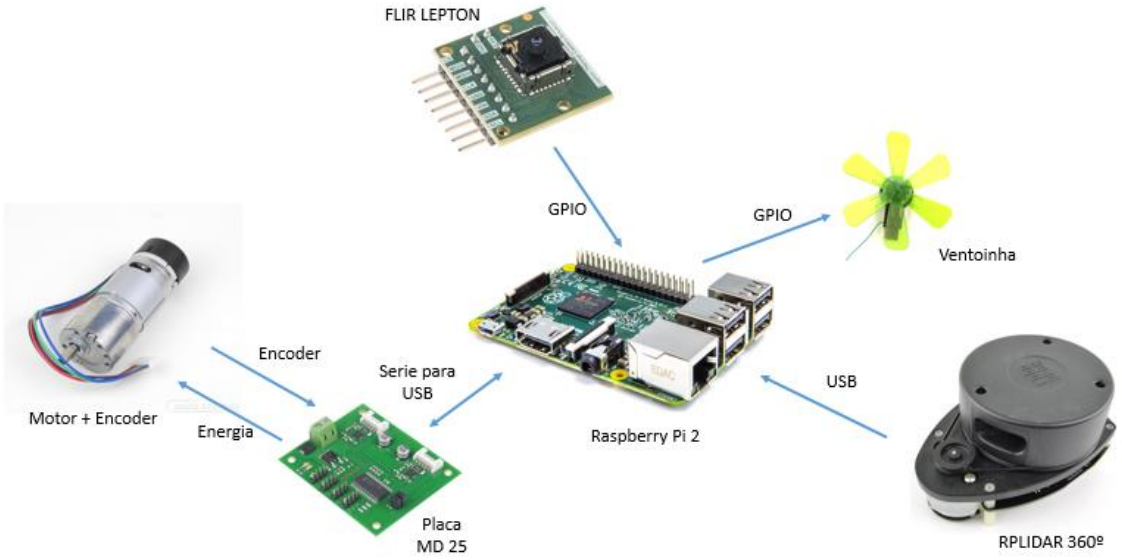


Figura 2.23 - Arquitetura de Hardware.

O RPI está ligado ao RPLIDAR 360° através de ligação USB, onde o RPI recebe do *rangefinder* um conjunto de pontos que caracterizam os obstáculos em seu redor. A câmara termográfica FL está ligada ao RPI através dos pinos GPIO e esta devolve um *array* com valores relacionados com a temperatura que está na sua área de visão.

Por outro lado, os motores têm uma ligação diferente dos componentes anteriores, pois estes estão acoplados aos encoders O RPI comunica com a placa MD25 enviando referências de velocidade para os motores e recebendo as contagens de picos de encoder. Por sua vez a placa MD25 gera sinais de tensão a aplicar aos motores e lê os respetivos sinais dos encoders.

Uma outra ligação está relacionada com a ventoinha (sistema de extinção de chama) sendo necessário apenas um comando por parte do RPI para atuar, não recebendo uma resposta de volta por parte da ventoinha.

2.4 Software

Usualmente o RPI é programado através da linguagem de programação *Python*. Esta linguagem é classificada como de alto nível orientada a objetos e vem instalada por defeito no RPI para o desenvolvimento de aplicações. É administrada pela *Python Software Foundation*, sendo um recurso com código aberto podendo até ser utilizada gratuitamente até para fins comerciais (Lutz & Ascher, 2004).

Tem como principais características a sua sintaxe concisa e clara, a portabilidade, ou seja, consegue correr em vários sistemas operativos tais como o Windows®, Linux®/Unix e Mac OS X®. Outro aspeto pelo qual é bastante utilizada, é a produtividade. O código em *Python* normalmente tem de ⅕ a ⅓ do tamanho do código equivalente em *C++* ou *Java* o que significa menor digitação, menor depuração e menor manutenção após o desenvolvimento. Outro fator deve-se aos programas em *Python* serem executados imediatamente, sem as etapas de compilação e vinculação de outras ferramentas.

A programação em *Python* é facilitada devido à existência de fóruns de discussão na *Internet*, que permitem encontrar informações e tutoriais sobre a estrutura da linguagem, funções e bibliotecas disponíveis. A versão utilizada para o desenvolvimento deste robô foi o *Python* 2.7.

Na programação em *Python* destacam-se as seguintes bibliotecas utilizadas: *cv2*, *numpy*, *math*, e *pylepton*. *Cv2* é uma biblioteca de processamento de imagem que permite tratar a imagem recebida pela câmara termográfica de acordo com o objetivo desejado. As bibliotecas *numpy* e *math* são bibliotecas matemáticas que concedem a capacidade de trabalhar com matrizes e de operações matemáticas, como por exemplo, funções trigonométricas. A biblioteca *pylepton* é uma biblioteca projetada com o intuito de permitir uma ligação entre a câmara termográfica FLIR LEPTON ® e o RPI.

Outro *software* utilizado foi o Matlab® com o objetivo de implementar o sistema de navegação onde ao contrário dos outros sistemas o seu processamento é realizado num PC (*Personal Computer*). O Matlab® é uma ferramenta informática, interativa e de alta performance, orientada à execução de tarefas que envolvem cálculos numéricos. Este *software*, para além de disponibilizar uma linguagem de programação própria, fornece um ambiente de computação com excelentes capacidades gráficas e com um vasto conjunto de funções, organizadas segundo diversas áreas científicas (Moraes & Vieira, 2013).

O Matlab® é uma ferramenta que armazena dados em variáveis baseadas em matrizes. Têm a particularidade de poder atribuir dados às matrizes, sem que tenha de fazer uma pré-definição de variáveis. Esta característica e o grande conjunto de funções internas da ferramenta permite

solucionar problemas computacionais de uma forma mais rápida, simples e compacta do que quando desenvolvidas numa das linguagens de programação mais tradicionais como *C*, *Fortan* ou *Pascal*.

Apesar de existirem funções internas no Matlab®, foi necessário recorrer a outras *toolboxes.* De modo a conseguir implementar o desejado, entre elas estão as *toolboxes* "*Robotics*" e "*Machine Vision*" criadas por Corke (2011).

3 ESTUDOS PRELIMINARES

Neste capítulo serão abordados estudos/ensaios realizados com os sensores e atuadores, de modo a permitir um maior aprofundamento do conhecimento sobre o funcionamento do *hardware*, a sua comunicação e programação. Estes estudos serão realizados antes de se efetuar os ensaios com todo o *hardware* acoplado no robô. Este capítulo está dividido em quatro secções: a primeira aborda o controlo dos motores através de comunicação com o RPI (sistema de locomoção); a segunda aborda a utilização da câmara FL para detetar chamas (sistema de deteção de chama); a terceira aborda a ligação entre o RPI e a ventoinha e por fim a obtenção de mapas através do sensor *rangefinder* RPLIDAR 360° e o tratamento dos dados adquiridos (sistema de navegação).

3.1 Sistema de Locomoção

Este estudo consiste em efetuar a comunicação entre o RPI e os motores. Para tal desenvolveu-se um robô com a funcionalidade de seguir uma linha de cor vermelha (Figura 3.1). Este protótipo foi equipado com uma *webcam* de ligação USB, RPI, os motores e as respetivas alimentações (*powerbank* e bateria).

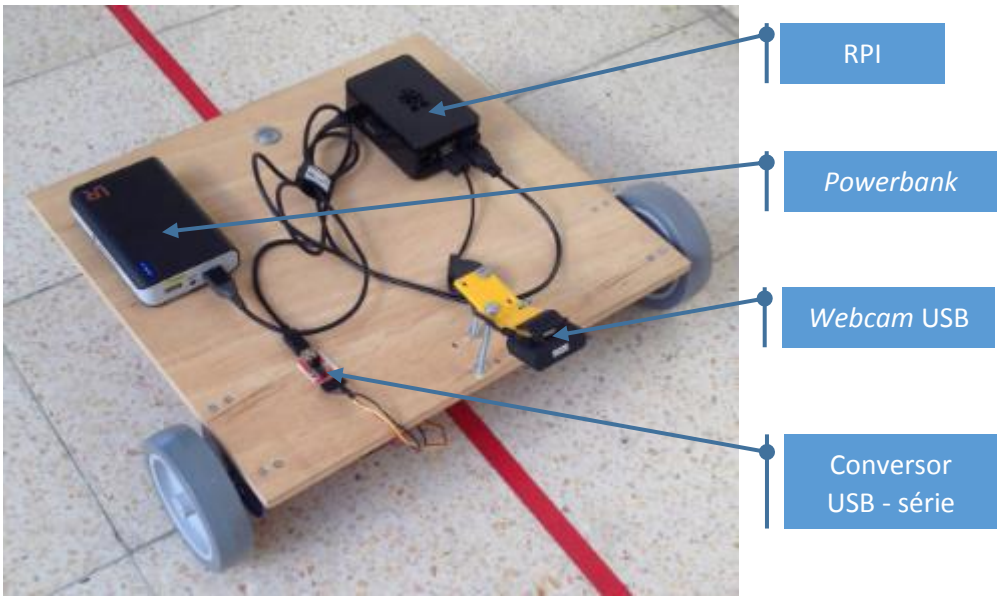


Figura 3.1 - Robô seguidor de linha.

A comunicação entre os motores e o RPI foi efetuada através da placa controladora MD25 que tem como função receber comandos por parte do RPI e transmiti-los aos motores. Para esse

efeito foi realizada uma ligação série entre o motor e a placa controladora e depois um conversor série - USB para a transmissão de informação da placa controladora para o RPI (Figura 3.2).

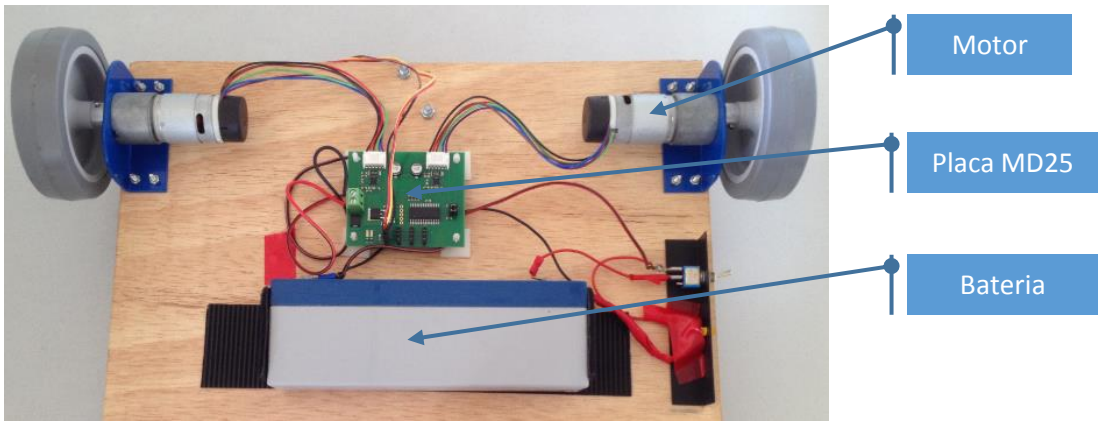


Figura 3.2 - Ligações efetuadas entre placa controladora, motores e a sua alimentação.

A comunicação entre o RPI e a placa MD25 foi realizada através de envio de comandos específicos (pré-definidos) por parte do RPI de modo a que a placa compreende-se a instrução dada. Estes comandos iniciam-se com o envio de um byte de sincronização com o valor 0 e, em seguida, o próximo comando por outros dados de bytes. A placa MD25 irá responder caso o comando seja aplicável. Na lista de comandos de utilizados foram utilizados os seguintes:

Comando	Nome	Bytes enviados	Bytes recebidos	Descrição
0x23	GET ENCODER 1	2	4	Contagem do encoder do motor 1
0x24	GET ENCODER 2	2	4	Contagem do encoder do motor 2
0x31	SET SPEED 1	3	0	Definir velocidade para o motor 1
0x32	SET SPEED 2	3	0	Definir velocidade para o motor 2
0x35	RESET ENCODERS	2	0	Põe ambos os encoders a zero

Tabela 3.1 – Lista de comandos utilizados para a placa MD25.

De modo a que o robô seguisse a linha foi necessário realizar um tratamento à imagem recebida pela *webcam* (*frame*), neste caso utilizou-se um algoritmo de segmentação por *threshold* na matriz RGB da imagem original. Caso os valores da matriz forem superiores a um determinado valor estipulado, esses valores são substituídos por 1 caso seja inferior são substituídos por 0, criando uma nova matriz só com valores de 0 e 1 (binarização).

Com esta nova matriz, obtém-se uma imagem a preto e branco (*mask*) que é interpretada pelo robô e que o permite "ver" a trajetória correta (Figura 3.3).

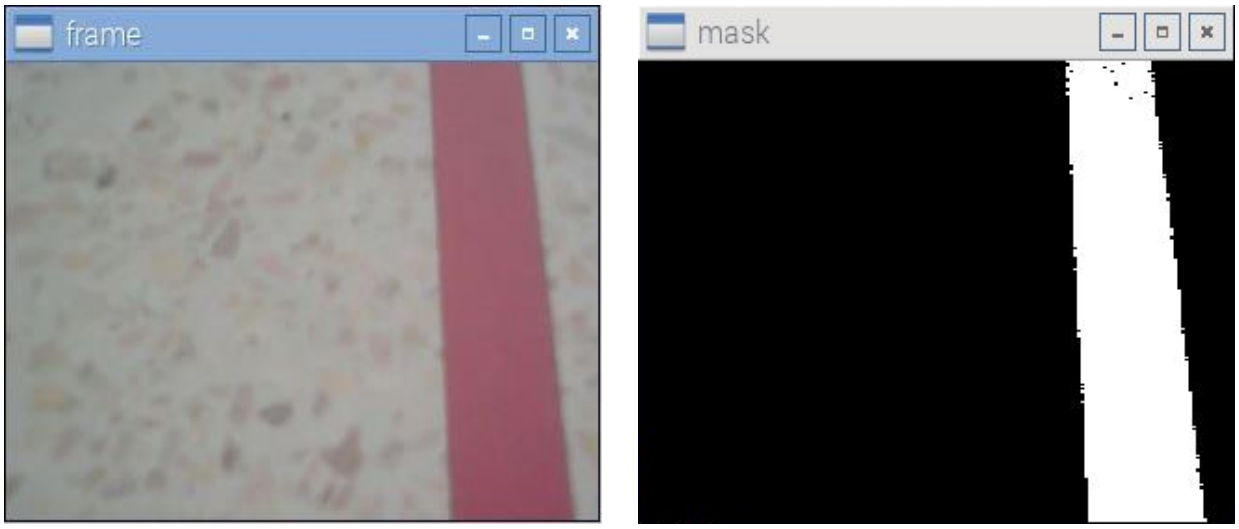


Figura 3.3 – Imagens iniciais: captada pela *webcam* (à esquerda), após binarização (à direita).

De modo a conseguir controlar os motores e definir as velocidades a serem dadas a cada roda com o objetivo de definir a postura do robô, recorreu-se ao cálculo da coordenada do centróide referente à linha branca da imagem binária (*mask*).

Para realizar o objetivo desejado (controlar a postura do robô) apenas é necessário ter em conta a coordenada da direção em x. Esta coordenada é dada pela seguinte expressão:

$$x_c = \frac{\sum_{i=1}^{n} A_i \cdot x_i}{\sum_{i=1}^{n} A_i} \qquad (3.1)$$

Em que $A_i$corresponde à área do componente i e x_i corresponde ao centróide do componente i.

A metodologia utilizada para o controlo dos motores consiste em alinhar o centro da imagem captada pela *webcam* com o centróide da linha, com base na dispersão entre ambos é dada instruções aos motores para corrigir o alinhamento e assim a postura do robô (ver Figura 3.4).

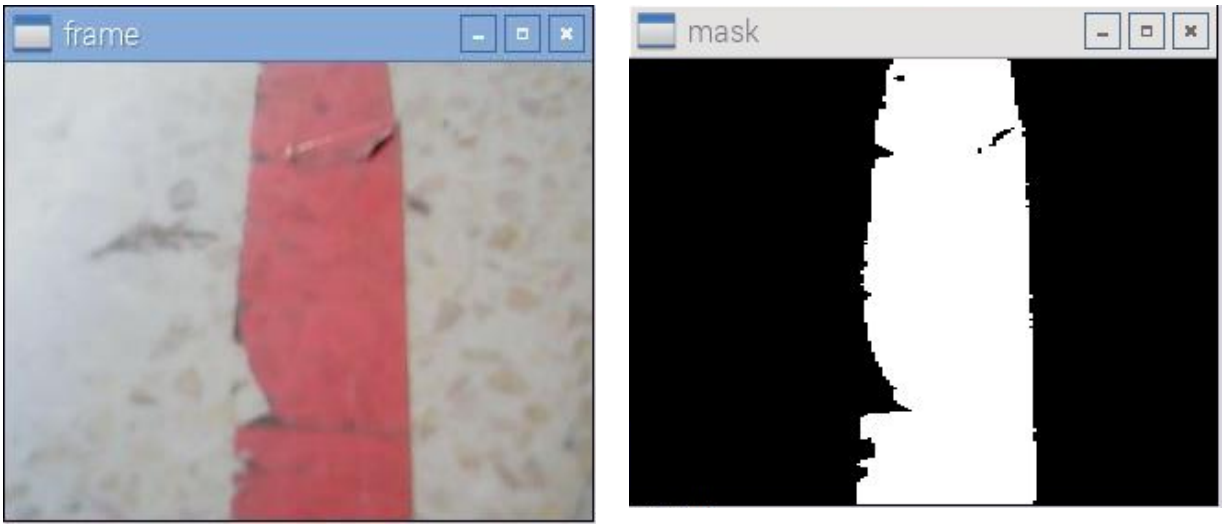


Figura 3.4 - Imagens após correção de postura: captada pela *webcam* (à esquerda), após binarização (à direita).

3.2 Sistema de Deteção de Chama

Este estudo consiste na utilização da câmara termográfica FL para a deteção de chama, através de comandos provenientes do RPI. A deteção da chama através da FL ocorre devido à existência de uma relação entre a informação *output* da FL e a grandeza física temperatura. Para realizar a comunicação entre a FL e o RPI utilizou-se os pinos GPIO - Figura 3.5.

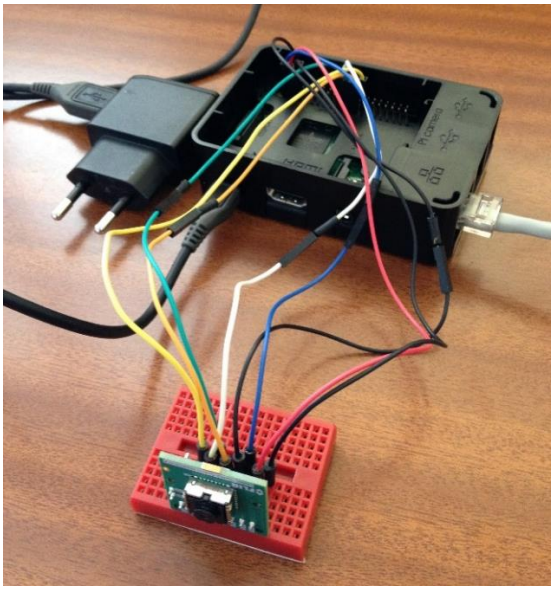


Figura 3.5 - Ligações entre RPI e FL.

O *output* da FL é uma *array* com valores relacionados com a temperatura que está na sua área de visão, através de programação transforma-se a *array* numa imagem a preto e branco, *raw*

image. Aplicando um mapeamento de cores à *raw image*, obtém-se uma imagem térmica baseada em cores frias (azul) e cores quentes (vermelho) - Figura 3.6.

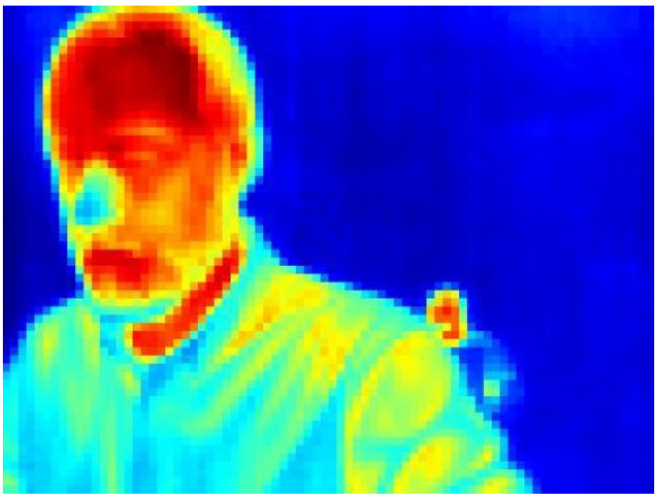


Figura 3.6 - Exemplo de uma imagem obtida da FL com mapeamento de cores.

Apesar de se obter uma imagem termográfica (variando de azul até vermelho) a escala é relativa. Pois a cor azul e vermelha vai estar sempre associada à temperatura mínima e máxima respetivamente que a câmara captar, logo esta informação só por si, não é suficiente.

Foi necessário descobrir um valor de separação (*threshold*), de modo a que caso o valor na *array* seja superior ou igual ao valor de referência, o valor da *array* passará a 1, caso contrário passará a 0 (binarização). Obtendo assim uma imagem binária no qual o branco representa o valor de temperatura elevado (comparando com o meio envolvente) e o preto o restante.

De modo a perceber qual o valor de separação foi necessário efetuar ensaios, que consistiram em apontar a câmara para uma pessoa e registaram-se vários valores de *output* da mesma (mínimos e máximos). O ensaio foi repetido mas com uma vela em frente à câmara para o mesmo efeito (Figura 3.7). Assim foi possível compreender em que gama de valores se encontra uma chama.

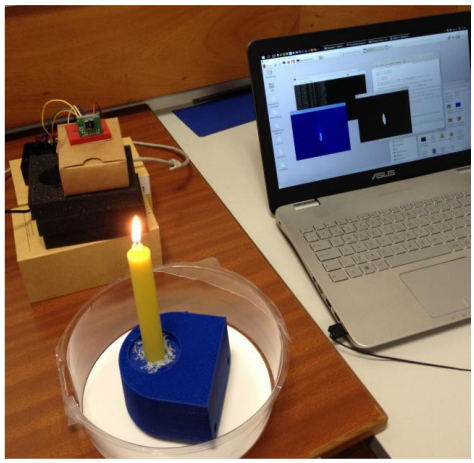


Figura 3.7 - *Setup* utilizado nos ensaios.

	***Output* Mínimo**	***Output* Máximo**
	7995	8521
	7992	8519
	7991	8518
	7993	8518
	7994	8518
	7990	8517
	7994	8517
	7995	8518
Média	7993	8518

Tabela 3.2 - Resultados mínimos e máximos obtidos sem chama.

	***Output* Mínimo**	***Output* Máximo**
	8008	15657
	8007	15661
	8010	15658
	8012	15660
	8012	15659
	8014	15659
	8014	15661
	8013	15659
Média	8011	15659

Tabela 3.3 - Resultados mínimos e máximos obtidos com chama.

Analisando a Tabela 3.2e Tabela 3.3 conclui-se que a temperatura máxima corporal de uma pessoa comum corresponde a um *output* de 8518, enquanto a de uma chama corresponde a 15659 (valores aproximados).

Com base nestes resultados pode- se concluir que não existe uma relação direta entre o *output* da FL e a temperatura real do meio envolvente – isto porque se uma pessoa possuir uma temperatura corporal de 36°C (8518), o valor mínimo na sala deveria ser 15°C (durante os ensaios mediu-se esta temperatura com um multímetro com termopar). Estes 15°C corresponderiam a um output, assumindo relação linear, de aproximadamente 3549. Porém o valor mínimo registado foi de 7993. Assim, conclui-se que não é possível obter a temperatura real da imagem termográfica.

Realizou-se ensaios semelhantes com o intuito de ver a qualidade da imagem da FL consoante a distância da chama de vela, tendo sido fixado o ponto de referência em 9000 e as distâncias estudadas foram 0.3, 1, 2 e 3m. As Figura 3.8--Figura 3.11 ilustram estes resultados.

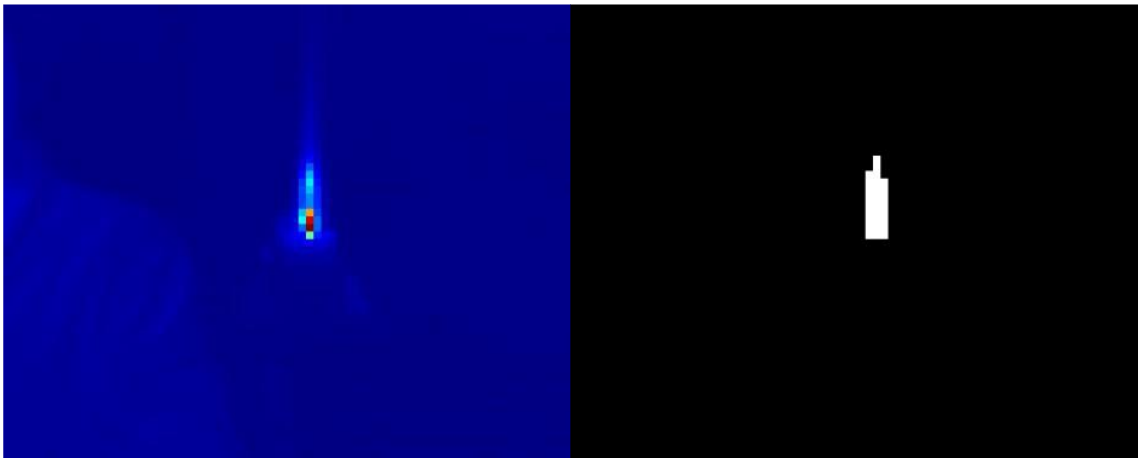


Figura 3.8 - Imagem termográfica (à esquerda) e imagem binária (à direita) com ponto de referência igual a 9000 e 0.3m de distância.

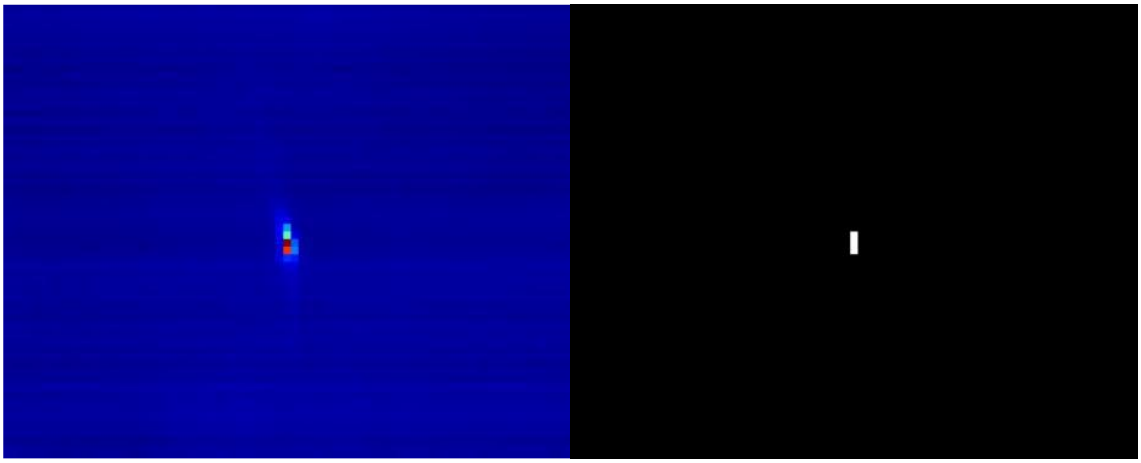


Figura 3.9 - Imagem termográfica (à esquerda) e imagem binária (à direita) com ponto de referência igual a 9000 e 1m de distância.

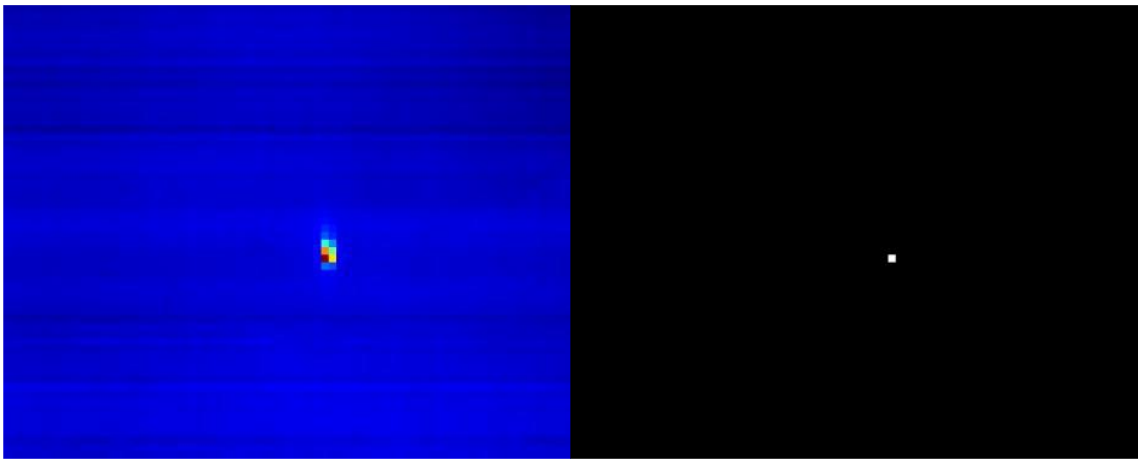


Figura 3.10 - Imagem termográfica (à esquerda) e imagem binária (à direita) com ponto de referência igual a 9000 e 2m de distância.

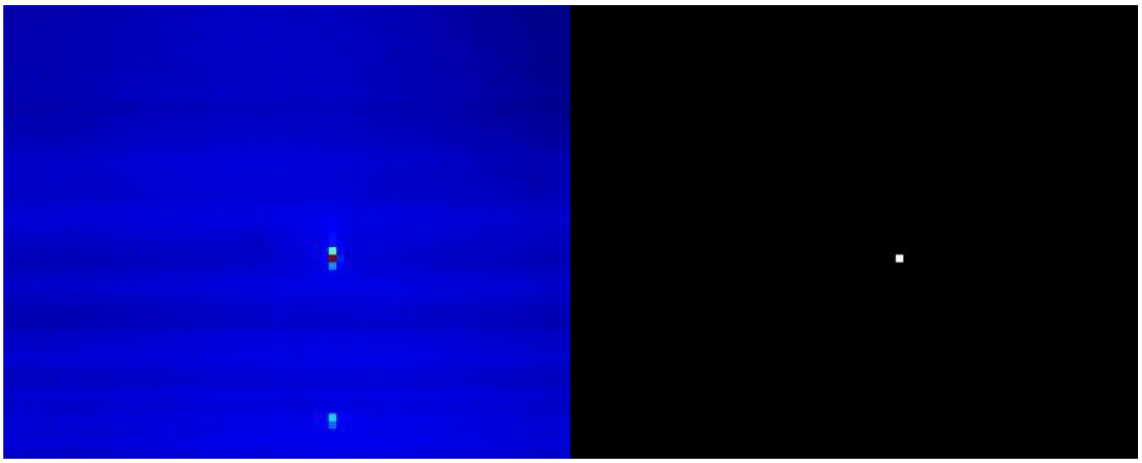


Figura 3.11 - Imagem termográfica (à esquerda) e imagem binária (à direita) com ponto de referência igual a 9000 e 3m de distância.

Analisando as Figuras anteriores (Figura 3.8--Figura 3.11), conclui-se que a câmara FL é um pouco limitada para distâncias entre 2 e 3 metros. Nestes casos a fonte de calor da chama concentra-se maioritariamente num reduzido ponto. Apesar da limitação, não terá influência no labirinto pois um quarto só poderá ter no máximo 1m de comprimento e/ou largura. Em todo o caso, na imagem binária continua a aparecer um *pixel* superior ao ponto de referência utilizado, ou seja, continua a haver deteção de chama.

3.3 Sistema de Extinção de Chama

O presente estudo pretende efetuar as ligações entre o RPI, a ventoinha e respetiva alimentação e por fim, verificar o cumprimento do seu principal objetivo (extinção da chama). Para a

alimentação da ventoinha recorreu-se à bateria dos motores (12V). Porém a ventoinha necessita de apenas de 5V de alimentação, ou seja, os 12V provenientes da bateria dos motores irão danificar a ventoinha. Para resolver esse problema recorreu-se a um díodo de Zener. Outro aspeto importante no sistema é a necessidade de que a ventoinha seja ligada apenas quando for desejado, ou seja, é necessário um interruptor programável. Para esse efeito recorreu-se a um transístor de potência de junção bipolar (BJT - *Bipolar Junction Transistor*) do tipo NPN – Figura 3.12.

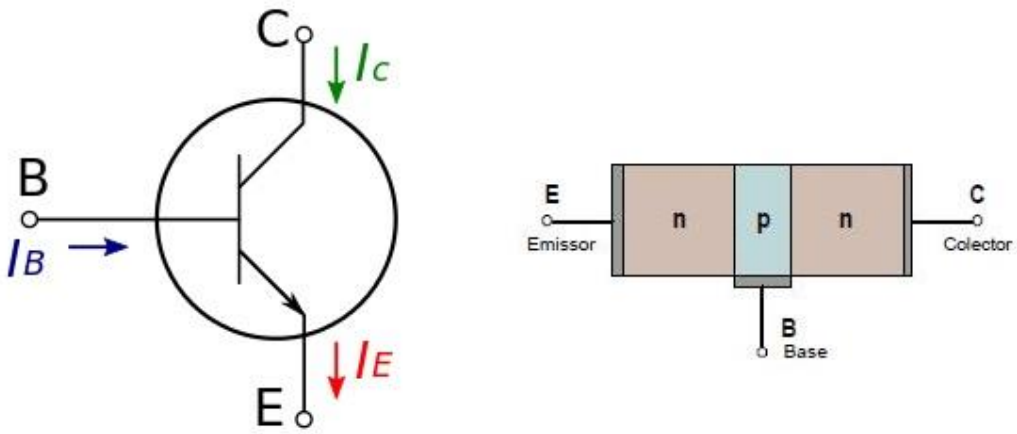


Figura 3.12 – Esquerda: Símbolo atribuído ao componente transístor e a convenção das correntes. Direita: Estrutura do transístor do tipo NPN.

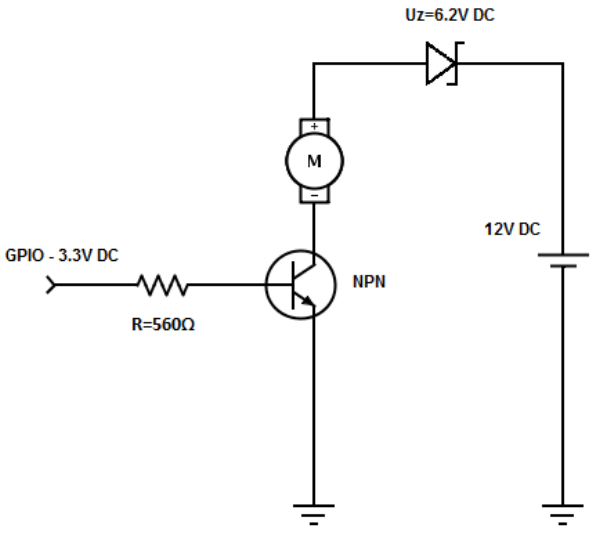


Figura 3.13 - Esquema de ligações associadas ao sistema de extinção de chama.

Através da Figura 3.13 verifica-se que o motor da ventoinha (M) é alimentado pela bateria dos motores de 12V, passando antes por um díodo de Zener. Induzindo uma queda de tensão de 6.2V, o que resulta numa tensão de 5.8V a chegar ao motor da ventoinha.

Porém a alimentação só ocorre quando o transístor recebe um sinal via GPIO proveniente do RPI, caso contrário o transístor encontra-se ao corte não permitindo a passagem de corrente. O sinal proveniente do RPI passa primeiro por uma resistência antes de chegar ao transístor. A resistência ao estar em série com a base do transístor limita a corrente que lhe chega, de tal forma que não seja superior à que possa suportar, evitando assim o risco de queimar o transístor.

3.4 Sistema de Navegação

O sistema de navegação inclui vários sistemas: perceção, localização, planeamento e controlo. Neste estudo será dado enfase ao sistema de perceção, onde se irá efetuar a comunicação com o sensor *rangefinder* RPLIDAR 360°. Para esse efeito foram realizados alguns estudos/ensaios de modo a entender a sua receção e transmissão de dados e respetivo tratamento dos dados provenientes. Devido à complexidade de programar um sistema de navegação, optou-se por realizar os primeiros estudos de comunicação com o PC, e desenvolver um simulador com recurso ao *software* Matlab®.

A ligação entre o RPLIDAR 360° e o PC foi realizada por uma porta USB através de um adaptador próprio (Figura 3.14) que permite converter os dados serie UART para USB (RoboPeak, 2014).

Figura 3.14 Ligação entre RPILDAR 360° e o PC.

O RPLIDAR 360° fornece um vetor com informação de cada um dos pontos da nuvem de pontos obtida pela medição realizada em coordenadas polares (distância e ângulo) - Figura 3.15.

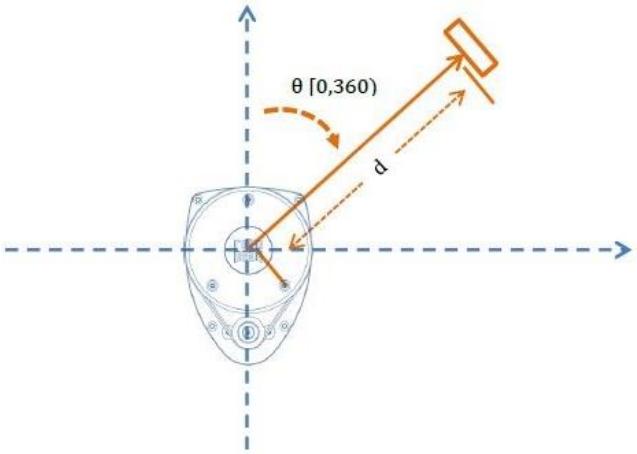


Figura 3.15 - Referencial utilizado pelo *rangefinder* RPLIDAR 360°. Fonte: (RoboPeak, 2014).

Através da projeção das coordenadas polares para coordenadas cartesianas é possível obter um gráfico da nuvem de pontos.

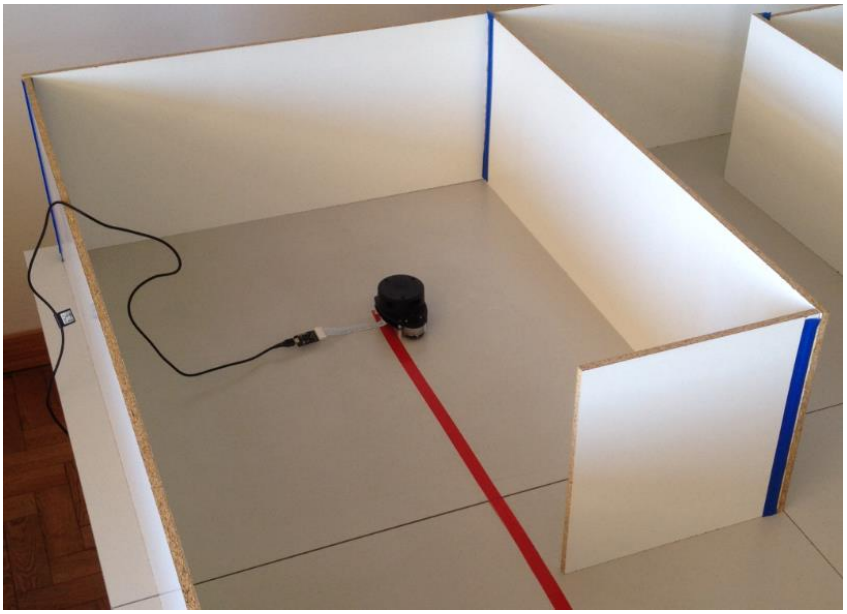


Figura 3.16 – Cenário de simulação utilizado (quarto).

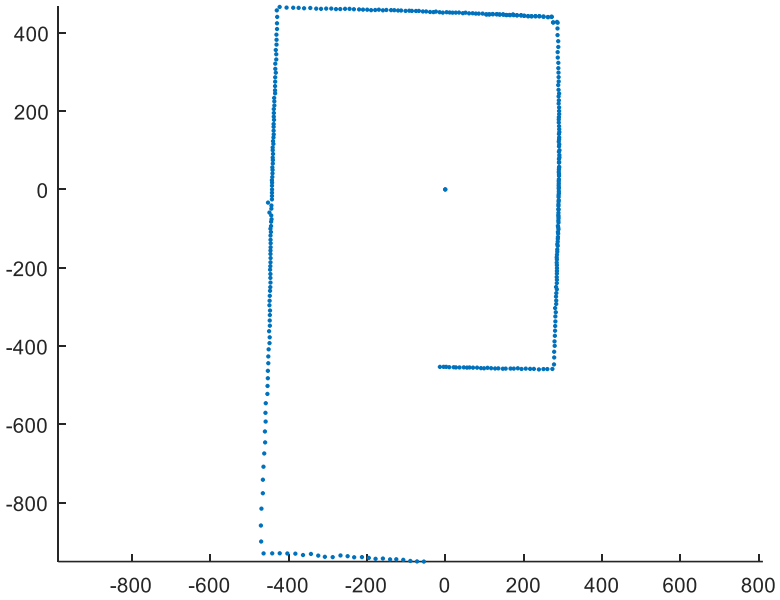


Figura 3.17 - Nuvem de pontos obtida no quarto de simulação.

Através da visualização da Figura 3.17 é possível retirar algumas conclusões, primeiro, o ponto que está isolado a meio do quarto (coordenadas 0,0) representa a posição do sensor em relação ao ambiente, segundo, à medida que a distância ao sensor aumenta, o espaçamento entre pontos também aumenta, o que pode induzir em erros pois existe a possibilidade de serem confundidos com ruido, algo que se deve ter em conta.

Tendo o conhecimento que neste caso de estudo o ambiente é constituído por paredes lisas, estas podem ser representadas por segmentos de retas. Com base nesse conceito, a solução para evitar a acumulação do ruido é a criação de linhas (reconhecimento de linhas) através dos pontos dados. Para esse efeito foi utilizado o algoritmo *Split and Merge*.

O algoritmo *Split and Merge* é um dos mais populares na extração de linhas que teve origem na visão computacional (Nguyen, Martinelli, Tomatis, & Siegwart, 2005).

Este algoritmo é definido da seguinte forma:

1. Inicio: Definir conjunto s_1 consistido por N pontos. Colocar s_1 numa lista L
2. Encontrar uma linha adequada para o próximo conjunto s_i em L
3. Detetar ponto P com a máxima distância d_P da linha
4. Se d_P for inferior ao valor de separação, continuar (ir para o passo 2)
5. Caso contrário, realizar divisão em s_i no ponto P em s_{i1} e s_{i2}, substituir s_i em L por s_{i1} e s_{i2}, continuar (ir para passo 2)
6. Quando todos os conjuntos (segmentos) em L forem verificados funde segmentos colineares.

De modo a obter uma reta que melhor se ajusta ao conjunto de pontos é utilizado o método numérico dos mínimos quadrados.

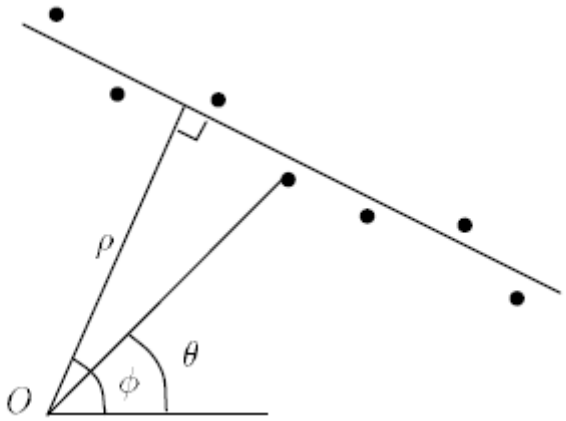


Figura 3.18 - Parâmetros utilizados no ajuste da reta à nuvem de pontos pelo método dos mínimos quadrados. Fonte: (Lu & Milios, 1994)

O ajuste de uma reta a um conjunto de pontos (x_i, y_i) na vizinhança de tamanho n é definido pelos parâmetros ρ (distância normal desde a origem até à reta) e ϕ (direcção da normal à reta) (ver Figura 3.18) que minimizam o seguinte erro:

$$E_{fit} = \sum_{i=1}^{n} (x_i \cos\phi + y_i \sin\phi - \rho)^2 \quad \textbf{(3.2)}$$

Uma solução de forma fechada pode ser derivada da seguinte forma:

$$\phi = \frac{1}{2} tan^{-1} \frac{-2S_{xy}}{S_{y^2} - S_{x^2}} \quad (3.3)$$

$$\rho = \overline{x}\, cos\, \phi + \overline{y}\, sin\, \phi \quad (3.4)$$

$$min_{(\phi,\rho)}(E_{fit}) = \frac{1}{2}\left(S_{x^2} + S_{y^2} - \sqrt{4S_{xy}{}^2 + \left(S_{y^2} - S_{x^2}\right)^2}\right) \quad (3.5)$$

Onde $\bar{x}$ e $\bar{y}$ são as médias ponderadas dos valores de x:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (3.6)$$

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \quad (3.7)$$

E S_{x^2}, S_{y^2} e S_{xy} são parâmetros do método dos mínimos quadrados.

$$S_{x^2} = \sum_{i=1}^{n} (x_i - \overline{x})^2 \quad (3.8)$$

$$S_{y^2} = \sum_{i=1}^{n} (y_i - \overline{y})^2 \quad (3.9)$$

$$S_{xy} = \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y}) \quad (3.10)$$

Com base na reta obtida pelo método dos mínimos quadrados, verifica-se qual o ponto onde o modulo da distância normal à reta d_i é maior (Figura 3.19), que pode ser calculada da seguinte forma:

$$d_i = \rho_i \cos(\theta_i - \phi) - \rho \quad (3.11)$$

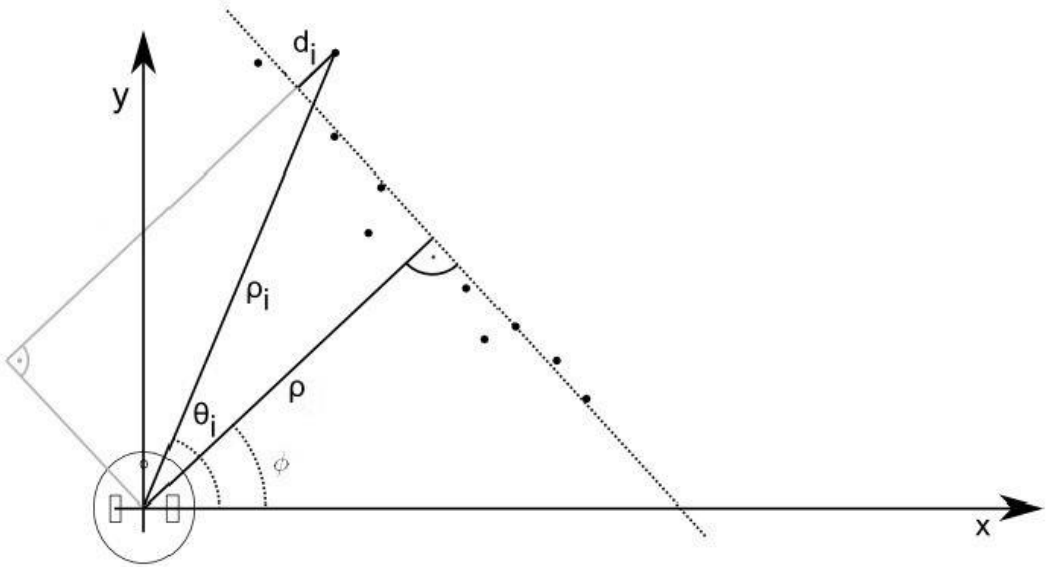


Figura 3.19 – Verificação da distância à reta calculada pelo método dos mínimos quadrados. Fonte: (Correll, 2016)

Sabendo o valor da maior distância normal à reta, realiza-se um *threshold* de modo a verificar se o valor obtido é ou não superior ao valor de referência, caso seja inferior acaba o processo sendo essa a última linha, caso seja superior, no ponto em questão será efetuado um "corte" (*split*) na reta dividindo-a em duas retas - Figura 3.20. Tendo agora duas novas retas para efetuar o algoritmo novamente.

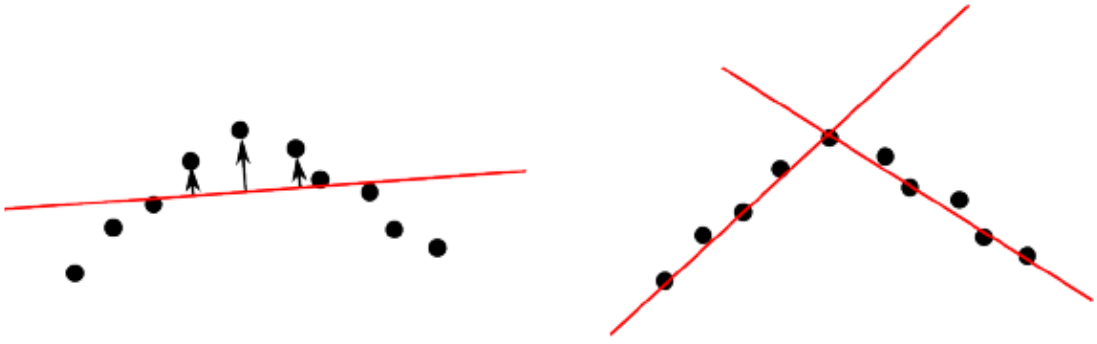


Figura 3.20 - Exemplo de um *split*. Fonte: (Correll, 2016)

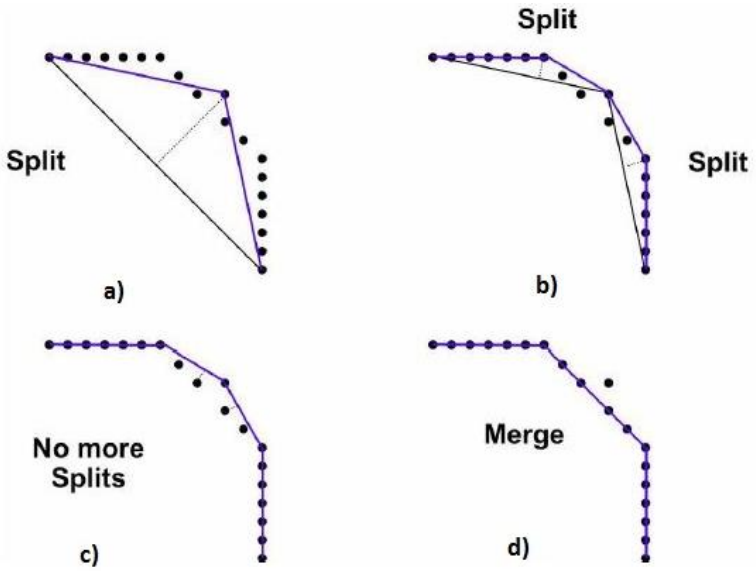


Figura 3.21 - Exemplo completo do método *Split and Merge*. Fonte: (Tardós, 2002)

A Figura 3.21 representa um exemplo completo do algoritmo *Split and Merge*: (*a*) começa por realizar um *split* no ponto onde existe maior distância normal à reta, (*b*) de seguida realizando um *threshold*, (*c*) conclui-se que é necessária uma nova operação de *split* em cada uma das novas retas, (*d*) por fim, não sendo necessário um novo *split*, realiza-se um *merge* entre os segmentos.

Aplicando o algoritmo *Split and Merge* à Figura 3.17 obtém-se:

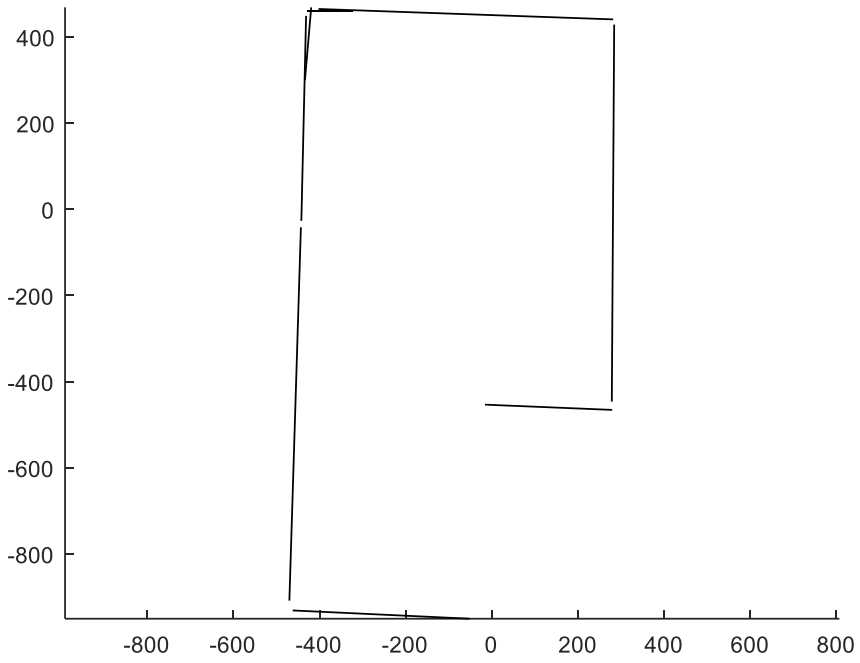


Figura 3.22 – Mapa obtido após aplicação do algoritmo *Split and Merge*.

Através da Figura 3.22 conclui-se que o algoritmo *Split and Merge* foi aplicado com sucesso, obtendo-se resultados bastante satisfatórios. Comprovando que a abordagem de extração de linhas no labirinto é válida. Com base nessa informação, concluiu-se que o sensor RPLIDAR 360° tem a capacidade de realizar um bom mapeamento, conseguindo uma leitura de qualidade do ambiente, praticamente em tempo real.

4 SISTEMA DE NAVEGAÇÃO

No presente capítulo será abordado em detalhe o sistema de navegação que está incorporado no robô. Este sistema é responsável pela navegação e localização do robô, ou seja, permite ao robô movimentar-se de forma segura e eficiente, recorrendo à informação do sensor *rangefinder* RPLIDAR 360° ® e odometria. Após a obtenção do mapa, torna-se possível determinar se o robô está num corredor ou dentro de um quarto ou até mesmo numa intersecção de corredores. Este tipo de informação será importante para que o robô possa tomar decisões.

Este capítulo começa por abordar o tema da cinemática do robô móvel e da localização por odometria, passando depois pelo processo de planeamento. Este último depende dos processos atrás referidos: localização e o mapeamento.

4.1 Cinemática do robô móvel

A cinemática é o ramo da mecânica clássica que se dedica à descrição do movimento de sistemas mecânicos. Na robótica móvel, é necessário entender o comportamento mecânico do robô, de modo a projetar robôs móveis apropriados para as tarefas e também entender como se cria *software* de controlo para uma instância de *hardware* do robô móvel.

O processo de compreender os movimentos de um robô começa com a descrição da contribuição de cada roda para o movimento, pois cada roda tem um papel no movimento do robô. Da mesma forma, cada roda também impõe restrições sobre o movimento do robô.

Na presente dissertação utiliza-se um robô com sistema diferencial. O robô é modelado como um corpo rígido com rodas, que opera num plano horizontal. O *chassis* do robô tem três graus de liberdade: dois para a posição no plano (x, y) e um para a orientação ao longo do eixo vertical (θ), com constrangimentos não-holonómicos de um sistema diferencial. A referência dos eixos de inércia do robô tem por base o seu chassis de construção geométrica e ortogonal. Assim sendo, todos os restantes graus de liberdade das diversas juntas assim como inclinação das rodas podem ser ignorados.

4.1.1 Odometria

A localização é um dos aspetos fundamentais de um robô móvel. A localização e orientação do robô devem ser obtidas, por exemplo, para que o robô se mova da posição atual para a meta. Na presente dissertação um dos métodos utilizados para a localização é o cálculo da posição através de odometria, que mede o movimento do robô.

Através de encoders incrementais é possível medir a rotação das rodas, o que, por sua vez, permite calcular a posição do robô e a sua orientação, integrando o modelo cinemático. A cinemática de um robô de tração diferencial é descrita pelo seguinte sistema de modelo contínuo:

$$\begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \end{pmatrix} = \int_0^t \begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{pmatrix} dt = \int_0^t \begin{pmatrix} v(t)\cos\theta \\ v(t)\sin\theta \\ \omega(t) \end{pmatrix} dt \qquad (4.1)$$

Onde $x(t)$, $y(t)$ e $\theta(t)$ correspondem à postura no instante t, $\dot{x}(t)$, $\dot{y}(t)$ correspondem à decomposição nos eixos cartesianos da velocidade linear e $\dot{\theta}(t)$ à velocidade angular. Tendo em conta que o robô será comandado por um controlador discreto, onde as leituras dos sensores se mantêm constantes entre dois tempos de amostragem, discretização ZOH (*Zero Order Holder*). Considerando um período de amostragem T, a posição é obtida pela incrementação ao longo de tempos de amostragem consecutivos. Assim a postura no instante k é dada por:

$$\begin{pmatrix} x(k) \\ y(k) \\ \theta(k) \end{pmatrix} = \sum_{q=0}^{k} \begin{pmatrix} \dot{x}(q) \\ \dot{y}(q) \\ \dot{\theta}(q) \end{pmatrix} T \qquad (4.2)$$

A posição no instante k é obtida incrementando o deslocamento nesse instante com os anteriores:

$$\begin{pmatrix} x(k) \\ y(k) \\ \theta(k) \end{pmatrix} = \sum_{q=0}^{k-1} \begin{pmatrix} \dot{x}(q) \\ \dot{y}(q) \\ \dot{\theta}(q) \end{pmatrix} T + \begin{pmatrix} \dot{x}(k) \\ \dot{y}(k) \\ \dot{\theta}(k) \end{pmatrix} T \qquad (4.3)$$

Resultando a seguinte expressão:

$$\begin{pmatrix} x(k) \\ y(k) \\ \theta(k) \end{pmatrix} = \begin{pmatrix} x(k-1) \\ y(k-1) \\ \theta(k-1) \end{pmatrix} + \begin{pmatrix} v(k)\cos\theta \\ v(k)\sin\theta \\ \omega(k) \end{pmatrix} T \qquad (4.4)$$

Que pode ser reescrito na forma de sistema:

$$\begin{cases} x(k) = x(k-1) + v(k)\cos\theta\, T \\ y(k) = y(k-1) + v(k)\sin\theta\, T \\ \theta(k) = \theta(k-1) + \quad \omega(k)\, T \end{cases} \qquad (4.5)$$

A Equação 4.5 representa o cálculo da posição e orientação do robô ($x(k), y(k)$ e $\theta(k)$) no instante k com base na posição e orientação anterior, velocidade $v(k)$ e orientação $\theta(k)$, num intervalo de discretização T.

De modo a calcular a velocidade, recorre-se aos constrangimentos cinemáticos das rodas. Para uma roda *standard* os constrangimentos cinemáticos correspondem a que qualquer rotação da roda leva a um movimento para a frente ou para trás, não havendo movimento lateral ou escorregamento. De acordo com estes constrangimentos, o valor da velocidade em cada roda obtém-se simplesmente pela conversão da velocidade de rotação da roda na velocidade tangencial.

$$\begin{cases} v_{left} = \dfrac{(\varphi(k) - \varphi(k-1))_{left}}{T}\, r \\ v_{right} = \dfrac{(\varphi(k) - \varphi(k-1))_{right}}{T}\, r \end{cases} \quad \textbf{(4.6)}$$

Sendo φ o ângulo de rotação, em radianos da roda esquerda ou direita no instante k ou $k-1$ e r o raio da roda (ver Figura 4.1).

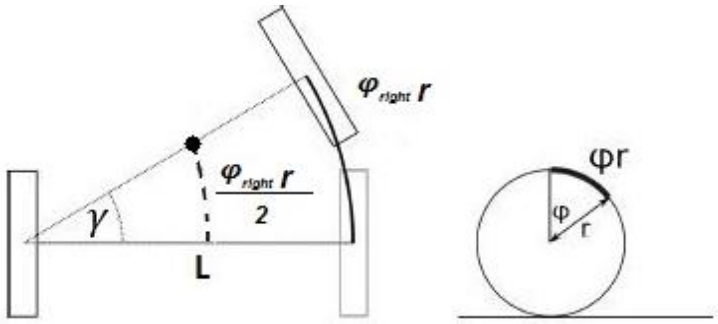


Figura 4.1 – Esquerda: Exemplo de uma viragem à esquerda. Direita: Uma roda de raio r que realiza uma rotação de φ(adaptado). Fonte: (Correll, 2016)

Cada uma das duas rodas motoras contribui para a velocidade no centro do robô - onde o sistema de coordenadas está fixo. Para tal, calcula-se a contribuição de cada uma das rodas, assumindo que as rodas restantes se mantêm não acionadas (Correll, 2016). Neste exemplo, a distância percorrida pelo ponto do centro, $s(k)$,é exatamente a meio das percorridas por cada roda - Figura 4.1. Podemos, portanto, escrever:

$$s(k) = \frac{\varphi_{left}(k)\, r}{2} + \frac{\varphi_{right}(k)\, r}{2} \quad \textbf{(4.7)}$$

Derivando ambos os lados da Equação 4.7 obtém-se:

$$v(k) = \frac{v_{left}(k)}{2} + \frac{v_{right}(k)}{2} \quad \textbf{(4.8)}$$

Em relação à orientação do robô, aplicando o mesmo raciocínio e considerando que a roda esquerda não está a ser atuada, ao rodar a roda direita cria-se uma rotação no sentido anti-horário. Sabendo a distância entre as rodas do robô, L é possível escrever:

$$\gamma(k)\, L = \varphi_{right}(k) r \quad \textbf{(4.9)}$$

Isolando o termo γ e derivando ambos os lados da Equação 4.9 obtém-se:

$$\dot{\gamma}(k) = \frac{\dot{\varphi_{right}}(k)\, r}{L} \Leftrightarrow \omega_{right} = \frac{v_{right}(k)}{L} \quad \textbf{(4.10)}$$

Adicionando a velocidade de rotação da roda esquerda e tendo em conta que a rotação da roda esquerda é no sentido horário é possível escrever:

$$\omega(k) = \frac{v_{right}(k)}{L} - \frac{v_{left}(k)}{L} \quad \textbf{(4.11)}$$

Obtém-se assim uma expressão para a velocidade angular no instante k.

4.1.2 Sistema Diferencial

Na seguinte análise, é utilizada a abordagem da cinemática de um robô de tração diferencial, descrita na obra de Siegwart, Nourbakhsh e Scaramuzza (2011).

Um robô móvel com sistema diferencial pode ser descrito através da relação entre uma referência global no plano e a referência local do robô. Os eixos de inércia X_I e Y_I correspondem à referência global do plano de inércia $I = \{X_I, Y_I\}$, centrado na origem. A posição P do robô define o seu ponto de referência no chassis, e é descrita por dois eixos locais ortogonais $\{X_R, Y_R\}$, o segundo coincidente com o eixo das rodas motoras e o primeiro passante pelo do eixo vertical da roda de apoio - Figura 4.2.

A posição global do ponto P é dada pelas coordenadas x e y no referencial $\{X_I, Y_I\}$ assim como pela diferença angular θ entre os referenciais locais e globais.

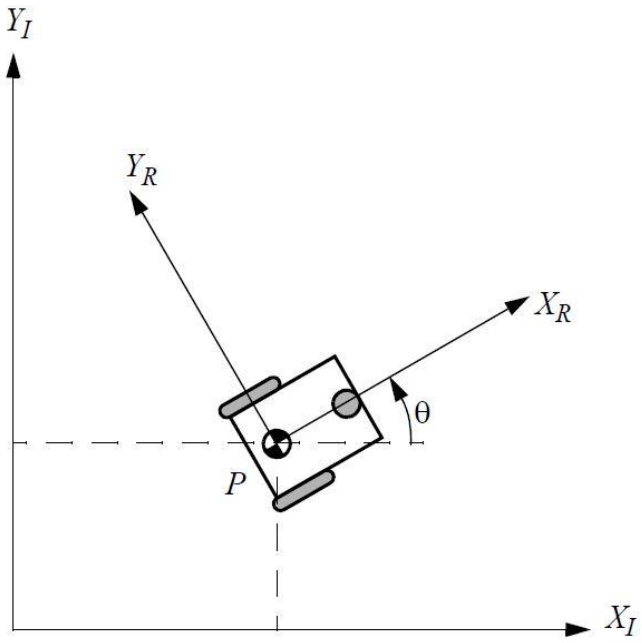


Figura 4.2 - Referências locais e globais do robô móvel. Fonte: (Siegwart, Nourbakhsh, & Scaramuzza, 2011).

O vetor de posição do robô móvel no referencial de base I designa-se por $\boldsymbol{\xi_I}$.

$$\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \quad \textbf{(4.12)}$$

Para descrever o movimento do robô em relação ao referencial global é necessário relacionar o movimento dos eixos do referencial global com os eixos do referencial local, que dependem do valor da orientação θ. Esta relação é descrita através da matriz de rotação:

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(4.13)}$$

4.1.3 Cinemática de um Sistema Diferencial

A cinemática de um robô de tração diferencial descrita no referencial I é dada pela Equação 4.14, onde $\dot{x}$ e $\dot{y}$ são as velocidades lineares nas direções X_I e Y_I e o ângulo θ a orientação do robô.

$$^I\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad \textbf{(4.14)}$$

No sistema de locomoção diferencial a velocidade linear, $v(t)$, é dada pela média aritmética das velocidades lineares das duas rodas – Equação 4.8. Enquanto a velocidade angular, $\omega(t)$, é dada pela Equação 4.11.

Criando um sistema de equações com as Equações 4.7 e 4.10:

$$\begin{cases} v(t) = \dfrac{v_{left}(t) + v_{right}(t)}{2} \\ \omega(t) = \dfrac{v_{right}(t) - v_{left}(t)}{L} \end{cases} \Leftrightarrow \begin{cases} v_{left}(t) = v(t) - \dfrac{L\ \omega(t)}{2} \\ v_{right}(t) = v(t) + \dfrac{L\ \omega(t)}{2} \end{cases} \quad \textbf{(4.15)}$$

Através do sistema de Equações, obteve-se o sistema de equações 4.15 que permite determinar a correspondente velocidade linear a enviar a cada roda, tendo uma lei de controlo que determine as velocidades lineares e angulares do robô.

4.1.4 Controlo de Posição

O controlo de posição é um componente do sistema de navegação que permite controlar a postura do robô consoante a distância e orientação à posição pré-definida (*goal*).

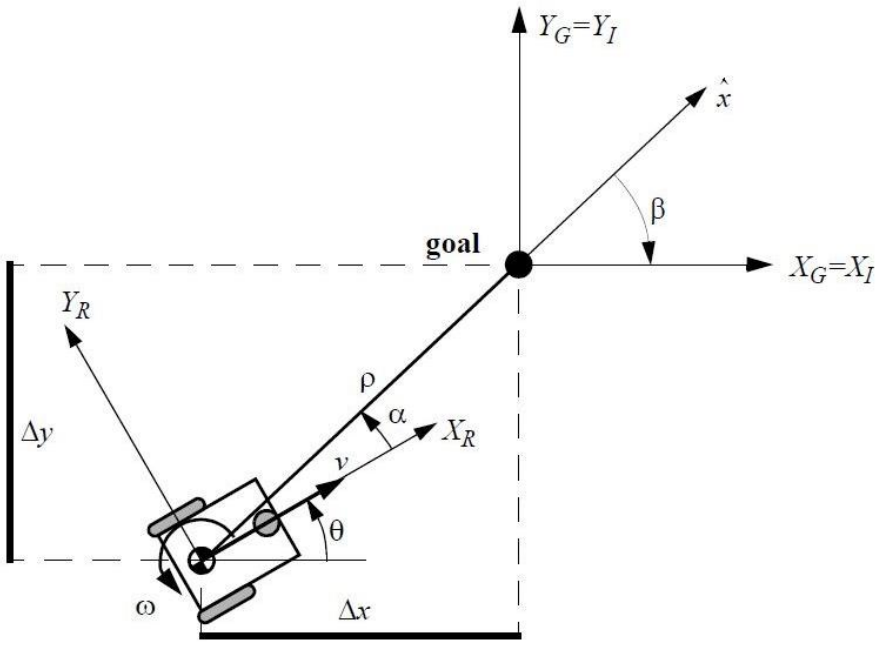


Figura 4.3 – Cinemática do robô móvel e as respetivas variáveis. Fonte: (Siegwart, Nourbakhsh, & Scaramuzza, 2011)

Admita-se que se pretende deslocar o robô de uma posição e orientação inicial $[x, y, \theta]_{ref}$ arbitrária para uma posição $[x, y, \theta]$ pré-definida (*goal*), conforme se pode visualizar na Figura

4.3. O ângulo α define o ângulo entre o eixo de referência X_R do robô e o vetor $\hat{x}$, que une o centro do eixo das rodas P com a posição da meta. β é o ângulo entre o vetor $\hat{x}$ e a configuração de destino, que neste exemplo tem a direção do eixo de inércia X_I. A distância entre P e a posição da meta designa-se por ρ. Estas variáveis podem ser facilmente calculadas através de uma análise geométrica.

Segundo a abordagem de Siegwart, Nourbakhsh e Scaramuzza (2011), considera-se a transformação de coordenadas em coordenadas polares com a sua origem na posição da meta:

$$\rho = \sqrt{\Delta x^2 + \Delta y^2} \quad \textbf{(4.16)}$$

$$\alpha = -\theta + \operatorname{atan2}(\Delta x, \Delta y) \quad \textbf{(4.17)}$$

$$\beta = -\theta - \alpha \quad \textbf{(4.18)}$$

O que gera um novo sistema, nas novas coordenadas polares, que pode ser representado na forma matricial:

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos\alpha & 0 \\ \dfrac{\sin\alpha}{\rho} & -1 \\ -\dfrac{\sin\alpha}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad \textbf{(4.19)}$$

O modelo cinemático acima representado é válido quando a direção α do robô é para a frente, ou seja, $\alpha \in I_1$ sendo I_1:

$$I_1 = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \quad \textbf{(4.20)}$$

Caso robô se desloque para trás, a velocidade passa a ser definida por $v = -v$, e a direção do robô, $\alpha \in I_2$, sendo I_2:

$$I_1 = \left[-\pi, \frac{\pi}{2}\right] \cup \left[\frac{\pi}{2}, \pi\right] \quad \textbf{(4.21)}$$

O que leva a alterações na matriz do sistema de controlo:

$$\begin{bmatrix}\dot{\rho}\\ \dot{\alpha}\\ \dot{\beta}\end{bmatrix} = \begin{bmatrix}\cos\alpha & 0\\ -\dfrac{\sin\alpha}{\rho} & 1\\ \dfrac{\sin\alpha}{\rho} & 0\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \quad \textbf{(4.22)}$$

4.1.5 Erro

A aplicação de uma lei de controlo de posição implica a existência de um erro, no qual será baseado o cálculo da ação de controlo. O vetor que representa o erro da posição atual no plano de referência do robô$\{X_R, Y_R, \theta\}$ é definido por **e**, sendo x, y e θ as coordenadas da meta.

$$e = [x, y, \theta]^T_{ref} \quad \textbf{(4.23)}$$

As ações de controlo de posição do robô são obtidas através da multiplicação da matriz de ganhos do controlador, **K** e o vetor de erros **e**.

$$K = \begin{bmatrix}k_{11} & k_{12} & k_{13}\\ k_{21} & k_{22} & k_{23}\end{bmatrix} \quad \text{com} \quad k_{i,j} = k(t, e) \quad \textbf{(4.24)}$$

$$\begin{bmatrix}v(t)\\ \omega(t)\end{bmatrix} = \mathbf{K}\,\mathbf{e} = \mathbf{K}\begin{bmatrix}x\\ y\\ \theta\end{bmatrix}_{ref} \quad \textbf{(4.25)}$$

De modo a que o controlo de $v(t)$ e $\omega(t)$ conduzam o erro para zero:

$$\lim_{t\to\infty} e(t) = 0 \quad \textbf{(4.26)}$$

4.1.6 Lei de Controlo

Os sinais de controlo v e ω são concebidos de modo dirigir o robô a partir da sua posição inicial até à meta. As leis de controlo utilizadas são as seguintes:

$$v = k_\rho\,\rho \quad \textbf{(4.27)}$$

$$\omega = k_\alpha\,\alpha + k_\beta\,\beta \quad \textbf{(4.28)}$$

Ao aplicar as leis de controlo e através da Equação 4.22 obtém-se o sistema de controlo em anel fechado, que pode ser representado na forma de matricial:

$$\begin{bmatrix}\dot{\rho}\\ \dot{\alpha}\\ \dot{\beta}\end{bmatrix} = \begin{bmatrix}-k_\rho\ \rho\ \cos\alpha\\ k_\rho\ \sin\alpha - k_\alpha\ \alpha - k_\beta\ \beta\\ -k_\rho \sin\alpha\end{bmatrix} \quad \textbf{(4.29)}$$

O sistema não possui nenhuma singularidade quando $\rho = 0$ e tem um único ponto de equilíbrio quando $(\rho, \alpha, \beta) = (0,0,0)$ fazendo com que o robô se desloque até a meta. Os ângulos α e β são definidos no intervalo $(-\pi, \pi)$. O sinal de v é constante, ou seja é sempre positivo se $\alpha(0) \in I_1$e negativo se $\alpha(0) \in I_2$. Tal implica que o robô segue sempre na mesma direção nas manobras de aproximação à meta, sem inverter a direção do movimento.

4.1.7 Estabilidade Local

O sistema de controlo em anel fechado é estável se verificar as seguintes condições:

$$k_\rho > 0\ ;\ k_\beta < 0\ ;\ k_\alpha - k_\rho > 0 \quad \textbf{(4.30)}$$

Prova:

Linearizando a posição do equilíbrio $(\cos x = 1\ ;\ \sin x = 1)$ a Equação 4.29 pode ser escrita da seguinte forma:

$$\begin{bmatrix}\dot{\rho}\\ \dot{\alpha}\\ \dot{\beta}\end{bmatrix} = \begin{bmatrix}-k_\rho & 0 & 0\\ 0 & -(k_\alpha - k_\rho) & -k_\beta\\ 0 & -k_\rho & 0\end{bmatrix} \begin{bmatrix}\rho\\ \alpha\\ \beta\end{bmatrix} \quad \textbf{(4.31)}$$

Sendo $\mathbf{A}$ a matriz:

$$A = \begin{bmatrix}-k_\rho & 0 & 0\\ 0 & -(k_\alpha - k_\rho) & -k_\beta\\ 0 & -k_\rho & 0\end{bmatrix} \quad \textbf{(4.32)}$$

O sistema é estável se todos os polos do sistema tiverem parte real negativa, que se pode verificar através dos valores próprios da matriz $\mathbf{A}$.

O polinómio característico da matriz $\mathbf{A}$ é:

$$\left(\lambda + k_\rho\right)\left(\lambda^2 + \lambda\left(k_\alpha - k_\rho\right) - k_\rho k_\beta\right) \quad \textbf{(4.33)}$$

As suas raízes têm componente real negativa se:

$$k_\rho > 0 \ ; \ -k_\beta > 0 \ ; \ k_\alpha - k_\rho > 0 \qquad \textbf{(4.34)}$$

O que prova as condições de estabilidade local anteriormente referidas.

Caso seja necessário um controlo de posição mais robusto, é aconselhado aplicar uma condição de estabilidade mais conservadora, de modo a garantir a que o robô não mude de direção ao aproximar-se da meta.

$$k_\rho > 0 \ ; \ k_\beta < 0 \ ; \ k_\alpha + \frac{5}{3} k_\beta - \frac{2}{\pi} k_\rho > 0 \qquad \textbf{(4.35)}$$

4.2 Planeamento

Esta secção apresenta um método para planeamento de trajetórias que, integrado com o sistema de localização, perceção de obstáculos e de controlo apresentados previamente, permitirá a navegação segura do robô.

O primeiro passo do planeamento de trajetórias consiste na transformação do modelo do mundo físico num mapa que o robô consiga interpretar, também conhecido como mapeamento. Na literatura existem três tipos de mapas:

- Métrico: representação bidimensional do espaço e dos objetos no ambiente ao nível do ponto de vista do sensor (Kanniah, Ercan, & Calderon, 2014).
- Topológico: representação espacial de forma gráfica, que descreve os locais e objetos de interesse como nós (*nodes*) e suas relações espaciais como linhas (*edges*) (Košnar,2011).
- Híbrido: representação semelhante à topológica com a particularidade de utilizar informações métricas (dimensões, distâncias, etc.).

Os mapas métricos são representados normalmente através de um plano dividido em células de dimensões iguais (*grid*) como ilustra a Figura 4.4. Nestes mapas, cada célula representa uma porção fixa do espaço e armazena um valor que indica o estado desta célula como, por exemplo, a presença ou a ausência de um obstáculo (Thrun, 1998).

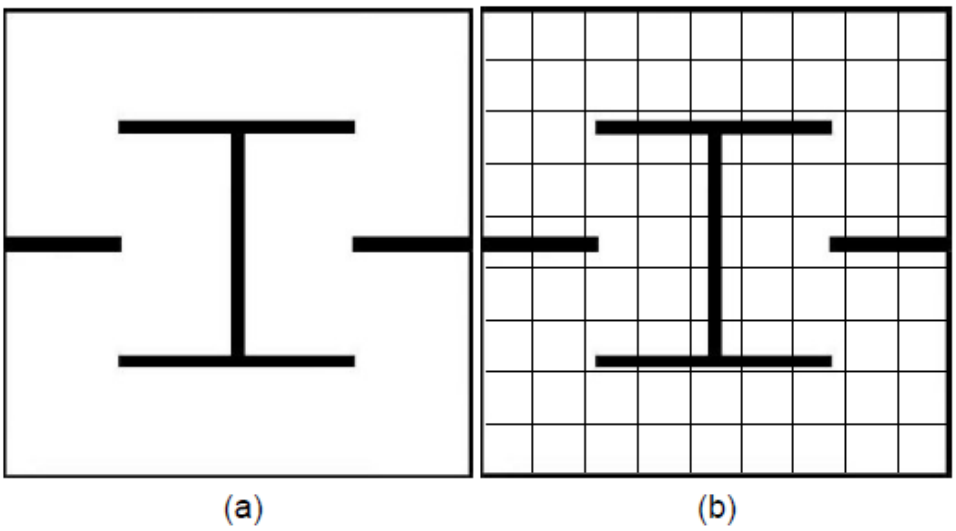


Figura 4.4 - (a) Representação de um ambiente não mapeado, (b) representação de um mapa métrico. Fonte: (Oliveira, 2010)

Mapas topológicos, também designados por mapas relacionais, fornecem as relações entre os vários locais ou objetos presentes no ambiente, sob a forma de um gráfico. Os mapas topológicos, de forma geral, registam informações sobre determinados elementos ou locais do ambiente, por exemplo uma entrada de um quarto, chamados marcos (*landmarks*).

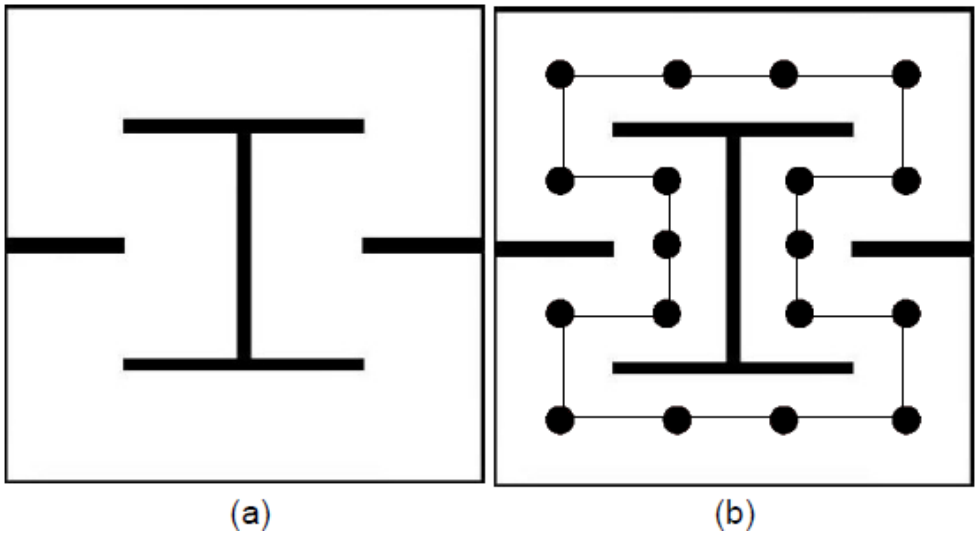


Figura 4.5 - (a) Representação de um ambiente não mapeado, (b) representação de um mapa topológico. Fonte: (Oliveira, 2010)

De uma forma resumida, os mapas métricos captam as propriedades geométricas do ambiente, enquanto os mapas topológicos descrevem a conectividade entre os diferentes locais, ou seja, os mapas topológicos representam o ambiente como uma lista de locais significativos que são

ligados através de arcos. Como as abordagens são distintas, cada uma apresenta as suas vantagens e desvantagens, referidas brevemente de seguida.

Os mapas métricos podem ser bastante precisos, consoante a precisão da grelha, contudo a sua complexidade dificulta, geralmente, um planeamento de trajetórias eficiente. Por outro lado, dificulta também a resolução de problemas em grandes ambientes internos (Thrun, 2002).

Os mapas topológicos, em contrapartida, podem ser utilizados eficientemente na determinação de trajetórias. Porém, também se torna difícil manter e gerar mapas topológicos em grandes ambientes, principalmente se os dados sensoriais momentâneos forem substancialmente ambíguos.

Para esta dissertação utilizaram-se mapas híbridos, ou seja, mapas com representação topológica em que as suas ligações (*edges*) contêm informação da distância métrica.

4.2.1 Criação do Mapa Topológico

Segundo Rawlinson & Jarvis (2008) os métodos baseados em diagramas / grafos de Voronoi generalizados (*Generalized Voronoi Diagram* - GVD) são os mais utilizados para a geração de mapas híbridos que contenham informações métricas e topológicas. Apesar de ser um método adequado existem algumas dificuldades associadas, entre as quais se destaca o *loop closure* (Choset & Nagatani, 2001).

4.2.1.1 Diagrama de Voronoi Generalizado

O GVD é um método do tipo *Road Map* que tende a maximizar a distância entre o robô e os obstáculos do mapa. O diagrama consiste nas arestas formadas pelos pontos que são equidistantes de dois ou mais obstáculos, sendo que quando os obstáculos no espaço de configuração são polígonos, o diagrama de Voronoi consiste em linhas retas e/ou segmentos parabólicos. Para a determinação do gráfico calcula-se para cada ponto no espaço livre, a sua distância ao obstáculo mais próximo (Siegwart, Nourbakhsh, & Scaramuzza, 2011).

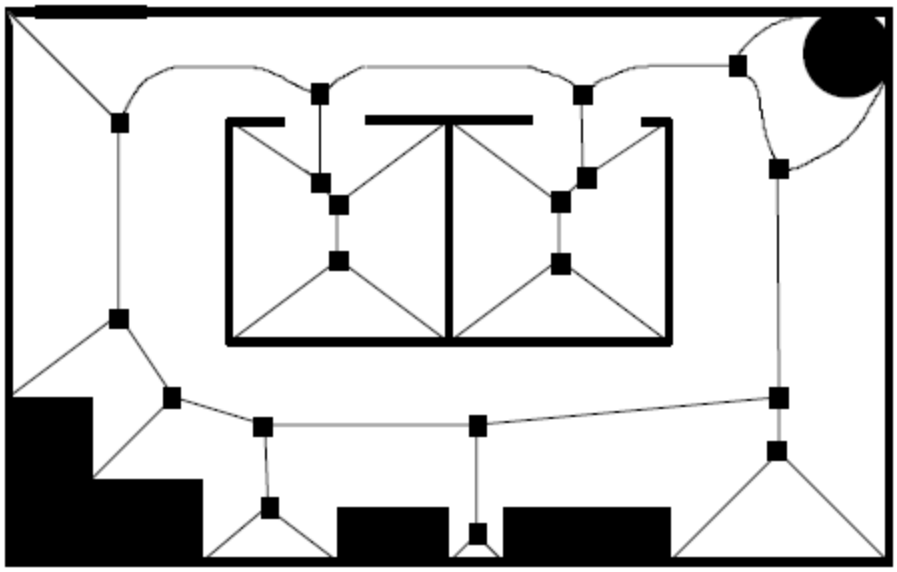


Figura 4.6 - Exemplo de um diagrama de Voronoi generalizado aplicado a um mapa já explorado. Fonte: (Murphy, 2000)

Através da Figura 4.6, verifica-se que a linha (*Voronoi edge*) passa no meio dos corredores e aberturas. O ponto onde mais do que duas arestas de Voronoi se intersectam é conhecido como um vértice de Voronoi (*Voronoi vertex*). Ou seja, num mapa baseado no GVD, as arestas são equivalentes às estradas e os vértices a cruzamentos. Tal facilita a tarefa do robô no seguimento de um caminho gerado a partir de um GVD, uma vez que existe uma estratégia de controlo local implícita que coloca o robô equidistante de todos os obstáculos (Murphy, 2000).

O GVD caracteriza-se por conter as seguintes propriedades: acessibilidade, departibilidade e conetividade. A acessibilidade está associada ao facto de existir um caminho entre o ponto inicial do robô móvel (*start*) e um ponto pertencente GVD. A departibilidade garante que existe um caminho entre o ponto final (*goal*) e um ponto pertencente ao GVD. A conetividade garante que existe um caminho entre o ponto inicial (*start*) e final (*goal*) (Choset, et al., 2005).

Segundo Choset, et al. (2005) o GVD pode ser definido de acordo com o seguinte modelo matemático:

Considerando um espaço de trabalho $\mathcal{W}$ com obstáculos $\mathcal{O}$, o espaço configurado para um obstáculo é definido por um conjunto de posições q que um obstáculo $\mathcal{WO}_i$ intercetado pelo robô $R(q)$. Assim o correspondente espaço dos obstáculos é definido por:

$$\mathcal{Q}_i = q \in \mathcal{Q} | R(q) \bigcap \mathcal{WO}_i \neq 0 \qquad \textbf{(4.36)}$$

E o espaço livre é definido como o conjunto de todos os pontos que não são intercetados por nenhum obstáculo:

$$\mathcal{Q}_{free} = \mathcal{Q} \backslash \left(\bigcup_i \mathcal{Q}_i \right) \quad \textbf{(4.37)}$$

O GVD divide o espaço livre em regiões onde as posições q são mais próximas de um obstáculos i do que os restantes h. Essas regiões, denominadas por regiões de Voronoi são definidas como:

$$\mathfrak{F}_i = \{ q \in \mathcal{Q}_{free} \mid d_i(q) \leq d_h(q) \; \forall h \neq i \} \quad \textbf{(4.38)}$$

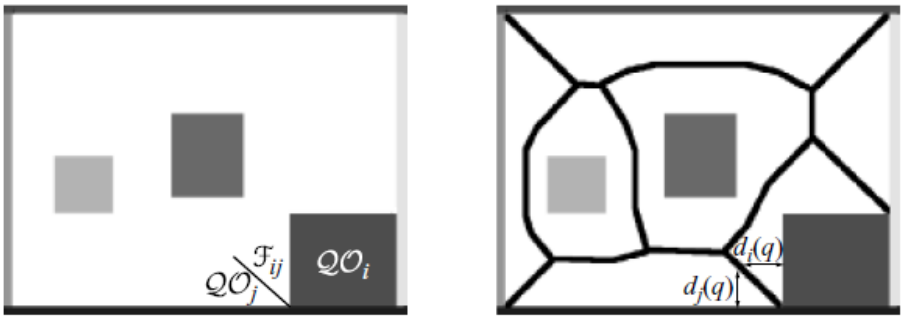


Figura 4.7 – Exemplo da criação de um GVD, com o segmento $\mathfrak{F}_{ij}$, os obstáculos $\mathcal{QO}_i$ e $\mathcal{QO}_j$ e as respetivas distâncias. Fonte: (Choset, et al., 2005)

Onde $d_i(q)$ é a distância a um obstáculo $\mathcal{QO}_i$ de q. O componente essencial para construção do GVD é o conjunto de pontos equidistantes aos obstáculos $\mathcal{QO}_i$ e $\mathcal{QO}_j$ definido com o termo *two-equidistant surface* denotada por $S_{ij} = \{ x \in \mathcal{Q} \mid \left(d_i(q) - d_j(q) \right) = 0 \}$. A superfície (*two-equidistant surface*) atravessa outros obstáculos, sendo necessário restringir o conjunto de pontos aos que são equidistantes a $\mathcal{QO}_i$ e $\mathcal{QO}_j$ e que os têm como os obstáculos mais próximos. A estrutura restringida (*two-equidistant faces*) pode ser denominada por:

$$\mathfrak{F}_{ij} = \{ q \in S_i \mid d_i(q) \leq d_h(q) \; \forall h \}^2 \quad \textbf{(4.39)}$$

A união de todos os pontos (*two-equidistant faces*) forma o GVD:

$$GVD = \bigcup_i \bigcup_j \mathfrak{F}_{ij} \quad \textbf{(4.40)}$$

4.2.1.2 Loop Closure

O *loop closure* consiste em detetar quando o robô revisita um local após ter descoberto uma nova área. Ou seja, um *loop closure* positivo ocorre quando o robô reconhece o local como anteriormente visitado – Figura 4.8. O *loop closure* é crucial para melhorar a robustez dos algoritmos de SLAM (quer topológico, quer métrico). Tal deteção permite aumentar a precisão da estimação da posição atual do robô, o que implica uma melhoria no desempenho do SLAM.

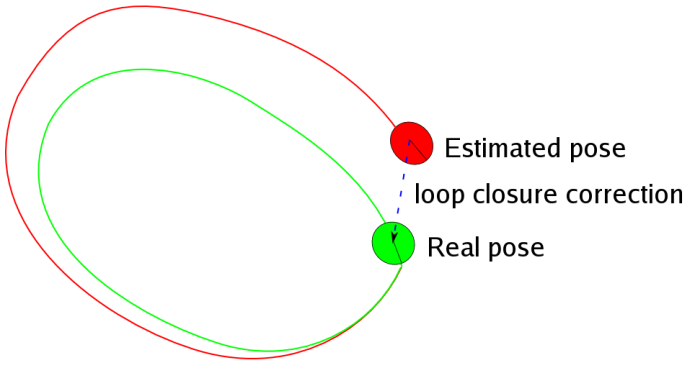


Figura 4.8 - Correção da posição através de *loop closure*. Fonte: (Baillie, 2004).

Existem várias abordagens para detetar um *loop closure*: através de visão (Campos, Correia, & Calado, 2013), ou de *scans* de um *laser rangefinder*, sendo esta a que será implementada neste trabalho. O "*scan matching*" é uma abordagem bastante utilizada, que consiste em associar os *scans* de *laser* às posturas do robô. *Scans* sequenciais são registradas para cada postura, que resulta numa trajetória de posturas do robô e um mapa obtido através de *scans* de *laser*, agora alinhados (Ho & Newman, 2007). Para realizar o *loop closure* recorreu-se ao algoritmo ICP (*Iterative Closest Point*).

O algoritmo ICP surgiu no início dos anos 90 para registar *range data* para modelos de objetos em CAD (*Computer-Aided Design*). O seu objetivo consistia em encontrar a melhor transformação que minimizasse a distância entre as duas nuvens de pontos (Correll, 2016).

Em robótica, o algoritmo ICP é aplicado para alinhar *scans* de *laser* 2D. Por exemplo, a transformação que minimiza o erro entre duas aquisições consecutivas do ambiente após movimento do robô. O algoritmo associa a cada ponto da aquisição ao ponto mais próximo, em coordenadas cartesianas, à aquisição anterior ou superfície a ser alinhada – Figura 4.9 - (Kelly, 2013)

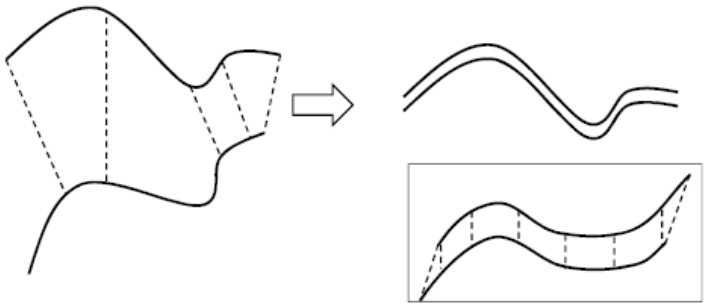


Figura 4.9 - Aplicação do algoritmo ICP. Fonte: (Kelly, 2013).

Porém este é um problema complicado, pois é necessário perceber quais os pontos das duas aquisições consecutivas que são correspondentes, quais os pontos que são *outliers* (discrepantes) (devido ao ruido proveniente dos sensores), e que pontos precisam ser descartados, pois nem todos os pontos se sobrepõem em ambas as aquisições (Correll, 2016).

Segundo Segal, Hähnel, & Thrun (2009) o conceito-chave do algoritmo ICP pode ser sumarizado em dois passos:

1. Calcular as correspondências entre as duas aquisições;
2. Calcular a transformação que minimiza a distância entre pontos correspondentes;

A repetição iterativa destes dois passos resulta tipicamente numa convergência para a transformação desejada. É preciso notar que, devido ao facto de os dados violarem normalmente a suposição de uma completa sobreposição, é necessário adicionar um *threshold* máximo para fazer coincidir dois pontos, d_{max}. Este *threshold* tem em conta o facto de alguns pontos não terem qualquer correspondência na segunda aquisição.

Em grande parte das implementações do algoritmo ICP, a escolha do d_{max} representa um *trade-off* entre a convergência e precisão. Um valor baixo resulta numa má convergência; um valor elevado provoca correspondências incorretas. O algoritmo generalizado para ICP pode ser escrito da seguinte forma (Segal, Hähnel, & Thrun, 2009):

Input: Duas nuvens de pontos: $\mathrm{A} = \{a_i\},\ \mathrm{B} = \{b_i\}$

Uma transformação inicial: T_0

Output: A transformação correta, T, que alinha A e B

$T \leftarrow T_0$;

Enquanto não convergir realizar

para $i \leftarrow 1$ até N realizar

$m_i \leftarrow$ ProcuraPontoPróximoEmA ($T\ .\ b_i$);

se $||m_i - T\ .\ b_i|| \leq d_{max}$ então

$\omega_i \leftarrow 1$;

senão

$\omega_i \leftarrow 0$;

fim

fim

$T \leftarrow \arg\min\{\sum_i \omega_i\ ||T\ .\ b_i - m_i||^2\}$;

fim

Para a aplicação do algoritmo ICP seguiu-se a referência da obra de Corke (2011). Dado duas nuvens de pontos $\mathbf{M_i}$, $\mathbf{D_j}$, $\in \Re^2$, $i,j \in 1 \ldots N$. Deseja-se obter a postura relativa $\boldsymbol{\xi}$ tal que:

$$\boldsymbol{D_i = \xi\ .M_i} \qquad \textbf{(4.41)}$$

Assume-se que para cada ponto de $\mathbf{D_i}$, o ponto correspondente em $\mathbf{M_j}$ é o que está mais perto, ou seja, o que minimiza $\left|\mathbf{D_i} - \mathbf{M_j}\right|$. O passo seguinte consiste em calcular os centróides das nuvens de pontos.

$$\overline{M} = \frac{1}{N_M}\sum_{i=1}^{N_M} \mathbf{M_i} \qquad \textbf{(4.42)}$$

$$\overline{D} = \frac{1}{N_D}\sum_{i=1}^{N_D} \mathbf{D_i} \qquad \textbf{(4.43)}$$

A partir da qual se retira a translação:

$$\mathbf{t} = \bar{D} - \bar{M} \quad (4.44)$$

De seguida calcula-se a matriz $\mathbf{W}$:

$$\mathbf{W} = \sum (\mathbf{M_i} - \bar{M})(\mathbf{D_i} - \bar{D})^T \quad (4.45)$$

Através da decomposição em valores singulares obtém-se:

$$\mathbf{W} = \mathbf{U}\sum\mathbf{V^T} \quad (4.46)$$

Na qual é obtida a matriz de rotação:

$$\mathbf{R} = \mathbf{V}\mathbf{U^T} \quad (4.47)$$

Segundo Corke (2011), a estimação da postura relativa entre as duas nuvens de pontos corresponde a:

$$\boldsymbol{\xi} \sim (\mathbf{R}, \mathbf{t}) \quad (4.48)$$

4.2.2 Algoritmo de Dijkstra

De modo a obter-se um plano eficaz é necessário determinar o melhor caminho, o qual está normalmente associado ao caminho mais curto. O algoritmo de Dijkstra, concebido pelo cientista da computação holandês Edsger Dijkstra (Dijkstra, 1959), soluciona o problema da escolha do caminho mais curto num grafo (Siegwart, Nourbakhsh, & Scaramuzza, 2011).

O algoritmo de Dijkstra consiste em ponderar com pesos (relacionados com o comprimento) os vários caminhos (neste caso, as arestas do diagrama de Voronoi) possíveis entre a sua localização e a meta (*goal*). O algoritmo escolhe o vértice mais próximo para adicionar ao conjunto da solução, ou seja, irá calcular o caminho de custo mínimo entre os vértices de um grafo.

Segundo Thulasiraman, Arumugam, Brandstädt, & Nishizeki (2016) o algoritmo de Dijkstra pode ser apresentado da seguinte forma:

***Input*:** Grafo de conexões $G = (V, E)$ com comprimentos $w(e) = w(i,j) \geq 0$ para cada aresta $e = (i,j)$.

***Output*:** Caminho mais curto e seu comprimento entre o vértice 1 e todos os outros vértices.

Inicio

$\lambda\,(1) \;=\; 0$;

$\lambda\,(i) \;=\; \infty$, para cada $i \;\neq\; 1$;

$PRED(i) \;=\; i$, para cada vértice i;

$PERM(i) \;=\; 0$, para cada vértice i;

$S_0 \;=\; \emptyset$;

para $k \;=\; 1, 2, \ldots, n$ fazer

Seja u_k um vértice que não é considerado permanente e tem o menor valor de λ;

$PERM(u_k) = \;1$;

$S_k = \; S_{k-1} \cup \{u_k\}$;

para cada aresta $e \;=\; (u_k, j)$ fazer

se $\lambda\,(j) \;>\; \lambda\,(u_k) \;+\; w(u_k, j)$, então

$\lambda\,(j) = \lambda\,(u_k) \;+\; w(u_k, j)$ e $PRED(j) = \; u_k$;

fim

fim

fim

O algoritmo começa por estimar o peso (ou distância) entre todos os vértices ao vértice inicial 1. Essa estimativa para o vértice i será denominada por $\lambda(i)$. Inicialmente, o algoritmo de Dijkstra começa por atribuir $\lambda(1) = 0$ e $\lambda(j) = \infty$ para todo $j \neq 1$. Inicialmente nenhum vértice é considerado permanente. Um vértice i não permanente com o valor mínimo de λ é considerado permanente, passando a ser denotado por u_i. Todas as arestas (u_i, j) sendo j não permanente, são examinados para verificar se violam a condição ótima:

$$\lambda(i) \;+\; w(i,j) \;\geq\; \lambda(j) \quad \text{para todo} \quad e \;=\; (i,j) \qquad \textbf{(4.49)}$$

Se existir uma aresta (u_i, j), então o valor de $\lambda(j)$ é atualizado para $\lambda(j) = \lambda(i) + w(u_i, j)$ e o antecessor de j é definido para u_i. O algoritmo termina quando todos os vértices forem considerados permanentes, assumido que é possível chegar a todos os vértices através do vértice inicial. Ou seja, ao terminar o algoritmo os valores de $\lambda\ (i)$ serão finitos (Thulasiraman, Arumugam, Brandstädt, & Nishizeki, 2016).

No algoritmo descrito acima é utilizada uma *array* $PERM$ para indicar quais os vértices que são considerados permanentes. Começando com $PERM(v) = 0$ para todo o v. $PRED$ é uma *array* que guarda a sequência com que os vértices foram considerados permanentes. Se um vértice v for considerado permanente então v, $PRED(v)$, $PRED(PRED(v))$, . . ., 1 são os vértices que definem o caminho mais curto entre o vértice v e 1.

4.2.3 Exploração baseada em fronteiras

A exploração baseada em fronteiras, tal como o nome indica, é uma abordagem que utiliza as deteções de fronteiras para explorar o ambiente. A ideia essencial consiste em ganhar a maior informação possível sobre o ambiente movendo o robô para os limites (*boundary*) entre o espaço aberto e o território desconhecido (Yamauchi, 1997).

Quando um robô se move para uma fronteira, tem a capacidade de observar o espaço inexplorado e adicionar nova informação ao seu mapa. Como resultado, o território mapeado expande, avançando a fronteira entre o conhecido e desconhecido. Ao mover-se sucessivamente para as fronteiras, o robô aumenta progressivamente o seu conhecimento do ambiente até este ter sido completamente explorado.

Segundo o estudo de Yamauchi (1997), cria-se uma grelha de ocupação, onde cada célula na grelha é classificada. Essa classificação surge da comparação entre a probabilidade de ocupação da célula com a probabilidade inicial que é atribuída a todas as células com o valor de 0.5. Cada célula é classificada como:

- Aberta: probabilidade de ocupação < probabilidade inicial;
- Desconhecida: probabilidade de ocupação = probabilidade inicial;
- Ocupada: probabilidade de ocupação > probabilidade inicial.

As células com uma probabilidade de ocupação baixa são representadas por um espaço branco; células com probabilidade de ocupação desconhecida são representadas por um ponto pequeno; células com uma probabilidade de ocupação elevada são representadas por pontos grandes (ver Figura 4.10).

Para a localização das fronteiras entre o espaço aberto e desconhecido é utilizado um processo análogo de deteção de bordas e extração de regiões. Qualquer célula aberta adjacente a uma célula desconhecida é identificada como uma célula do limite da fronteira. As células adjacentes às células do limite são agrupadas numa região fronteiriça. Qualquer região fronteiriça acima de um valor mínimo (aproximadamente o tamanho do robô) é considerada uma fronteira. Após a deteção da(s) fronteira(s), no caso de existirem várias fronteiras possíveis, opta-se pela fronteira mais próxima que ainda não tenha sido visitada.

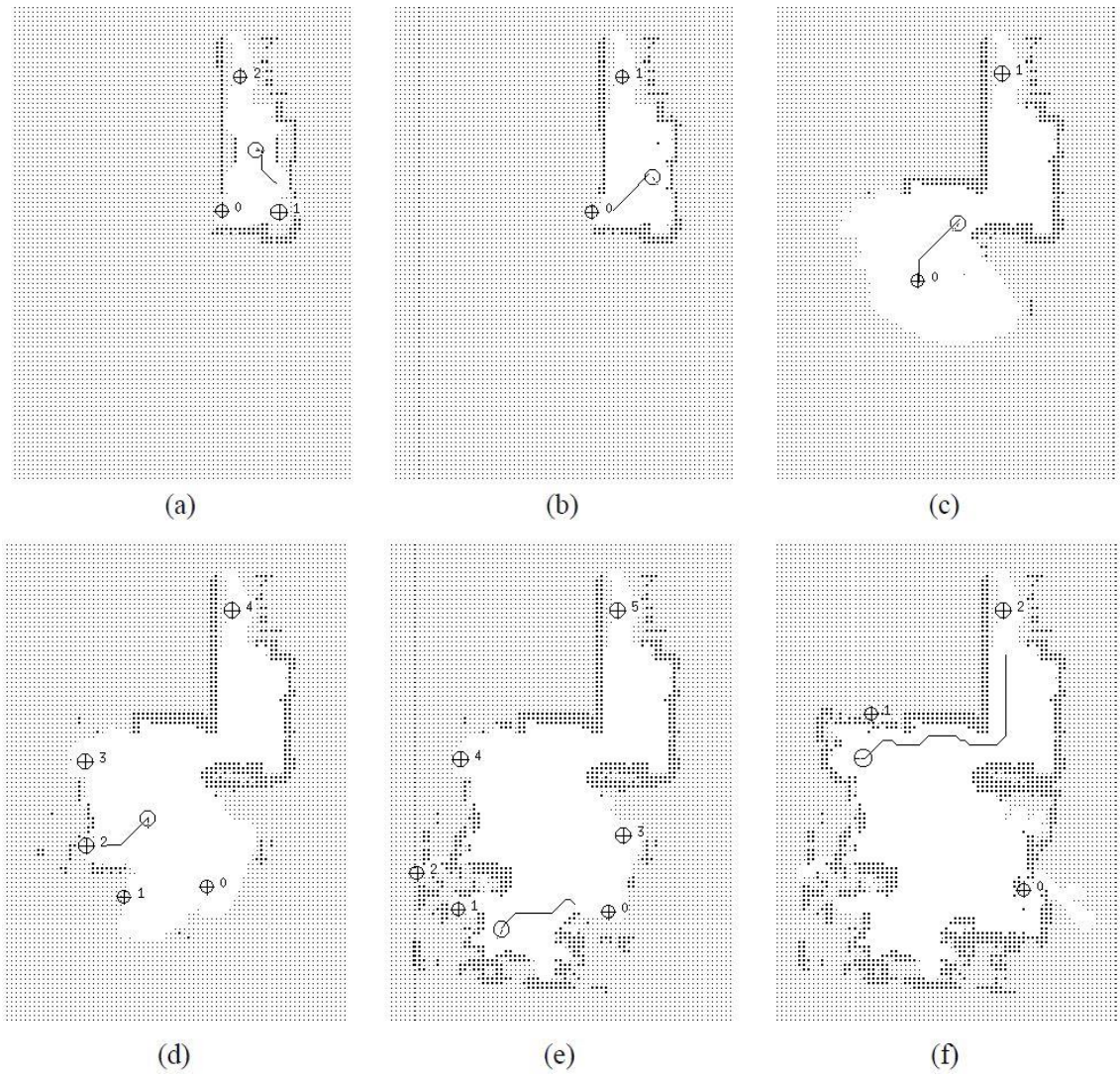


Figura 4.10 – Exemplo de uma exploração baseada em fronteiras. Fonte: (Yamauchi, 1997)

A Figura 4.10 representa um exemplo de uma exploração baseada em fronteiras. (*a*) Com base no mapa inicial, verifica-se a existência de três *boundaries*,$\bigoplus_0$,$\bigoplus_1$ e $\bigoplus_2$, optando pela deslocação para $\bigoplus_1$. (*b*) Chegando à zona pré-definida, conclui-se que se trata de uma zona fechada, tendo agora duas opções $\bigoplus_0$ e $\bigoplus_1$(antiga $\bigoplus_2$), optando-se pela zona $\bigoplus_0$. (*c*) Ao alcançar a nova zona pré-definida, regista-se um aumento na informação sobre o ambiente,

obtendo-se assim uma nova zona para explorar. O processo de exploração baseado em fronteiras é iterativo acabando quando não existirem zonas para explorar. Nessa altura, obteve-se um mapa completo do ambiente referente às suas fronteiras.

4.3 Algoritmo de Exploração e Mapeamento

O algoritmo de exploração e mapeamento desenvolvido é esquematizado na Figura 4.11, que resume os passos envolvidos numa iteração. Sem perda de generalidade, assume-se que existe um mapa k da iteração anterior. O passo 1 do algoritmo identifica as fronteiras existentes nesse mapa e calcula um caminho para a fronteira mais próxima; no passo 2 executa-se o movimento correspondente ao plano anterior; no passo 3 faz-se a captura de *scan* na posição atual e no passo 4 calcula-se a incerteza entre posições, verifica-se a existência de revisitação e, se necessário, corrige-se a estimativa da trajetória; finalmente, no passo 5 alinha-se o *scan* atual com os mais próximos por forma a validar as fronteiras presentes nos *scans*.

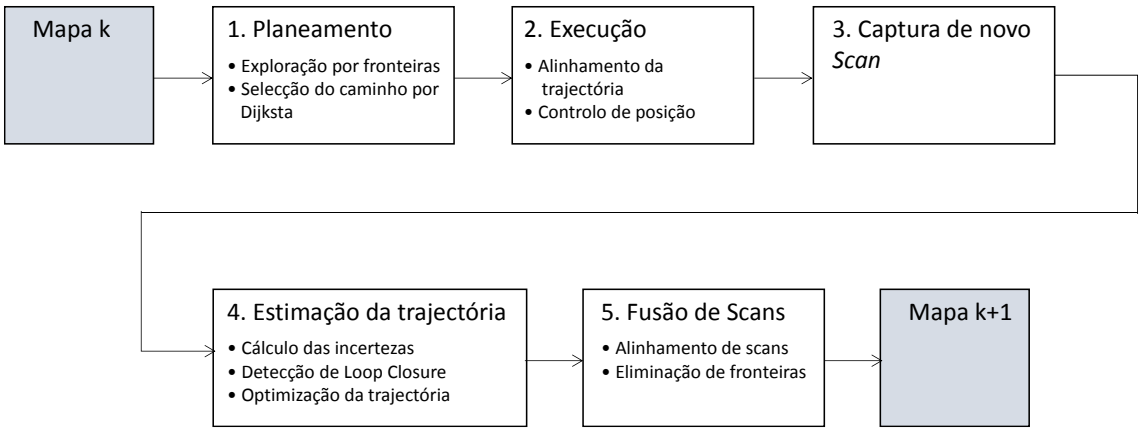


Figura 4.11 - Esquema do algoritmo de exploração e mapeamento desenvolvido.

A apresentação dos conceitos e etapas envolvidos no algoritmo não seguirá a ordem daquela sequência mas, por uma questão de sistematização e inteligibilidade, será feita por ordem de relevância e precedência para a explicação dos conceitos subsequentes. Assim, começará por apresentar-se o mapa híbrido que sustenta o algoritmo; de seguida descrevem-se os fundamentos do Pose-SLAM (passo 4), posteriormente aborda-se a fusão de *scans* (passo 5) e termina-se a secção com o planeamento e execução (passos 1 e 2).

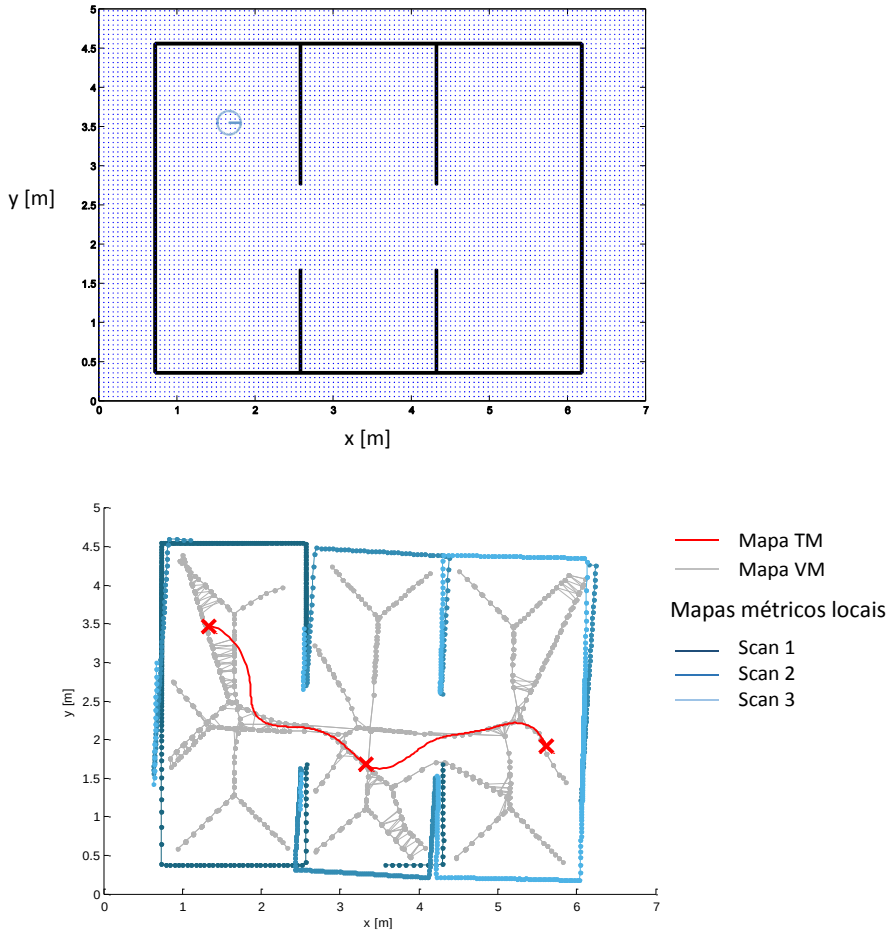


Figura 4.12 - Exemplo de um mapa híbrido construído pelo algoritmo de exploração e mapeamento. Em cima: estrutura real do ambiente, em baixo: componentes do mapa híbrido.

4.3.1 Mapa híbrido

O sistema de exploração, navegação e mapeamento que foi desenvolvido assenta numa representação híbrida do ambiente. Esta representação é composta por um conjunto de mapas métricos locais e ligações topológicas entre eles (ver Figura 4.12).

4.3.1.1 Mapas métricos locais

Os mapas métricos locais (a azul na Figura 4.12) estão intimamente ligados com o tipo de sensor usado na exploração do ambiente, o laser *rangefinder*. Cada mapa local consiste no conjunto de pontos obtidos numa leitura do *rangefinder*, conjunto que designaremos *scan* e representaremos pela expressão S^r indicando que é uma leitura feita no referencial do robô. Dada uma estimativa da posição do robô no momento em que essa leitura foi feita, é possível transformar as coordenadas do *scan* do referencial do robô para o referencial do mundo, obtendo-se S^0. Em cada iteração k do algoritmo de exploração é adquirido um novo *scan*, referido porS_k^0.

4.3.1.2 Mapas topológicos

O algoritmo proposto faz uso de dois tipos de representações topológicas: a primeira (a vermelho na Figura 4.12) consiste nas ligações entre as posições do robô registadas até ao momento, representando a acessibilidade entre duas posições. Assim, este grafo inclui as ligações entre posições consecutivas do robô e ainda as ligações decorrentes da deteção de revisitação (*loop closure*). Esta representação inclui também informação métrica: a das posturas estimadas e da incerteza a elas associada. Por isso, esta representação é fundamental no algoritmo para avaliar a posição relativa entre duas posições, bem como a sua incerteza, e para a correção das estimativas de postura através do algoritmo de Pose-SLAM. Este mapa será designado TM (*Trajectory Map*) porque fundamentalmente descreve a topologia da trajetória realizada e a sua estimativa métrica.

A segunda componente topológica do mapa (a cinzento na Figura 4.12) é construída com base nos diagramas de Voronoi que são extraídos de cada mapa local. Concretamente, este mapa topológico, designado por VM (*Voronoi based Map*), é um grafo não direcional que contém todos os nós dos diagramas de Voronoi locais e as respetivas ligações. A estas são acrescentadas ligações adicionais que ligam nós de diferentes diagramas locais. São estas ligações que transformam um conjunto de diagramas de Voronoi independentes num mapa topológico completo do ambiente. Para se estabelecerem estas ligações procedeu-se da forma descrita de seguida.

Para cada novo *scan* extraiu-se o seu diagrama de Voronoi e identificou-se o conjunto $\mathbf{C_k}$ dos *scans* anteriores contra os quais é feita a fusão do *scan* atual (ver secção 4.3.3 sobre a forma de cálculo deste conjunto). De seguida, os *scans* desse conjunto são alinhados com o *scan* atual, usando o método ICP, e as transformações obtidas por esse meio são aplicadas aos respetivos diagramas de Voronoi. Depois de alinhados os diagramas de Voronoi determina-se, para cada nó do diagrama atual, o nó mais próximo entre todos os nós dos restantes diagramas. Para nós cuja distância é inferior a um *threshold*, adiciona-se uma ligação, que o une ao respetivo nó mais próximo.

4.3.2 Pose-SLAM

Nesta secção descrever-se-ão brevemente os conceitos básicos que fundamentam o algoritmo de mapeamento Pose-SLAM. A principal referência bibliográfica que documenta este algoritmo é o tutorial de Grisetti, Kummerle, Stachniss, & Burgard, (2010).

4.3.2.1 Conceitos básicos

No âmbito do Pose-SLAM, a trajetória do robô é estimada através da minimização da função de custo (Grisetti, Kummerle, Stachniss, & Burgard, 2010)

$$\hat{\mathbf{x}} = \operatorname{argmin} \sum_{ij} \mathbf{e}_{\mathbf{ij}}' \boldsymbol{\Omega}_{\mathbf{ij}} \mathbf{e}_{\mathbf{ij}} \quad \textbf{(4.50)}$$

onde $\hat{\mathbf{x}}$ é a variável gaussiana multidimensional dos estados estimados, $\mathbf{e}_{\mathbf{ij}}$ é o erro entre uma medição de postura relativa do robô e a diferença entre estados estimados e $\boldsymbol{\Omega}_{\mathbf{ij}}$ é a matriz de informação relativa à medição.

O vetor de estados $\mathbf{x}$ resulta da concatenação dos vetores de estados de cada iteração, $\mathbf{x}_{\mathbf{k}}$, os quais contêm as coordenadas de postura do robô em SE(2):

$$x = [x_1, x_2, \dots, x_n]' \quad \textbf{(4.51)}$$

$$x_k = [x_k, y_k, \theta_k]' \quad \textbf{(4.52)}$$

Para se realizar a otimização, o sistema gaussiano que representa a trajetória é construído na chamada forma canónica, onde os parâmetros são o vetor de informação, $\mathbf{b}$, e a matriz de informação $\mathbf{H}$, que se relacionam com o vetor média, $\mathbf{x}$, e a matriz de covariância, $\boldsymbol{\Sigma}$, da representação por momentos, através de (Koller & Friedman, 2009):

$$\mathbf{b} = \boldsymbol{\Sigma}^{-1}\mathbf{x} \quad \textbf{(4.53)}$$

$$\mathbf{H} = \boldsymbol{\Sigma}^{-1} \quad \textbf{(4.54)}$$

De acordo com Grisetti, Kummerle, Stachniss, & Burgard, (2010), o vetor de informação $\mathbf{b}$ e a matriz de informação $\mathbf{H}$ são calculados por:

$$\mathbf{b} = \sum_{ij} \mathbf{e}_{\mathbf{ij}}^{\mathbf{t}} \boldsymbol{\Omega}_{\mathbf{ij}} \mathbf{J}_{\mathbf{ij}} \quad \textbf{(4.55)}$$

$$\mathbf{H} = \sum_{ij} \mathbf{J}_{\mathbf{ij}}' \boldsymbol{\Omega}_{\mathbf{ij}} \mathbf{J}_{\mathbf{ij}} \quad \textbf{(4.56)}$$

Nestas expressões $\mathbf{J_{ij}}$ é o jacobiano de $\mathbf{e_{ij}}$ e as matrizes de informação $\mathbf{\Omega_{ij}}$ são dadas pela inversas das matrizes de covariância $\mathbf{P_{ij}}$ entre duas posturas.

Neste trabalho, as matrizes de covariância $\mathbf{P_{ij}}$ são calculadas em cada iteração do algoritmo de exploração. Durante a execução do movimento que leva o robô até à fronteira seguinte, a matriz de covariância é calculada por (Corke, 2011):

$$\mathbf{P}(t) = \mathbf{F}_{\boldsymbol{x}}\mathbf{P}(t-1)\mathbf{F'}_{\boldsymbol{x}} + \mathbf{F}_{\boldsymbol{v}}\mathbf{V}\mathbf{F'}_{\boldsymbol{v}} \quad \textbf{(4.57)}$$

onde t designa o instante de tempo contado a partir do início do movimento e $\mathbf{P}(0)$ é inicializado como uma matriz nula, o que corresponde a admitir-se que no início a incerteza é zero. $\mathbf{F}_{\boldsymbol{x}}$ e $\mathbf{F}_{\boldsymbol{v}}$ são os Jacobianos da cinemática direta do robô relativos às variáveis de estado e de atuação, respetivamente.

Por forma a obter-se a incerteza relativa entre quaisquer duas posturas é necessário ter acesso à matriz de covariância global, $\mathbf{\Sigma}$. De acordo com a relação entre as representações na forma de momentos e na forma canónica das gaussianas multivariável, Σ e H relacionam-se por:

$$\mathbf{\Sigma} = \mathbf{H}^{-1} \quad \textbf{(4.58)}$$

Assim, e uma vez estimada a matriz $\mathbf{H}$, $\mathbf{\Sigma}$ é obtida pela inversão daquela matriz.

4.3.2.2 Estimação da distância entre duas posições

Sejam $\mathbf{p_i}$ e $\mathbf{p_j}$ os vetores de posição do robô nas iterações i e j:

$$\mathbf{p_i} = [x_i, y_i]', \mathbf{p_j} = \left[x_j, y_j\right]' \quad \textbf{(4.59)}$$

A diferença entre as duas posições é dada por:

$$\mathbf{d_{ij}} = \mathbf{p_i} - \mathbf{p_j} \quad \textbf{(4.60)}$$

Uma vez que o vetor de estados $\mathbf{x}$ é uma variável gaussiana, e dado que $\mathbf{d_{ij}}$ pode ser escrita como uma transformação afim daquela variável, $\mathbf{d_{ij}}$ é também uma variável gaussiana, com média:

$$\mathbf{\mu_{d_{ij}}} = \mathbf{\mu_{p_i}} - \mathbf{\mu_{p_j}} \quad \textbf{(4.61)}$$

e covariância:

$$\mathbf{\Sigma_{ij}} = \mathbf{A_{ij}}\,\mathbf{\Sigma}\,\mathbf{A'_{ij}} \qquad \textbf{(4.62)}$$

Na última expressão $\mathbf{A_{ij}}$ designa a matriz de coeficientes da transformação afim subjacente à Equação 4.60.

$$\mathbf{A_{ij}} = \begin{bmatrix} \cdots & \overset{3(i-1)+1}{1} & & \cdots & \overset{3(j-1)+1}{-1} & & \cdots \\ \cdots & & 1 & \cdots & & -1 & \cdots \end{bmatrix}. \qquad \textbf{(4.63)}$$

4.3.2.3 Correção da estimativa

Durante a exploração do ambiente, o vetor de estados é estendido em cada nova iteração, com a concatenação das variáveis de estado correspondentes à postura mais recente:

$$\mathrm{x} = [\mathrm{x}', \mathrm{x_k}]' \qquad \textbf{(4.64)}$$

De modo semelhante, a matriz de informação é acrescida da linha e coluna correspondentes ao estado atual e é atualizada segundo a (Equação 4.56). Nesta atualização introduzem-se apenas os constrangimentos triviais, aqueles que descrevem a informação conhecida entre duas posturas consecutivas, medida pela odometria. Esta informação adicional não obriga à correção da estimativa da trajetória, pois a nova informação não se relaciona com as posturas passadas, para além da $k-1$. Contudo, quando é detetada a revisitação de uma posição estamos perante constrangimentos que obrigam à correção das estimativas de $\mathbf{x}$. De facto, devido aos erros existentes, quer na estimativa de $\mathbf{x}$, quer na postura relativa medida durante a revisitação, estas serão, em algum grau, inconsistentes, o que justifica a re-estimação de $\mathbf{x}$.

Como demonstrado em Grisetti, Kummerle, Stachniss, & Burgard, (2010), a solução de (Equação 4.50) passa por resolver o sistema de equações

$$\mathbf{H}\Delta\mathrm{x} = \mathbf{b} \qquad \textbf{(4.65)}$$

Nesta expressão $\Delta\mathrm{x}$ é a variação a aplicar ao vetor de estados estimado que conduzirá à solução ótima. Tratando-se este problema de minimização de um problema não-linear, devido às operações envolvidas no cálculo de $\mathbf{e_{ij}}$, a otimização é feita de forma iterativa, através do método de Gauss-Newton. Segundo este método, $\Delta\mathrm{x}$ é calculado repetidamente e aplicado a $\hat{\mathbf{x}}$,

por forma a obter-se uma nova estimativa. Devido à natureza não linear do problema, em cada iteração os erros e Jacobianos são calculados utilizando os valores da estimativa mais recente.

4.3.3 Fusão entre scans

Para efeitos de fusão de informação entre mapas locais define-se, para cada mapa, o conjunto daqueles relativamente aos quais a fusão é válida. A razão por que não se faz a fusão com todos os restantes mapas locais prende-se com os erros inerentes às estimativas de posição do robô. De facto, uma vez que a estimativa de posição tem erros associados, e que os mapas locais dependem desta, através da transformação de coordenadas do referencial do robô para o referencial do mundo, os mapas locais têm também erros associados. Por esta razão, a posição relativa entre mapas contém também incerteza, que será tanto maior quanto maior for o percurso do robô que os separa, já que os erros se acumulam com o espaço percorrido. Devido a esta incerteza, a fusão de informação entre mapas locais que se encontram afastados, ou mesmo próximos mas ligados por um percurso longo entre eles, pode não ser fiável.

O critério usado para a seleção dos *scans* a incluir em $\mathbf{C_k}$ é baseado na matriz de covariância $\mathbf{\Sigma_{ik}}$, que descreve a incerteza entre um *scan* i e o *scan* k (ver secção 4.3.2.2 para a forma de cálculo desta matriz). Designando por λ_{i1} e λ_{i2} os dois valores próprios desta matriz, a selecção é feita impondo que estes valores devem ser menores que um limite pré-definido, th_f, que estabelece a incerteza máxima admissível numa direção:

$$C_k = \{i \in \mathbb{N} | i < k \wedge \lambda_{i1}, \lambda_{i2} < th_f\} \qquad \textbf{(4.66)}$$

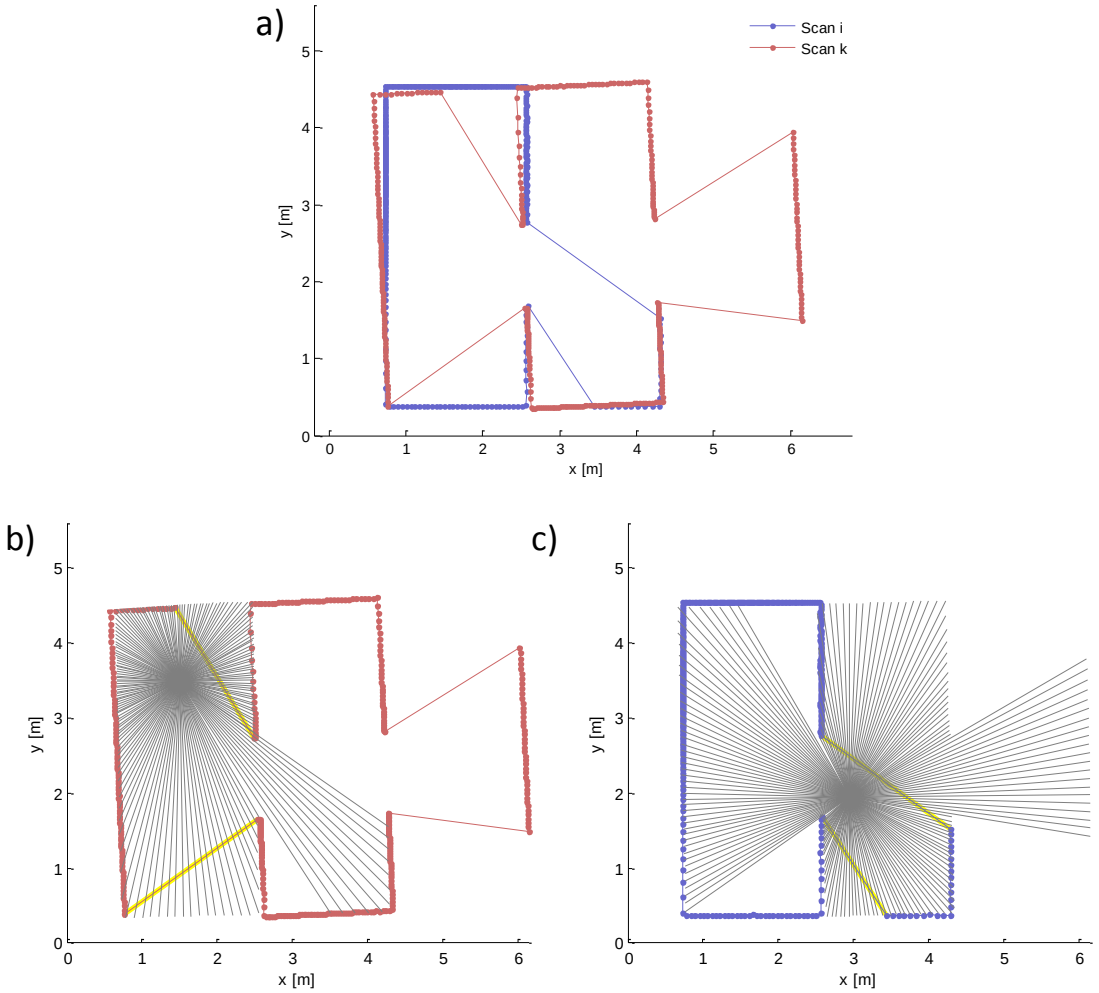


Figura 4.13 - Exemplo de alinhamento de *scans*: a) *scans* i e k, b) validação das fronteiras do *scan* k, c) validação das fronteiras do *scan* i. As fronteiras a amarelo do *scan* $k(i)$, em b (c), são eliminadas, por serem intersectadas por feixes do *scan* $i(k)$.

A fusão entre *scans* tem por objetivo verificar quais os segmentos de cada *scan* que são limites válidos do ambiente e quais são fronteiras para regiões já observadas. Com esse fim, determina-se, para cada par i,k, a matriz de transformação A_{ik}, que alinha o *scan* i com o *scan* k, usando o método ICP. De seguida os pontos do scan i são transformados por A_{ik} (ver Figura 4.13.c) e determina-se a intersecção entre os feixes do *rangefinder* emitidos na posição k com os segmentos do scan i. Os segmentos para os quais existe intersecção são eliminados, pois correspondem a limites para uma região observada na posição k. O mesmo procedimento é realizado para a verificação dos segmentos do *scan* k contra os feixes emitidos na posição i (Figura 4.13.b). Este procedimento leva à eliminação de segmentos do *scan* k.

4.3.4 Planeamento e Execução

Na etapa de planeamento determina-se um plano de movimento para aproximar o robô da fronteira mais próxima. O primeiro passo nesta etapa é o da determinação dos nós do mapa VM cuja posição métrica é mais próxima de cada fronteira existente. De seguida, calcula-se o caminho sobre este mapa que leva o robô da posição atual a cada um desses nós. A fronteira que corresponde ao caminho mais curto é selecionada e o caminho respetivo é guardado para execução.

Na fase de execução não é possível usar a posição métrica dos nós como referência de posição para o robô, devido aos erros existentes na estimativa de posição. Para contornar esta dificuldade recorre-se ao conceito de mapa local. Durante a execução do movimento determina-se o mapa local que corresponde ao nó de referência e alinha-se o respetivo *scan* com o *scan* captado pelo robô. Essa transformação é aplicada ao nó de referência obtendo-se assim uma posição de referência válida.

5 VALIDAÇÃO EXPERIMENTAL E RESULTADOS OBTIDOS

Neste capítulo é abordado o trabalho realizado após os estudos preliminares já referenciados no Capitulo 3. Será realizado um estudo sobre o sistema de deteção e extinção de chama do robô. No final será apresentado o desenvolvimento de um simulador de navegação, que permite testar o conceito em diferentes cenários num ambiente virtual.

5.1 Sistema de Deteção e Extinção de Chama

No presente ensaio é proposto verificar o sistema de deteção e extinção de chama anteriormente retratados. Para a realização do ensaio partiu-se do pressuposto que o robô entrou no quarto no qual está a chama. Após esse instante ocorre a situação que se pretende ensaiar.

5.1.1 Descrição do algoritmo

Para representar o algoritmo por trás do sistema de deteção e extinção de chama recorre-se a um fluxograma - Figura 5.1. Começando pela chegada de informação por parte da FL, o robô depara-se com a seguinte questão: “Existe chama no quarto?”. Caso a resposta seja negativa (condição lógica falsa), o robô vai averiguar se a variável flag tem o valor 1 associado. Caso a resposta seja negativa (condição lógica falsa) será atribuído o valor 1 à variável flag, entrando depois no “Modo de Varrimento”. Este modo consiste em rodar o robô 90° para a sua esquerda procurando pela chama, e depois 180° para a sua direita. Se após a rotação o robô não detetar chama, conclui-se que não existe chama no quarto, terminando assim o algoritmo (condição lógica verdadeira para flag = 1).

Caso a resposta à questão “Existe chama no quarto?” seja positiva (condição lógica verdadeira), o robô terá uma nova questão a responder: “Está suficientemente próxima?”. Caso a resposta seja positiva (condição logica verdadeira), o robô irá para os motores das rodas e atuar a ventoinha durante 10 segundos. Após os 10 segundos da atuação da ventoinha, o robô irá verificar através da FL se ainda existe chama. Caso ainda exista o robô irá repetir as instruções anteriores, caso não detete chama, conclui que extinguiu a chama e termina o algoritmo.

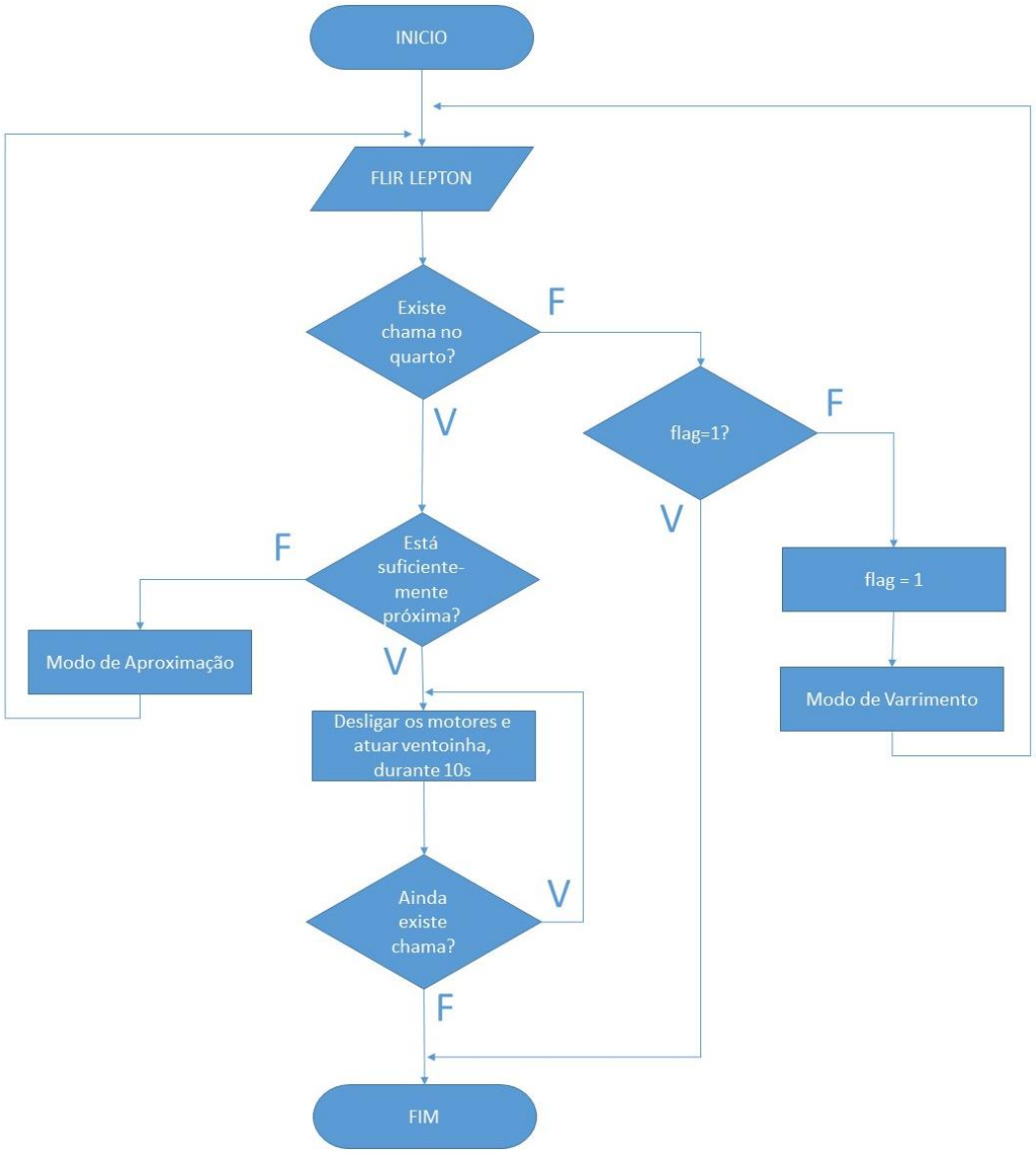


Figura 5.1 - Fluxograma do sistema de Deteção e Extinção de chama.

Caso a resposta à questão "Está suficientemente próxima?" seja negativa (condição lógica falsa) o robô entra no Modo de Aproximação. Este modo, semelhante ao seguidor de linha abordado anteriormente, consiste em aproximar o robô da chama, através do alinhamento do centróide dos *pixéis* brancos obtida pela FL com o meio da imagem obtida pela FL. Sendo que a atuação nos motores das rodas é baseado no erro entre esse alinhamento, conforme apresentado na secção 3.1.

5.1.2 Ensaios Realizados

No primeiro ensaio considera-se que o robô está a meio do quarto e a vela encontra-se num canto do quarto - Figura 5.2.

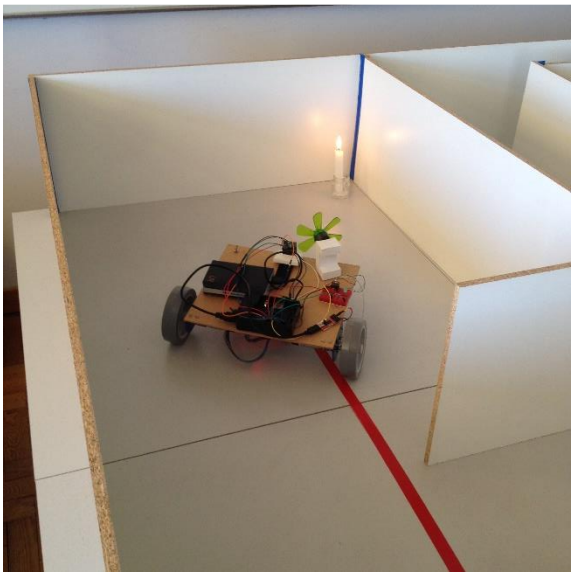


Figura 5.2 - Cenário do primeiro ensaio.

No segundo ensaio considera-se que o robô está na entrada do quarto e a vela situa-se a meio do quarto - Figura 5.3.

Figura 5.3 - Cenário do segundo ensaio.

No terceiro ensaio considera-se que o robô está na entrada do quarto e a vela situa-se num canto do quarto (aumento da distância) - Figura 5.4.

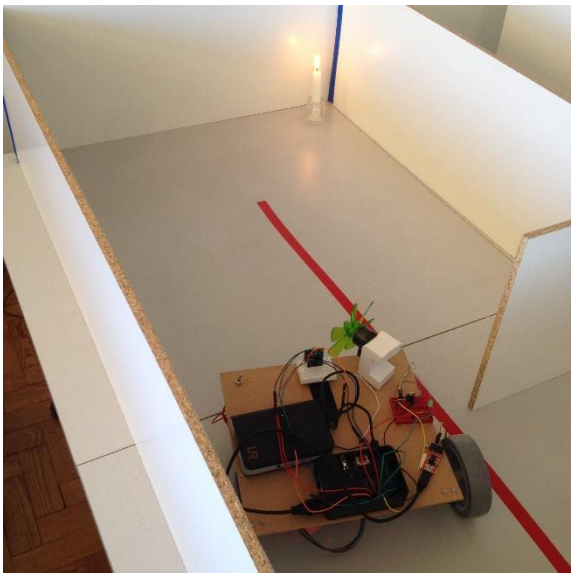


Figura 5.4 - Cenário do terceiro ensaio.

No quarto ensaio considera-se que o robô está situado na entrada do quarto sem conhecimento da localização da vela - Figura 5.5. Ou seja, inicialmente, a vela não está na linha de visão da câmara termográfica.

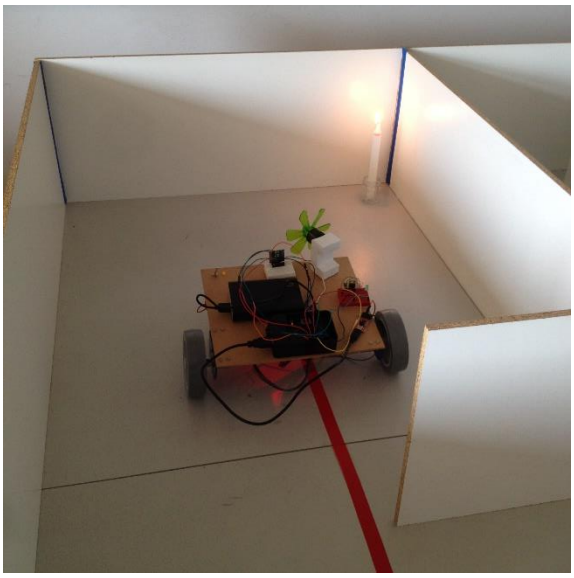


Figura 5.5 - Cenário do quarto ensaio.

Após a realização de 20 ensaios em cada cenário obteve-se uma taxa de sucesso de 100%, ou seja, em todos os 80 ensaios realizados, o robô conseguiu extinguir a chama. Um outro fator que foi tido em conta foi a distância à qual o robô parava para extinguir a chama. Considerando um raio de segurança de 10 cm de modo a garantir que o robô não tombe a vela e uma distância

máxima de 20 cm de modo a garantir que robô esteja a uma distância suficiente que lhe permita extinguir a chama. Com base nos ensaios obteve-se os seguintes resultados:

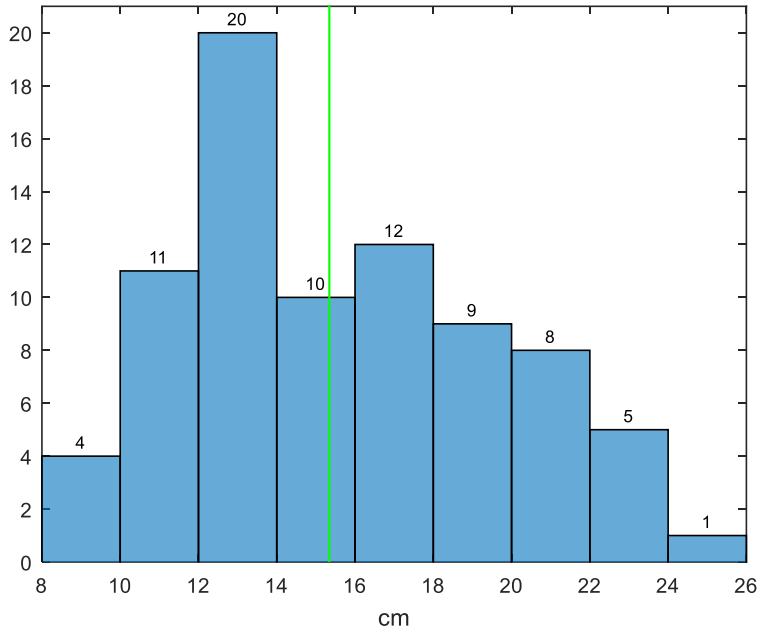


Figura 5.6 - Valores obtidos da distância de paragem do robô (cm).

Através da Figura 5.6 é possível verificar que a amostra tem uma média aproximada de 15.3 cm (linha verde). Em relação aos limites é possível concluir que foi observada 4 vezes uma distância de paragem inferior a 10 cm (correspondente a 5%). Para valores superiores a 20 cm verificou-se 14 observações (correspondente a 17.5%). Importa salientar que das 14 observações, 5 encontram-se no intervalo entre 20 cm e 20.5 cm, não sendo muito distante da referência máxima – Figura 5.7.

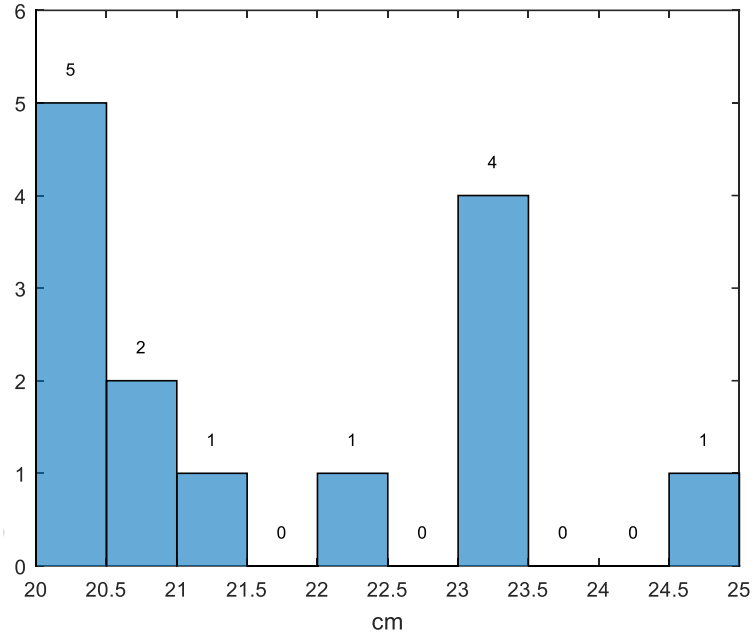


Figura 5.7 – Valores obtidos no intervalo de 20 cm a 25 cm.

Com base nos ensaios realizados é possível afirmar que o sistema implementado consegue cumprir o seu principal objetivo (extinção da chama), pois registou-se uma eficácia de 100%. Em relação à distância de segurança pré-definida é possível afirmar que existe uma probabilidade de 77.5% do robô parar no intervalo desejado. Os dados completos do ensaio situam-se no Anexo 2.

5.2 Simulador

Como estudo preliminar de navegação, a navegação do robô foi testada num ambiente de simulação. Este ensaio foi realizado numa aplicação desenvolvida em Matlab® para testar algoritmos de exploração, navegação e mapeamento.

A principal vantagem associada à criação de ambientes de simulação são os custos associados em comparação com um ambiente e protótipo físico. Através do simulador é possível verificar de um modo trivial a correta implementação dos algoritmos num ambiente virtual.

O simulador é composto por dois locais distintos: o primeiro é o gráfico onde se insere o mapa e o segundo é uma barra de comandos com os botões de interface – Figura 5.8. Para o utilizador criar o cenário (mapa), dirige-se ao botão "*Add Walls*", o que permite ao utilizador criar paredes através de dois cliques em pontos distintos dentro do gráfico, criando um reta entre esse dois pontos escolhidos.

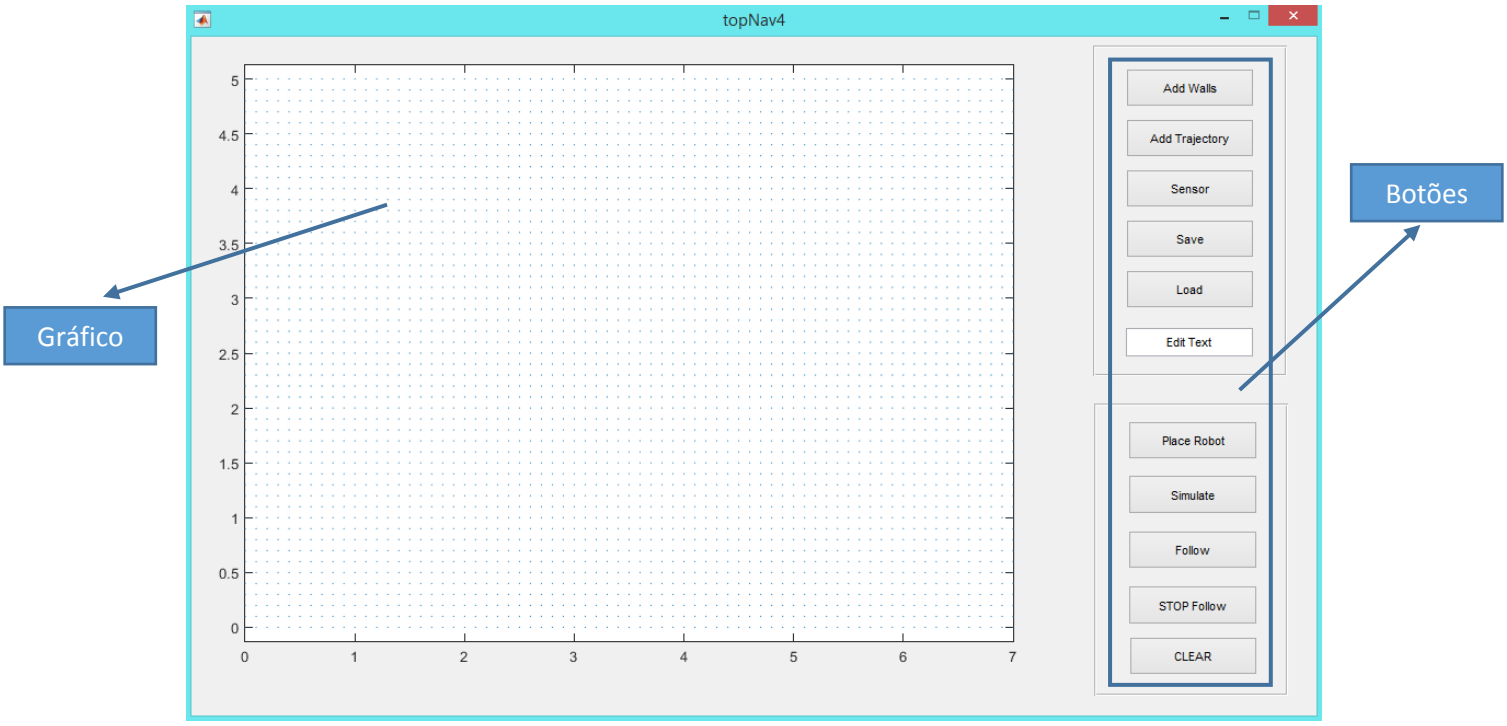


Figura 5.8 – Interface gráfica do simulador.

Após a conclusão da criação do mapa, o próximo passo é colocar a posição do robô através do botão "*Place Robot*". A sua colocação tal como a parede é definida por dois cliques, o primeiro define a posição do robô no mapa e o segundo a sua orientação.

Tendo a postura inicial do robô e o ambiente a ser explorado definidos, o utilizador tem a possibilidade de guardar esses dados, para uma eventual repetição futura do ensaio. Para esse efeito existem os botões "*Save*" e "*Load*", para guardar e carregar os dados iniciais. Para colocar o nome do ficheiro é utilizada a barra de escrita que na Figura 5.8 está com o nome "*Edit Text*". Para iniciar a simulação dirige-se ao botão "*Follow*", e o robô de forma autónoma irá explorar o ambiente até o algoritmo de exploração terminar.

O primeiro ensaio, devido à sua simplicidade, vai incidir em explicar o funcionamento do simulador e os seus *outputs*. O segundo ensaio tem como objetivo demonstrar a capacidade do robô voltar a uma fronteira que foi identificada mas não explorada devido a optar por uma fronteira mais próxima. O terceiro tem como propósito mostrar a funcionalidade do robô ao realizar o *loop closure* para melhorar a estimação da sua posição.

5.2.1 Ensaio 1

Para primeiro ensaio, o ambiente escolhido é relativamente simples, sendo constituído por um quarto e um corredor - Figura 5.9. Nesta figura o robô é representado pela circunferência e a sua orientação pelo traço no interior da mesma.

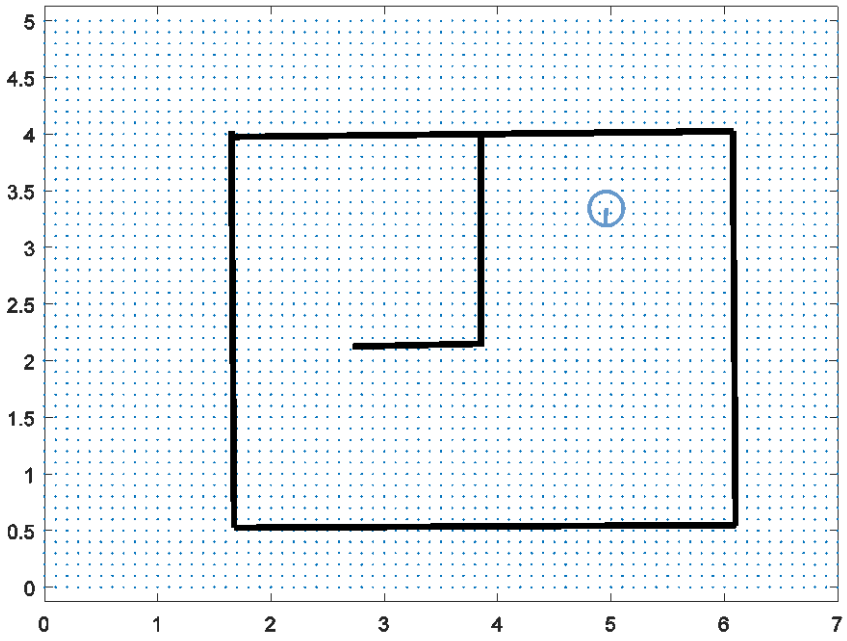


Figura 5.9 – Condições iniciais do ensaio 1.

No começo da simulação, o robô ativa o sensor *rangefinder* e obtém um mapa local. Com base nessa informação verifica se existe uma fronteira para explorar – Figura 5.10.

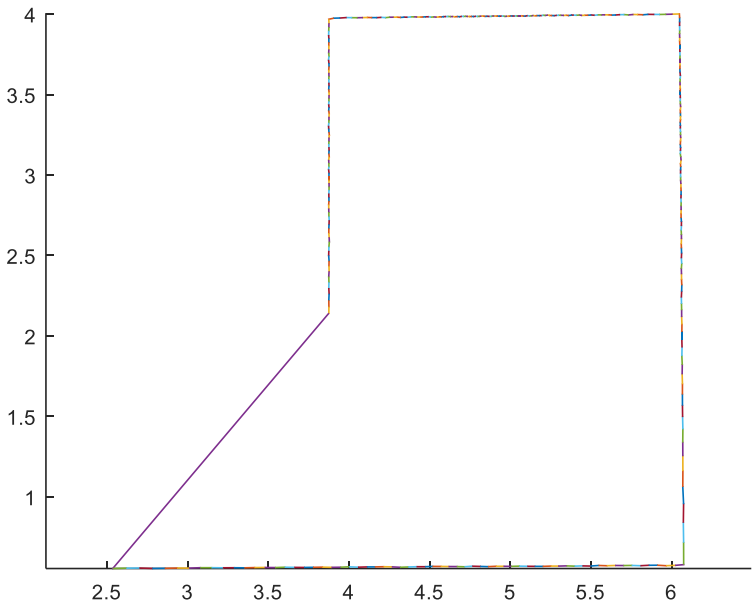


Figura 5.10 – Mapa de fronteiras (ensaio 1).

Através da Figura 5.10, verifica-se que a fronteira localiza-se no segmento de reta diagonal a roxo. Sendo esse o ponto para qual o robô se irá deslocar – Figura 5.11.

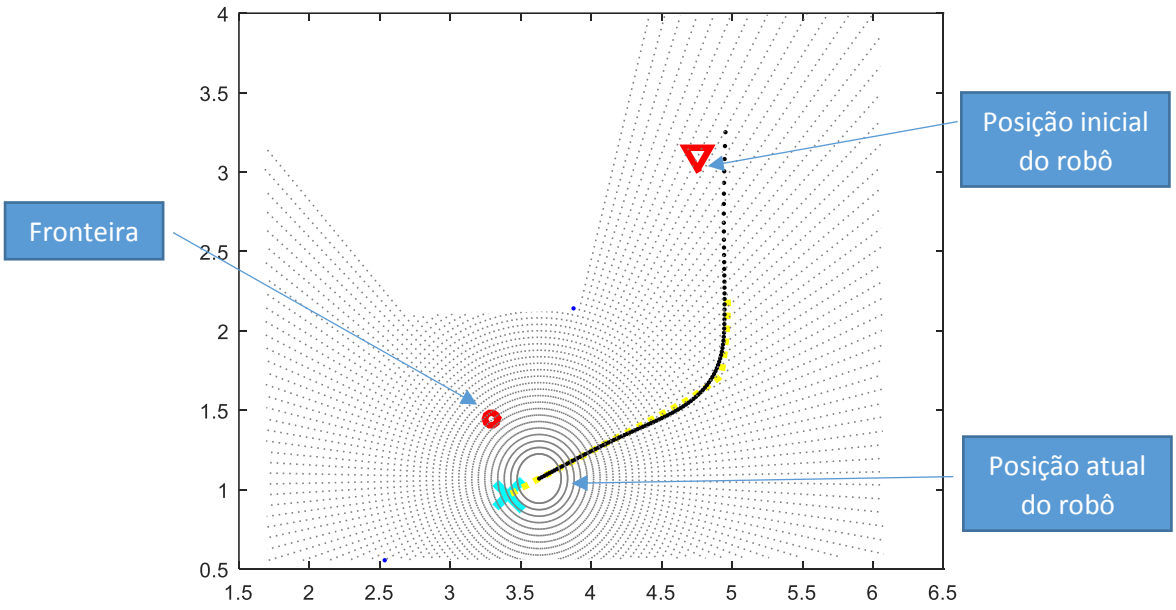


Figura 5.11 – Deslocamento do robô e a respetiva leitura obtida pela *rangefinder*.

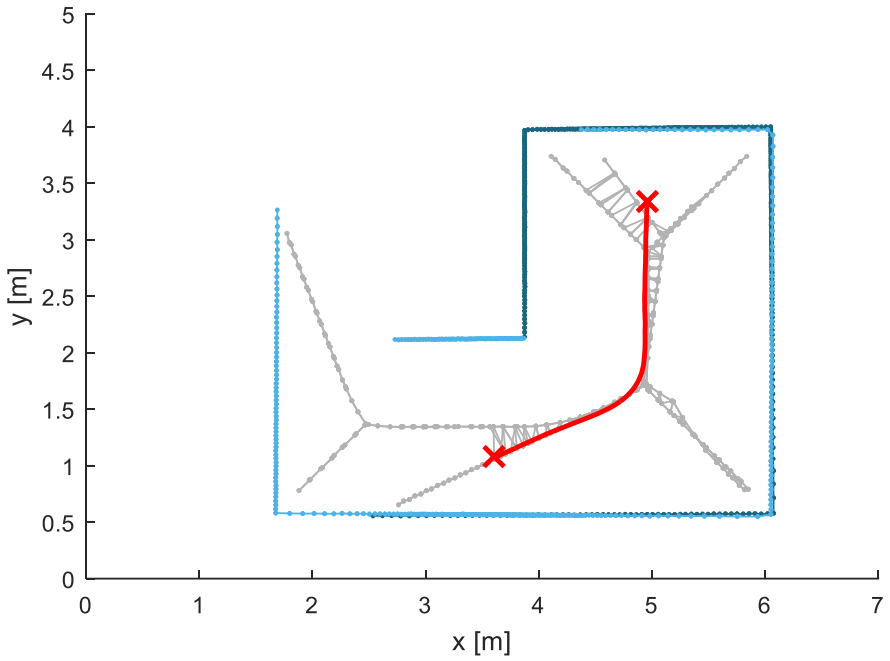


Figura 5.12 – Mapa híbrido obtido (parcial).

Através da Figura 5.12 observa-se o mapa métrico baseado nos pontos obtidos pelo *rangefinder* (a azul) e o mapa topológico gerado através do GVD. (a cinzento). É possível ainda observar a linha a vermelho que corresponde à trajetória percorrida pelo robô. Chegando ao ponto de destino o robô realiza uma nova leitura no *rangefinder*, obtendo novas informações- Figura 5.13:

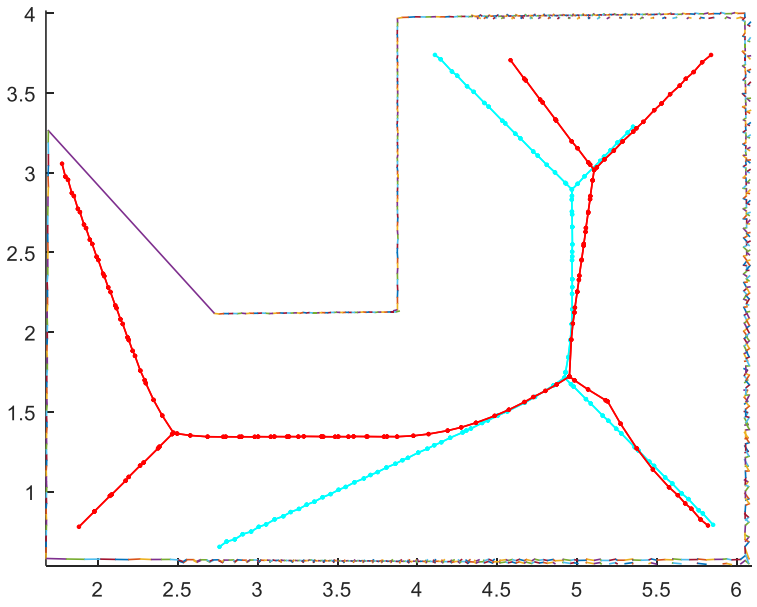


Figura 5.13 – Junção do novo mapa de fronteiras com o anterior e respetivos GVD locais.

Através da Figura 5.13 verifica-se a existência de uma nova fronteira, sendo esse o novo ponto de destino do robô – Figura 5.14.

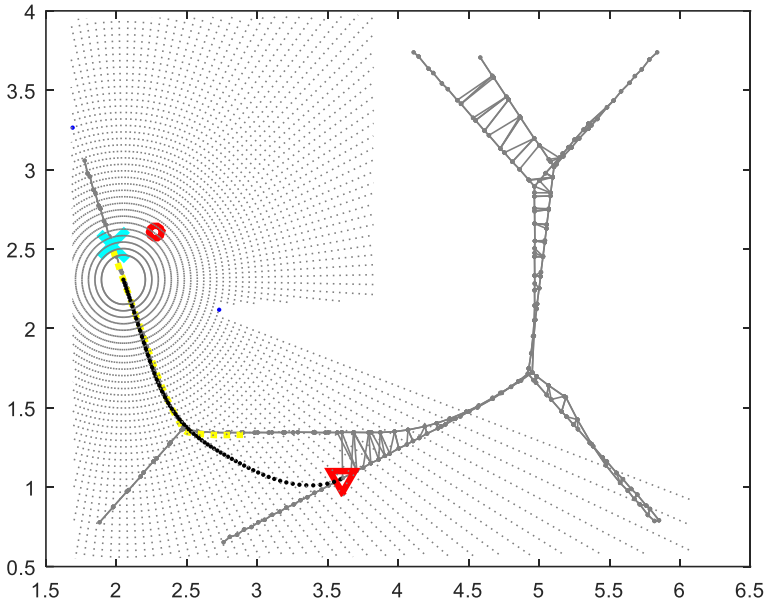


Figura 5.14 – Deslocamento do robô e fusão dos GVD locais obtidos anteriormente (ensaio 1).

Chegando ao ponto de destino, o robô realiza uma nova leitura - Figura 5.15.

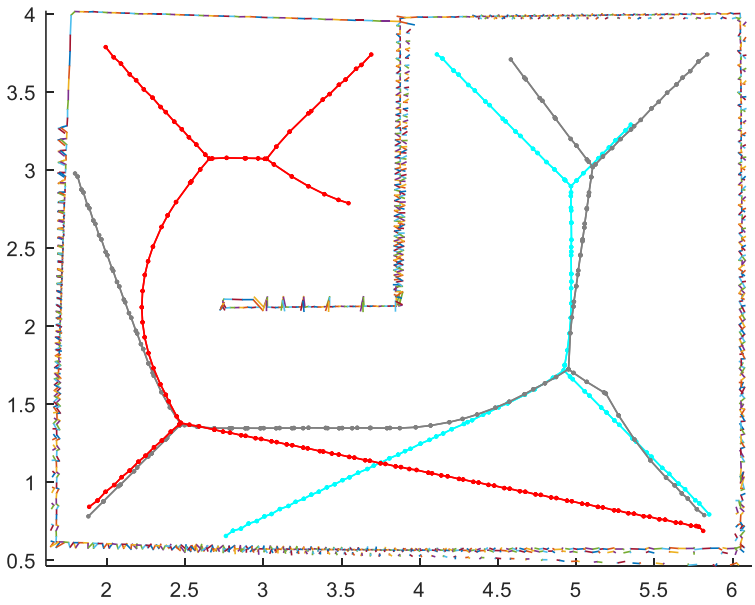


Figura 5.15 – Junção final dos mapas de fronteiras e GVD locais não fusionados (ensaio 1).

Na Figura 5.15 observam-se vários GVD, sendo que o vermelho corresponde ao robô na posição final, o cinzento à posição intermédia e o azul-claro à posição inicial. Na Figura 5.16 é possível observar a fusão dos vários GVD descritos anteriormente.

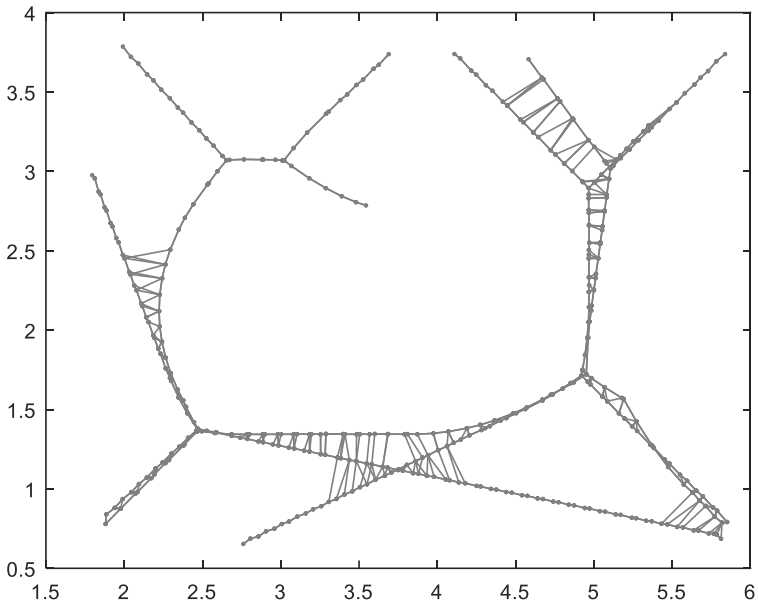


Figura 5.16 – GVD final, resultado das várias fusões entre GVDs locais (ensaio 1).

Por fim, não havendo mais fronteiras para explorar, o algoritmo é terminado, assim como a simulação. Obtendo um mapa híbrido final:

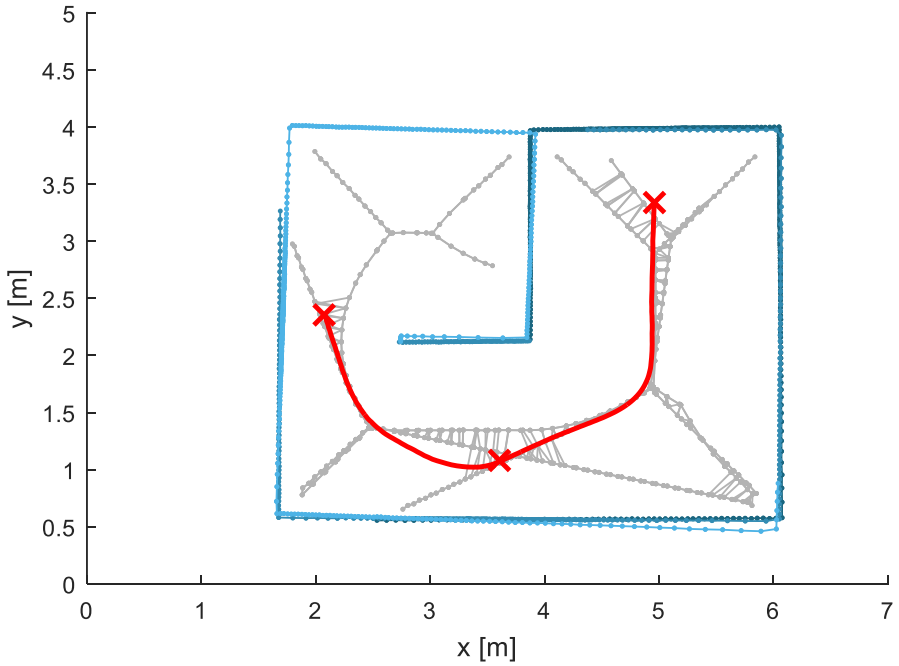


Figura 5.17 – Mapa híbrido final (ensaio 1).

Na Figura 5.17 é possível observar o mapa híbrido final e a trajetória percorrida pelo robô com as suas posições de paragem. Com base neste exemplo verifica-se que o simulador apresenta um algoritmo de exploração e navegação válido e que o mapa final obtido assemelha-se bastante ao criado pelo utilizador inicialmente.

5.2.2 Ensaio 2

O segundo ensaio consiste num ambiente mais complexo que o anterior, sendo mais próximo do modelo utilizado na competição. É constituído por três quartos e um corredor que os liga - Figura 5.18.

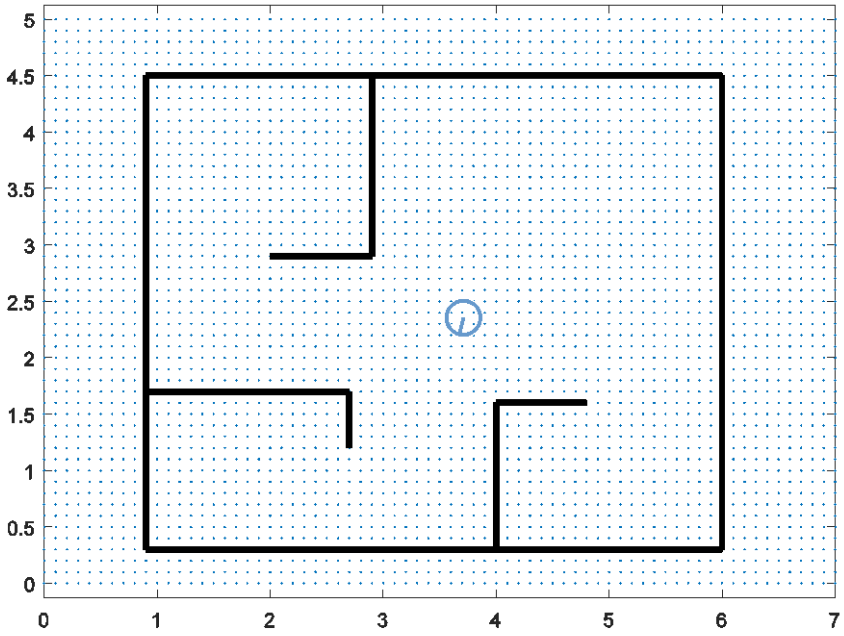


Figura 5.18 – Condições iniciais do ensaio 2.

Como este ensaio consiste num ambiente mais complexo, a sua completa exploração contém muitos *outputs*. Por essa razão serão apenas apresentados os *outputs* iniciais e finais.

Tal como no ensaio anterior, o robô inicia com uma leitura do *rangefinder* de modo a procurar fronteira(s) para se deslocar – Figura 5.19.

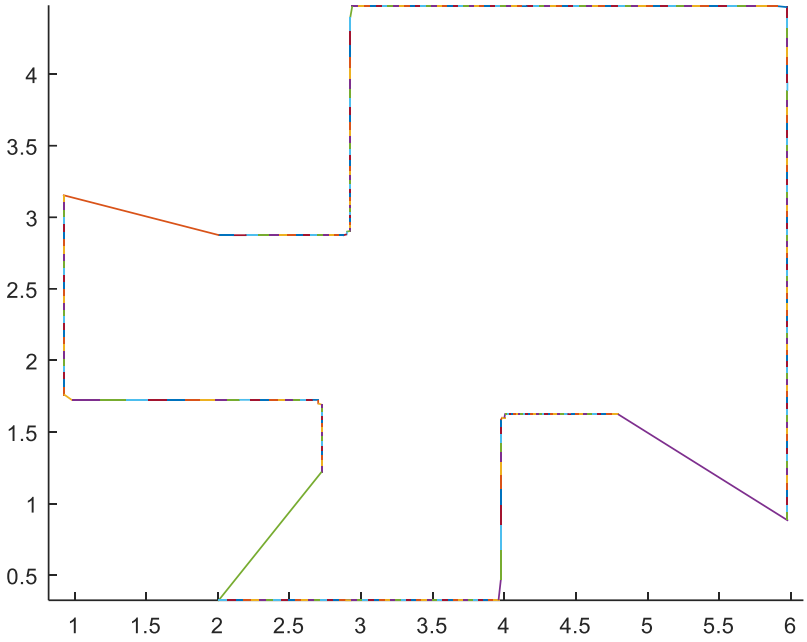


Figura 5.19 – Mapa de fronteiras (ensaio 2).

Através da Figura 5.19 verifica-se que existem três fronteiras, optando-se pela que está mais próxima do robô - Figura 5.20.

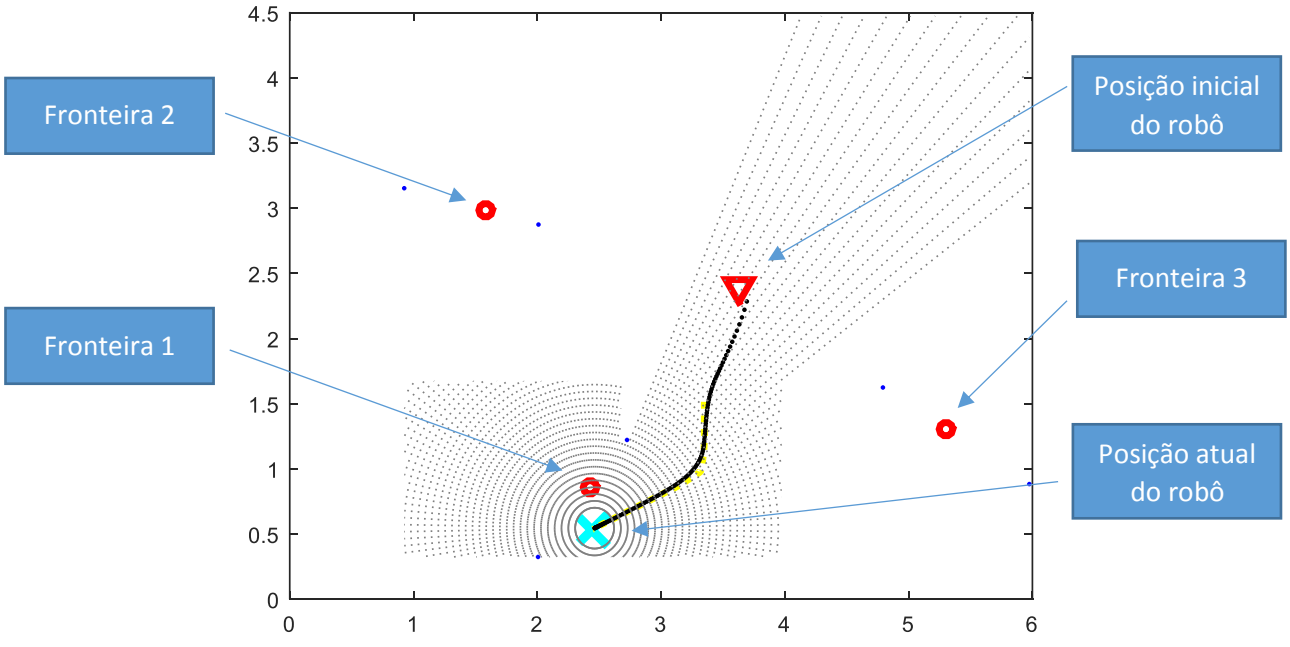


Figura 5.20 - Deslocamento do robô e a respetiva leitura obtida pela *rangefinder* (ensaio 2).

Terminado o algoritmo obteve-se o seguinte mapa:

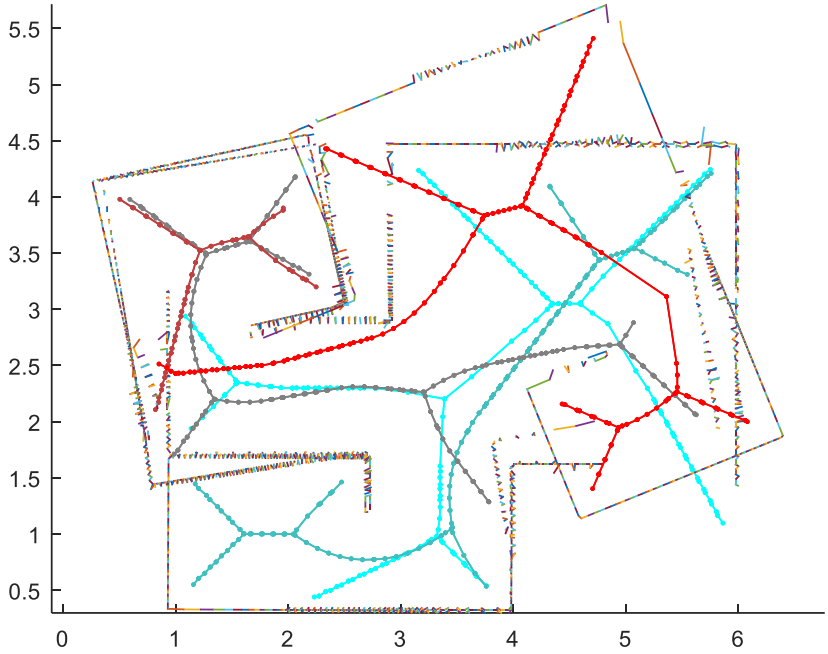


Figura 5.21 – Junção final dos mapas de fronteiras e GVD locais não fusionados (ensaio 2).

Através da Figura 5.21 é possível observar os vários GVD locais (não fusionados) e o mapa métrico que aparentemente não estão corretos (não correspondem ao mapa real). Tal deve-se ao facto de o mapeamento local ser atribuído no referencial mundo na posição estimada do robô através de odometria. Como se sabe, a estimativa da odometria tem erros associados, que são tanto maiores quanto maior a distância percorrida e as variações de direção realizadas. Por essa razão os mapas locais estão, neste ensaio, mais desviados do *groundtruth* métrico, algo que era menos notório no ensaio anterior, já que a distância percorrida era menor. Note-se, no entanto, que apesar da discrepância entre a informação métrica estimada e o *groundtruth*, o mapa topológico mantém a informação relevante que permite navegar no mapa corretamente.

Na Figura 5.22 observa-se o GVD final que corresponde à fusão dos vários GVD locais.

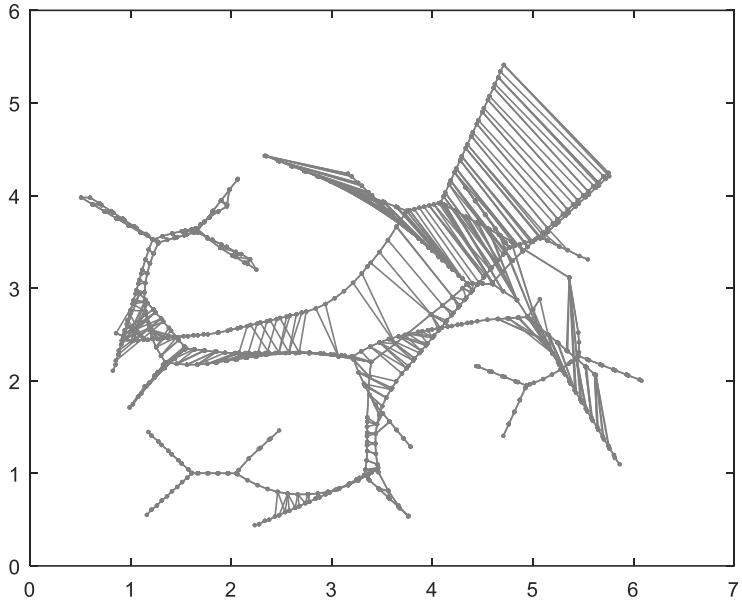


Figura 5.22 – GVD final, resultado das várias fusões entre GVDs locais (ensaio 2).

Na Figura 5.23 observa-se o mapa híbrido final (GVD final e mapa de fronteiras) e a respetiva trajetória percorrida (segundo odometria). Com base na Figura é possível observar que o primeiro e segundo *scans* são bastantes coerentes com o mapa real, sendo que a discrepância começa a ocorrer à chegada à terceira posição.

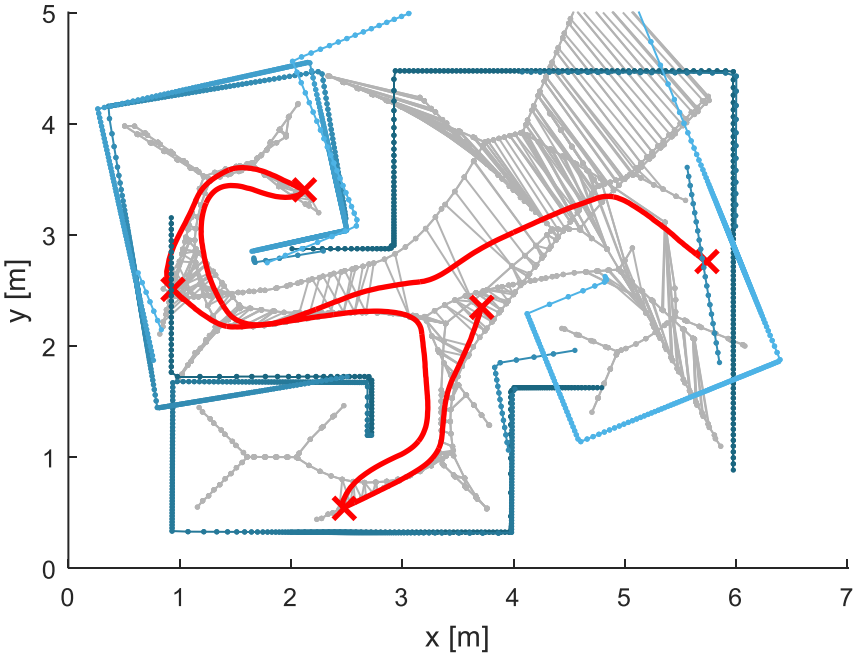


Figura 5.23 – Mapa híbrido final (ensaio 2).

Exemplo disso é a posição que o robô ocupa no referencial do mundo na terceira posição em que está parado, pois segundo a odometria, é coincidente com uma parede anteriormente detetada. O que vai em conta com o que foi referido anteriormente, ao aumentar a distância percorrida e as variações na direção o erro da estimação de posição por odometria aumenta.

O interessante a retirar da Figura 5.23 é que apesar do erro de odometria ser significativo, o robô consegue explorar todos os quartos do cenário. Pois é possível observar que todos os quartos (as suas formas) estão na Figura 5.23. Tal só é possível com a presença do robô no quarto e respetiva leitura. Pode-se concluir também que apesar dos mapas (métricos) não serem coerentes com o modelo real, o robô tem acesso a todos os espaços através do mapa topológico criado. É relevante notar também o facto de o robô, após ter concluído que chegou a um beco (neste caso um quarto), dirige-se para a fronteira mais próxima que foi detetado logo na primeira iteração.

5.2.3 Ensaio 3

O terceiro ensaio consiste num ambiente simples que é constituído por um único corredor quadrangular - Figura 5.24 – e é demonstrativo da capacidade de deteção de revisitação (*loop closure*) e da consequente correção do mapa através do método Pose-SLAM.

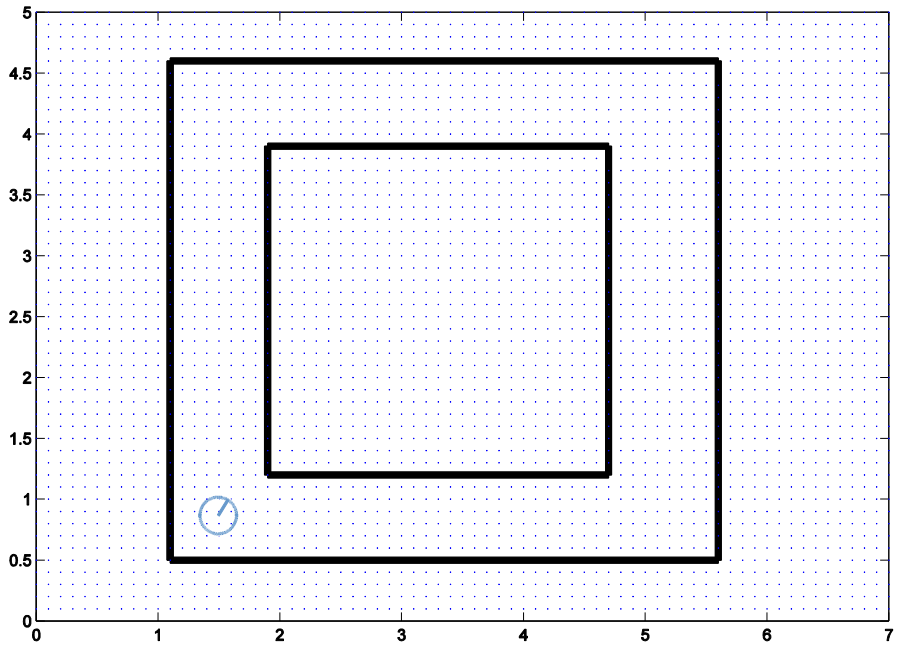


Figura 5.24 - Condições iniciais do ensaio 3.

Após a conclusão do algoritmo de exploração obteve-se os seguintes resultados:

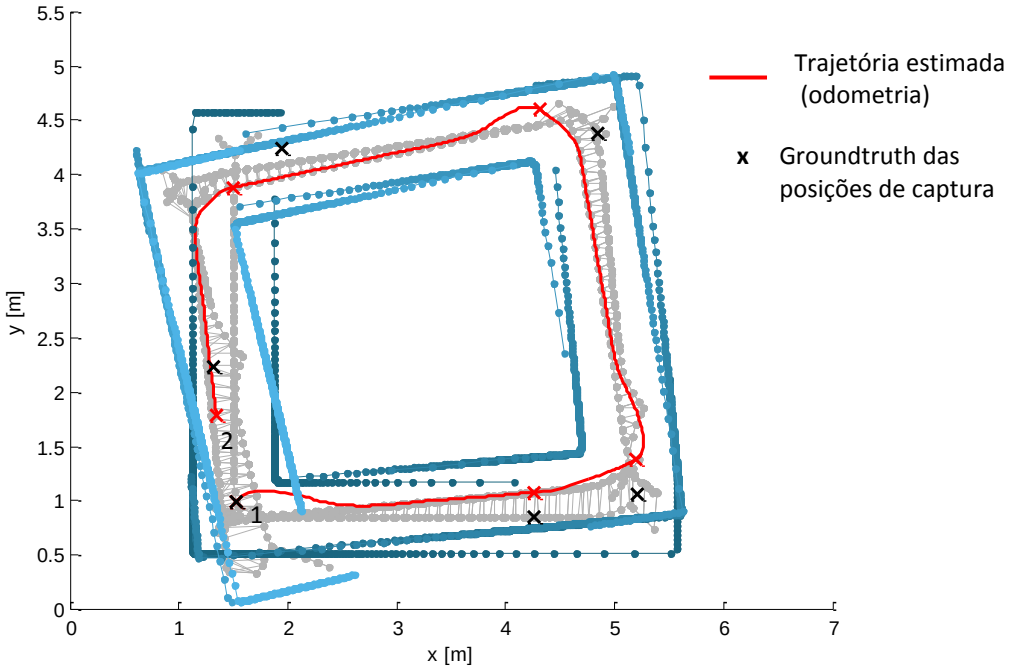


Figura 5.25 – Mapa final sem a correção por Pose-SLAM.

Na Figura 5.25 é possível observar a exploração total do ambiente por parte do robô. Também é possível notar o aumento do desfasamento entre a posição de captura estimada por odometria e a posição real de captura (*groundtruth*) com o aumento da distância percorrida. Verifica-se também um desfasamento nos mapas métricos, pela razão explicada anteriormente.

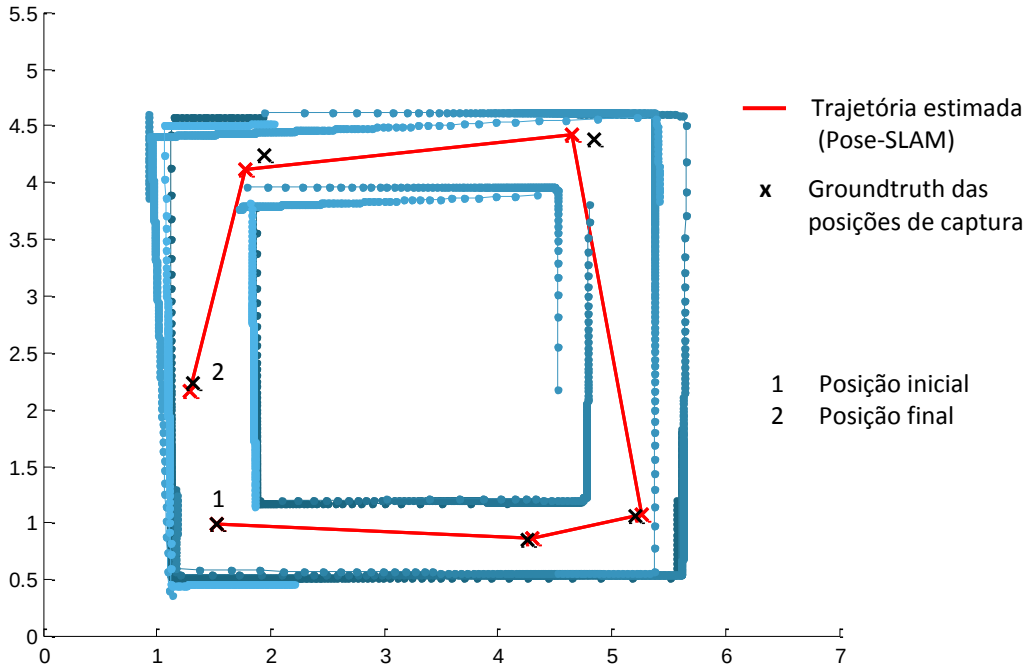


Figura 5.26 - Mapa final com a correção por Pose-SLAM.

Na Figura 5.26 apresenta-se a correção das posições de captura e dos *scans* associados, após a aplicação do Pose-SLAM. Para este efeito, utilizou-se a informação adicional sobre a revisitação da posição 1 pela posição 2. Este constrangimento levou a que a correção à trajetória apresente melhor aproximação ao *groundtruth* na primeira, e última posição de captura. Como consequência da correção da posição do robô através do *loop closure*, os *scans* associados a cada posição de captura também são corrigidos. Contudo, nas posições intermédias entre a inicial e a final essa correção é menos precisa já que não há informação adicional acerca destas – o *loop closure* acrescenta informação apenas em relação à primeira e última posição.

6 CONCLUSÕES E TRABALHO FUTURO

A presente dissertação demonstra o desenvolvimento de um robô autónomo com a funcionalidade de combate a incêndios num ambiente de simulação. Apresentou-se um sistema de deteção de chama, que permite a captura de imagem termográfica e, após o seu tratamento, isolar a chama. Após a realização dos ensaios conclui-se que o sistema deteta a chama com facilidade até a 2 m de distância. Aliado a este sistema, apresentou-se um sistema de extinção com uma ventoinha, sendo esta acionada pela unidade de processamento com o auxílio de um transístor NPN. Aglomerando este dois sistemas e o sistema de locomoção, realizaram-se ensaios de extinção de chamas. Com base nos ensaios, verificou-se que o sistema previamente descrito tem uma taxa de sucesso de 100%. Verificou-se ainda uma probabilidade de 77.5% do robô parar para extinguir a chama no raio de segurança definido pelo autor. Estes resultados permitiram concluir que a implementação dos dois sistemas é eficaz e válida.

A criação do simulador, permitiu realizar vários em ensaios em diversas situações, (diferentes localizações iniciais do robô e formas do labirinto). O simulador é bastante realista pois tem em conta o ruído associado aos diversos sensores. O sistema de navegação foi desenvolvido com base no Pose-SLAM. A sua exploração é baseada em fronteiras, que consiste em movimentar o robô em direção às fronteiras não exploradas. O planeamento da sua trajetória é realizado através do GVD local que é criado a cada iteração do robô, criando assim mapas topológicos das variadas zonas. O alinhamento dos mapas (fronteiras e topológico) após uma iteração do robô foi conseguido através do algoritmo ICP, permitindo assim a fusão da informação de iterações sucessivas. Uma outra caraterística do simulador consiste em que o robô seja capaz de voltar a uma fronteira não explorada anteriormente identificada, independentemente da sua distância. No simulador verificou-se também a sua funcionalidade referente ao controlo de posição através do *loop closure*.Com base nos vários ensaios realizados foi possível concluir que modelo proposto é válido.

No futuro propõe-se a criação de um sistema de extinção de chama mais realista, recorrendo por exemplo, a uma bomba ou a um aspersor. Outra proposta para o futuro consiste em realizar um estudo estrutural do *chassis* robô, com o objetivo de verificar se a alteração para outros materiais (por exemplo impressão 3D) é viável.

Deverá também iniciar-se a implementação do sistema de navegação no RPI, com o objetivo de realizar a fusão dos vários sistemas (navegação e deteção e extinção de chama), ensaiar com o protótipo experimental no cenário real. A prossecução destas linhas de trabalho dará continuidade ao projeto de criar um robô para participar na competição nacional “Robô Bombeiro”.

Referências

Arkin, R. C. (1998). *Behavior-Based Robotics.* London: The MIT Press.

Baillie, J.-C. (2004). *Loop Closure Detection.* Retrieved Outubro 2016, from École Nationale Supérieure de Techniques Avancées (ENSTA-ParisTech): http://cogrob.ensta-paristech.fr/loopclosure.html

Bekey, G. A. (2005). *Autonomous Robots: From Biological Inspiration to Implementation and Control.* Cambridge: The MIT Press.

Bernardino, R. (2016). *Robô Móvel para Máquinas-ferramentas* (Tese de Mestrado). Instituto Superior de Engenharia de Lisboa, Portugal

Campos, F., Correia, L., & Calado, J. (2013). Loop Closure Detection with a Holistic Image Feature. *Portuguese Conference on Artificial Intelligence* (pp. 247-258). Springer Berlin Heidelberg.

Carreira, F., Calado, J., & Cardeira, C. (2011). A Mobile Robot Navigation Planning in a Human Populated Environment. *Proceedings of Robotica 2011, the 11th International Conference on Autonomous Robot Systems and Competitions*, (pp. 15-20). Lisboa, Portugal. Retrieved from http://www.dem.ist.utl.pt/~cardeira/papers/Robotica2011Proceedings_Fcarreira.pdf

Carreira, F., Calado, J., Cardeira, C., & Oliveira, P. (2015). Enhanced PCA-Based Localization Using Depth Maps. *Intelligent & Robotic Systems*, 341-360.

Carreira, F., Canas, T., Silva, A., & Cardeira, C. (2006). i-Merc: A Mobile robot to Deliver Meals inside Health Services. *Proceedings of RAM 2006, the IEEE Conference on Robotics, Automation and Mechatronics*, (pp. 1-8).

Carreira, F., Christo, C., Valério, D., Ramalho, M., Cardeira, C., Calado, J., & Oliveira, P. (2012). 2D PCA-based Localization for Mobile Robots in Unstructured Environments. *Proceedings of IROS 2012, the IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3767-3868). Vila-moura: IEEE.

Choset, H., & Nagatani, K. (2001). Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization Without Explicit Localization. *IEEE Transactions on Robotics and Automation* (pp. 125 - 137). IEEE.

Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., & Thrun, S. (2005). *Principles of Robot Motion.* London, England: The MIT Press.

Corke, P. (2011). *Robotics, Vision and Control: Fundamental Algorithms in MATLAB.* Berlin: Springer.

Correll, N. (2016). *Introduction to Autonomous Robots.* Boulder, Colorado.

Davoult, B., & Lanne, B. (n.d.). *First INtelligent Extinguisher (FINE) Robot*. Retrieved Abril 2016, from Tuvie: http://www.tuvie.com/first-intelligent-extinguisher-fine-robot/

Dijkstra, E. (1959). A note on two problems in connexion. *Numerische mathematik*, (pp. 269-271).

Dobrow, A. (n.d.). *OLE, the Fire Fighting Beetle*. Retrieved Abril 2016, from Gearfuse: http://www.gearfuse.com/ole-the-fire-fighting-beetle/

Dumiak, M. (2008, Março 31). *The Firefighting Robot*. Retrieved Abril 2016, from Popular Science: http://www.popsci.com/scitech/article/2008-03/firefighting-robot?image=0

Fauske, K. M., Gustafsson, F., & Hegrenæs, Ø. (2007). Estimation of AUV dynamics for sensor fusion. *10th International Conference on Information Fusion* (pp. 1 - 6). Quebec: IEEE.

Figueiredo, J. (2015). *Navegação de robô móvel em ambiente com humanos* (Tese de Mestrado). Instituto Superior de Engenharia de Lisboa, Portugal

FLIR. (2014). FLIR LEPTON Long Wave Infrared (LWIR) Datasheet.

Goris, K. (2005) *Autonomous Mobile Robot Mechanical Design* (Tese de Mestrado) Vrije Universiteit Brussel, Belgium

Grisetti, G., Kummerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, (pp. 31-43).

Hawkins, M. (2012, Junho 9). *Simple Guide to the RPi GPIO Header and Pins*. Retrieved Maio 2016, from Raspberry Pi Spy: http://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/#prettyPhoto

Haynes, H., & Molis, J. (2015). *US Firefighter Injuries-2014.* National Fire Protection Association.

Herndon, J., & Haley, D. (2001). Pioneer Robot Testing Program and Status.

Hiremath, S., van der Heijden, G., van Evert, F., Stein, A., & ter Braak, C. (2014). Laser range finder model for autonomous navigation of a robot in a maize field using a particle filter. *Computers and Electronics in Agriculture*, (pp. 41-50).

Ho, K., & Newman, P. (2007). Detecting Loop Closure with Scene Sequences. *International Journal of Computer Vision, 74*(3), 261–286.

Junior, V. (2006). *Arquitetura Hibrida para Robôs Móveis Baseada em Funções de Navegação com Interação Humana* (Tese de Doutoramento). Escola Politécnica da Universidade de São Paulo, Brasil

Kanniah, J., Ercan, M., & Calderon, C. (2014). *Practical Robot Design Game Playing Robots.* CRC Press.

Kelly, A. (2013). *Mobile Robotics, Mathematics, Models, and Methods.* New York, USA: Cambridge University Press.

Kim, Y.-D., Kim, Y.-G., Lee, S.-H., Kang, J.-H., & An, J. (2009). Portable Fire Evacuation Guide Robot System. *Intelligent Robots and Systems*, (pp. 2789-2794). St. Louis, USA.

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning series).*

Košnar, K. (2011) *Robot Mapping for large environments* (Tese de Doutoramento) Czech Technical University. Czech Republic

Lu, F., & Milios, E. (1994). Robot Pose Estimation in Unknown Environments byMatching 2D Range Scans. *Computer Vision and Pattern Recognition.*

LUF Fire - Fighter. (n.d.). *LUF 60™ – Operation Area*. Retrieved Abril 2016, from LUF Fire - Fighter: http://www.luf60.at/ff-luf60-einsatzgebiet/?L=1

Lutz, M., & Ascher, D. (2004). *Learning Python* (2nd ed.). Sebastopol, California, USA: O'Reilly & Associates, Inc.

Marques, C., Cristóvão, J., Lima, P., Frazão, J., Ribeiro, I., & Ventura, R. (2006). RAPOSA: Semi-Autonomous Robot for Rescue Operations. *Intelligent Robots and Systems.* Beijing, China: IEEE.

Moon, Y. (2002). SONY AIBO: THE WORLD'S FIRST ENTERTAINMENT ROBOT. *Interactive Marketing (John Wiley & Sons).*, 73-90.

Moraes, V., & Vieira, C. (2013). *MATLAB – Curso Completo.* FCA - Editora de Informática.

Murphy, R. R. (2000). *Introduction to AI Robotics.* London, England: The MIT Press.

Naval Research Laboratory (NRL). (2014, Março 24). *NRL Autonomy Lab Hosts Shipboard Fire Robotics Consortium*. Retrieved Abril 2016, from Naval Research Laboratory (NRL): http://www.nrl.navy.mil/media/news-releases/2014/nrl-autonomy-lab-hosts-shipboard-fire-robotics-consortium

Nguyen, V., Martinelli, A., Tomatis, N., & Siegwart, R. (2005). A Comparison of Line Extraction Algorithms using 2D Laser Rangefinder for Indoor Mobile Robotics.

Oliveira, J. (2010) *Um sistema integrado para navegação autónoma de robôs móveis* (Tese de Mestrado) Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, Brasil

Pieri, E. R. (2002). *CURSO DE ROBÓTICA MÓVEL.* Florianópolis: Universidade Federal de Santa Catarina.

PORDATA, Base de Dados de Portugal Contemporâneo. (2016, Fevereiro 26). *Incêndios florestais e área ardida – Continente*. Retrieved from PORDATA: http://www.pordata.pt/Portugal/Inc%c3%aandios+florestais+e+%c3%a1rea+ardida+%e2%80%93+Continente-1192

Raspberry Pi Foundation. (n.d.). *Raspberry Pi*. Retrieved Maio 2016, from Raspberry Pi: https://www.raspberrypi.org/

Raspbian. (n.d.). *Welcome to Raspbian*. Retrieved Maio 2016, from Raspbian: https://www.raspbian.org/FrontPage

Rawlinson, D., & Jarvis, R. (2008). Topologically-directed navigation. *Robotica*, 189-203.

Revista Segurança. (2015, Novembro 11). *Incêndios em edifícios na cidade de Lisboa*. Retrieved Julho 2016, from revista segurança: http://www.revistaseguranca.com/incendios-em-edificios-na-cidade-de-lisboa/

Ribeiro, M. I. (1999, Outubro). Localização em Robótica Móvel - Odometria. Lisboa, Portugal. Retrieved Agosto 2016, from http://users.isr.ist.utl.pt/~mir/cadeiras/robmovel/Odometry.pdf

RoboPeak. (2014). RPLIDAR: Introduction and Datasheet.

Robot Electronics. (2016). *RD02 - 12v robot drive*. Retrieved Julho 2016, from http://www.robot-electronics.co.uk/products/drive-systems/drive-kits/rd02-12v-robot-drive.html

Rodrigues, D. P. (2010). *Técnicas de Navegação.* Campinas: Universidade Estadual de Campinas.

Sankari, J., & Imtiaz, R. (2016). Automated Guided Vehicle (AGV) for Industrial sector. *Proceedings of the 10th International Conference on Intelligent Systems and Control, ISCO 2016.*

Secchi, H. (2012, Abril). Uma Introdução aos Robôs Móveis. Retrieved Junho 2016, from http://www.obr.org.br/wp-content/uploads/2013/04/Uma_Introducao_aos_Robos_Moveis.pdf

Segal, A., Hähnel, D., & Thrun, S. (2009). Generalized-ICP. *Robotics: Science and Systems V.* Seattle, USA.

Siegwart, R., Nourbakhsh, I., & Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots.* London: The MIT Press.

Tan, C., Liew, S., Alkahari, M., Ranjit, S., Said, M., Chen, W., . . . Sivarao. (2013). Fire Fighting Mobile Robot: State of the Art and Recent Development. *Basic and Applied Sciences, 7(10)*, 220-230.

Tardós, J. D. (2002, August). Data Association in SLAM. Stockholm, Sweden. Retrieved Setembro 2016, from http://www.cas.kth.se/SLAM/Presentations/mingo-slides.pdf

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence, 99*, 21-71.

Thrun, S. (2002). Robotic Mapping: A Survey. *School of Computer Science Carnegie Mellon University.*

Thulasiraman, K., Arumugam, S., Brandstädt, A., & Nishizeki, T. (2016). *Handbook of Graph Theory, Combinatorial Optimization, and Algorithms.* Boca Raton: CRC Press.

Upton, E., & Halfacree, G. (2012). *Raspberry Pi® User Guide.* West Sussex, United Kingdom: John Wiley & Sons Ltd.

Yamauchi, B. (1997). A Frontier-Based Approach for Autonomous Exploration. *International Symposium on Computational Intelligence in Robotics and Automation* (pp. 146-151). Monterey, California, USA: IEEE.

Anexos

Anexo 1 – Desenhos Técnicos

Anexo 2 – Tabelas de valores de distância (Sistema de Extinção de Chama)

Valores obtidos do Cenário 1:

Ensaio	d [cm]	Ensaio	d [cm]	Ensaio	d [cm]	Ensaio	d [cm]
1	11.5	6	8.5	11	10.5	16	11
2	12.5	7	11.5	12	13.5	17	12.5
3	11.5	8	10	13	9	18	12
4	9	9	12.5	14	13.5	19	12.5
5	10	10	13	15	13.5	20	12

d_{min}	8.5	$h_{inicial}$	17.5cm	$d_{méd}$
d_{max}	13.5	h_{final}	17cm	11.5

Tabela A2.1 - Valores obtidos do Cenário 1.

Valores obtidos do Cenário 2:

Ensaio	d [cm]	Ensaio	d [cm]	Ensaio	d [cm]	Ensaio	d [cm]
1	10	6	10.5	11	18	16	19
2	13	7	13	12	15	17	20
3	9.5	8	16.5	13	18	18	21
4	10	9	14	14	20	19	23
5	10.5	10	15	15	20	20	25

d_{min}	9.5	$h_{inicial}$	17cm	$d_{méd}$
d_{max}	25	h_{final}	15.5cm	16.05

Tabela A2.2 - Valores obtidos do Cenário 2.

Valores obtidos do Cenário 3:

Ensaio	d [cm]	Ensaio	d [cm]	Ensaio	d [cm]	Ensaio	d [cm]
1	20	6	16	11	13	16	14
2	18	7	23	12	13.5	17	12.5
3	15	8	17.5	13	22	18	14.5
4	16	9	13.5	14	14	19	20
5	16.5	10	13.5	15	12.5	20	23

d_{min}	12.5	$h_{inicial}$	17.5cm	$d_{méd}$
d_{max}	23	h_{final}	16cm	16.4

Tabela A2.3 - Valores obtidos do Cenário 3.

Valores obtidos do Cenário 4:

Ensaio	d [cm]	Ensaio	d [cm]	Ensaio	d [cm]	Ensaio	d [cm]
1	15.5	6	12.5	11	19	16	19
2	15.5	7	17.5	12	20.5	17	19
3	15.5	8	23	13	16.5	18	17.5
4	18	9	17	14	20.5	19	16.5
5	12	10	17	15	19	20	17.5

d_{min}	12	$h_{inicial}$	17.5cm	$d_{méd}$
d_{max}	23	h_{final}	14.5cm	17.425

Tabela A2.4 - Valores obtidos do Cenário 4.