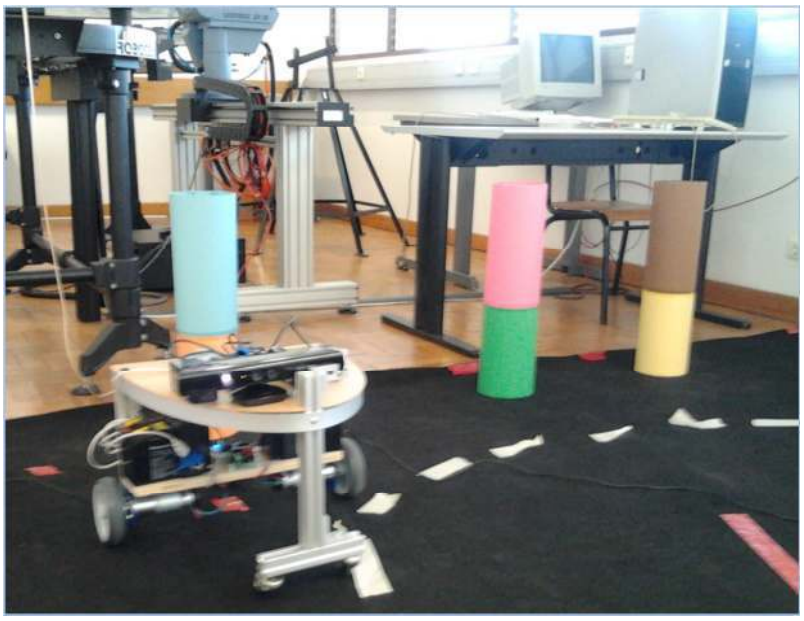
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Mecânica



Navegação de robô móvel em ambiente com humanos

JOÃO CARLOS MARTINHO FIGUEIREDO
Mestre em Engenharia Civil

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

Orientador: Dr. João Manuel Ferreira Calado
Co-orientador: Mestre Fernando Paulo Neves da Fonseca Cardoso Carreira
Co-orientador: Dr. Francisco Mateus Marnoto de Oliveira Campos

Júri:
Presidente: Dr. Joaquim Infante Barbosa
Vogal: Dr. Carlos Baptista Cardeira
Vogal: Dr. Francisco Mateus Marnoto de Oliveira Campos

Dezembro, 2015

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA
Área Departamental de Engenharia Mecânica
ISEL

Navegação de robô móvel em ambiente com humanos

JOÃO CARLOS MARTINHO FIGUEIREDO
Aluno n.º 21954

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

Orientador: Dr. João Manuel Ferreira Calado
Co-orientador: Mestre Fernando Paulo Neves da Fonseca Cardoso Carreira
Co-orientador: Dr. Francisco Mateus Marnoto de Oliveira Campos

Júri:
Presidente: Dr. Joaquim Infante Barbosa
Vogal: Dr. Carlos Baptista Cardeira
Vogal: Dr. Francisco Mateus Marnoto de Oliveira Campos

Dezembro, 2015

Dedicatória

…À minha Esposa, pela força que me deste neste regresso à vida Universitária.

Agradecimentos

Ao orientador Professor Doutor João Calado, por ter aprovado este desafio e especialmente por ter aceitado a minha candidatura para o Mestrado de Engenharia Mecânica.

Aos Co-orientadores Professores Fernando Carreira e Francisco Campos, pelo apoio, suporte e disponibilidade que demonstraram desde o primeiro dia, e sem os quais a presente dissertação não seria possível.

Resumo

A robótica é um tema que desde a sua introdução tem capturado o interesse e a imaginação das pessoas. A sua aplicação menos fantasiosa e mais realista pode ser vista na indústria. No entanto, um dos maiores desafios encontra-se em criar um robô com capacidade de navegação autónoma, incluindo o reconhecimento e desvio de pessoas, em ambientes estruturados. A robótica móvel baseada no controlo por visão tem-se desenvolvido nos últimos 30 anos. Ao longo deste período, diversos modelos matemáticos foram apresentados com o objectivo de resolver os múltiplos desafios que o controlo de uma unidade robótica autónoma apresenta. A visão robótica, tal como no ser Humano, permite à unidade uma compreensão do mundo onde se insere. No caso de o ambiente ser partilhado com pessoas, a necessidade de interpretar e separar uma pessoa de um objecto é fundamental, para a segurança física e psíquica do ser Humano. Esta dissertação propõe o estudo e desenvolvimento de um robô móvel autónomo apto para navegar em espaços com pessoas. Os trabalhos realizados focaram-se na localização no espaço, reconhecimento de pessoas e planeamento e controlo de trajectórias.

Com este conjunto de soluções pretende-se contribuir com um sistema autónomo robusto capaz de interagir em segurança em ambientes de trabalho partilhados com pessoas, com elevado grau de confiança de modo a que seja possível implantar num ambiente industrial.

Palavras-chave

Robótica
Visão e controlo
Navegação
Mapas de estrada probabilísticos
Campos de Potência Artificiais
Transformação de características de imagem com escala invariável (SIFT)

Pesquisa acelerada e robusta de características de imagem (SURF)

Histograma de Gradientes Orientados (HOG)

Robótica probabilística

Filtro de Kalman

Filtro de Partículas

Localização de Monte Carlo

Kinect

Imagem RGB-D

Abstract

Robotics is a subject known to capture the imagination of people. It can be seen in a realistic fashion in many fields of modern industry. Current robotics challenges can be found in the deployment of autonomous mobile robots when operating in Human shared environments. Vision based mobile robotics is a subject being studied for over 30 years. Robotic Vision, supports the perception of the surrounding world, including the discrimination of objects from people, therefore it is paramount to gurantee the physical and psychological safety of Humans. This work presents a study and development of a mobile robot with the ability to detect people, and to modify the navigation path accordingly.

This thesis aims to provide a robust autonomous robot system that can safely operate in Human shared environments, thus being suitable to a real industrial situation.

Keywords

Robotics
Vision & Control
Navigation
Probabilistic Roadmaps
Scale Invariant Feature Transform (SIFT)
Speeded Up Robust Features (SURF)
Histogram of Oriented Gradients (HOG)
Probabilistic Robotics
Kalman filter
Particle Filters
Artificial Potential Fields
Monte Carlo Localization (MCL)
RGB-D Image

Lista de Símbolos

X,Y,Z	Coordenadas cartesianas dos eixos de referência	
X_i, Y_i	Eixos de inércia globais	
P	Posição cartesiana genérica do robô	
X_R, Y_R	Eixos de referência locais do robô, centrados em P	
ξ	Vector de postura do robô (posição e ângulo)	
R(θ)	Matriz de rotação ortogonal	
v	Velocidade linear do robô	$m.s^{-1}$
ω	Velocidade angular do robô	$rad.s^{-1}$
φ_i	Escorregamento da roda i	
θ	Ângulo de orientação do robô com um eixo de inércia global	rad
ρ	Distância euclidiana entre a posição P do robô e a posição desejada	m
α	Ângulo entre eixo local do robô X_R e um eixo de inércia global X_i	rad
β	Ângulo entre o vector ρ e o eixo de inércia global X_i	rad
t	Tempo	s
l	Eixo medido desde o centro do robô (P) à roda	m
J	Matriz jacobiana	
K	Matriz de Ganho	
e	Erro de postura	
N_f	Número de rodas fixas	
N_s	Número de rodas com direcção (*Steer*)	
$\beta_s(t)$	Ângulo de direcção da roda, função do tempo	rad
C_{1f}	Matriz de graus liberdade para rodas fixas	$[N_f x 3]$
C_{1s}	Matriz de graus liberdade para rodas com direcção	$[N_s x 3]$
q	Ponto coordenadas genéricas	
$F(q)$	Força no ponto q	N.m
$U(q)$	Campo potências na posição q	
$\nabla U(p)$	Gradiente de um vector U na posição q	
$Bel(x)$	Representação do espaço-estado *Belief*	
x_t	Variável de estado no instante tempo t	
o_t	Variável representativa da observação	
a_t	Variável da acção odométrica	
x^i	Amostra que representa um estado hipotético	
m	Número de amostras x^i	
w	Peso representativo da importância da amostra	
v_g	Velocidade linear para alcançar a meta (*goal*)	m
θ_g	Velocidade angular para alcançar a meta (*goal*)	rad
K_i	Ganho de amplificação da odometria	$A.V^{-1}$

Nomenclatura

BBF	*Best-Bin-First*
CVM	*Curvature Velocity approach*
CFK3	*Complementary Kalman Filters3*
DoG	*Difference of Gaussians*
EAPF	*Evolutionary Artificial Potential Field*
EKF	*Extended Kalman filter*
GLOH	*Gradient Location and Orientation Histogram*
HOG	*Histogram of Oriented Gradients*
IMH	*Internal Motion Histograms*
MCL	*Monte Carlo Localization*
MBH	*Motion Boundary Histograms*
MOEA	*Multi Objectives Evolutionary Algorithm*
PCA	*Principal Component Analysis*
PRM	*Probabilistic Roadmap Method*
RRT	*Rapidly exploring Random Tree*
RNA	Rede Neuronal Artificial
RMSE	*Root Mean Square Error*
SIFT	*Scale Invariant Feature Transform*
SURF	*Speeded Up Robust Features*
SVM	*Support Vector Machine*
UKF	*Unscented Kalman Filter*
VRLD	*Visual Region Localization Descriptor*
WMR	*Wheel Mobile Robots*

Índice

Índice de Figuras

Índice de Tabelas

Introdução

A presente dissertação aborda o tema da navegação de um robô móvel em ambientes estruturados e na presença de Humanos. O objectivo é a criação de um robô[1] autónomo capaz de planear uma trajectória inserida num mapa conhecido do mundo e, de seguida, navegar de um ponto de partida até a uma meta, reconhecendo referências visuais para se auto localizar. O processo fica completo com a capacidade de detectar visualmente pessoas e corrigir o caminho, de modo a contornar em segurança as mesmas. Dado que existem múltiplas componentes de controlo necessários para o robô ser efectivamente autónomo, a presente dissertação foi proposta nas seguintes etapas:

1) Elaboração de um sistema de detecção de referências exteriores utilizando a visão (Kinect®), incluindo os ensaios experimentais.
2) Investigação e estudo da localização em ambiente estruturado, com a criação de um modelo que permita ao robô localizar-se no mapa.
3) Investigação e estudo sobre métodos de percepção de pessoas, com a elaboração de um sistema de percepção de pessoas utilizando a visão (Kinect®), incluindo os ensaios experimentais para validar as distâncias de segurança;
4) Navegação do robô móvel no local mapeado garantindo distâncias de segurança física e psíquica das pessoas presentes no espaço;

Cada etapa proposta está assente na área de estudo da Robótica, especificamente localização, visão robótica, robótica probabilística, teoria de controlo, sistemas de navegação e detecção de pessoas.

[1] O termo robô aparece da obra de ficção científica de 1921, "Rossum's Universal Robots" do escritor Checo Karl Capek. Nesta obra, os robôs são introduzidos como pessoas artificiais ou andróides e a palavra "robô" deriva do termo checo "escravo". Em 1950 o escritor Isaac Asimov apresenta inúmeras séries de ficção sobre robôs e explora a problemática da interacção robô – Homem, nomeadamente na questão moral.

Capítulo I

1. Estrutura do documento

Capitulo I

Contém o enquadramento do tema da navegação robótica em ambientes estruturados e na presença de Humanos. Segue-se a estrutura do documento. Inclui o estado da arte dos diversos tópicos abordados na dissertação, como a localização, planeamento de trajectórias, navegação e detecção de pessoas. Conclui-se com a contribuição da dissertação para o presente tema.

Capitulo II

É dedicado ao sistema da localização, fazendo uma análise do detector de referências (*beacons*). Faz-se igualmente uma abordagem ao ruído intrínseco no sensor visual e a sua correcção através de redes neuronais artificiais. Conclui-se com a apresentação do estimador MCL (*Monte-Carlo Localization*), com o qual se estima probabilisticamente a postura (posição e ângulo) do robô num ambiente estruturado. Segue-se a descrição dos algoritmos desenvolvidos, os ensaios experimentais e respectivos resultados.

Capitulo III

Aborda a detecção de Humanos, através da análise de modelos de descritores de características na imagem e da sua junção para a detecção de pessoas. Apresentam-se igualmente os algoritmos desenvolvidos e os ensaios realizados para a detecção de humanos, com os respectivos resultados.

Capitulo IV

Incide sobre a navegação robótica. São apresentados os tipos de navegação, a cinemática de robôs móveis, a teoria de controlo e o planeamento de trajectória, no qual se inclui a teoria dos campos de potências artificiais. A secção de ensaios aborda os algoritmos desenvolvidos para cada um dos temas e inclui os resultados experimentais obtidos. Conclui-se este capítulo com o algoritmo geral da dissertação, onde se combinam todos os temas previamente tratados com vista a atingir os objectivos propostos na dissertação.

1.1. Estado da Arte na localização em ambientes estruturados

A localização do robô em ambientes estruturados é composta por diversos temas, que se cruzam, dependendo do autor. Nesta perspectiva, propõe-se uma revisão temática do estado da arte.

A localização do robô em ambientes estruturados pode ser abordada pelo seu problema mais simples, que é a localização quando a postura inicial é conhecida e se procura apenas compensar o erro incremental da odometria, usando para isso referências exteriores comparadas com um mapa conhecido. Siegwart e Nourbakhsh (2004) fazem uma análise completa a esta problemática, colocando em evidência os desafios da localização, assim como as respectivas soluções propostas. Destacam-se o ruído dos sensores e o seu efeito, o modelo de erro para a estimação odométrica da posição, a incerteza da postura no movimento e as hipóteses para a representação do estado *Belief* (postura estimada pelo robô) num ambiente estruturado.
Esta representação pode ser feita através de estimadores, nomeadamente o EKF (*Extended Kalman Filter*), MCL (*Monte-Carlo Localization*) ou SLAM (*Simultaneous Localization And Navigation*).

S.Y.Chen (2012) apresenta um estudo dos 30 anos de aplicação do filtro de Kalman na visão robótica. O autor incide nos últimos 5 anos, para a robótica visual na localização, vigilância, estimativa, pesquisa, exploração, navegação, manipulação, mapeamento e modelação. O autor destaca igualmente métodos aplicados com o filtro de Kalman, nomeadamente o estudo de L. Teslic (2010) para redução do ruído na estimativa da matriz de co-variância.

Van Nguyen *et al* (2012) fazem recurso de uma rede neuronal com lógica *fuzzy* para executar o ajuste contínuo da convergência do algoritmo EKF. A combinação da rede neuronal com lógica *fuzzy* cria um novo filtro FNN-EKF (*fuzzy neural network based extended Kalman filter*) para resolução dos problemas de localização em espaços fechados (*indoor*) para a robótica móvel em ambientes não estruturados. O artigo faz uso da experiência do filtro FNN-EKF aplicado a um robô de tracção diferencial ao ajustar *online* o processo de cálculo das co-variâncias do ruído, de modo a optimizar (reduzir) o erro de navegação gerado entre o caminho real e o percorrido pelo robô.

S. Li e P. Ni (2010) propõem a utilização do filtro de Kalman com raiz quadrada para introduzir estabilidade no algoritmo de localização e mapeamento simultâneo SLAM (*Simultaneous Localization And Navigation*).

O presente trabalho propõe a utilização do estimador MCL para a localização do robô.

Thrun, Bugard e Fox (2005) apresentam um trabalho de referência sobre a robótica probabilística, no qual está incluído o modelo (MCL) para a localização do robô. O problema da localização global acontece quando o robô não conhece a sua localização inicial, logo a estimativa do erro de posição não pode ser assumida. Esta situação conduz a um segundo problema, designado por *robot kidnapping*, onde a estimativa da posição do robô num determinado instante pode não convergir para a posição real do mesmo. O estimador deve ter a capacidade de convergir para a posição onde o robô foi colocado, detectando efectivamente o "rapto". A estabilidade de um sistema de localização depende da capacidade de recuperação do algoritmo perante estas duas problemáticas. Os autores demostram a capacidade do estimador MCL ultrapassar estes problemas.
Os desafios encontrados na utilização do estimador MCL levam a que os autores apresentem o algoritmo *Mixture*-MCL, baseado no algoritmo MCL, com o objectivo de fazer a integração de duas amostras distintas e complementares para gerar a estimativa (postura) do estado futuro do robô. As vantagens deste algoritmo para a localização robótica são a eficiência computacional, a versatilidade ao abranger sistemas dinâmicos não-lineares, a adaptabilidade na independência das plataformas de computação e a robustez no cálculo com número limitado de amostras.

Theodoros Theodoridis *et al* (2013) apresentam um estudo para localização robótica numa cadeira de rodas, integrando a tecnologia de captura de imagem de profundidade (3D) com o filtro de partículas robusto Monte Carlo. Os autores propõem um localizador de região por visão (*Visual Region Localization Descriptor*) com aquisição tridimensional do mapa recorrendo ao sensor *Kinect* e posterior localização com recurso ao filtro de partículas *Mixture*-MCL. O "descritor" faz a segmentação do mapa de profundidade do sensor numa grelha composta por 36 regiões, onde a profundidade de cada região é associada a uma célula-coluna que permite determinar a distância a ser medida no modelo do filtro de partículas. O trabalho experimental na cadeira de rodas

robótica assenta em 3 casos de localização distintos: a) Modelo com filtro *Mixture*-MCL e sensor odométrico. b) Modelo com *Mixture*-MCL e sensor de sonar. c) Modelo com *Mixture*-MCL e sensor de profundidade *Kinect*. O estudo comparativo demonstra a eficiência e confiança do modelo tridimensional com sensor de profundidade para localização robótica na cadeira de rodas.

A localização do robô proposto na presente dissertação depende de referências exteriores, normalmente pesquisadas por sensores visuais com capacidade de captação de imagens de cor e de profundidade, ou seja RGB-D (*Red, Green, Blue – Depth*). Em alternativa a esta abordagem alguns autores recorrem a padrões naturais do ambiente para localizar o robô (Campos, Correia e Calado, 2015), mas, neste caso, esta capacidade fica dependente da existência de referências suficientemente discriminativas.

Biswas e Veloso (2012) abordam o problema do volume de informação gerado pela robótica baseada em visão RGB-D. Para contornar este problema, propõem o filtro FSPF (*fast sampling plane filtre*) como algoritmo para redução da nuvem de pontos da amostragem tridimensional, separando as amostras por nível de profundidade e classificando-as em planos 3D. Os pontos na proximidade são agrupados nos planos de vizinhança em função de uma margem de erro (*outliner points*). Um segundo filtro faz a projecção dos pontos nos planos para 2D, aplicando-se de seguida o algoritmo de localização. A amostragem total da nuvem de pontos é processada em 2D. O algoritmo FSPF não necessita de informação adicional em formato RGB, baixando o tempo de utilização do CPU em 16% para recolha de imagens a 30 *frames* por segundo.

Wen-Tsai Huang *et al* (2012) demonstram duas técnicas para a localização robótica móvel em ambientes fechados. Na primeira técnica, são utilizadas imagens para captura de marcadores fixos no tecto em posições conhecidas, para calcular o posição e postura do robô, segundo um método global. Na segunda técnica, monta-se uma câmara RGB-D no robô a qual é utilizada para adquirir imagens a cores e profundidade do ambiente, segundo um método local. O movimento relativo do robô é calculado baseado no registo da nuvem de pontos tridimensional e características SURF entre pares de imagens consecutivas. A combinação dos dois métodos e a robustez resultante é demostrada pelos autores na sua obra.

F. Carreira, *et al* (2014) apresentam um trabalho de validação experimental para localização robótica em ambientes fechados com recurso a visão de profundidade, utilizando dados corrompidos. O problema abordado investiga a quantidade de erros acumulados devida à introdução de dados corrompidos na imagem de profundidade capturada. A visão é obtida por um sistema composto por um sensor de profundidade 2D, apontado para o tecto. Neste artigo, os autores abordam a perda de sinal IR (infra vermelho) que não é detectado pelo transdutor de profundidade. Para contornar esta falha de informação, implementam uma extensão do algoritmo PCA (*Principal Component Analysis*). O PCA converte a informação capturada num banco de dados baseado num espaço ortogonal, de elevada compressão, para lidar com ambientes não estruturados. Os autores demonstram que a metodologia PCA permite: a) A compressão da base de dados dos sinais adquiridos, b) Remoção de informação corrompida capturada pela câmara de profundidade IR e c) Criação de novos vectores próprios para comparação das imagens de profundidade reconstruídas com a base de dados PCA. Os ensaios de robustez incluem a introdução de falhas de informação artificiais nas imagens captadas. Ultrapassa-se assim a limitação criada pela perda de informação, validando a utilização do sensor de profundidade para operar em ambientes não estruturados, com condições de iluminação variáveis.

A robótica probabilística é igualmente um tema em amplo desenvolvimento, pelo que se destacam o trabalho de M. H. Li, *et al* (2007). Estes introduzem um filtro de partículas para implementação de SLAM monocular em ambientes fechados. Combinam o modelo UKF (*Unscented Kalman Filter*) para amostragem de novos estados, por integração do estado observado. O filtro EKF é posteriormente aplicado para estimativa do posicionamento das referências observadas.

N. Ganganath and H. Leung (2012) apresentam um estudo sobre os diversos transdutores utilizados actualmente para navegação e localização autónoma para robôs WMR (*Wheel Mobile Robots*). No estudo, os autores demonstram que a navegação com recurso exclusivo a odómetria (*encoder*) produz erros de localização exponenciais ao longo do tempo. A utilização simultânea de odometria e da câmara de profundidade *Microsoft Kinect*® como sensores num filtro EKF permite optimizar o algoritmo SLAM com precisão acrescentada na localização de referências. O resultado da experiência é a fusão do filtro EKF com o filtro de partículas (algoritmo Monte Carlo), resultando numa

diminuição drástica do erro de localização RMSE (*root mean square error*), por redução da incerteza na postura do estado futuro.

A optimização paramétrica com recurso a rede neuronal para melhorar o algoritmo SLAM é proposta por K. Jeong-Gwan *et al* (2010) através do filtro NN-EKF para compensação de erros odométricos resultantes da medição por *encoders*. O filtro NN (*neural network*) adapta-se aos erros sistemáticos do movimento do robô, sem conhecimento do estado anterior, pela capacidade de aprendizagem apresentada pelo algoritmo. A experiência com modelos reais em condições reais é obtida com o recurso a modelos múltiplos, ou seja utilização de múltiplos filtros para tarefas substancialmente complexas.
O caso da navegação e localização SLAM para um robô subaquático ilustra a utilização de dois filtros EKF, conforme demonstram A. Mallios *et al* (2010).
O primeiro filtro estima o caminho local percorrido pelo robô e a probabilidade de incerteza no *scan* adquirido pelo sonar, fazendo a estimativa de posicionamento no mapa e corrigindo as imagens acústicas capturadas pelo movimento do robô. O segundo filtro EKF estima e regista as posturas (estados) do *scan*.

Rodrigues, Carreira, Cardeira e Calado (2013) apresentam um sistema de localização de robôs móveis em ambientes não estruturados. Neste artigo, os autores demonstram a capacidade do robô se auto localizar em ambientes não estruturados, sem necessidade de reconhecimento de *beacons* ou de outras referências exteriores. O sensor de posição baseado em PCA é filtrado por uma grelha probabilística Bayesiana combinada com um filtro de Kalman linear para estimar a postura global do robô móvel. Este método, para além do baixo custo computacional que apresenta, permite a auto-localização do robô móvel em ambientes fechados, com precisão e capacidade de operação em diferentes condições de iluminação.

Das diversas obras sobre o tema da localização robótica em ambiente estruturado, propõem-se para a presente dissertação os autores (i) Siegwart e Nourbakshsh, com a obra "*Introduction to Autonomous Mobile Robots*" (2004) e (ii) Thrun, Fox e Burgard, com a obra "*Probabilistic Robots*" (2005), como as principais referências bibliográficas para a elaboração de um estimador MCL.

1.2. Estado da Arte na detecção de humanos

A presente dissertação tem como um dos seus objectivos o desenvolvimento de um detector de Humanos com visão RGB-D. A detecção de humanos é feita com recurso a descritores e classificadores de imagem. Um descritor pesquisa a imagem por determinadas características, posteriormente /analisadas pelo classificador, que determina se estas correspondem, ou não, ao objectivo da detecção. Estes algoritmos têm sido desenvolvidos nos últimos anos. Propõe-se uma revisão histórica, separando os diversos modelos de detecção de humanos, para melhor compreensão dos temas.
Os descritores de imagem com histogramas com campos receptivos foram inicialmente apresentados por Swain e Ballard (1991) com uma performance aceitável no reconhecimento de objectos, comparando diferentes histogramas RGB e ignorando qualquer relação espacial entre características de imagem em pontos diferentes. A utilização de um descritor para reconhecer objectos leva a que Lowe (1999) apresente uma nova técnica baseada na transformada de Hough que consiste na utilização de imagens triplas para acumulação de provas no processo de escolha e emparelhamento dos objectos que representem pontos de interesse e respectivos classificadores de imagem. Esta integração dos diferentes componentes faz com que o método do descritor de características na imagem SIFT *(Scale Invariant Feature Transform)* esteja na base de diversos modelos matemáticos para a detecção de objectos ou pessoas numa imagem.

Schiele e Crowley (2000) evoluem o conceito dos histogramas com campos receptivos para descrever características da imagem apresentando histogramas com derivadas parciais de primeira ordem ou combinação de magnitudes de gradientes e respostas Laplacianas processadas em múltiplas escalas. Linde e Lindebeg (2004, 2012) refinam este tipo de aproximação para um modelo mais geral, o histograma de campos receptivos compostos, combinando múltiplas derivadas de Gauss ou invariantes diferenciais, ambas calculadas a partir de parelhas de imagem de escala cinza ou RGB. Este método comprova ser eficiente ao tratar histogramas de elevada dimensão assim como histogramas de fila composta complexa, com elevada performance em ambos.

O sucesso dos descritores de imagem SIFT na detecção de objectos é motivador de novos trabalhos. Mikolajczyk e Zisserman (2004) fazem a extracção de características de imagem com histogramas orientados em função da fronteira da mesma. Estas

características foram combinadas com sete sub-detectores, resultando num detector de pessoas em imagens estáticas, que se demostra ser robusto perante o fenómeno de oclusão.

Dalal e Triggs (2005) aperfeiçoaram a técnica de descrição visual de objectos com recurso ao SIFT denso, apresentando um algoritmo baseado numa malha substancialmente mais refinada no domínio da imagem, resultando num número alargado de características na imagem local. Este método permite uma detecção mais robusta de pessoas, quando comparado com uma malha mais esparsa.
Tendo como base a elevada discriminação dos histogramas de gradientes orientados altamente dependentes da posição, os autores desenvolvem um descritor de imagem constituído por um conjunto de histogramas de gradientes orientados, calculados sobre uma grelha contida no domínio da imagem. O classificador HOG (*Histograms of Oriented Gradients*) é um descritor de características regionais na imagem. É mais correcto afirmar que o HOG se assemelha ao histograma com campos receptivos, definidos em sub-regiões do domínio da imagem. Sobre este, o operador HOG evolui ao (i) incluir dependência na posição das imagens por ser composto por um conjunto de histogramas de menor dimensão definidos nas sub-regiões locais e (ii) ser formado a partir dos histogramas de orientação locais ao invés de derivadas parciais ou invariantes diferenciais. Contudo, o descritor de características HOG não é normalizado na orientação (o SIFT é) logo não é rotacionalmente invariante, mas é normalizado no contraste da imagem (o SIFT não é).

Mikolajczyk e Cordelia (2005) propõem um descritor de características GLOH (*Gradient Location and Orientation Histogram*) que tem como base o descritor de características SIFT original do qual partilha um histograma dependente da posição do gradiente de orientações na vizinhança do ponto de interesse. O GLOH difere contudo ao (i) ser processado numa malha polar-logarítmica ao invés de rectangular, (ii) recorrer a 16 células para quantificar o gradiente de direcções, em oposição às 8 células do SIFT e (iii) usar a análise de componentes principais (PCA) para reduzir a dimensão do classificador de imagens. No geral verifica-se uma vantagem de performance para o GLOH em imagens estruturadas e vantagem para o SIFT em imagens com texturas, conforme demonstrado pelos autores.

O descritor SURF (*Speeded Up robust features*) foi apresentado por Bay *et al* (2006, 2008) como uma variação do descritor SIFT com o qual partilha um vector de características, obtido a partir de respostas similares a campos receptivos, na vizinhança de um ponto de interesse. O classificador SURF destaca-se nos seguintes aspectos: a) Baseia-se na propagação de *Haar wavelengths*, ao invés das aproximações diferenciais da pirâmide de imagem (SIFT), b) o ponto de interesse é obtido por aproximação do máximo espaço-escala do determinante da matriz Hessiana por contraste com o operador de Laplace e c) as entradas no vector de características são processadas como somas e somas absolutas das derivadas de primeira ordem $\sum L_x, \sum |L_x|, \sum L_y, \sum |L_y|$. Experimentalmente demonstra-se que o operador SURF tem uma performance comparável ao operador SIFT, com vantagem de velocidade computacional para o SURF.

O trabalho de referência de Dalal e Triggs, "*Histograms of Oriented Gradients for Human Detection*" (2005) é proposto como base de trabalho das componentes algorítmicas do processo de detecção de humanos da presente dissertação.

1.3. Estado da arte na navegação robótica

A navegação é um tema fundamental para qualquer sistema de controlo de um robô móvel. Propõe-se de seguida uma revisão com critério histórico, dando enfâse à temática dos campos de potências, por ser um componente fundamental do sistema do robô móvel apresentado nesta dissertação.

Simmons e Koenig (1995) apresentam uma obra sobre navegação robótica probabilística em ambientes parcialmente observáveis. Os autores abordam a problemática dos robôs necessitarem de sistemas de navegação robustos para poderem operar autonomamente durante longos períodos de tempo. Apresentam resultados para um sistema que utiliza modelos de Markov para observações parciais de modo a fazer o seguimento do robô em ambiente do tipo escritório. O robô é comandado sob o modelo de acção orientada por objectivos, baseado numa distribuição probabilística da sua localização, incluindo o grau de incerteza dos sensores e actuadores. O modelo inclui a integração de um mapa topológico com informação métrica aproximada. A robustez do conjunto é demonstrada pelos autores.

Desouza (2002) faz a revisão da navegação robótica móvel com base em visão. O autor separa a navegação em dois componentes, a realizada dentro de portas e a feita no exterior. Para cada componente, faz-se a divisão em temas específicos com a diferença entre ambientes estruturados e não-estruturados. Dentro de ambientes *indoor* estruturados, fazem-se analises separadas entre modelos espaciais topológicos ou geométricos. Para os ambientes não-estruturados, o autor analisa a navegação com recurso a métodos *optical flow* para reconhecimento de objectos específicos no espaço.

Tsourveloudis *et al* (2002) propõem um campo de potências electrostático para planeamento de trajectória do robô móvel *Nomad 200*. O campo de potências electrostático é combinado com uma lógica *fuzzy* de dupla camada e implementado num sistema de navegação em tempo real, num ambiente dinâmico bidimensional. Inicialmente o ambiente é mapeado numa rede de resistências eléctricas. Um campo potencial electrostático é posteriormente adicionado através da injecção de corrente na rede. O caminho de máxima corrente na rede corresponde ao caminho óptimo no ambiente. A primeira camada do motor de inferência de lógica *fuzzy* faz a fusão das leituras dos sensores de colisão do robô, segundo as 4 direcções principais (frente, traz, esquerda, direita). A segunda camada actua no desvio de potenciais colisões contra obstáculos dinâmicos, mantendo em simultâneo o seguimento de trajectória gerada pelo campo potencial electrostático.

Cosío e Castañeda (2004) apresentam um trabalho sobre navegação autónoma de um robô móvel baseada num campo de potências artificial melhorado com um algoritmo genético. Nos métodos de campo de potências convencionais, o robô é atraído para a meta e rejeita diversos obstáculos. Ao depender de apenas um único campo de potências, podem ocorrer armadilhas, onde o campo não consegue produzir um resultado que contorne obstáculos de maior dimensão. Os autores apresentam um método que gera múltiplos pontos de atracção auxiliares que ajudam o robô a evitar obstáculos grandes ou espaços confinados. A configuração do campo de potências óptimo e determinada pelo algoritmo genético. As simulações baseiam-se em 3 configurações distintas de obstáculos, cada uma com 10 trajectórias diferentes, com uma taxa de sucesso de 93%.

Huanga *et al* (2006) propõem uma nova abordagem à navegação robótica guiada por visão, baseada no modelo de navegação humana. Este modelo utiliza as direcções relativas da meta e obstáculos, a distância à meta e a largura angular dos obstáculos para calcular um campo de potências sobre a direcção do robô. O campo controla a aceleração angular e dirige o robô em direcção à meta, ao mesmo tempo que o afasta dos obstáculos. Dado que a direcção é controlada directamente, este método adapta-se bem à navegação local em robôs não-holonómicos[2], pois produz trajectórias suaves e com uma curvatura contínua. Este modelo foi desenvolvido com base em visão RGB *sem* informação de profundidade, mas funciona com outros tipos de sensores. O robô utilizado nos ensaios experimentais foi do tipo tracção diferencial.

Apresenta-se uma revisão histórica de outras problemáticas na navegação robótica, nomeadamente na presença de obstáculos ou em ambientes dinâmicos.

Benavidez (2011) apresenta uma estrutura para um sistema de navegação com seguimento de alvos para robôs móveis. O autor utiliza uma câmara Kinect RGB-D com um sistema x86 com *software* Ubuntu Linux. O controlador lógico usa uma rede lógica *fuzzy* para executar o desvio de obstáculos e fazer o seguimento de alvos (*tracking*). Introduz inovação ao enviar a informação recolhida para um servidor com uma lógica de aprendizagem baseada numa rede neuronal para reconhecimento de padrões, seguimento de objectos assim como uma estratégia a longo prazo de planeamento de trajectórias e optimização de processos. O objectivo geral é a criação de um sistema robótico com capacidade de autonomia em ambientes exteriores.

Chi-Pang Lam *et al* (2011) apresentam um algoritmo de navegação que tem em consideração os estados de humanos e robôs para alcançar uma coexistência harmoniosa entre ambos. A navegação robótica na presença de humanos é um tema pouco estudado no campo da robótica. Ao navegar num ambiente com humanos, o robô dever prestar atenção, para além da trajectória e obstáculos, à presença de humanos. Os autores propõem regras que garantem uma navegação segura e suave na presença de humanos, através do método de navegação HCSN (*Human-Centered Sensitive Navigation*). O

[2] Em robótica, um sistema é não-holonómico se o número de graus de liberdade controláveis for inferiores ao número total de graus de liberdade. Por exemplo, um robô do tipo triciclo não consegue executar estacionamento paralelo.

método considera que tanto os humanos como os robôs possuem zonas sensíveis, tendo os humanos zonas de segurança física e psicológica. Estas zonas são modeladas de acordo com prioridades pré-estabelecidas, cujo resultado origina robôs com movimentos socialmente aceitáveis.

Carreira, Calado e Cardeira (2011) apresentam um sistema de planeamento para navegação robótica, com capacidade de desvio de pessoas, baseado em campos potenciais. O planeador processa a posição, orientação e diferentes distâncias de conforto e adopta um comportamento que evita a colisão, assim como adopta um movimento socialmente aceitável. Tem a particularidade de introduzir duas forças repulsivas relacionadas com a segurança física e de visibilidade (psicológica).

Raja e Pugazhenthi (2012) fazem uma revisão dos sistemas de planeamento de trajectória para robôs móveis. Partindo do pré-requisito de um planeamento que evite colisões, os autores fazem uma análise dos tipos de planeadores existentes e dos avanços propostos em ambientes *offline* assim como *online*. Os autores apresentam algoritmos de optimização que podem ser utilizados em paralelo com algoritmos clássicos, incluindo a capacidade de resolver problemas do tipo tempo polinomial não-determinístico (*NP-hard*). Concluem com uma identificação de factores que consideram ser fundamentais no desenvolvimento de um algoritmo computacionalmente eficiente para o planeamento de trajectórias.

Lau *et al* (2013) apresentam um estudo sobre a representação eficiente de modelos espaciais, em formato tipo grelha, para a navegação robótica em ambientes dinâmicos. Abordam os sistemas de navegação mais comuns e introduzem um novo algoritmo que consegue lidar com súbitas alterações na navegação, como a introdução de obstáculos móveis ou a passagem por áreas não mapeadas. Na obra, os autores incluem algoritmos incrementais para fazerem o *update* de algoritmos de navegação conhecidos.

Carreira, Calado, Cardeira e Oliveira (2015) apresentam um artigo sobre uma classe de filtros complementares para a fusão de três sensores, denomidado de *Complementary Filters3* (CF3). Com o objetivo de obter estimativas óbtimas, os autores abordam a problemática da estimação com a utilização de técnicas optimizadas do filtro de Kalman linear. Esta nova classe de filtros *Complementary Kalman Filters3* (CKF3) é aplicada

para estimar a atitude de um robô móvel, fundindo os sinais fornecidos por três sensores: giroscópio, bussula digital e odometria, para ambos no domínio contínuo e discreto do tempo.

Para além dos estudos supra mencionados, existem outros na área de algoritmos de navegação. Para a presente dissertação propõem-se as obras de (i) Siegwart e Nourbakhsh (2004) no capítulo "*mobile robot kinematics*" como referência teórica para cinemática de movimento de robôs móveis.

1.4. Trabalho realizado

Um robô móvel para poder cumprir os objectivos definidos deve possuir a capacidade de sentir o mundo físico que o rodeia, planear de acordo como essa informação e agir de modo a alcançar a sua meta. A palavra-chave nesta afirmação é planear, que implica "pensar" no processo óptimo que conduza ao objectivo, tendo em consideração obstáculos móveis não previstos no momento da partida, por exemplo pessoas, que afectam o processo de execução da trajectória inicialmente calculada. Uma máquina autónoma sem planeamento implica um sistema reactivo incapaz de se adaptar.

"a goal oriented machine that can sense, plan and act"[3]

Navegar num ambiente implica um conjunto de passos essenciais, passando pela localização, planeamento de trajectória, navegação e controlo. Cada um destes termos necessita de ser estudado numa ordem lógica progressiva. Na base está a cinemática do robô, com os necessários modelos matemáticos que traduzem a sua realidade física. O planeamento de trajectórias possibilita a navegação do robô através de um caminho. Navegar implica que o robô sabe onde está, ou seja, possui capacidade de se localizar relativamente ao mundo.

Existem vários tipos de robô no mercado, desde os mais simples feitos "em casa" aos mais complexos, de elevado custo, que incluem sensores de visão e profundidade, sensores de choque, motores e sistema de controlo, totalmente integrados.

[3] *in* Peter Corke (2011), " Robotics, Vision and Control: Fundamental Algorithms in MATLAB", *Volume 73, capitulo 5, pp. 88*

Para a presente dissertação, o modelo escolhido é o disponível no laboratório de robótica do ISEL-ADEM (ver figura 1.1). Este consiste num modelo de duas rodas motorizadas diferenciais e terceira roda de apoio livre, apoiadas num chassis de alumínio para suporte do PC e que contém a placa de processamento de sinal, bateria e portas de comunicação. O conjunto é complementado com a *Microsoft* Kinect® para captura de imagens RGB-D, com o objectivo de detectar referências (*beacons*) e pessoas.

Na presente dissertação desenvolve-se um sistema de navegação assente na localização probabilística e no planeamento de trajectória com campo de potências. Este possui a capacidade de detectar humanos na sua trajectória e, em tempo real, processar um novo caminho para alcançar a meta, contornando as pessoas de modo a garantir a sua segurança física e psicológica.

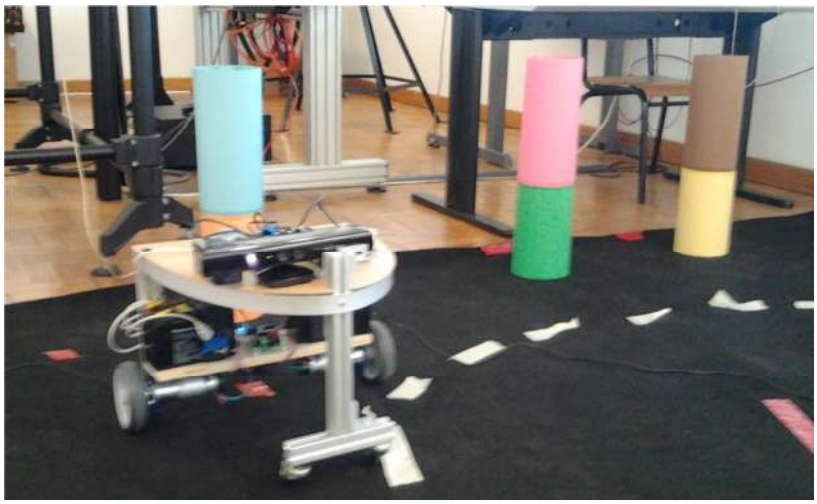


Figura 1.1 – Robô móvel com Kinect, incluindo zona de ensaios e *beacons*

Pretende-se igualmente com a presente dissertação, criar um modelo que possa ser utilizado numa hipotética aplicação comercial. Em breve nota de exemplo, podem enaltecer-se as vantagens de um robô de transporte de peças a operar num ambiente industrial, ao ter capacidade de distinguir pessoas de objectos e agir de acordo, evitando-se possíveis acidentes e aumentado a eficiência da operação, oferecendo um superior grau de confiança das pessoas sobre o robô.

Capitulo II

2. Sistemas de localização

Um dos objectivos do robô móvel proposto é o de, em qualquer instante, saber qual é a sua localização no mundo que o rodeia. Para tal, o robô observa esse mundo através de sensores de visão como a Microsoft Kinect e procura localizar objectos ou pessoas. A utilização de um sistema GPS (*global positioning system*) seria uma hipótese muito útil, em teoria, para resolver a problemática da localização do robô proposto. Na prática tal não se verificou pois o sistema não funciona dentro de edifícios, tornando-o inviável para robôs de pequena dimensão em ambientes fechados (*indoor*).
Recorrendo ao sistema de visão da Kinect, é possível desenvolver duas funcionalidades úteis à operacionalidade do robô: a primeira é a de distinguir uma pessoa de um objecto (ex.: cadeira) e a segunda é a de discriminar, também, entre os objectos conhecidos, aqueles que interessam para a sua localização. Ultrapassados estes dois obstáculos, estará disponível um conjunto de informações que indicam a posição de determinados objectos relativamente ao robô, permitindo assim que este estime a sua postura no mundo. A este conjunto de processos designa-se por sistema de localização.

Neste capítulo abordam-se os temas que contribuem para o sistema de localização do robô. O primeiro refere-se à detecção de *beacons*, a qual é resolvida através da pesquisa de determinados padrões de cor, em função de padrões de referência específicos, incluindo a sua posição no espaço tridimensional. Estas observações são posteriormente refinadas através de uma rede neuronal artificial.
O segundo tema é o da localização por métodos de Monte-Carlo[4] (MCL), os quais, através dos dados recebidos e processados dos sensores, estimam probabilisticamente qual a posição do robô no mundo. O capítulo termina com os ensaios elaborados e respectivas conclusões para cada um dos temas.

[4] Os métodos de Monte Carlo são normalmente empregues quando se pretende simular sistemas com elevado número de graus de liberdade e incerteza nas entradas, ou seja quando é irrealista tentar processar um resultado exacto com um algoritmo determinístico. O método foi desenvolvido para o projecto Manhattan, durante a 2ª Grande Guerra em Los Alamos, pelos matemáticos John Von Neuman, Stanislaw Ulam e Nicholas Metropolis. O nome do projecto faz referência aos casinos de Monte Carlo por espelhar o carácter de jogo do método.

2.1. Detecção de *Beacons*

O método proposto tem por objectivos a detecção de *beacons* e determinação da sua posição no espaço tridimensional. Esta informação é posteriormente fornecida ao sistema de localização do robô, contribuindo para a correcção das estimativas de posição.

A detecção inicia-se com a Kinect, que captura as imagens de cor e profundidade do mundo. Decorre um processo de identificação de regiões de cor que coincidam com as cores de referência existentes nos alvos. O resultado indica claramente, para cada alvo detectado, a posição de seu centro na imagem. Segue-se o processo de transformação desta informação em coordenadas cartesianas, para poder extrair as posições *xx*, *yy* e *zz* do *beacon* detectado. O processo conclui-se com a correcção da informação através de uma rede neuronal, que pode ser enviado para o estimador MCL, conforme ilustra a figura 2.1.

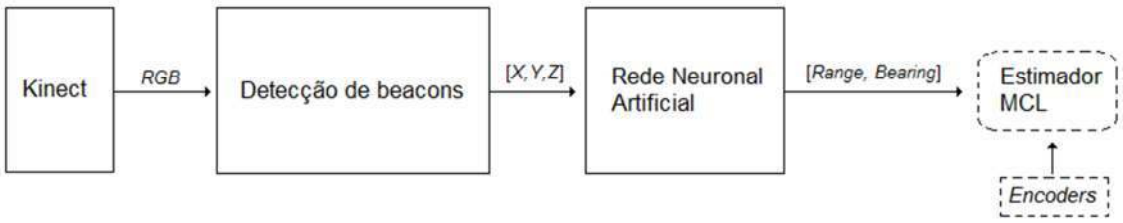


Figura 2.1 – Esquema simplificado da detecção de *beacons*

Condicionalismos:

- O *hardware* proposto implica a utilização de um bloco de correcção entre as imagens de cor obtidas pela câmara RGB e a profundidade obtida pelo sensor de infravermelhos *Depth*, antes da aplicação do método da detecção de *beacons*.

- Existem diferenças na informação de profundidade recolhida pelo sensor de infravermelhos e a sua correspondente real no espaço, logo aplica-se uma rede neuronal artificial para corrigir esta situação.

Método proposto

Segue-se a apresentação em mais detalhe de método matemático proposto para a detecção de *beacons*.

2.1.1. Correcção das imagens do sensor RGB-D

A fusão das imagens $\mathbf{RGB}_{n,m}$ e $\mathbf{Depth}_{n,m}$ (ver figura 2.2) através do módulo *NID_CVST_Depth* do Simulink, produz três matrizes, com a relação da posição dos pixéis em RGB e correspondente na imagem de profundidade corrigidos.

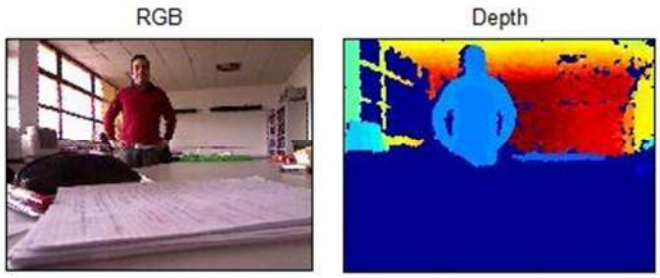


Figura 2.2 – Ensaio com Kinect, a) Imagem RGB, b) Imagem de profundidade (*Depth*)

O objectivo da correcção é eliminar erros entre a posição de um dado pixel *ij* na imagem de cor e a sua correspondência na imagem de profundidade. O algoritmo tem em consideração nos cálculos a distancia entre a câmara de RGB e *Depth* da Kinect, de modo a não introduzir erros de perspectiva.

A matriz $\mathbf{I}_{ij}, i = 1, \dots, n \; e \; j = 1, \dots, m$ contém a imagem RGB corrigida. A matriz $\mathbf{D}_{ij}$ contém a imagem *Depth* corrigida. A matriz $\mathbf{XYZ}_{ijk}$, $k = 1,..,3$ é uma matriz de dimensão 640x480 com 3 camadas adicionais k que representam a informação do sensor de profundidade para os eixos *xx*, *yy* e *zz*.

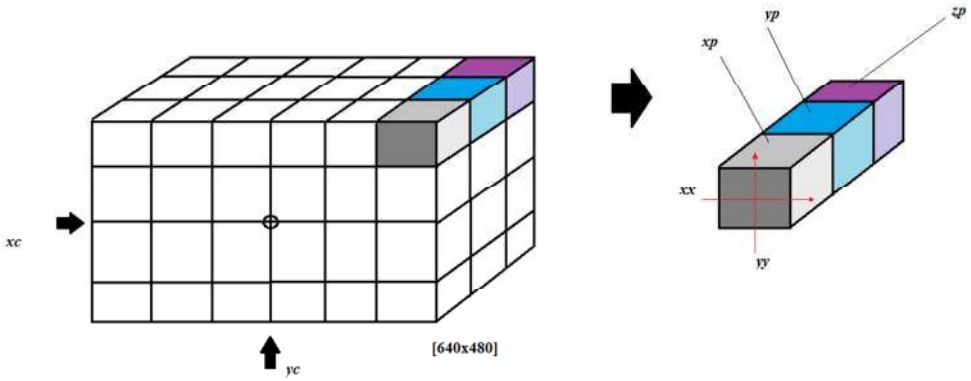


Figura 2.31 – Representação da matriz XYZ, com as componentes *xp*, *yp* e *zp*.

Internamente corrige a diferença de pixéis *p linhas, q* colunas, onde o pixel *ij* na imagem *depth* não coincide com o pixel *ij* na imagem RGB, aparecendo como um desfasamento *i=i+p* e *j=j+q*. Segue-se a transformação destes dados numa nuvem de pontos tridimensional, onde cada pixel *ijk* contém a informação das distâncias

relativamente à posição da Kinect. Ou seja, em cada ponto, existem dados de distância, em metros, para o eixo *xx* horizontal, eixo *yy* vertical, ambos correspondentes à imagem bidimensional a cores e o eixo *zz*, que corresponde à profundidade. O resultado é a matriz **XYZ**, com as componentes *xp*, *yp* e *zp* (ver figura 2.3).

2.1.2. Identificação de referências

A detecção é feita através de procura de padrões de cor na imagem RGB, separando-se os alvos em q combinações distintas. A cor de referência é sempre a primeira, vertical e no topo do cilindro, logo para o método proposto, esta é identificada como cor principal e a segunda, inferior, a cor secundaria. A percentagem dos valores RGB que definem uma determinada cor de referência é definida pelo vector $\mathbf{cor}_{ref} = [R_{ref}, G_{ref}, B_{ref}]$ sendo $c = \mathbf{cor}_{ref},\ c \in \mathbb{N}$ o índice das cores. As combinações de cores designam-se por $q,\ q \in \mathbb{N}$. O número que identifica um alvo designa-se por $p,\ p \in \mathbb{N}$. Notar que para a presente dissertação são utilizados p=6 alvos, com q=6 combinações, os seja Vermelho-verde (q=1), Laranja-azul (q=2), Verde-vermelho (q=3), Azul-laranja (q=4), Castanho-amarelo (q=5) e Amarelo-castanho (q=6).

Imagem **RGB** com $m \times n$ pixéis, onde $i = 1, \dots, n$ e $j = 1, \dots, m$

Imagem **RGB** corrigida, $\mathbf{IMG}_{ij} = [\mathbf{R}_{ij}\ \mathbf{G}_{ij}\ \mathbf{B}_{ij}],\ R, G, B \in \{0, 1, 2, \dots, 255\}$

Cor RGB de referência $\mathbf{cor}_{ref} = [R_{ref}\ G_{ref}\ B_{ref}],\ R_{ref}, G_{ref}, B_{ref} \in \{0, 1, 2, \dots, 255\}$

2.1.3. Normalização

É fundamental salientar que este método depende muito das condições de iluminação presentes, e o mesmo objecto vai ter diferentes percentagens nas combinações RGB dependendo da hora do dia, iluminação exterior ou outros factores que possam afectar a iluminação do local de ensaios. Múltiplos testes permitem identificar um intervalo de confiança onde é possível identificar a cor, para permitir a rápida modificação do padrão de pesquisa caso se verifiquem falhas na captura. Para o exemplo de cor de referência vermelha, os intervalos [0.85 0.20 0.45] e [0.70 0.40 0.50] permitem algumas combinações, de modo a adaptar a detecção às condições de iluminação locais.

O processo de normalização inicia-se com um escalar que indique o grau de luminosidade do local.

$$lum = \sqrt{\mathbf{R}_{ij}^2 + \mathbf{G}_{ij}^2 + \mathbf{B}_{ij}^2} \tag{2.1}$$

Define-se um intervalo para o valor de luminosidade. Este intervalo é determinado por ensaio, e pode ser alterado em função das condições de iluminação locais. Para a presente dissertação, utilizou-se o valor mínimo 0,1.

$$lum = \begin{cases} 0{,}1 \; se \; lum < 0{,}1 \\ lum \; se \; lum \; \geq 0{,}1 \end{cases} \tag{2.2}$$

Segue-se a normalização individual das componentes primárias de cor da imagem RGB.

$$norm_{i,j,jRed} = \frac{\mathbf{R}ij}{lum} \; , \; norm_{i,j,Green} = \frac{\mathbf{G}ij}{lum} \; , \; norm_{i,j,Blue} = \frac{\mathbf{B}ij}{lum} \tag{2.3}$$

O resultado é a transformação da imagem original **Img** na imagem normalizada **Imd**. Ambas são compostas por uma matriz tridimensional, com as dimensões dos pixéis *i* e *j* e a dimensão de cor (*ref*).

$$\mathbf{Imd}_{\boldsymbol{i,j,ref}} = \begin{bmatrix} norm_{1,1} & norm_{1,j} \\ norm_{j,1} & norm_{i,j} \end{bmatrix}_{ref} \tag{2.4}$$

2.1.4. Identificação do alvo

O processo de identificação de um alvo de uma determinada cor principal na imagem implica um conjunto de passos de modo a eliminar falsas detecções. O caso mais comum pode ser a presença de outros objectos de cor idêntica no campo de visão que não têm correspondência aos alvos de referência no mapa. O método de identificação de seguida apresentado procura identificar regiões candidatas a alvos, testar e validar, passando de candidato a detecção positiva. O processo inicia-se com a análise de cada pixel *ij* na imagem normalizada *Imd.*

i) Determinar a distância euclideana entre a cor detectada e a cor de referência.

$$d_{ij} = \sqrt{(R_{ref} - \mathbf{R}_{ij})^2 + (G_{ref} - \mathbf{G}_{ij})^2 + (B_{ref} - \mathbf{B}_{ij})^2} \tag{2.5}$$

ii) Definir o intervalo para a cor, onde $d_{ref}(c)$ é um escalar que representa uma distância mínima aceitável entre a cor obtida e a cor de referência *c*. O valor de d_{ref} resulta de um processo de ensaio e é ajustado em função das condições de iluminação locais.

O resultado é a variável *bw* cuja informação é binária [0,1] logo permite a rápida identificação de regiões na imagem.

$$bw_{ij}(c) = \begin{cases} 0, se\ d_{ij} > d_{ref}(c) \\ 1, se\ d_{ij} \leq d_{ref}(c) \end{cases} \quad (2.6)$$

iii) Segue-se o processo de detecção de regiões cuja cor esteja dentro do intervalo de pesquisa. *m* identifica o número de regiões (*blobs*) detectadas com a mesma cor, x_{mj} o centro da região m em xx, y_{mj} o centro do região detectada *m* em yy, w_{mj} a largura (*width*) da região e h_{mj} a altura (*height*) da região.

$$\mathbf{BLOB}_{m,j}(c) = \begin{bmatrix} x_1 & y_1 & w_1 & h_1 \\ x_m & y_m & w_m & h_m \end{bmatrix}, j = 1, \dots, 4 \quad (2.7)$$

Por região *m* de cor *c* detectada, designa-se o indice que identifica numericamente a região $u = indice_m(c),\ u \in \mathbb{N}$, a area da região $area_u = (w_u \times w_u)$, o centroide da região $\boldsymbol{ct}_u = (x_u, y_u)$.

iv) Com *m* regiões de cor *c* detectadas, é necessário seleccionar duas cuja combinação coincida com as cores de referência do alvo a detectar. Os alvos têm a combinação de cores definida, designa-se esta combinação pela matriz *Target,* $\mathbf{TG}_q = [c_1, c_2],\ q = 1, \dots, 6$ onde o índice q indica sempre a cor principal de referência. A pesquisa da referência $\mathbf{TG}_q$ na imagem é feita através do cálculo da menor distância euclideana entre pares de manchas $\mathbf{BLOB}_{mj}(c)$ onde $c \in \mathbf{TG}_q$. O vector resultante *beacon,* $\mathbf{bc}_u$ tem dimensão *u* indicativa do número de regiões detectadas.

$$\mathbf{bc}_u = \sqrt{[x_m(TG_{q,1}) - x_m(TG_{q,2})]^2 + [y_m(TG_{q,1}) - y_m(TG_{q,2})]^2} \quad (2.8)$$

v) O método proposto funciona com alvos com combinação de cores distinta de modo a poder referenciar posições específicas no mundo. Tal implica que, para a presente dissertação, não deve existir na imagem dois ou mais alvos com a mesma combinação de cor. Se existir, pode estar-se perante uma falsa detecção por combinação casual de cores de outros objectos. Um controlo eficaz para esta situação é proposto pela limitação da distância $a \in \mathbb{R}^+$ entre pares de cores dos candidatos a alvo, detectados na imagem.

$$\mathbf{bc}_u = \begin{cases} 0, & se\ bc_u > a \\ bc_u, & se\ bc_u \ll a \end{cases} \quad (2.9)$$

vi) Os candidatos a alvos detectados necessitam de passar por outro teste, para garantir que o alinhamento dos centroides das regiões coincide com a posição vertical dos alvos cilíndricos. O alvo tem o centroide da cor de referência sempre por cima do centroide da cor secundaria, com ambos os centroides alinhados no mesmo eixo vertical. Define-se um intervalo escalar para filtrar as regiões detectadas, nos eixos horizontal *dx* e vertical *dy* da imagem.

$$dy_j = \sqrt{[ct_u(c_{1j}) - ct_u(c_{2j})]^2},\ c_{1j} \neq c_{2j} \tag{2.10}$$

$$dx_i = \sqrt{[ct_u(c_{1i}) - ct_u(c_{2i})]^2},\ c_{1i} \neq c_{2i} \tag{2.11}$$

vii) Com as distâncias *dx* e *dy* pode-se testar a verticalidade do candidato $\mathbf{bc}_u(TG)$.

$$\mathbf{bc}_u = \begin{cases} 0, & se\ dx > \frac{1}{2}.dy \\ bc_u, & se\ dx \leq \frac{1}{2}.dy \end{cases} \tag{2.12}$$

viii) Os candidatos filtrados pela verificação anterior passam a ser detecções positivas do alvo q. Para q alvos de referência cria-se um ciclo de programação de 1…r que repita os passos 3º → 7º. O resultado deste ciclo é a matriz *detected beacons.*

$\mathbf{DB}_{r,s}, r \in \mathbb{N}, s \in \mathbb{N}$

$$\mathbf{DB}_{r,s} = \begin{bmatrix} x_{1,1} & y_{1,2} \\ \vdots & \vdots \\ x_{r,s} & y_{r,s} \end{bmatrix} \tag{2.13}$$

2.1.5. Transformação da informação

A transformação da posição na imagem de cada alvo detectado em coordenadas cartesianas no espaço implica a comparação de posições entre os pixéis $[x_{rs}, y_{rs}]$ da matriz $\mathbf{DB}_{rs}$ com a matriz $\mathbf{XYZ}_{ijk},\ k = \{1,2,3\}$. Resultam 3 escalares com a informação da profundidade *xx*, *yy* e *zz*, para cada alvo detectado.

$$\begin{aligned} Xbc &= \mathbf{XYZ}_{i,j,1}(\mathbf{DB}_{rs}), i = r, j = s, k = 1 \\ Ybc &= \mathbf{XYZ}_{i,j,2}(\mathbf{DB}_{rs}), i = r, j = s, k = 2 \\ Zbc &= \mathbf{XYZ}_{i,j,3}(\mathbf{DB}_{rs}), i = r, j = s, k = 3 \end{aligned} \tag{2.14}$$

Apesar da informação da altura do centroide do alvo detectado *Zbc* estar disponível, esta não contribui em nada para a detecção dos alvos no modelo proposto de duas cores por

alvo. Caso o número de combinações q aumente, esta informação permite diferenciar diferentes alturas de centroides, para aumentar a distinção entre combinações.

2.1.6. Resultado do modelo proposto para a detecção de *beacons*

O vector de posição dos *beacons,* $\mathbf{bp_q} = [Xbc_q, Ybc_q]$ guarda a informação da posição do alvo q no espaço, nas dimensões *xx* e *yy*.

A aplicação do presente método não implica que a câmara esteja centrada obrigatoriamente na posição [0,0] do robô. A variável *kinect offset* permite que a câmara, quando colocada no robô, possa ter um posicionamento excêntrico $ko = [ko_x, ko_y]$. O vector de posição dos *beacons*, afectado do excêntrico da câmara designa-se por *beacon kinect.*

$$\mathbf{bk}_q = [Xbk_q + ko_x, Ybk_q + ko_y] \tag{2.15}$$

A matriz que contém a informação de todos os vectores de posição designa-se por *sensor data* ou seja

$$\mathbf{SD}_q = \begin{bmatrix} Xbk_1 & Ybk_1 \\ \vdots & \vdots \\ Xbk_q & Xbk_q \end{bmatrix} \tag{2.16}$$

2.2. Erro de medição do sensor visual

No decorrer dos ensaios de *Ground Truth*[5] do algoritmo de detecção de *beacons* verificaram-se algumas discrepâncias entre as observações feitas pelo método de estimação da posição e as posições reais. Pretende-se que esta informação seja usada pelo sistema de localização, contudo, a qualidade do estimador está directamente ligada à confiança na informação recolhida pelos sensores. Tendo em conta que os erros obtidos estavam associados a problemas de falta de calibração do sensor, tornou-se então necessário corrigir os dados calculados pelo detector de *beacons*. Essa correcção feita através de uma rede neuronal artificial, que se apresenta de seguida.

[5] Na aprendizagem automática, o termo *Ground Truth* refere-se à verificação da precisão dos dados de treino para um determinado modelo de aprendizagem. A validação é comprovada através de métodos estatísticos.

2.2.1. Rede Neuronal Artificial

Uma Rede Neuronal Artificial (RNA) pode ser definida como uma estrutura de processamento (rede), passível de implementação em dispositivos electrónicos. É composta por um número de unidades interligadas (neurónios artificiais), sendo que cada unidade apresenta um comportamento específico de entrada/saída. Este é determinado por (1) uma função de transferência, (2) ligações com outras unidades dentro de um raio de vizinhança e (3) entradas externas, caso existam.

Citando Albert Nigrin (1993), "*uma RNA é um circuito composto por uma grande quantidade de unidades simples de processamento inspiradas no sistema nervoso humano*". Outra boa definição é dada por Simon Haykin (1999), onde *"uma RNA é um sistema paralelo e distribuído, composto por unidades de processamento simples que possuem uma capacidade natural de armazenar e utilizar conhecimento"*.

Uma rede neural artificial pode é normalmente projectada através da definição de:

1. Um tipo de nós artificiais;
2. Um padrão de conectividade entre os nós, ou seja, de uma *arquitectura* para a rede;
3. Um método de determinação dos parâmetros livres da rede, denominado de *algoritmo de aprendizagem* ou *treino*.

Para a correcção da posição dos *beacons*, é necessário uma rede que receba os dados polares da distância e ângulo entre o alvo e a câmara e filtre estes dados produzindo valores corrigidos. Para este efeito, propõe-se uma rede neuronal com apenas 1 camada escondida, o que se verificou ser suficiente para reduzir significativamente o erro de posição.

2.2.2. Rede *Feed Forward* com uma camada escondida

Este tipo de rede neuronal artificial multicamada consiste numa camada de entrada, uma camada escondida e uma camada de saída. A actividade resultante propaga-se através da rede, camada por camada, até que a resposta seja produzida na camada de saída, como ilustrado pela figura 2.4.

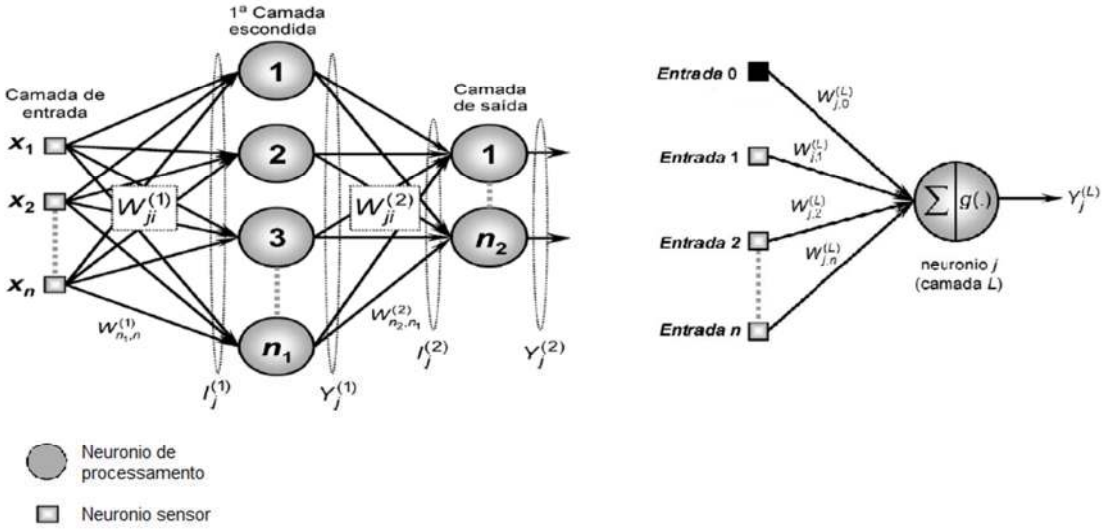


Figura 2.4 – (a) Rede neuronal artificial *feed forward*, (b) detalhe, *in* Salomão J. e Andreão R., *Inteligência Artificial*, pp. 14

Geralmente os nós de entrada são lineares. São também denominados por neurónios sensoriais. Esta rede é denominada *feed forward* porque a propagação do sinal ocorre apenas da entrada X_n para a saída $Y_j^{(L)}$, ou seja, é feita apenas no sentido directo.

O objectivo da rede é ter capacidade de treino. O algoritmo de retro propagação faz esse treino, ajustando os pesos $W_{ji}^{(L)}$ entre as camadas ocultas L através de um processo iterativo. Na fase *forward*, um padrão é apresentado à rede e passa pelas diversas camadas até que uma resposta seja produzida na camada de saída. Estas passagens entre camadas geram erros δ_j, calculados na fase seguinte. Esta é a fase *backward*, onde a saída obtida é comparada com a saída desejada para o padrão de treino. Esta comparação resulta num erro, que é propagado para trás, desde a camada de saída até à camada de entrada. Os pesos das camadas internas são ajustados em função do respectivo erro.

A variável $W_{ji}^{(L)}$ representa a matriz dos pesos da conexão entre ji neurónios em L camadas

$$W_{ji}^{(L)} = \begin{bmatrix} w_{10} & w_{11} & \cdots & w_{1i} \\ \vdots & \vdots & \ddots & \vdots \\ w_{j0} & w_{j1} & \cdots & w_{ji} \end{bmatrix}, j = 1, \dots, m, i = 1, \dots, n, L = 1, \dots l \qquad (2.17)$$

$I_j^{(L)}$ são os vectores cujos elementos expressam a entrada ponderada pelos pesos do j neurónio da camada L. Na presente dissertação é utilizada uma camada.

Camada escondida

$$I_j^{(1)} = \sum_{i=0}^{n} w_{ji}^{(1)}.x_i \Leftrightarrow I_j^{(1)} = w_{j,0}^{(1)}.x_0 + w_{j,1}^{(1)}.x_1 + \cdots + w_{j,n}^{(1)}.x_n \quad (2.18)$$

Camada de saída

$$I_j^{(2)} = \sum_{i=0}^{n1} w_{ji}^{(2)}.Y_i^{(1)} \Leftrightarrow I_j^{(2)} = w_{j,0}^{(2)}.Y_0^{(1)} + w_{j,1}^{(2)}.Y_1^{(1)} + \cdots + w_{j,n1}^{(2)}.Y_{n1}^{(1)} \quad (2.19)$$

$Y_j^{(L)}$ são os vectores cujos elementos expressam a saída do *j* neurónio da camada *L*.

$$Y_j^{(1)} = g\left(I_j^{(1)}\right),\ Y_j^{(2)} = g\left(I_j^{(2)}\right) \quad (2.20)$$

Algoritmo do processo de treino de retro propagação

1) A camada de saída $Y_j^{(L)}$ calcula o erro da rede δ_j.

2) Faz-se a correcção dos pesos, $\Delta\omega_{ji} = \alpha\delta_j X_i$ (2.21)

3) A camada de saída envia o erro para a última camada oculta.

4) A camada oculta estima e calcula o seu erro, $\delta_j = f'(u_j).\sum \delta_i w_{ji}$ (2.22)

5) Cada unidade actualiza os seus pesos, $w_{ji}(new) = w_{ji}(old) + \Delta w_{ji}$ (2.23)

2.2.3. Rede neuronal para correcção de dados de profundidade

Para o sistema de localização proposto no subcapítulo seguinte, são necessárias coordenadas polares (distancia, ângulo) entre o robô e o alvo.

Esta rede fornece ao sistema de localização as distancia e os ângulos corrigidos dos alvos detectados. O valor da distância euclideana designa-se por *r* (*range*) e *v* é a distância máxima admissível, em metros. O ângulo entre o eixo yy e o centroide do alvo designa-se por *b*, *bearing,* expresso em radianos.

A função *nn_fun_polar* contém o cálculo da rede neuronal *forward feed.* Recebe os dados do sensor de profundidade em coordenadas cartesianas *xx*, *yy* (catetos) e calcula os valores de saída da posição em coordenadas polares, *r* (hipotenusa) e *b* (ângulo).

O cálculo das coordenadas em sistema polar é:

O primeiro passo converte as coordenadas cartesianas na sua equivalente polar.

$$r_{sensor} = \sqrt{Xbk_q^2 + Ybk_q^2} \tag{2.24}$$

$$b_{sensor} = \tan^{-1}\left(\frac{Xbk_q}{Ybk_q}\right) \tag{2.25}$$

Os vectores inNP_r e inNP_b representam a informação dos alvos capturados pelo sensor, função dos intervalos polares $r \in [0, v]$ e $b \in [0, \pi]$.

$$\text{inNP}_r = \left(\frac{r_{sensor}}{v}\right), r \in [0, v] \tag{2.26}$$

$$\text{inNP}_b = \left(\frac{b_{sensor}}{\pi}\right) + \frac{1}{2}, b \in [0, \pi] \tag{2.27}$$

A transformação dos valores de entrada do sensor inNP_r, inNP_b pela rede neuronal devolve os vectores de saída outNP_r e outNP_b.

Como estes valores estão dentro dos intervalos *r* e *b*, têm de ser convertidos novamente para fora do intervalo.

$$\text{dcp}_r = \text{outNP}_r \times v \tag{2.28}$$

$$\text{dcp}_b = \left(\text{outNP}_b - \frac{1}{2}\right) \times \pi \tag{2.29}$$

De modo a uniformizar os cálculos na lógica seguida anteriormente, cria-se a matriz $\mathbf{DCP}_{q,2}$ que contem os resultados da rede neuronal polar, função dos alvos detectados *q*.

$$\mathbf{DCP}_{q,2} = \begin{bmatrix} dcp_{1,1} & dcp_{1,2} \\ \vdots & \vdots \\ dcp_{q,r} & dcp_{q,b} \end{bmatrix} \tag{2.30}$$

O processo de transformação da função rede neuronal polar (*nn_fun_polar*) pode ser resumindo como $SD_q \xrightarrow{nn\_fun\_polar} dcp_q$ e fornece informação fundamental para o sistema de localização.

2.2.4. Cálculo do erro da rede

O erro associado à rede neuronal foi calculado comparando medições manuais de alvos no mapa com os dados obtidos pelo sensor. Desenvolveu-se um algoritmo semelhante ao anterior, dedicado apenas a este tipo de ensaios.

A matriz $\mathbf{M}_q$ contém as coordenadas cartesianas xm_q e ym_q dos alvos de referência no mapa, necessárias para a atribuição do padrão desejado na fase de treino.

$$r_{map} = \sqrt{xm_q^2 + ym_q^2} \tag{2.31}$$

$$b_{map} = \tan^{-1}\left(\frac{xm_q}{ym_q}\right) \tag{2.32}$$

Os vectores inNM_r e inNM_b representam a informação da posição dos alvos no mapa, função dos intervalos polares $r \in [0, v]$ e $b \in [0, \pi]$.

$$\text{inNM}_r = \left(\frac{r_{map}}{v}\right), r \in [0, v] \tag{2.33}$$

$$\text{inNM}_b = \left(\frac{b_{map}}{\pi}\right) + \frac{1}{2}, b \in [0, \pi] \tag{2.34}$$

Com esta informação utiliza-se o modelo de treino da rede neuronal, em matlab. Esta recebe os valores de entrada inNM_r, inNM_b e calcula a função de transformação da rede neuronal artificial.

2.3. Localização com o algoritmo Monte Carlo

A detecção de pelo menos dois *beacons* permite obter a postura do robô, por triangulação das posições dos *beacons*. Torna-se necessária a utilização de um localizador que permita a fusão sensorial das observações dos beacons com a odometria do veículo. Entre os diversos modelos matemáticos que existem para a localização, nesta dissertação escolheu-se o estimador MCL (*Monte Carlo Localization*) por ser robusto perante estimativas iniciais imprecisas e basear-se num mapa do mundo pré-definido.

A problemática da localização consiste na estimativa da postura do robô $Bel(x_t)$ em relação a um mapa conhecido do ambiente. O mapa contém a informação relativa às características[6] cujas posições devem ser detectadas pelo sensor, segundo uma distribuição da medição (z_t). Um dos objectivos para um sistema da localização é a capacidade para localização global, ou seja a localização do robô desde o instante *t=0*.

[6] Tradução do Inglês *features*. Estas características são os alvos (*beacons, landmarks*) no mapa, que servem de referência ao sistema de localização, com informação da posição [x, y].

A figura 2.5 ilustra o processo de localização e a incerteza da postura.

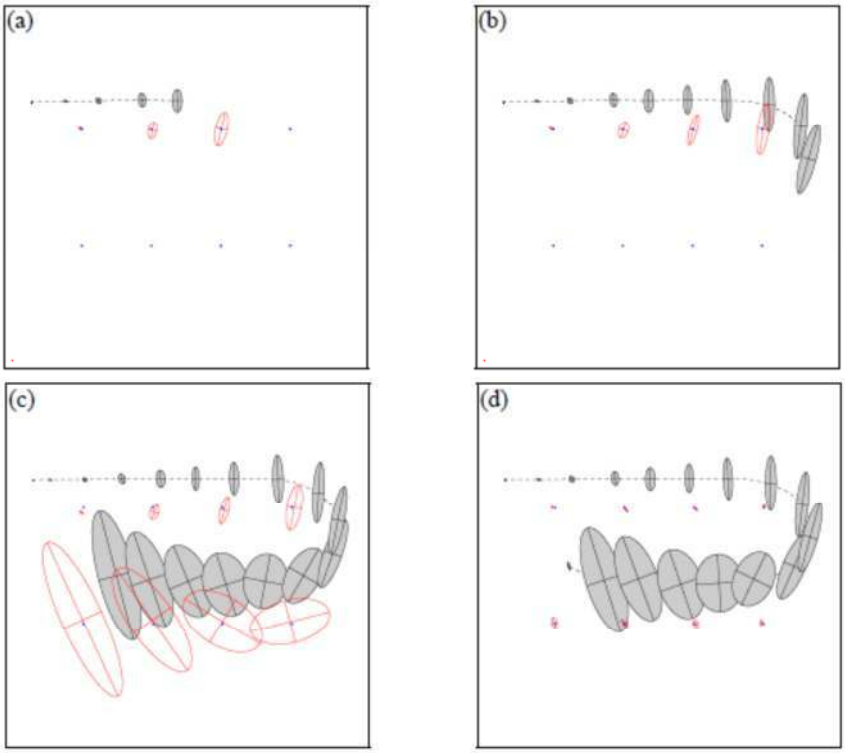


Figura 2.5 - De (a) a (c) a incerteza da posição do robô aumenta, assim como a incerteza relativa às referências encontradas. Em (d), o robô capta novamente a 1ª referência, logo a incerteza de *todas* as referências diminui, assim como diminui a incerteza referente ao seu estado. (*in* Thrun, *Probabilistic Robotics,* 2005, pp.251)

2.3.1. Estimador *Monte Carlo Localization*

Thrun, Bugard e Fox (2005) apresentam o estimador MCL para a localização do robô na sua obra de robótica probabilística. Ao contrário do EKF, o estimador MCL é capaz de estimar a postura do robô no mundo, a partir de uma postura inicial desconhecida, garantindo estimativas globalmente estáveis.

O MCL tem na sua génese a utilização de um filtro de partículas. Este filtro é uma aplicação não-paramétrica do filtro de Bayes, onde as posteriores $Bel(x_t)$ são representadas por *n* amostras aleatórias finitas dos estados (x_t), denominadas partículas. O estimador MCL adiciona partículas aleatórias segundo uma distribuição uniforme, onde cada partícula possui uma postura $[x, y, \theta]$ e à qual se atribui um factor de importância (peso) $w_t^{(i)}$ função das observações actuais do sensor (z_t).

O problema de localização global resolve-se com a aleatoriedade das partículas e com a versatilidade deste tipo de representação, que permite a descrição de distribuições *Belief* multimodais complexas.

Fazendo uma ilustração (figura 2.6) do estimador MCL por passos, tem-se:

1 – Fase de Predição, primeira estimativa da postura, função das acções de controlo actuais e observações anteriores.

2 – Fase de Actualização, onde a primeira estimativa da postura é actualizada em função do filtro de partículas. O filtro contém a distribuição das amostras cujo peso é determinado pelas observações actuais do sensor.

3 – Resultado, fase onde se determina qual a postura $[x, y, \theta]$ estimada do robô em relação as *landmarks* no mapa conhecido do mundo. Resulta do valor máximo da distribuição, ou seja aquele que produz menor incerteza na localização do robô.

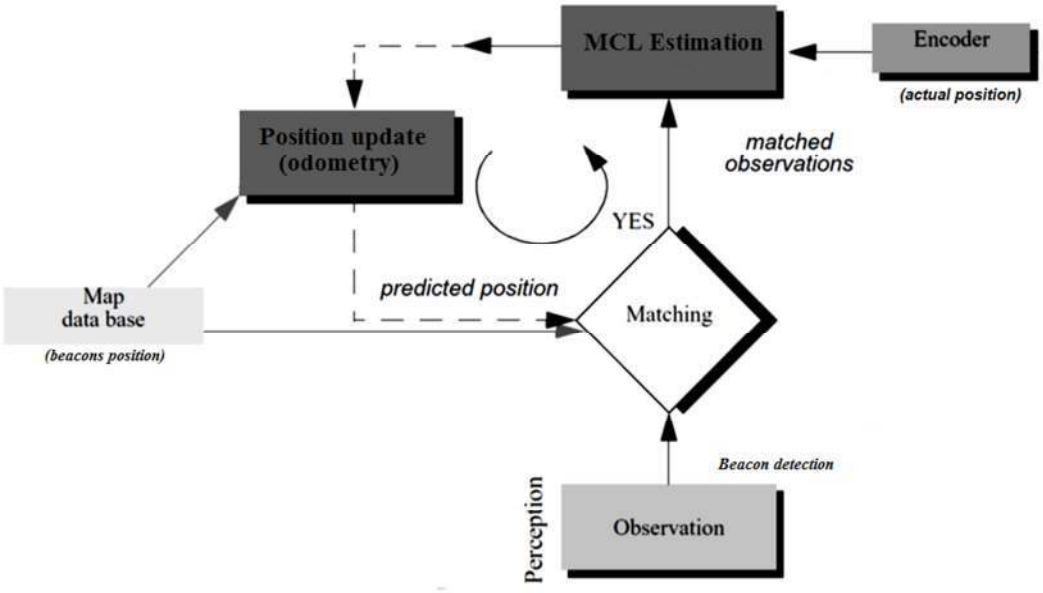


Figura 2.6 – Diagrama de funcionamento do estimador MCL,
***in* Siegwart *et al*, *Introduction to Autonomous Mobile Robots*, pp. 234**

O MCL representa o estado de probabilidade de localização do robô *Belief* através de um conjunto de partículas com pesos. Podem apontar-se como principais vantagens do modelo MCL:

1) Inclui toda a informação proveniente dos sensores e da dinâmica do movimento.
2) O alvo observado não implica nenhuma restrição à forma como a distribuição é processada.

3) O tempo computacional do algoritmo é focado nas regiões de maior densidade dos pesos, ou seja onde a probabilidade de localização do alvo é predominante.
4) O número de partículas ou tempos de amostragem do algoritmo podem ser ajustados em função das capacidades do equipamento disponível.
5) Resolve os problemas de localização global e rapto do robô.
6) Permite sistemas não-paramétricos com distribuições multimodais complexas.

2.3.1.1. Abordagem matemática ao estimador MCL

O MCL é um filtro de Bayes recursivo que estima a postura posterior do robô através de uma distribuição do conjunto de amostras fornecidas pelo filtro de partículas, função das observações recolhidas dos sensores e da comparação com referências do mapa conhecido. Assume a condição de Markov, ou seja os dados dos estados passado não condicionam o estado futuro, uma vez conhecido o estado presente.

O filtro de Bayes estima uma densidade probabilística no espaço de estado condicionada pelos dados da observação. Esta posterior define-se como "*Belief*" na literatura,

$$Bel(x_t) = p(x_t|d_{0..t}) \tag{2.35}$$

onde x_t é o estado no tempo *t* e $d_{0..t}$ a sequência de dados recolhidos ao longo do tempo. Os robôs móveis normalmente possuem sensores para a percepção (z_t), por exemplo para a observação do mundo. Possuem igualmente motores actuados por uma acção de controlo (u_t), para gerar movimento.

A posterior $Bel(x_t)$ integra a observação (z_t) e a odometria (u_t).

$$Bel(x_t) = p(x_t|z_t, u_{t-1}, z_{t-1}, u_{t-2}, \dots, z_0) \tag{2.36}$$

Assume-se que as observações e a odometria chegam com uma sequência alternada. A posterior é calculada com base na predição inicial e com a odometria $p(u_t)$, à qual se segue a fase de actualização com a distribuição das observações $p(z_t)$.
Não existe garantia de que na inicialização do robô haja informação sobre a sua postura inicial. Quando $Bel(x_{t=0}) = unknow$, está-se perante um problema de localização global, onde a postura inicial do robô é desconhecida. Para ultrapassar este obstáculo, o estimador MCL assume uma distribuição uniforme como estado inicial.

O MCL determina a posterior na fase de actualização. Aplicando a regra de Bayes à equação (2.36) obtém-se

$$Bel(x_t) = \frac{p(z_t|x_t,u_{t-1},\ldots,z_0).p(x_t|u_{t-1},\ldots,z_0)}{p(z_t|u_{t-1},\ldots,z_t)} \quad (2.37)$$

dado que o denominador é uma constante, para a instante, a equação fica

$$Bel(x_t) = \eta.p(z_t|x_t,u_{t-1},\ldots,z_0).p(x_t|u_{t-1},\ldots,z_0) \quad (2.38)$$

onde η é a constante de normalização,

$$\eta = p(z_t|u_{t-1},\ldots,z_0)^{-1} \quad (2.39)$$

Segundo o pressuposto de Markov, pode simplificar-se a equação (2.38),

$$Bel(x_t) = \eta.p(z_t|x_t).p(x_t|u_{t-1},\ldots,u_0) \quad (2.40)$$

Fazendo a integração da equação (2.40) para o tempo *t-1*, obtém-se a equação recursiva completa do filtro de Bayes.

$$Bel(x_t) = \eta.p(z_t|x_t)\int p(x_t|x_{t-1},u_{t-1}).Bel(x_{t-1})\,dx_{t-1} \quad (2.41)$$

A equação 2.41 é a base do estimador MCL. Define um estimador recursivo para o estado de sistemas com observações parciais.

Segue-se a aplicação do necessário modelo do movimento das observações.

2.3.1.2. Modelo probabilístico do movimento e das observações

O modelo de movimento $p(x'|x,u)$ é a generalização probabilística da cinemática do robô. Ambas as posturas *x* (presente) e *x'*(futuro) são representações de 3 variáveis, duas cartesianas (x e y) e uma terceira θ correspondente ao ângulo da orientação. O valor da acção *u* representa a odometria ou uma lei de controlo, ambas responsáveis por uma mudança na postura do robô. As equações da cinemática descrevem fisicamente o tipo de robô (linear, diferencial) ou seja, devolvem a postura expectável em *x'* num ambiente ideal sem ruído, após ser actuado por *u*. No mundo real é necessário considerar o erro aleatório associado ao movimento, logo a postura *x'* é não determinística. O modelo de movimento probabilístico $p(x'|x,u)$ descreve a densidade da postura futura perante possíveis cenários de *x e u*. O ruído gerado pelo movimento é

escrito como uma distribuição probabilística de média centrada, para as componentes de rotação e translação.

O modelo da observação utiliza esta informação para escrever uma distribuição probabilística da percepção $p(z|x)$ que se decompõe em 3 partes:

1) Cálculo do valor que um sensor sem ruído deveria gerar.
2) Cálculo da co-variância do modelo de ruído do sensor.
3) Fusão das múltiplas leituras oriundas da pesquisa do sensor numa única variável de densidade.

Assume-se então a postura do robô como x e a leitura individual do raio emitido pelo sensor z_i, com α_i o seu ângulo (*bearing*). A leitura ideal sem ruído do sensor designa-se por $g(x, \alpha_i)$.

Para a localização, providencia-se ao robô um mapa cartesiano do ambiente de navegação. Pode então definir-se a probabilidade da observação z_i, função da postura x como

$$p(z_i|x) = p(z_i|g(x, \alpha_i)) \tag{2.42}$$

A equação (2.42) é uma densidade centrada em $g(x, \alpha_i)$ que representa a distância correcta medida.

A figura 2.7 demonstra uma experiencia de Thurman *et al*, com um sensor *rangefinder*, para ensaiar a densidade da percepção $p(z|x)$. O pico corresponde à distância $g(x, \alpha_i)$ que o sensor deveria captar num ambiente ideal sem ruído.

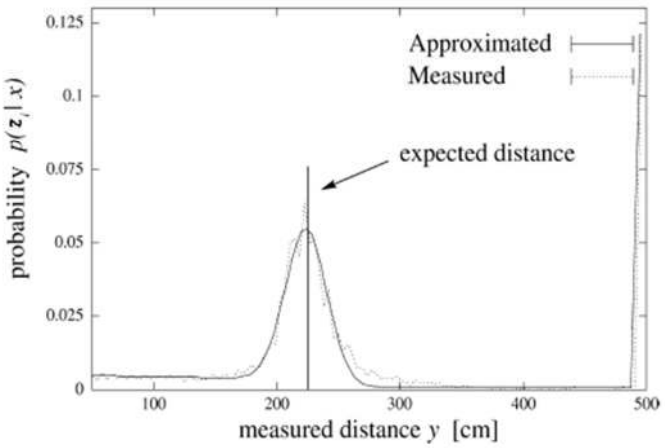


Figura 2.7 – Exemplo de uma distribuição das leituras z_i do sensor (*in* Thrun *et al*, *Robust Monte Carlo localization for mobile robots*, pp.107)

Segue-se a integração multiplicativa das densidades individuais $p(z_i|x)$ e a aplicação da condição de Markov (independência do estado presente dos passados ou futuros) o que resulta na equação

$$p(z|x) = \prod_i p(z_i|x). \quad (2.43)$$

Definidos os modelos probabilísticos do movimento e das observações, apresenta-se na secção seguinte a sua integração no filtro de partículas.

2.3.1.3. Sumário MCL

- O algoritmo de localização MCL representa a postura do posterior com recurso a partículas. O rácio entre o número de partículas utilizadas e o respectivo custo computacional está amplamente estudado.
- O MCL consegue localizar globalmente o robô e a adição de partículas aleatórias no filtro resolve o problema do rapto do robô.
- Ultrapassa as limitações do estimador EKF, ao fazer a representação de distribuições *Belief* multimodais complexas, ao não assumir condições paramétricas no cálculo da densidade posterior.

2.4. Ensaios do sistema de localização

2.4.1. Ensaios do detector de *beacons*

O algoritmo *detecta_beacons* contém o código de detecção de *beacons* baseado na visão RGB, com correspondente captura de profundidade e cálculo das distâncias ao alvo, traduzidas em variáveis com valores reais, necessários para a localização do robô (Figura 2.8). Verificaram-se diferenças entre os resultados captados pelo sensor e as reais posições dos alvos no local de ensaios, pelo que foi implementada uma RNA para a sua correcção. Efectuaram-se testes de *ground truth* para determinar se a implementação da rede melhorou os resultados obtidos.

O objectivo dos ensaios passa pela determinação de um envelope de segurança relativamente aos valores de profundidade capturados pela Kinect, nos eixos cartesianos *xx*, *yy*, *zz* de modo a que o processo de navegação do robô possa reflectir com maior exactidão o mundo que o rodeia.

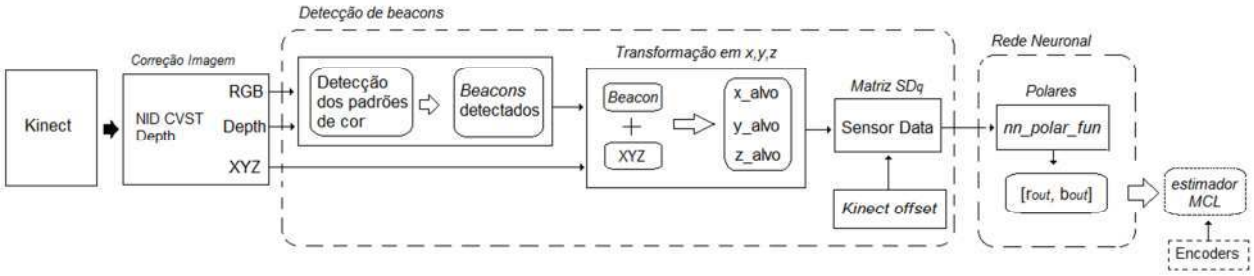


Figura 2.8 – Diagrama de funcionamento do algoritmo de detecção de *beacons*.

Estrutura de cálculo

1) Identificação dos alvos através de padrões RGB.
2) Transformação da informação, através da conversão do centro (*pixel*) de cada alvo em profundidade (*xyz*).
3) Condensação dos resultados numa matriz de dimensão normalizada, de modo a que os dados seja facilmente acessíveis por diferentes subprogramas numa lógica comum.
4) Filtragem dos resultados por uma rede neuronal.

2.4.2. Ensaios efectuados

Foram feitos 250 ensaios para determinar o grau de qualidade da informação recolhida com o algoritmo de detecção de *beacons*. Estes decorreram em 2 fases distintas, essencialmente antes da aplicação da rede neuronal artificial, onde se fizeram 134 ensaios e depois da aplicação da correcção da RNA, com 148 ensaios de verificação.

Foi ainda feita uma fase inicial de testes, para afinação de diversos parâmetros do algoritmo e onde se verificou a necessidade de corrigir a discrepância nos dados entre a imagem RGB e a imagem de profundidade. Embora não sendo facilmente visível, existe uma diferença entre a imagem RGB_{ij} e $Depth_{ij}$, para idênticas posições dos pixéis *ij*. A figura 2.9 ilustra este problema, onde os pontos que representam a detecção deveriam, em teoria, apresentar as mesmas posições *ij*. O módulo NID_CVST_Depth foi aplicado para resolver este problema.

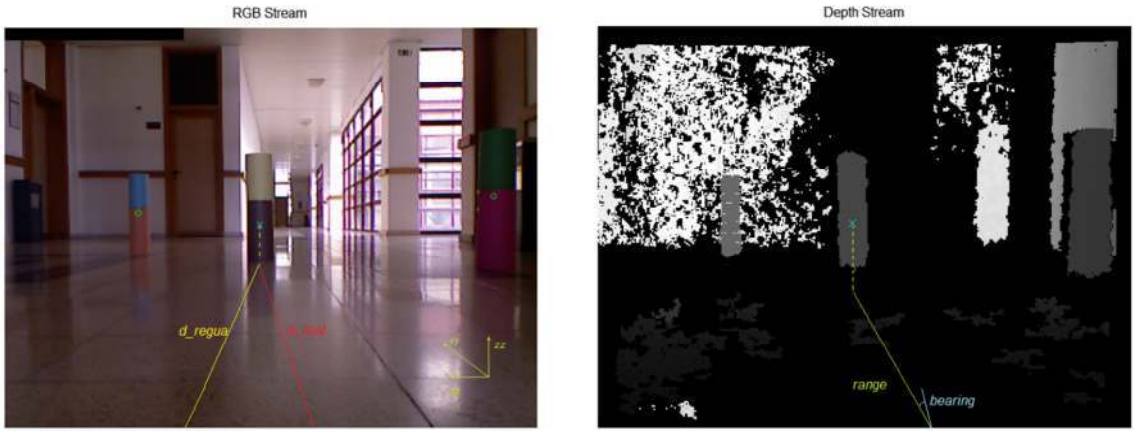


Figura 2.9 – Exemplo de um ensaio na fase inicial de testes. (a) Imagem RGB e (b) Imagem de profundidade

2.4.2.1. Transformação das imagens da Kinect

O módulo *NID_CVST_Depth* faz a fusão das imagens RGB e *Depth* da Kinect numa matriz RGB-D. É importante referir que o sensor de infravermelhos tem limitações físicas para a captura de luz reflectida, funcionando no intervalo [0,50 – 9,00] metros e dependendo especificamente da textura do material alvo. Se o sensor for apontado para uma janela de vidro, o sinal de luz é reflectido em múltiplas direcções que não o sensor, logo o resultado não é um valor numérico, mas sim um vector vazio (*Not a Number* no Matlab). A manipulação inicial de dados conclui-se com a troca das coordenadas *yy* e *zz*, já que o módulo exporta *zz* como profundidade, no entanto na literatura *zz* indica a altura.

2.4.2.2. 1ª Fase - Ensaios antes da aplicação da RNA

Para cada alvo detectado é lida uma posição cartesiana, que deve ser convertida em polar, pois o algoritmo MCL necessita da distância e do ângulo relativos ao robô. Os ensaios feitos na fase inicial demostraram diferenças entre a posição produzida pelo sensor e a real posição do alvo, pelo que se decidiu aplicar uma RNA para correcção. Definido o objectivo do ensaio, criou-se um modelo para evitar que possíveis desvios de ângulo da Kinect influenciassem o resultado (Figura 2.10).

Posicionou-se o robô com a Kinect na posição [2.00, 0.00] e o alvo de referência fixo na posição [2.00, 4.30]. Notar que a Kinect devolve valores relativos à sua posição, logo a correcta leitura teórica do alvo fixo deve devolver [0.00, 4.30]m directamente do sensor, ao qual posteriormente se acrescenta o cálculo da posição relativa ao ponto [0, 0] no mapa, passando o resultado a [2.00, 4.30]m.

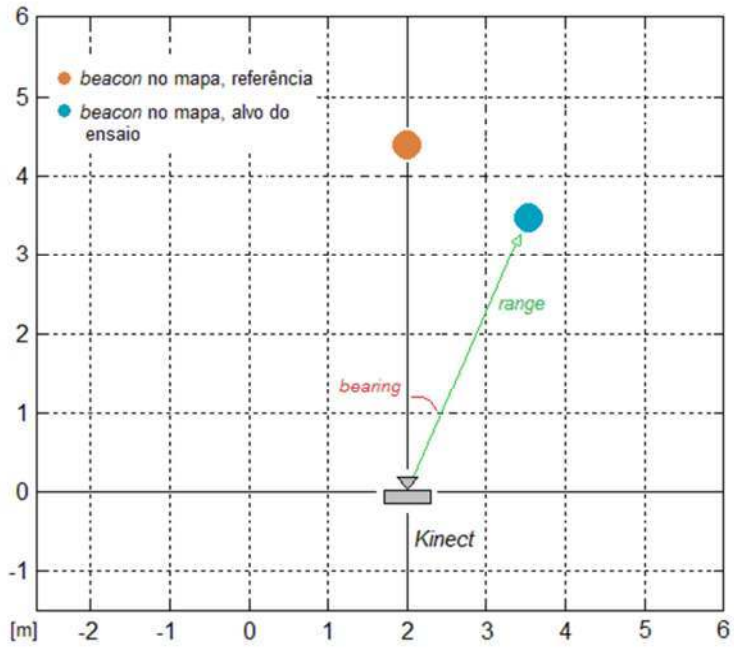


Figura 2.10 – Exemplo da 1ª fase de ensaios da detecção de *beacons~*

Com ambos sensor e alvo fixos, garante-se que a orientação da kinect está alinhada com o eixo xx. Posteriormente é colocado um alvo variável em diferentes posições na sala. É fundamental salientar que a Kinect tem um potencial de detecção de profundidade de aproximadamente 9.0m, no entanto limitou-se a distância dos alvos a um máximo de 5.0m. Verificou-se que para distâncias superiores é possível detectar os mesmos, mas a probabilidade das falsas detecções sobe consideravelmente.

A tabela 2.1 exemplifica uma experiência onde se obtém a leitura das posições do alvo de referência e alvo variável, em metros. As colunas do alvo fixo contêm os valores medidos fisicamente na sala com recurso a fita métrica, relativamente ao referencial do mundo. O alvo variável segue a mesma lógica, onde cada experiencia é feita para uma nova posição. O algoritmo devolve os valores do alvo variável detectado em coordenadas cartesianas [x, y].

Tabela 2.1 – Amostra de resultados dos ensaios com a Kinect, sem rede neuronal

Alvo fixo - referência				Alvo variável			
Posição medida		Posição estimada		Posição medida		Posição estimada	
x,ref	*y,ref*	*Xref*	*Yref*	*x,var*	*y,var*	*XX,var*	*YY,var*
2,000	4,300	0,007	4,270	1,000	4,400	-0,849	4,804
2,000	4,300	0,007	4,270	0,500	4,400	-1,409	5,013
2,000	4,300	0,007	4,323	0,500	4,200	-1,361	4,674

No total realizaram-se 134 experiências, com alvos em diferentes posições dentro dos intervalos $xx \in [-4, 4]$ e $yy \in [0.5, 5]$ que representa o espaço físico abrangido pelo sensor de visão. Conclui-se, através destes ensaios, que a média do erro entre as observações do sensor e as reais medidas no local é de: $\bar{\varepsilon}_{xx} = 14{,}36\ \%$ e $\bar{\varepsilon}_{yy} = 6{,}13\ \%$ em coordenadas cartesianas, e em coordenadas polares de: $\bar{\varepsilon}_{range} = 4{,}86\ \%$ e $\bar{\varepsilon}_{bearing} = 17{,}21\ \%$ .Nesta experiência não foi ainda aplicada a rede neuronal.

2.4.2.3. 2ª Fase - Ensaios *ground truth* com adição da RNA

A segunda fase de ensaios foi dedicada à correcção dos dados da Kinect através da rede neuronal. Para optimizar o tempo de laboratório e garantir a veracidade dos dados, criou-se um programa de testes semelhante ao conduzido na 1º fase, mas com a diferença de devolver simultaneamente as posições do alvo de antes e depois da rede neuronal.

O cálculo da rede neuronal foi feito com a base de dados recolhida da 1ª fase de experiências no laboratório. Com este conjunto é possível criar no Matlab um código que normalize os dados e garanta que os limites fiquem contidos entre [0,1] de modo a que a rede neuronal possa ser processada.

Efectivamente considerou-se uma profundidade de 5.00m como o limite dos valores a serem processados pela rede neuronal. Ensaios com entradas fora deste intervalo (*dist* > 5.00m) fazem com que a normalização ultrapasse o intervalo $x = [0, 1]$, $y = [0, 1]$ e os resultados demostraram constantemente uma disparidade com a realidade.

O primeiro código denomina-se *nn_test_polar* e recolhe a informação das 134 experiências decorridas na 1ª fase, gravadas no ficheiro *ground_truth_network*. Os valores de entrada são distâncias *r=range* e ângulos *b=bearing*. Para facilitar o código, aos valores de entrada chamam-se *in* e os de saída *out*.

2.4.2.4. Função *nn_test_polar*

Ao correr esta função vai criar-se uma nova rede neuronal artificial com 1 camada escondida, neste exemplo *robot_polar_net.net1* (figura 2.11).

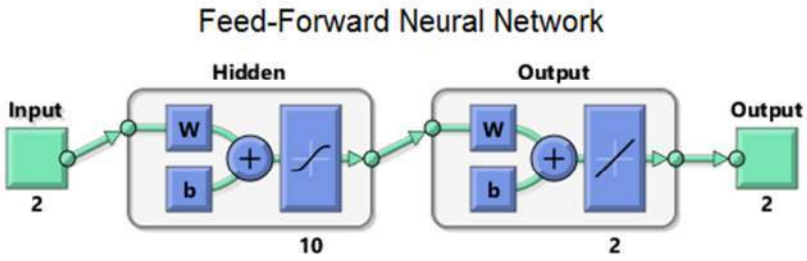


Figura 2.11 - Modelo da rede neuronal artificial

Esta rede é aplicada no final do algoritmo de detecção de *beacons*, e devolve 2 pares de saídas, uma cartesiana, $sensor_{cart,NN} = [xx_{out,NN}, yy_{out,NN}]$ e uma polar, $sensor_{polar,NN} = [range_{out,NN}, bearing_{out,NN}]$. A saída em coordenadas cartesianas é um simples cálculo matemático e apenas serve para verificar no laboratório os valores obtidos, por ser mais simples medir coordenadas cartesianas. A tabela 2.2 contém um excerto dos resultados obtidos. No total foram feitos 134 ensaios para treino da rede e 148 ensaios de *ground truth* para verificação do grau de confiança da rede neuronal.

Tabela 2.2 – Ensaio com rede neuronal *nn_polar_fun*

Alvo fixo - referencia				Alvo variável					
Medido		Estimado		Medido		Estimado, cartesiano		Estimado, polar	
x,ref	*y,ref*	*x,ref*	*y,ref*	*x,var*	*y,var*	*XX,out,nn*	*YY,out,nn*	*r,outNN*	*b,outNN*
1,500	3,000	-0,012	2,977	0,400	4,000	-1,116	3,933	4,036	-0,261
1,500	3,000	-0,006	2,978	0,550	4,000	-0,978	3,978	4,057	-0,234
1,500	3,000	-0,006	2,978	0,650	4,000	-0,869	3,973	4,044	-0,214
1,500	3,000	-0,006	2,951	0,800	4,000	-0,703	3,921	3,984	-0,183

O alvo variável percorre uma área que está contida no intervalo $x = [-4, 4]$ e $y = [0, 5]$ e dentro do do campo de visão físico do sensor. Conclui-se nesta 2ª fase, de 146 ensaios de *ground truth*, que a média do erro de leitura entre o alvo de referência e o alvo variável, com a aplicação da rede neuronal, é $\bar{\varepsilon}_{xx,NN} = 9{,}23\%$ e $\bar{\bar{\varepsilon}}_{yy,NN} = 3{,}33\%$ para coordenadas cartesianas A média do erro com aplicação da RNA em coordenadas polares é $\bar{\varepsilon}_{range} = 2{,}06\%$ e $\bar{\varepsilon}_{bearing} = 9{,}33\%$.

2.4.2.5. Análise estatística dos resultados do ensaio de *ground truth.*

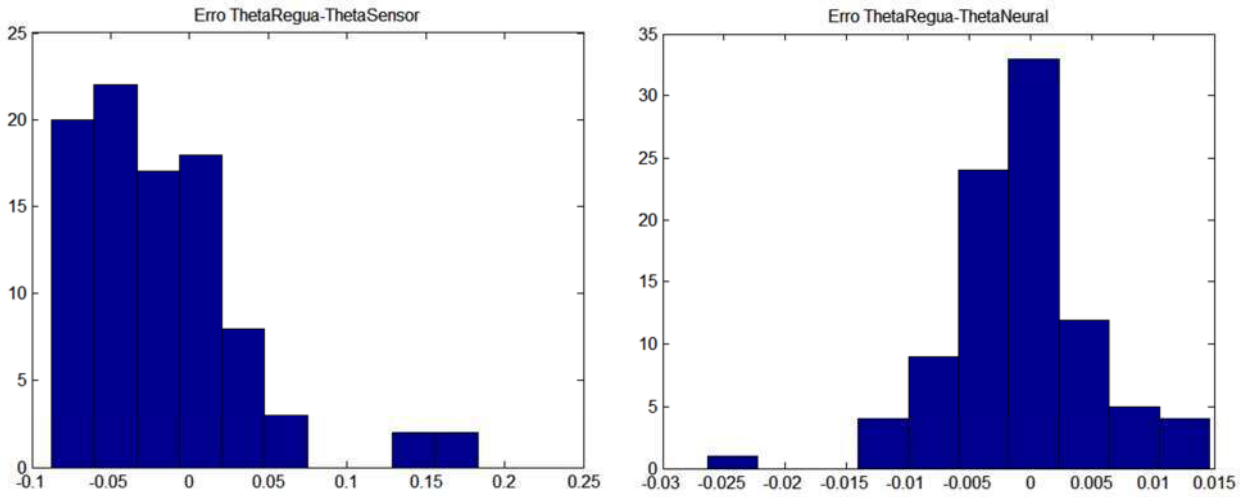


Figura 2.12 – a) Distribuição do erro do ângulo antes da rede, b) corrigido.

O objectivo final da rede e reduzir o erro associado ao fazer uma distribuição gaussiana centrada em zero com a menor variância possível. A figura 2.12 apresenta o resumo da alteração da distribuição do erro do ângulo (*bearing*) antes e depois da aplicação da rede neuronal *nn_polar_fun*. Esta diminuição do erro é fundamental para as leituras do ângulo, pois é através destas que o estimador MCL faz os cálculos.

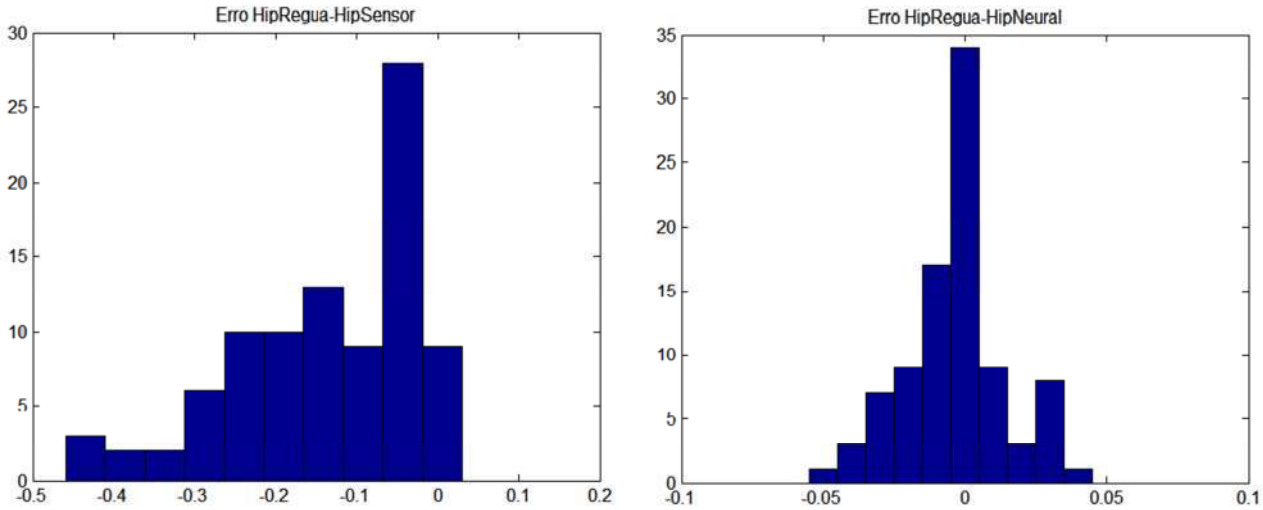


Figura 2.13 – a) Distribuição do erro da distância antes da rede, b) corrigido.

A figura 2.13 apresenta o resultado para os dados da distância (*range*) antes e depois da correcção com a rede neuronal. Tal como na experiencia anterior, o resultado é a centralização do erro em zero com uma distribuição gaussiana e o menor valor possível

para a variância. O resultado das experiencias permite obter uma co-variância para a distancia $\sigma^2 = 0.1486$ e outra para o ângulo $\psi^2 = 0{,}0564$.

Estas co-variâncias são fundamentais para fornecer ao modelo de localização MCL uma referencia para o estado *Belief* das leituras dos sensores.

2.4.3. Conclusões sobre os ensaios efectuados

A tabela 2.3 faz o resumo dos ensaios realizados na 1ª fase e na 2ª fase. Como se pode observar, as correcções resultantes da aplicação da RNA têm evidentes melhorias, tanto em coordenadas cartesianas como em coordenadas polares.

Tabela 2.3 – Comparação de resultados da aplicação da rede neuronal artificial

	Cartesianas		Polares			
Média do erro	*XX*	*YY*	*Range*	*Bearing*	Algoritmo RNA	Nº ensaios
Antes da RNA	14,36%	6,13%	4,86%	17,21%	*nn_fun*	134
Depois da RNA	9,23%	3,33%	2,06%	9,33%	*nn_polar_fun*	148
Diferença	-5,13%	-2,80%	-2,80%	-7,88%	Σ	**250**
Redução % do erro	**35,77%**	**45,66%**	**57,54%**	**45,80%**		

A aplicação da RNA conduz a uma redução do erro, nas coordenadas cartesianas e polares. Observe-se a distância polar (range) com um erro inicial de 4,86%, que, após aplicação a rede, se reduz para 2,06%. Isto implica uma redução percentual relativa de 57%. Para o ângulo, observa-se uma redução do erro em 45%. Assim, fica valida a utilização da rede neuronal artificial no algoritmo de detecção. Os dados dos ensaios da rede neuronal estão apresentados na secção "anexos".

Capitulo III

3. Detecção de Humanos

3.1. Detecção de Humanos com câmara RGB-D

Um dos objectivos propostos na presente dissertação é o de conferir ao robô a capacidade de detectar pessoas no seu campo de visão. Deve ainda evitar colisões e garantir uma distância física (e psicológica) segura para as pessoas.

A detecção de pessoas é feita com um algoritmo que analisa as imagens da câmara RGB-D. Essencialmente, o robô deve ter a capacidade de (i) detectar uma ou mais pessoas na imagem, (ii) determinar a distância às pessoas encontradas e (iii) alterar a trajectória de navegação inicial para uma trajectória segura.

O algoritmo proposto para a navegação na presença de Humanos, esquematizado na figura 3.1, consiste nos seguintes passos:

(1) A detecção positiva da pessoa, através da pesquisa de uma característica na imagem. Este passo implica a utilização de um algoritmo especializado na detecção de Humanos com sistemas de visão.

(2) Segue-se o programa (idêntico ao detector de *beacons*) que faz a fusão das imagens RGB e *Depth* e extrai as coordenadas cartesianas e polares entre o robô e a pessoa. Dado que o sensor RGB-D é o mesmo, existe a mesma necessidade de corrigir as imagens através da rede neuronal artificial. Com esta informação o robô acrescenta a posição da pessoa no seu mapa do mundo.

(3) Alteração da trajectória calculada pelo sistema de navegação, em função da posição da pessoa no mundo e do círculo de segurança definido em redor desta.

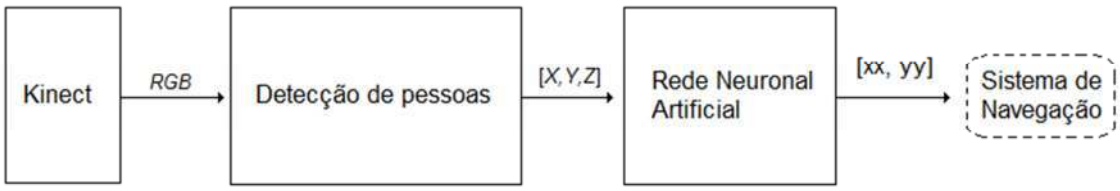


Figura 3.1 – Fluxo de informação relativa à detecção de Humanos

Existem diversas obras no campo da detecção de Humanos com sistemas de visão. Para a presente dissertação propõe-se a utilização do algoritmo *Vision.PeopleDetector* disponível no pacote *Computer Vision System* do Matlab, desenvolvido com base no descritor de imagens HOG (*Histogram of Oriented Gradients*) proposto por Dalal e Triggs (2005).

O HOG é um descritor de características (*feature descriptor*) para imagens estáticas. Este procura na imagem características que, no seu conjunto, formem padrões que se assemelhem aos padrões do modelo de treino do classificador SVM (*Support Vector Machine*). Neste caso, o padrão de treino é a silhueta humana, na posição vertical e desobstruída, incluindo poses paradas ou em movimento. É importante referir que este algoritmo não faz uso da profundidade, fazendo a pesquisa de características na imagem RGB e devolvendo a *percentagem* de certeza de detecção.

No subcapítulo modelo matemático faz-se uma abordagem mais teórica às características HOG.

3.1.1. Método proposto

Existe alguma semelhança entre a detecção de pessoas e a detecção de *beacons*, já que ambas são feitas recorrendo ao sensor Kinect e têm como objectivo detectar e devolver a posição tridimensional relativamente ao robô. Assim sendo, utilizam-se as mesmas variáveis que contêm a informação RGB e profundidade, corrigidas pelo módulo *nid_cvst_depth*. Para uma análise mais detalhada deste módulo, consultar a secção 2.1.1.

O processo de detecção de pessoas inicia-se com a entrada de informação RGB da matriz **IMG** no algoritmo *vision.PeolpleDetector*. Este algoritmo do Matlab divide-se em três partes, (i) o descritor de características HOG, (ii) o classificador SVM e (iii) matriz de resultados.

O descritor HOG (i) pesquisa a imagem **IMG** por características contidas numa janela. Dentro desta, a pesquisa compara blocos HOG com os blocos memorizados no algoritmo, que representam a silhueta humana, na posição vertical e desobstruída. A dimensão desta janela de pesquisa, nas opções *upright_128x64* para melhor resolução ou *upright_96x48* para melhor performance, define o contexto em redor da pessoa, ou seja a margem em pixéis, para o topo, base, esquerda e direita da pessoa. O descritor

HOG extrai *n* características que possivelmente indicam a presença de uma pessoa. O classificador treinado SVM (ii) processa esta informação e calcula uma resposta na forma de *pessoa* ou *não-pessoa*.

A variável vPD contém o resultado do código de Matlab do algoritmo *vision.PeopleDetector*, incluindo o modelo de treino utilizado *upright_128x64,* assim como diversos parâmetros para afinação dos resultados. A detecção de características na imagem $\mathbf{Img}_{ij}$ é feita com o comando *step* e gravada na matriz **BB**. O índice *p* representa o número de pessoas detectadas na imagem. A matriz de resultados $\mathbf{BB}_{p,4}$ contém a posição, em pixéis, de cada pessoa na imagem, onde uma pessoa é representada através de uma caixa ao redor da sua silhueta. Ou seja, $p = 1,..,np,\ np = número\ de\ pessoas$, onde *x* e *y* representam o canto superior esquerdo da caixa de envolvência, *w* (*width*) a largura da caixa e *h* (*height*) a altura da mesma. Se não houver detecções de pessoas na imagem $\mathbf{IMG}_{ij}$, resulta um vector vazio.

$$\mathbf{BB}_{p,4} = \begin{bmatrix} x_{1,1} & y_{1,2} & w_{1,3} & h_{1,4} \\ \vdots & \vdots & \vdots & \vdots \\ x_{p,1} & y_{p,2} & w_{p,3} & h_{p,4} \end{bmatrix},\ x, w \in [1,..,m],\ \ y, h \in \{1,..,n\}\, pixeis \qquad (3.1)$$

O algoritmo *vision.PeopleDetector* inclui a sub-rotina *scores*, que devolve o valor escalar de confiança $s_{p,1}$ para a matriz de resultados $\mathbf{BB}_{p,4}$, função de *p*. Este intervalo pode ser definido individualmente para cada detecção ou resultado da distribuição estatística do conjunto de detecções, se $p \geq 2$. O intervalo faz parte do conjunto de parâmetros para afinação de resultados. Este escalar é visível no topo de cada caixa de detecção.

O algoritmo de detecção pode ser afinado em função dos resultados obtidos na matriz $\mathbf{BB}_{p,4}$. Os parâmetros $size_{max}, size_{min}$ definem a dimensão máxima e mínima da janela de detecção do descritor HOG. Notar que a alteração dos valores pré-definidos pode resultar num aumento da área de detecção com correspondente aumento do número de falsas detecções ou a diminuição, onde não ocorre nenhuma detecção. O parâmetro *scale_factor* define a sensibilidade da resolução da detecção entre os intervalos $size_{max}$ e $size_{min}$. Normalmente definido pelo escalar 1.05. A redução deste escalar aumenta a sensibilidade, no entanto aumenta também o tempo de computação necessário para a detecção.

Na figura 3.2 pode observar-se o resultado da detecção de duas pessoas. Notar que as caixas de envolvência têm diferentes dimensões, assim como diferentes resultados *score*.

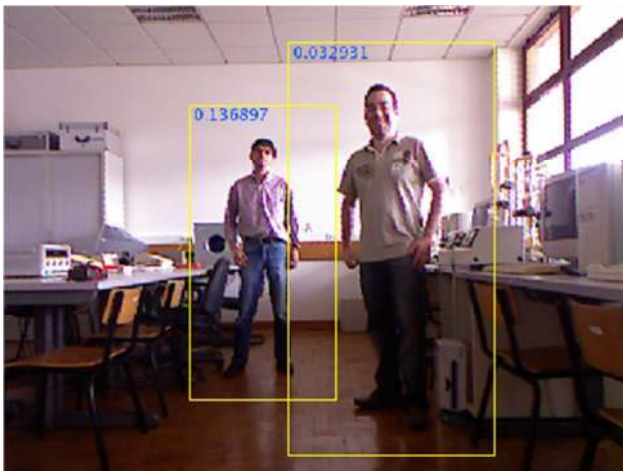


Figura 3.2 – Detecção de múltiplas pessoas, Laboratório de robótica, ISEL-ADEM

A matriz de resultados $\mathbf{BB}_{p,4}$ identifica a posição da pessoa detectada na imagem. Para conhecer a posição tridimensional da pessoa, é necessário obter o centroide da caixa, que efectivamente coincida com o centro físico da pessoa.

A matriz $\mathbf{PC}_{p,2}$ (*people centroid*) guarda a informação dos centroides, em pixéis, calculados por:

$$xc_{p,1} = \frac{w_{p,3}}{2} + x_{p,1},\ xc_{p,1} \in [m,n] \tag{3.2}$$

$$yc_{p,2} = \frac{h_{p,4}}{2} + y_{p,2},\ yc_{p,2} \in [m,n] \tag{3.3}$$

$$\mathbf{PC}_p = \begin{bmatrix} xc_{1,1} & yc_{1,2} \\ \vdots & \vdots \\ xc_{p,1} & yc_{p,2} \end{bmatrix} \tag{3.4}$$

A figura 3.3 ilustra um ensaio para a detecção de pessoas, na sala de trabalhos da Associação de Estudantes do ISEL. As duas imagens, RGB e *Depth* são captadas em simultâneo pela Kinect. A aplicação do detector resulta na imagem "*Detection*" onde é visível a detecção de uma pessoa. Notar que a detecção ocorre num ambiente com pavimento altamente reflectivo e contra luz. Mesmo nesta situação o algoritmo demonstra ser robusto, ao detectar a pessoa em posição vertical, desobstruída. Neste

exemplo, fica visível uma caixa de diálogo com posições $x = xc_p,\ y = yc_p$, em pixéis, para verificação do centroide $[xc, yc]$ calculado pela matriz $\mathbf{PC}_p$.

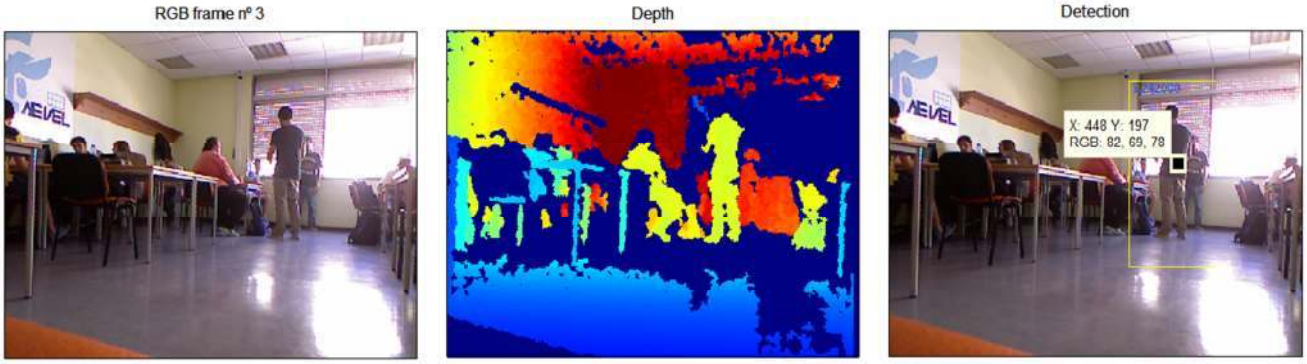


Figura 3.3 – (a) Imagem RGB, (b) gradiente de profundidades, (c) imagem com resultado da detecção.

3.1.1.1. Transformação da informação

A transformação da posição *ij* na imagem de cada pessoa detectada em coordenadas cartesianas no espaço implica a indexação da matriz $\mathbf{XYZ}_{ijk},\ k = 1,..,3$ com as coordenadas *i* e *j* dos centroides. Resultam 3 escalares com a informação da profundidade *xx*, *yy* e *zz*, para cada pessoa detectada.

$$xpc = \mathbf{XYZ}_{i,j,1}(\mathbf{PC}_p), i = r, j = s, k = 1 \tag{3.5}$$

$$ypc = \mathbf{XYZ}_{i,j,2}(\mathbf{PC}_p), i = r, j = s, k = 2 \tag{3.6}$$

$$zpc = \mathbf{XYZ}_{i,j,3}(\mathbf{PC}_p), i = r, j = s, k = 3 \tag{3.7}$$

3.1.1.2. Resultado da detecção de pessoas

O vector de posição **pd**, função das pessoas *p* detectadas, $\mathbf{pd_p} = [xpc_p,\ ypc_p]$ guarda a informação da posição da pessoa *p* no espaço, nas dimensões *xx* e *yy*. A matriz $\mathbf{PD}_p$ guarda as detecções de 1…*p*.

$$\mathbf{PD}_p = \begin{bmatrix} xpc_1 & ypc_1 \\ \vdots & \vdots \\ ypc_p & xpc_p \end{bmatrix} \tag{3.8}$$

A variável *kinect offset* permite o posicionamento excêntrico da câmara $ko = [ko_x,\ ko_y]$ relativamente aos eixos de inércia do robô. É necessário, pois fisicamente pode não ser possível alinhar o sensor com estes eixos. O vector de posição das pessoas,

afectado do excêntrico da câmara designa-se por *people kinect.* A matriz com o conjunto 1…p designa-se por $\mathbf{PK}_p$.

$$\mathbf{PK}_p = \left[xpc_p + ko_x\ ,\ ypc_p + ko_y\right] = [xpk_p,\ ypk_p] \tag{3.9}$$

A matriz que contem a informação de todos os vectores de posição designa-se por $\mathbf{SP}_p$ (*sensor people*).

$$\mathbf{SP}_p = \begin{bmatrix} xpk_1 & ypk_1 \\ \vdots & \vdots \\ xpk_p & ypk_p \end{bmatrix}, p = 1,..,np \tag{3.10}$$

Esta matriz contém um erro de posição, idêntico ao abordado na correcção de alvos na secção 2.2.3.1. Assim sendo, utiliza-se a mesma rede neuronal artificial, para corrigir estas posições.

Uma variante com informação polar desta matriz $\mathbf{SP}_{p,polar}$ é igualmente calculada, para situações de programação que necessitem dos dados em sistema polar, com (R) distância em metros e (β) o ângulo em radianos, entre o centro de inércia do robô e a pessoa detectada.

$$\mathbf{SP}_{p,polar} = \begin{bmatrix} Rpk_1 & \beta pk_1 \\ \vdots & \vdots \\ Rpk_p & \beta pk_p \end{bmatrix} \tag{3.11}$$

A figura 3.4 apresenta esquematicamente o modelo proposto.

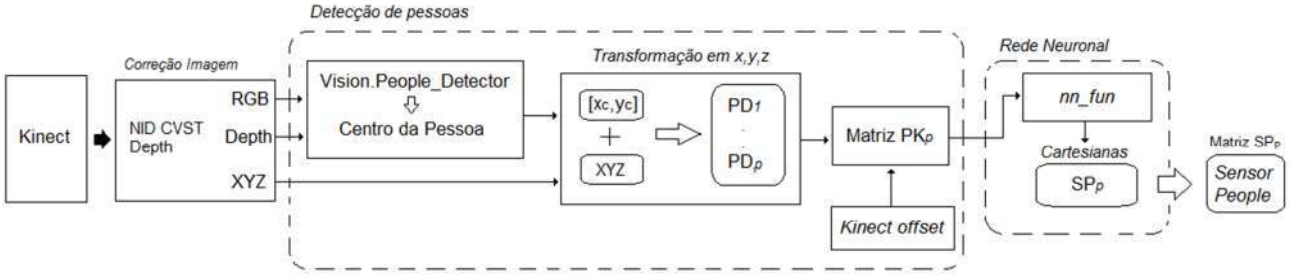


Figura 3.4 – Diagrama do detector de humanos proposto.

3.2. Modelo matemático para a detecção de Humanos

A detecção de humanos numa imagem é um processo complexo que inclui uma multitude de poses das pessoas, tanto paradas como em movimento, possivelmente num cenário (estático ou dinâmico) carregado de objectos e sob condições de iluminação variáveis. Na presente dissertação propõe-se o algoritmo de Detecção de Humanos HOG de Dalal e Triggs (2005). Este faz a detecção de pessoas na posição vertical e desobstruída. É composto por um descritor de características HOG e por um classificador de imagens linear SVM.

3.2.1. Descrição do método

A aparência de um objecto (ex.: pessoa) pode ser caracterizada através da distribuição de gradientes de intensidade locais (direcção da fronteira) mesmo que não exista conhecimento preciso da posição da fronteira ou do gradiente correspondente. Esta caracterização é feita através da divisão da imagem em pequenas regiões espaciais, ou células. Cada célula acumula histogramas de gradientes das direcções unidimensionais (orientação da fronteira) de todos os pixéis nela contidos. A junção destes histogramas locais produz o histograma dos gradientes de orientação da célula.

A invariância perante diferentes condições de iluminação (luz, sombra) é obtida através da normalização da imagem em regiões de maior dimensão (blocos) e a informação acumulada em histogramas de normalização. O resultado é aplicado na normalização das células dentro do bloco. Para cada imagem existem múltiplos blocos normalizados, denominados descritores HOG locais.

O método HOG emparelha todos estes blocos numa malha densa de blocos sobrepostos (janela) à qual é aplicado o classificador de características lineares SVM. O resultado de classificação é a indicação da presença ou não de um humano em cada janela da imagem.

3.2.2. Visão generalista do método

A identificação de uma pessoa numa imagem requer alguns passos (ver figura 3.5). Pode-se resumir este processo como:

- Receber a imagem RGB e normaliza-la perante diversas condições de iluminação.

- Evidenciar, num conjunto de pixéis (célula) determinadas características através da computação dos gradientes.
- Promover um sistema de votação dos pesos das células. Resulta um bloco com um histograma dos gradientes da orientação (HOG) locais.
- Repetir este processo na imagem e criar múltiplos blocos. Resulta uma sobreposição dos blocos HOG locais.
- Emparelhar todos os blocos na imagem e determinar uma janela de detecção.
- Filtrar o objecto dentro da janela através de um classificador linear SVM, treinado com a silhueta humana.
- O resultado produz a detecção de pessoa / não-pessoa

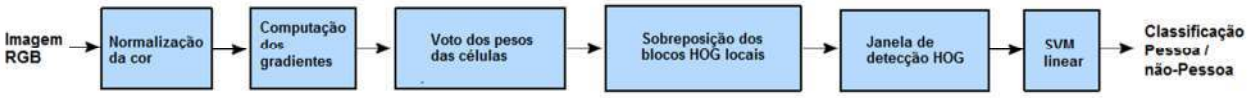


Figura 3.5 – Algoritmo de classificação de pessoas, *in* Dalal e Triggs, *Histograms of oriented gradients for human detection,* 2005

3.2.3. Descritor de características HOG

A componente estática da imagem é pesquisada através do algoritmo HOG, com algumas semelhanças ao descritor SIFT (*Scale Invariant Feature Transform*) proposto por Lowe (2004). O descritor HOG define uma malha refinada de células, onde cada célula contém um histograma local de orientação, determinado através do processo interno R-HOG (*Rectangular-HOG*).

Para cada pixel, o gradiente dos vectores de orientação é calculado e convertido num ângulo. O peso desta orientação é função da magnitude do gradiente. Cada célula possui pixéis com votos acumulados em função dos pesos. Durante o processo de votação o algoritmo faz a interpolação espacial, tanto linear como angular, sobre os blocos para reduzir os efeitos de *aliasing*[7]. Estas células são agrupadas em blocos que são posteriormente normalizados para garantir robustez do bloco quanto a invariância da iluminação.

A sobreposição espacial dos blocos é executada de modo a permitir que cada célula sobressaia múltiplas vezes durante diferentes normalizações. Este passo acelera o processo de decisão. Os histogramas normalizados de todos os blocos são concatenados

[7] Processo X, com amostragem discreta no tempo, $Y(t) = X(t)$, $t \in \Delta t \cdot Z$, origina perdas de informação nas altas frequências, difícil distinção de altas frequências com algumas baixas frequências. Depende do intervalo de amostragem Δt.

de modo a providenciar à janela de detecção um vector de aprendizagem para o descritor de imagem (ver figura 3.5).

3.2.3.1. Descritor SIFT

Para cada ponto de interesse da malha é processado um histograma de gradientes de direcção local na escala do ponto, sobre uma vizinhança em redor do ponto cujos gradientes de direcção são discretizados em 8 direcções (ver figura 3.6). A composição dos histogramas locais na malha de 4×4 com 8 direcções discretizadas resulta em 4×4×8=128 dimensões para cada ponto de interesse. O SIFT está optimizado para fazer a descrição (parelha) em malhas amplas e esparsas.

3.2.3.2. R-HOG

O rectangular-HOG é uma variação do descritor SIFT. Essencialmente faz a computação de uma malha densa da célula e sem ser definido pela orientação dominante do conjunto de pixéis dentro da célula. A orientação é definida por um parâmetro de escala β, unitário. Aplicando o mesmo processo a blocos de células obtém-se um vector que implicitamente define a posição espacial dos blocos relativamente à janela de detecção rectangular.

O modelo R-HOG está optimizado para fazer a descrição para malhas densas na forma espacial. Este tipo de descritor é o utilizado na presente dissertação.

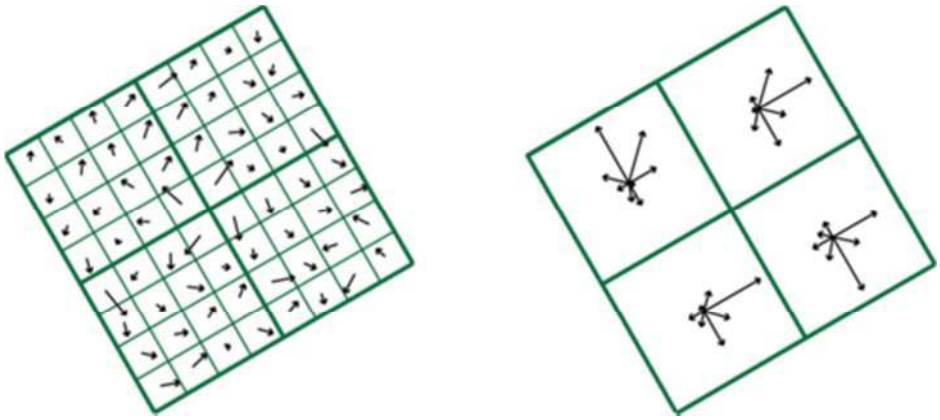


Figura 3.6 – (a) Gradientes na imagem, (b) Histograma das orientações, *in* Lindeberg, *Scholarpedia,* 2012

3.2.4. Modelo matemático do descritor HOG

Para uma dada imagem estática, pode formular-se o seguinte algoritmo:

- Definir uma malha dentro da célula *G,* com centro em $c = (x_c, y_c)$ e raio *r*.
- Para cada ponto *p* contido na sub-malha, $p \subset [m \times m]$ de *G*.
    - Calcular o gradiente $\nabla \mathcal{I}|p$ estimado em *p* como uma média dos pesos de $\nabla \mathcal{I}$, utilizando pesos bilineares centrados em *p*.
    - Adicionar o voto com o peso $\|\nabla \mathcal{I}\| = \frac{1}{\sqrt{2\pi}} \exp(-\frac{\|p-c\|^2}{r^2})$ ao histograma de orientação da célula, para a orientação de $\nabla \mathcal{I}$.
- Determinar o gradiente $\nabla \mathcal{I}_G$ de orientação do histograma (HOG) local da célula.
- Aplicar o mesmo processo a um bloco $B = G \times G$ células e determinar o HOG local $\nabla \mathcal{I}_B$ do bloco.
- Concatenar todos os blocos numa janela de detecção HOG
  $window_{ij} = B_{ij}(\nabla \mathcal{I}_B)||B_{ij}(\nabla \mathcal{I}_B)$

3.2.5. Classificador SVM linear

O SVM (*Support Vector Machine*) linear é um algoritmo de aprendizagem rápida para problemas de classificação de multi-classes em conjuntos de dados complexos. A componente linear implica uma rotina cujo peso computacional aumenta linearmente em função da dimensão do conjunto de dados, conforme proposto por Scholkopf e Smola (2002). Aplicando o SVM linear ao descritor de imagem consegue-se uma detecção fiável em tempo real, incluindo a possibilidade de manipular grandes quantidades de dados assim como robustez perante diferentes conjuntos de dados e parâmetros. Existem testes com SVM não lineares, com menores taxas de erro mas a custo de tempo de processamento do CPU, o que se prova ser inviável na detecção em tempo real.

O processo de aprendizagem do classificador proposto por Dalal e Triggs (2005) foi feito com base num conjunto de imagens de treino onde as amostras de imagem com pessoas são positivas e as amostras de imagem sem pessoas, negativas.

3.2.6. Desvio de pessoas na trajectória do robô

Uma parte importante do trabalho proposto na dissertação refere-se à capacidade do robô para alterar a trajectória e evitar a colisão com pessoas. Para tal deve: (i) reconhecer uma pessoa na imagem, (ii) criar um anel de segurança em redor desta de

modo a evitar a colisão. Este anel serve tanto para a segurança física como psicológica da pessoa, em especial se se considerar uma possível aplicação industrial, com um robô de maior dimensão. (iii) Alterar a trajectória sem perder a posição de destino.

Considerando a pessoa como um obstáculo dinâmico, torna-se necessário escolher um modelo matemático que permita alterar a trajectória do robô em função da posição da pessoa no mundo. Propõe-se a utilização de um modelo parcialmente inspirado na navegação em ambientes humanos densos, proposto por Trautman *et al* (2013).

O trabalho do autor aborda a navegação em ambientes densos. Este trabalho não se aplica directamente ao proposto nesta dissertação, no entanto contém o estudo sobre o que é um raio de segurança aceitável em torno de uma pessoa. Este varia conforme o número de pessoas por metro quadrado, mas é seguro afirmar que 0,50 p/m^2 é um bom ponto de partida. Este valor equivale a um raio de segurança de 0,80m. Para efeitos de ensaio, este raio, embora demostrado ser o socialmente aceitável, limita substancialmente a área livre pela qual o robô pode navegar nas condições do laboratório, pelo que todos os testes foram realizados com um raio de segurança de 0,50m. O robô aplica este parâmetro a cada pessoa detectada e actualiza a sua trajectória. Para situações que não sejam ensaios, o valor proposto pelo autor deve ser sempre a referência.

3.2.7. Método proposto

O método proposto para a presente dissertação utiliza a ideia de criação de um círculo de segurança com um raio pré-definido *r*. O detector de pessoas faz uma detecção positiva e produz uma posição. A esta posição associa-se o raio *r*. Estas duas informações são enviadas para o sistema de navegação robótica, que corrige o caminho. Este sistema de navegação será abordado com mais detalhe no capítulo seguinte. Assim sendo, a informação que o sistema de navegação tem que receber do detector de pessoas é: (i) a presença de uma pessoa (obstáculo) na trajectória calculada, (ii) a posição cartesiana dessa pessoa, (iii) o raio de segurança desejado.

Na figura 3.7 pode observar-se a detecção de uma pessoa[8]. Na imagem (a) são visíveis os *beacons*, assim como a pessoa. A imagem de profundidade (b) ilustra a

[8] Neste ensaio em particular, houve a necessidade de utilizar um objecto escuro (mochila) para criar contraste entre a camisa branca e o cenário branco.

profundidade, através de uma gama de cores. A imagem (c) *Detection* mostra o resultado do programa de detecção de pessoas, com uma identificação positiva, mesmo quando parcialmente tapada por um *beacon*. As matrizes SP_p e $SP_{p,polar}$ guardam respectivamente as posições cartesiana e polar da pessoa.

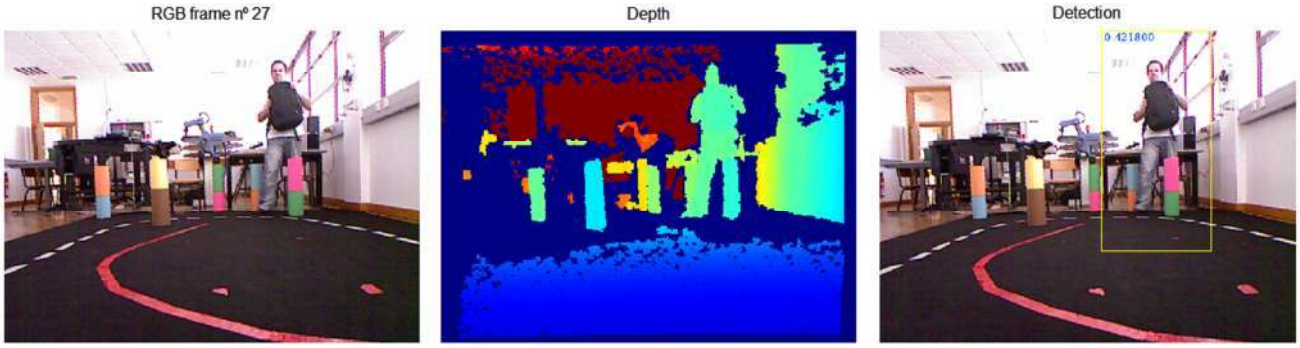


Figura 3.7 – Método para desvio de obstáculos.

O sistema de navegação actualiza o mapa com as posições das pessoas, verifica qual o raio de segurança desejado e calcula a nova trajectória. Para evitar o contacto com os *beacons*, também se define um raio de segurança mais pequeno, em torno destes. Na figura 3.8 é possível observar o mapa com os *beacons* detectados, actualizado com a posição da pessoa, incluindo os raios de segurança de ensaio ($r = 0{,}50m$) e socialmente aceitável ($R = 0{,}80m$, ver secção 3.2.6).

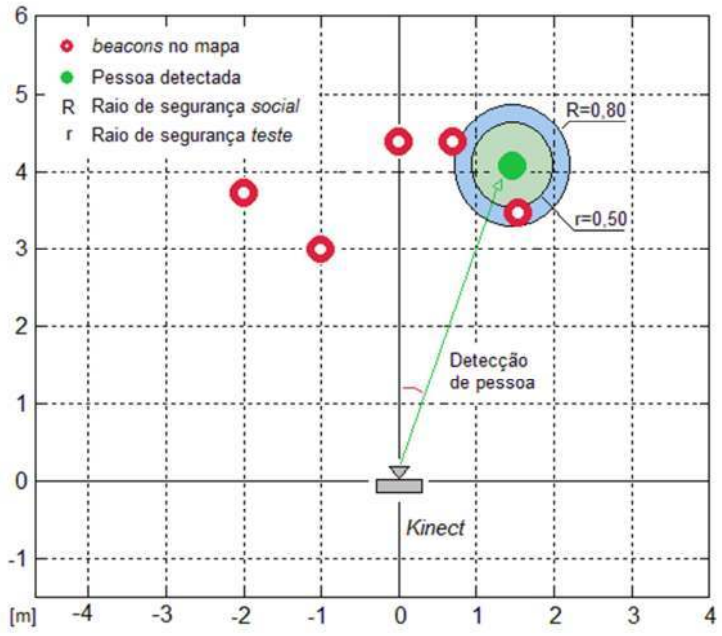


Figura 3.8 – Posição da pessoa no mapa, incluindo raios de segurança.

3.3. Ensaios na detecção de Humanos – Resultados experimentais

Para testar a robustez do algoritmo de detecção de pessoas proposto, foram realizadas 990 capturas de imagens, em 33 ensaios, de modo a afinar a operação de detecção. Estas experiências tiveram lugar dentro do recinto do ISEL: em locais pouco populosos como o Laboratório de Robótica e o hall do edifício de Mecânica (ISEL-ADEM), e em áreas com mais pessoas como o bar e a sala de estudo da associação de estudantes. O objectivo das diferentes localizações foi providenciar o máximo número de variáveis como diferentes condições de iluminação, distinção na reflexão dos vários materiais nos pavimentos e paredes, posições estáticas e em movimento das pessoas e distinção de posturas obstruídas por obstáculos. De salientar que os ensaios retratam capturas feitas com os algoritmos afinados para trabalharem em conjunto sem erros e com resultados fiáveis, de modo a que as variáveis como distâncias das pessoas, iluminação, posturas e outras possam ser analisadas sem corrupção dos dados por erros de programação dos algoritmos. Os ficheiros com os dados dos ensaios de seguida apresentados estão no CD que acompanha a presente dissertação.

3.3.1. Laboratório de Robótica (ISEL-ADEM)

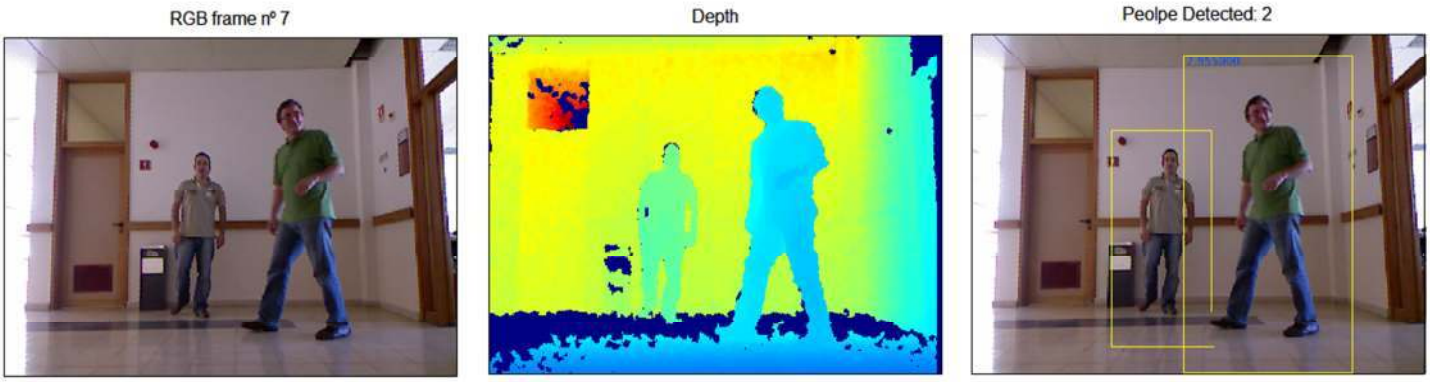


Figura 3.92 – *Frame* nº 7, *dataset* (dados_201474_1725.mat)

Neste local testaram-se múltiplas situações de posturas, obstáculos, distâncias entre outras variáveis que possam afectar a captura, efectivamente pesquisando os piores cenários. Os ensaios foram realizados considerando a Kinect colocada sobre a mesa ou apoiada numa postura similar à do robô móvel (Figura 3.9). Verifica-se que a distância da pessoa à câmara influencia a captura, com uma percentagem de detecção de 47,50%, onde a distância mínima de 2,00m de afastamento é necessária de modo a que toda a silhueta seja perceptível na imagem RGB. Mantendo uma distância igual ou superior, a

percentagem de captura sobe para 88,89%. A situação anterior com múltiplas pessoas em diversas posturas tem uma taxa de sucesso de 75,00%. Posturas laterais estáticas demonstram ser difíceis de capturar, com 47,83%. Posturas frontais ou de costas, incluindo movimento fazem subir a percentagem para 65,22%. Aplicando a câmara RGB-D numa altura semelhante ao robô, a 40cm do pavimento, a percentagem de capturas sobe para 92,86%. No total fizeram-se 330 capturas em 11 ensaios, com média de 67,77% de detecções positivas de e 0,30% de falsas detecções (figura 3.10).

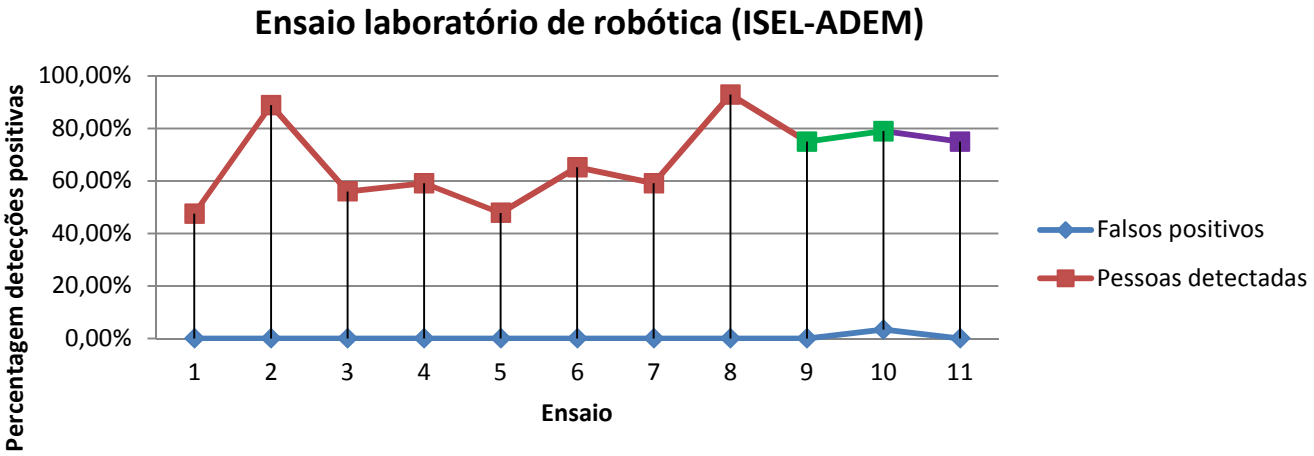


Figura 3.10 – Gráfico ensaios laboratório de robótica, 11 ensaios com 30 capturas cada

Tabela 3.1 – Resumo dos resultados do laboratório de robótica

Ensaio	Falsos positivos	Percentagem detecção pessoas
1	0,00%	47,50%
2	0,00%	88,89%
3	0,00%	56,00%
4	0,00%	59,09%
5	0,00%	47,83%
6	0,00%	65,22%
7	0,00%	59,09%
8	0,00%	92,86%
9	0,00%	75,00%
10	3,33%	78,95%
11	0,00%	75,00%
Σ total	**0,30%**	**67,77%**

3.3.2. Bar AeISEL

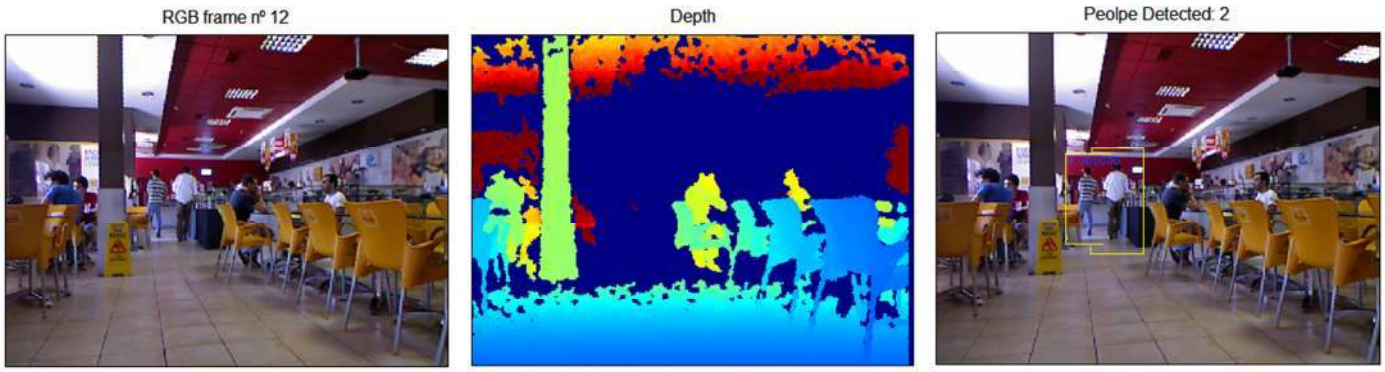


Figura 3.11 – *Frame* nº 12, *dataset* (dados_201479_1615.mat). Notar que a perda de informação de profundidade para distâncias > 9,00m não impede a detecção de pessoas.)

Este ensaio (figura 3.11) demonstra a robustez do algoritmo na detecção de pessoas em ambientes com iluminação solar e artificial, com elevado número de obstáculos, com capturas em de pessoas em posturas laterias em movimento e passagem muito rápida (1 *frame*). O primeiro ensaio origina um falso positivo no 6º *frame*, ao capturar uma cadeira. Este erro é isolado já que, mantendo a mesma posição da câmara, não se repete no restante ensaio. A percentagem de detecção neste mesmo ensaio é de 85,71% e os falsos positivos 16,67%. Verifica-se que a informação de profundidade devolve um conjunto de valores vazio, pois a parede em frente à câmara está a uma distância superior a 9,00m. De notar que este factor não impede que as capturas positivas com distâncias inferiores contenham informações *depth* e mesmo nos casos onde tal não existe, as capturas por RGB têm uma taxa de sucesso de 86,67%. Verificam-se igualmente diversas situações de sobreposição de pessoas na profundidade, cuja silhueta causa desafio ao algoritmo, com uma captura de 66,67%. Movimentos contínuos, em postura vertical devolvem taxa sucesso de 90%. No conjunto desta localização foram feitas 210 capturas em 7 ensaios, com média de 85,72% de detecção de pessoas e 2,38% de falsos positivos.

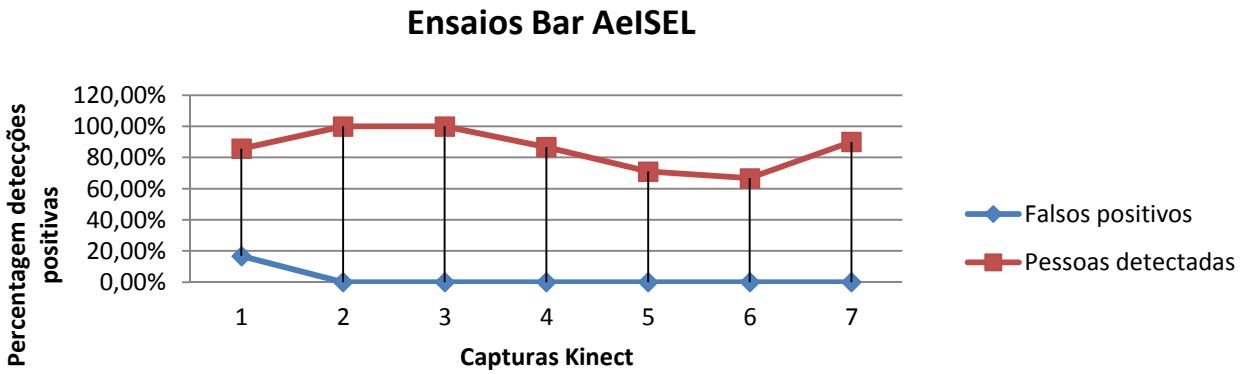


Figura 3.12 – Gráfico ensaios Bar AeISEL

Tabela 3.2 – Resumo dos resultados do bar AeISEL

Ensaio	Falsos positivos	Percentagem detecção pessoas
1	16,67%	85,71%
2	0,00%	100,00%
3	0,00%	100,00%
4	0,00%	86,67%
5	0,00%	70,97%
6	0,00%	66,67%
7	0,00%	90,00%
Σ total	**2,38%**	**85,72%**

3.3.3. Sala estudo AeISEL

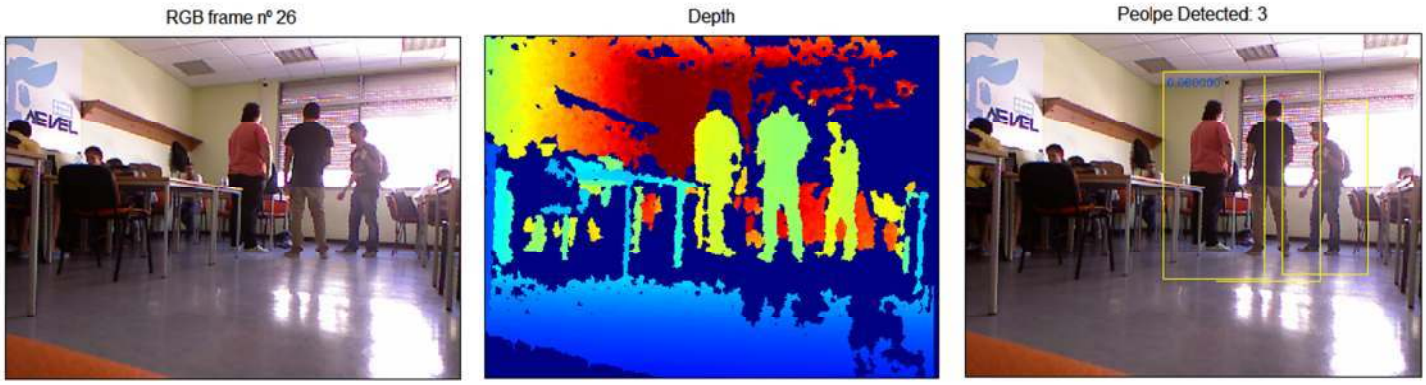


Figura 3.13 – *Frame* nº 26, *dataset* (dados_201479_1646.mat)

No primeiro ensaio (figura 3.13) é de notar que o pavimento altamente reflectivo com a câmara colocada em contra luz. Os falsos positivos que se obtiveram devem-se ao contraste da luz/sombra no chão causar uma silhueta similar à Humana, com valores de 13,33% erro. A detecção de pessoas com postura vertical e desobstruída devolve

capturas com 95,24% de sucesso, incluindo sombras. Descontando os faslos positivos, o número de capturas positivas corrigido é 82,54%. O reposicionamento da Kinect na cadeira para um ângulo mais favorável reduz a 100% o erro de falsos positivos por efeito luz/sombra. Verifica-se igualmente que as posturas influenciam a detecção, nomeadamente posturas em ¾ de costas, com 42,37% de capturas positivas. Nos cenários de posturas estáticas, com a pessoa detectada numa postura vertical faz subir a percentagem para 80,00%. Na situação em que 2 pessoas estão estáticas e sobrepostas originam uma silhueta complexa, com sucesso de detecção de 71,11%. Posturas com movimento e sobreposição momentânea garantem uma taxa de detecção de 73,47%. A média de detecções positivas deste ensaio foi de 74,51% e a percentagem de falsas detecções de 1,48%.

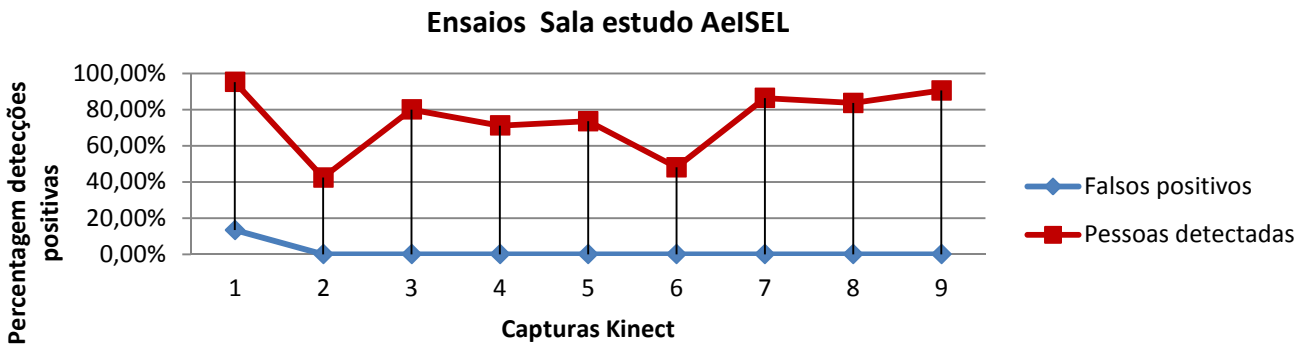


Figura 3.14 – Gráfico ensaio, sala estudo AeISEL

Tabela 3.3 – Resumo dos resultados da sala de estudo AeISEL

Ensaio	Falsos positivos	Percentagem detecção pessoas
1	13,33%	95,24%
2	0,00%	42,37%
3	0,00%	80,00%
4	0,00%	71,11%
5	0,00%	73,47%
6	0,00%	48,00%
7	0,00%	86,36%
8	0,00%	83,58%
9	0,00%	90,48%
Σ total	**1,48%**	**74,51%**

3.3.4. Ensaios Hall entrada Pavilhão de Mecânica (ISEL-ADEM)

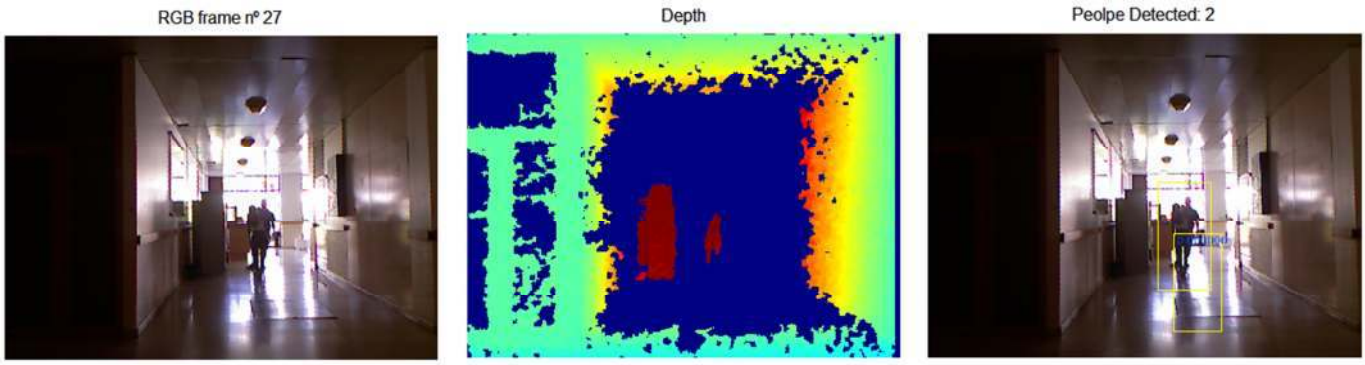


Figura 3.15 – *Frame* n.º 27, *dataset* (dados_2014715_1658.mat)

Esta localização demonstrou ser a mais complexa para o algoritmo, devido à elevada reflexão de luz no pavimento e luz ambiente dominada pelas janelas na entrada do local (figura 3.15). Foram testadas diversas posições da câmara RGB-D, inclusive contraluz directa. O 1º ensaio foi feito com a câmara apoiada na mão, com aproveitamento de 26 *frames*. Destes a taxa de detecção foi 42,86% dado à proximidade das pessoas da câmara, com respectiva silhueta cortada na imagem RGB. As situações 3 e 5 foram captadas em contraluz, sendo de todos os ensaios feitos em diversos locais, aqueles que mais falsos positivos originaram com 23,08% de erro na primeira captura e 48,39% na segunda. Verifica-se uma constante na detecção de uma silhueta de luz, similar a uma pessoa, o que explica estes resultados. Notar que a acumulação destes falsos positivos origina uma detecção de pessoas que não corresponde à realidade. Neste caso, utiliza-se um valor mínimo de aceitação para o parâmetro *scores*, que define o grau de incerteza da captura. O último ensaio, com nova localização da câmara, revela a percentagem de captura mais baixa de todas as experiencias com 4,00% de sucesso, devido ao elevado grau de reflexão de luz nesta localização em particular. A média de detecções positivas deste ensaio foi de 69,07% e a percentagem de falsas detecções de 11,91%.

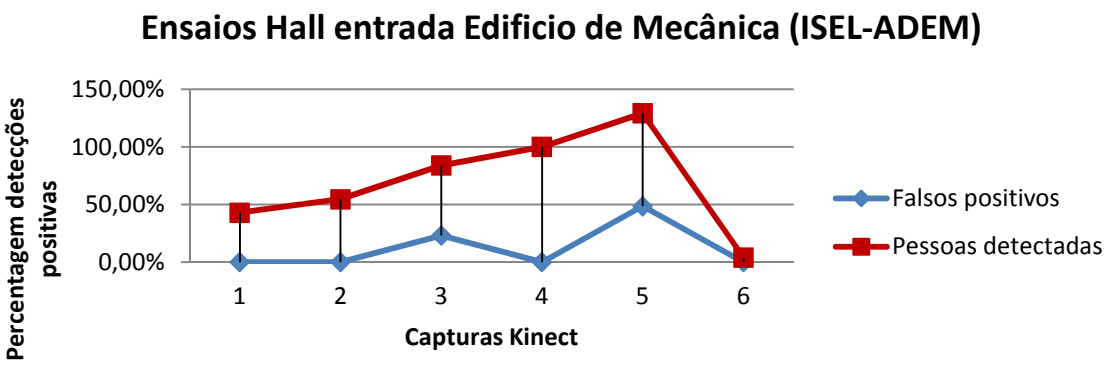


Figura 3.16 – Gráfico ensaio Hall entrada Edifício de Mecânica

Tabela 3.4 – Resumo dos resultados no Hall de entrada do Edifício de Mecânica

Ensaio	Falsos positivos	Percentagem detecção pessoas
1	0,00%	42,86%
2	0,00%	54,55%
3	23,08%	83,87%
4	0,00%	100,00%
5	48,39%	129,17%
6	0,00%	4,00%
Σ total	**11,91%**	**69,07%**

3.3.5. Conclusões dos testes de detecção de pessoas

O algoritmo de detecção de pessoas foi testado em 990 capturas divididas por 33 ensaios diferentes, em múltiplos cenários com inúmeras variáveis, para aferir o grau de confiança do mesmo. Os testes resultam numa média global de 73,65% de detecção positiva de pessoas com um erro por falsos positivos de 3,18%. O algoritmo prova ser robusto, dado que toda a experiencia foi assente na procura das piores situações, correspondentes a uma situação real de navegação do robô móvel. Se o objectivo for um ambiente de navegação controlado, com a iluminação e grau de reflexão de luz optimizadas, então facilmente se verificam médias de sucesso superiores a 90,00%.

O posicionamento das pessoas no mapa é feito com a posição cartesiana das pessoas detectadas, logo o grau de confiança pode ser definido pela média global dos ensaios de detecção, 73,65%. Existe a possibilidade de o programa detectar a pessoa, mas o sensor de profundidade não conseguir obter a informação tridimensional da mesma. Esta situação, embora possível, nunca ocorreu durante os ensaios efectuados.

Capitulo IV

4. Navegação robótica

Um robô móvel baseado em visão deve ter a capacidade de navegar autonomamente. Neste capítulo aborda-se o tema da Navegação robótica, incluindo tipos de navegação, cinemática dos robôs móveis, teoria de controlo, odometria, planeamento de trajectórias. Faz-se ainda a análise dos tipos de planeadores existentes na literatura e inclui-se o desvio de pessoas na trajectória. Apresentam-se as simulações e ensaios realizados. O capítulo conclui com o algoritmo geral da dissertação, onde se fundem todas as experiências realizadas para atingir as metas propostas com o robô móvel, com os respectivos resultados.

4.1. Sistemas de Navegação

A navegação[9] é um dos aspectos mais complexos de um robô móvel, que consiste no guiamento do robô até um objectivo. Entre os diversos estudos existentes na área de algoritmos de navegação, para a presente dissertação propõem-se: (1) a obra de Siegwart e Nourbakhsh (2004), como referência teórica e (2) a obra de Peter Corke (2011) como referência prática, para programação dos algoritmos de navegação.

Os robôs com sistema de controlo mais simples utilizam um método de navegação reactivo, reagindo ao meio que o rodeia. Entre os comportamentos mais típicos encontra-se o desvio de obstáculos, o seguimento de uma linha marcada no pavimento ou a reacção à intensidade de iluminação. Será correcto afirmar que o robô apenas sente e reage à informação do meio que o rodeia, sem qualquer conhecimento da sua posição no mundo.

Um sistema reactivo, por definição, não faz planeamento. Para que um robô móvel possa planear a sua trajectória, é necessária a navegação baseada em mapa. Este tipo de navegação é substancialmente mais complexo, pois implica que o robô móvel tenha: (1) conhecimento do mundo e das suas referências, representado através de um mapa, (2) conhecimento constante da sua posição e (3) um sistema de controlo que use as

[9] "O processo de conduzir o veículo de modo a que este alcance a meta desejada" *em* IEEE, Standard 172 – 1983

informações anteriores e providencie ao robô uma navegação desde a origem até ao destino desejado.

Sumarizando esta informação, a navegação baseada em mapa depende de quatro factores:

Percepção → O robô interpreta a informação recolhida pelos sensores.

Localização → O robô determina a sua posição no meio que o rodeia.

Computação → O robô calcula quais as acções necessárias para atingir o objectivo.

Actuação → O robô acciona os motores para seguir a trajectória calculada.

4.1.1. Tipo de sistemas de controlo para navegação

Existem múltiplos sistemas de controlo para suportar a navegação de robôs, distinguindo-se fundamentalmente entre a navegação comportamental e a navegação por mapa. Cada uma oferece vantagens e desvantagens, dependendo muito do objectivo para qual o robô móvel é projectado.

A navegação comportamental determina o movimento do robo como uma reação ao mundo que o rodeia. Esta reação é composta por: (i) informação do comportamento específico que deve ser seguido (ex. manter-se na area *n* ao seu redor), (ii) informação da posição dos obstáculos, com controladores baseados em lógica *fuzzy*, conforme ilustram Lowen *et al* (2012) ou controladores de rede neuronal, apresentados por Hunt *et al* (2012) e (iii) regra para deslocamento em espaços confinados, como por exemplo seguir a parede. A figura 4.1 ilustra um sistema de controlo comportamental. Os sensores captam informação do mundo. Segue-se o processamento, onde o sistema comportamental individualmente calcula as ações para um determinado objectivo, como evitar obstaculos ou seguir uma parede. O conjunto é posteriormente fundido em função dos pesos dos objectivos. Esta informação indica ao controlo cinematico qual a velocidade e ângulo a adoptar.

A obra Mothlag et al (2012) aborda um sistema de controlo comportamental para robôs móveis. Este tipo de navegação implica que cada mudança de local obrigue à criação de um novo programa comportamental que satisfaça os condicionalismos da nova area de trabalho.

Como não faz uso do mapa do ambiente, este tipo de navegação não será utilizado nesta dissertação, pois não oferece qualquer vantagem para o cálculo de trajectórias.

Figura 4.1 – Sistema de controlo para navegação comportamental

A navegação por localização consiste em controlar a deslocação do robô através de um caminho específico. Isto implica que o robô, ao longo da trajectória, se desloque da posição estimada para uma determinada posição no caminho. Para que este processo seja executado (ver figura 4.2), o robô necessita de um sistema de percepção, que reconheça referências no mundo a partir das quais determina a sua auto localização. Segue-se a fase de planeamento, onde o robô, com a estimativa da sua posição no mundo, a informação das posições do ponto de partida (pode ou não ser a posição actual) e da meta, planeia o caminho específico a seguir. Neste tipo de navegação, à medida que o robô se move no espaço, a sua postura vai sendo corrigida através de um controlador que, actuando nos motores das rodas, o conduz pelo caminho desejado até ao ponto de destino.

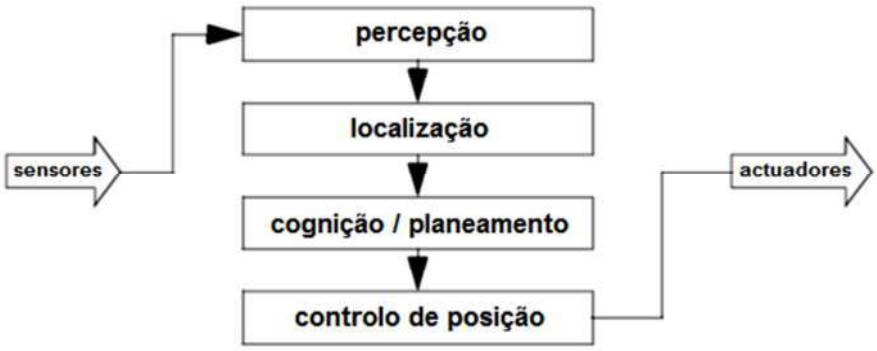


Figura 4.2 – Sistema de controlo para navegação baseada em mapa

A navegação baseada em mapa apresenta diversas vantagens:

- O conhecimento do mundo através de um mapa permite identificar obstáculos e zonas de passagem, sem recorrer a heurísticas simples como “toca e desvia”.
- Este conhecimento permite criar múltiplas trajectórias, que podem ser optimizadas numa trajectória ideal.

- A trajectória desejada pode ser definida através de um ponto de partida e meta, programada para oferecer capacidade de desvio de obstáculos não previstos e manter a estabilidade necessária para atingir a meta.

O sistema de navegação por mapa, em última analise, depende da construção interna que o robô faz do mundo e do grau de confiança que este possui relativamente aos dados capturados pelos sensores. Esta incerteza da estimativa da postura do robô é função do sistema de localização utilizado. Para esta dissertação, propôs-se o uso do estimador MCL (ver capítulo 2), que aplica um filtro de partículas para estimar a postura do robô.

4.2. Cinemática de robôs móveis

A cinemática do robô móvel é o estudo matemático do movimento. Neste estudo, analisam-se as relações geométricas entre cada elemento (ex. distância do eixo entre duas rodas) e as respectivas restrições aos graus de liberdade, função da construção do robô. Os parâmetros de controlo vão reger estas relações e determinar o comportamento do sistema.
A cinemática pode ser não-holonómica (ex. *differential drive*) ou holonómica (ex. *omniwheel robot*). A título de exemplo, um sistema não holonómico, por exemplo uma mota, tem o número de graus de liberdade inferior ao total de graus de liberdade. Já um *hovercraft*, cujo número de graus de liberdade é igual ao total de graus de liberdade, tem capacidade de deslizar lateralmente.

Para a presente dissertação utiliza-se o robô com sistema diferencial, disponível no laboratório de robótica do ISEL. Para a descrição da sua postura consideram-se 3 eixos ortogonais: dois eixos cartesianos horizontais para o posicionamento (x, y) e um eixo vertical para a orientação (θ). 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.

4.2.1. Sistema diferencial

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 referencia global do plano de inércia $I = \{X_I, Y_I\}$, centrado na origem (ver figura 4.3). A posição P do robô define o seu ponto de referência no chassis, e é descrita por dois eixos locais ortogonais $\{X_R, Y_R\}$, entre o eixo das rodas motrizes e entre o centro do eixo relativo à roda de apoio.

A posição global do ponto P é dada pelas coordenadas *x* e *y* assim como pela diferença angular θ entre os referenciais locais e globais.

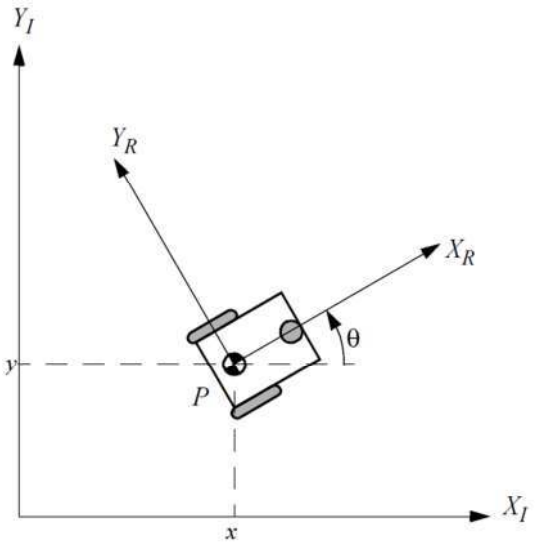


Figura 4.3 – Referencias locais e globais do robô móvel, *in* Siegwart, *Introduction to Autonomous Mobile Robots*, 2004, pp. 49

O vector de estados com a postura do robô móvel (equação 4.1) no referencial de base *I* designa-se por ξ_I.

$$\boldsymbol{\xi}_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{4.1}$$

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

$$\mathbf{R}(\theta) = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

4.2.2. Cinemática de um sistema diferencial

Nesta secção abordam-se as equações que definem a cinemática de um robô de tracção diferencial, descritas na obra de Siegwart e Nourbakhsh (2004).

A cinemática de um robô de tracção diferencial é definida pelo modelo apresentado na equação (4.3), onde $\dot{x}$ e $\dot{y}$ são as velocidades lineares nas direcções X_I e Y_I do plano de inércia *I* e o ângulo θ a orientação do robô, segundo o mesmo plano.

$$^{I}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix} \times \begin{bmatrix}v\\ \omega\end{bmatrix} \tag{4.3}$$

No sistema de locomoção diferencial a velocidade linear é dada pela média aritmética das velocidades lineares das 2 rodas:

$$V(t) = \frac{1}{2} \times \left[V_{right}(t) + V_{left}(t)\right] \tag{4.4}$$

Onde $V_{right}(t)$ e $V_{left}(t)$ são as velocidades lineares da roda direita e esquerda, respectivamente.

A velocidade angular é dada pela seguinte expressão:

$$\omega(t) = \frac{V_{right}(t) - V_{left}(t)}{L} \tag{4.5}$$

Sendo *L* a distância entre os pontos de contacto das rodas.

Sabendo-se que, durante o movimento angular, todos os pontos do eixo das rodas de tracção se deslocam em torno de um ponto (centro de rotação instantâneo) com a mesma velocidade angular, $\omega(t)$, é possível obter esse valor a partir da velocidade linear de cada uma das rodas.

$$\omega(t) = \frac{V_{right}(t)}{R + {}^{L}/_{2}} \quad , \qquad \omega(t) = \frac{V_{left}(t)}{R - {}^{L}/_{2}} \tag{4.6}$$

Onde R é raio de curvatura determinado por:

$$R = \frac{V(t)}{\omega(t)} \tag{4.7}$$

Assim, defindindo-se uma lei de controlo que determine as velocidades lineares e angulares do robô, é possível determinar a correspondente velocidade linear a enviar a cada roda:

$$V_{right}(t) = V(t) + \omega(t) \times \frac{L}{2} \ , \qquad V_{left}(t) = V(t) - \omega(t) \times \frac{L}{2} \tag{4.8}$$

4.2.3. Controlo de Postura

Com o controlo de postura é possível utilizar um sistema de navegação para mover a postura de referência ao longo do caminho planeado. Propõe-se o modelo de controlo de postura apresentado por Siegwart e Nourbakhsh (2004).

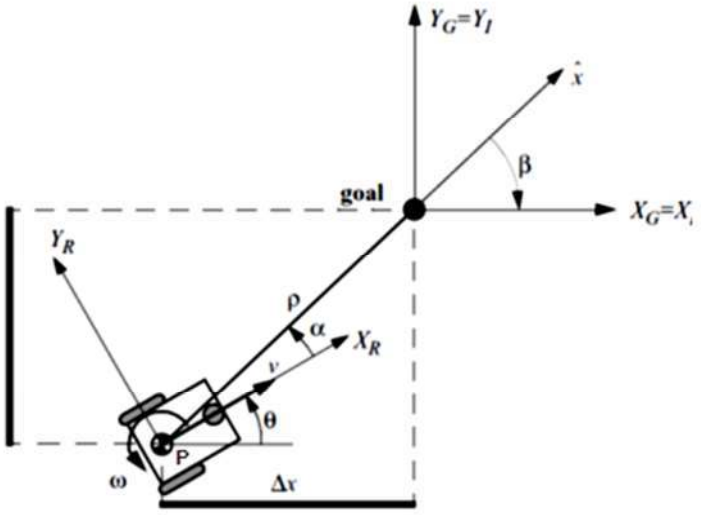


Figura 4.4 – Variáves e sistemas de eixos envolvidos na lei de controlo do robô, *in* Siegwart, *Introduction to Autonomous Mobile Robots*, 2004, pp. 83

Considere-se que se pretende deslocar o robô de uma posição e orientação inicial $[x, y, \theta]_{ref}$ arbitrária para uma postura $[x, y, \theta]$ pré-definida (*goal*), conforme se pode visualizar na figura 4.4. O ângulo α define o ângulo entre o eixo de referência do robô X_R e o vector $\hat{x}$, que une o centro do eixo das rodas *P* com a posição da meta. β define o ângulo da meta, entre o vector $\hat{x}$ e o eixo de inércia X_I. A distância entre *P* e a posição da meta designa-se por ρ.

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

$$\mathbf{e} = [x, \theta]_{ref}^{T} \tag{4.9}$$

A matriz **K** representa a matriz de ganhos do controlador. Aplicando esta matriz no sistema multivariável, através da multiplicação de **K** com vector de erros **e**, obtém-se as acções de controlo da postura do robô.

$$\mathbf{K} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \end{bmatrix} \text{ com } k_{ij} = k(t, e) \tag{4.10}$$

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \mathbf{K} \times \mathbf{e} = \mathbf{K}. \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{ref} \tag{4.11}$$

4.2.4. Sistema em coordenadas polares

Existem diversas vantagens em utilizar um sistema de coordenadas polares para descrever a cinemática de um robô de tracção diferencial. Considere-se a origem das coordenadas polares na posição da meta:

$$\rho = \sqrt{\Delta x^2 + \Delta y^2} \tag{4.12}$$

$$\alpha = -\theta + \text{atan2}(\Delta y, \Delta x) \tag{4.13}$$

$$\beta = -\theta - \alpha \tag{4.14}$$

A partir das equações (4.12), (4.13), (4.14) é possível obter o modelo cinemático do erro em coordenadas polares. Este pode ser escrito na forma matricial,

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos\alpha & 0 \\ \frac{\sin\alpha}{\rho} & -1 \\ -\frac{\sin\alpha}{\rho} & 0 \end{bmatrix} \times \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4.15}$$

Notar que este modelo cinemático é válido quando a direcção α do robô é para a frente, ou seja,

$$I_1 = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \quad \alpha \in I_1 \tag{4.16}$$

O robô também se pode deslocar para trás. Neste caso, a velocidade passa a ser definida por $v = -v$, e a direcção do robô $\alpha \in I_2$,

$$I_2 = \left\{-\pi, \frac{\pi}{2}\right\} \cup \left\{\frac{\pi}{2}, \pi\right\}, \quad \alpha \in I_2 \tag{4.17}$$

Quando o robô se desloca para trás, a matriz do sistema de controlo passa a ser

$$\begin{bmatrix}\dot{\rho}\\ \dot{\alpha}\\ \dot{\beta}\end{bmatrix} = \begin{bmatrix}\cos\alpha & 0\\ -\frac{\sin\alpha}{\rho} & 1\\ \frac{\sin\alpha}{\rho} & 0\end{bmatrix} \times \begin{bmatrix}v\\ \omega\end{bmatrix} \tag{4.18}$$

4.2.5. Lei de controlo

Os sinais de controlo *v* e *ω* definem o movimento do robô desde a sua posição inicial $(\rho_0, \alpha_0, \beta_0)$ até à meta. Aplicando uma lei de controlo, através das equações (4.19) e (4.20), é possível definir a cinemática do erro, em anel fechado, com a equação (4.21)

$$v = k_\rho . \rho \tag{4.19}$$

$$\omega = k_\alpha . \alpha + k_\beta . \beta \tag{4.20}$$

de onde resulta na matriz de controlo em anel fechado

$$\begin{bmatrix}\dot{\rho}\\ \dot{\alpha}\\ \dot{\beta}\end{bmatrix} = \begin{bmatrix}-k_\rho . \rho . \cos\alpha\\ k_\rho . \sin\alpha - k_\alpha . \alpha - k_\beta . \beta\\ -k_\rho . \sin\alpha\end{bmatrix} \tag{4.21}$$

O sistema não possui nenhuma singulariade quando $\rho = 0$ e equilibra-se 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 (-π,π). 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.

4.2.6. Estabilidade local

O sistema de controlo em anel fechado (4.21) é estável se verificar as condições

$$k_\rho > 0 \quad ; \quad k_\beta < 0 \quad ; \quad k_\alpha - k_\rho > 0 \tag{4.22}$$

Prova:

A matriz **A** representa o modelo em anel fechado . Linearizando a posição do equilibrio $(\cos x = 1\ ;\ \sin x = x)$ a equação pode ser escrita como:

$$\begin{bmatrix}\dot{\rho}\\ \dot{\alpha}\\ \dot{\beta}\end{bmatrix} = \underbrace{\begin{bmatrix}-k_\rho & 0 & 0\\ 0 & -(k_\alpha - k_\rho) & -k_\beta\\ 0 & -k_\rho & 0\end{bmatrix}}_{\mathbf{A}} \times \begin{bmatrix}\rho\\ \alpha\\ \beta\end{bmatrix} \tag{4.23}$$

O sistema é estável se todos os pólos do sistema tiverem parte real negativa, que se pode verificar atraves dos valores proprios da matriz **A**.

O determinante polinomial da matriz **A** é

$$\left(\lambda + k_\rho\right) \times \left\{\lambda^2 + \lambda.\left(k_\alpha - k_\rho\right) - k_\rho.k_\beta\right\} \tag{4.24}$$

e as suas raizes têm componente real negativa se $k_\rho > 0$; $-k_\beta > 0$; $k_\alpha - k_\rho > 0$, o que comprova a estabilidade local do control em anel fechado.

Para um controlo de posição robusto, aplica-se uma condição de estabilidade mais conservadora que garante que o robô não muda de direção quando se aproxima da meta

$$k_\rho > 0 \quad ; \quad k_\beta < 0 \quad ; \quad k_\alpha + \frac{5}{3}.k_\beta - \frac{2}{\pi}.k_\rho > 0 \tag{4.25}$$

Tal implica que $\alpha \in I_1$ para todo o *t*, quando $\alpha(0) \in I_1$ e $\alpha \in I_2$ para qualquer *t* e quando $\alpha(0) \in I_2$ respectivamente.
Com a cinemática do robô móvel descrita matematicamente, surge a necessidade de compreensão do planeamento de trajectórias e correspondentes algoritmos.

4.3. Planeamento de caminhos

Um dos objectivos do robô passa pela capacidade de seguir um determinado caminho entre a origem e o destino. Esta secção apresenta um método para planeamento de caminhos que, integrado com o sistema de localização, percepção de pessoas e de controlo apresentados previamente, permitirá a navegação segura do robô e das pessoas.
O primeiro passo do planeamento de trajectórias consiste na transformação do modelo do mundo físico num mapa, que o robô consiga interpretar. Na literatura existem vários modelos de planeamento, que podem ser generalizados em quatro tipos:

- Decomposição de células: faz o cálculo de múltiplos caminhos com pontos de origem e meta fixos.
- Mapa de estrada: calcula múltiplos pontos de início e metas, sempre com trajectórias fixas.

- Probabilísticos: geram pontos num mapa entre a origem e o destino e seleccionam probabilisticamente aqueles que oferecem o menor custo.
- Campos de potências: criam uma função matemática que representa um gradiente onde o ponto mais elevado é a origem e a meta tende para zero.

Para a presente dissertação faz-se uso do algoritmo do campo de potências para a criação de um sistema de navegação, por ser o método que melhor se adapta as necessárias alterações de trajectória perante a presença de pessoas.

4.3.1. Planeamento de trajectórias com Campo de Potências

O planeamento de trajectória com campos potenciais é executado através da criação de um campo de gradientes que se sobrepõe ao mapa, onde estão incluídos os pontos de partida e meta.

Fazendo uma analogia com a natureza, o campo de forças comporta-se como um vale onde os objectos são montanhas, os caminhos desde a origem cursos de água na base dos obstáculos e a meta o culminar destes caminhos, sendo o ponto mais atractivo.

O robô é discretizado como um ponto sob um campo de influências artificial *U(q)* e desloca-se pelo caminho que oferecer o menor custo. O ponto da meta comporta-se como um campo de atracção e os obstáculos como campos de repulsão, com valores de pico. A sobreposição de todas as forças existentes no mapa leva a que o robô escolha o caminho mais atractivo até à meta, evitando os obstáculos identificados no mapa. Estes obstáculos, sejam pessoas ou *beacons*, têm raios de segurança associados (consultar capítulo 3). As obras de Latombe (1991) e Siegwart (2004) providenciam as referências para o modelo matemático.

4.3.2. Modelo matemático

Fazendo a análise matemática, no caso mais simples, assume-se o robô móvel como um ponto e ignora-se a orientação (θ). O resultado é um campo de potências de duas dimensões *(x, y)*. Assumindo o diferencial deste campo como uma função *U(q)* é possível determinar a força artificial *F(q)* que actua na posição $q(x,y)$.

$$F(q) = -\nabla U(q) \tag{4.26}$$

onde $-\nabla U(q)$ é o gradiente do vector U na posição q

$$\nabla U = \begin{bmatrix} \frac{\partial U}{\partial x} \\ \frac{\partial U}{\partial y} \\ \frac{\partial U}{\partial z} \end{bmatrix} \tag{4.27}$$

O campo de potências que actua no robô é calculado através da soma do campo de atracção da meta com os campos de repulsão dos obstáculos

$$U(q) = U_{att}(q) + U_{rep(q)} \tag{4.28}$$

Do mesmo modo, as forças actuantes podem ser decompostas em atractivas e repulsivas

$$F(q) = F_{att}(q) - F_{rep}(q) = -\nabla U_{att}(q) - \nabla U_{rep}(q) \tag{4.29}$$

O campo de potências atractivo pode ser definido como uma função parabólica, descrita pela seguinte equação.

$$U_{att}(q) = \frac{1}{2} k_{att} \times \rho_{goal}^2(q) \tag{4.30}$$

onde k_{att} é um factor de escala positivo e $\rho_{goal}(q)$ a distância Euclidiana $\|q - q_{goal}\|$.

A força atractiva é determinada pela derivada do respectivo campo potencial e irá convergir para zero, à medida que o robô se aproxima da meta.

$$\begin{aligned} F_{att}(q) &= -\nabla U_{att}(q) \\ &= -k_{att} \times \rho_{goal}(q) \times \nabla \rho_{goal}(q) \\ &= k_{att} \times (q - q_{goal}) \end{aligned} \tag{4.31}$$

O campo de potências repulsivo é gerado como uma força com origem no centro de um determinado obstáculo conhecido. Este potencial deve aumentar proporcionalmente à medida que o robô se aproxima do obstáculo e atingir valores de pico quando o robô se encontra na sua vizinhança, mas não ter qualquer tipo de influência quando o robô está distante. Representa-se por:

$$U_{rep}(q) = \begin{cases} \frac{1}{2} k_{rep} \times (\frac{1}{\rho(q)} - \frac{1}{\rho_o})^2 & \text{se } \rho(q) < \rho_0 \\ 0 & \text{se } \rho(q) \geq \rho_0 \end{cases} \tag{4.32}$$

Analisando a equação temos k_{rep} como factor de escala, $\rho(q)$ é a menor distância entre q e o obstáculo e ρ_0 o limite (distância) de influência do obstáculo. A força repulsiva F_{rep} corresponde à derivada do campo repulsivo, dada pela equação (4.32).

$$F_{rep}(q) = -\nabla U_{rep}(q) =$$

$$= \begin{cases} k_{rep}.\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right) \times \frac{1}{\rho^2(q)} \times \frac{q - q_{obstaculo}}{\rho(q)} & \text{se } \rho(q) < \rho_0 \\ 0 \text{ se } \rho(q) \geq \rho_0 \end{cases} \quad (4.33)$$

A força resultante (4.27) actua sobre o ponto que representa o robô no mapa e faz com que este se desloque em direcção à meta ao mesmo tempo que escolhe caminhos que o afastem dos obstáculos.
Fazendo com que a velocidade do robô seja proporcional ao vector de forças é possível criar um movimento suave desde o ponto de partida até à meta.

4.3.3. Modelo proposto

O modelo matemático proposto implica o desenvolvimento de um algoritmo de planeamento baseado em campos de potências. Este algoritmo considera a existência de pessoas e objectos. Os *beacons* são considerados objectos, dos quais o robô se deve desviar para evitar colisões. O mesmo aplica-se às pessoas detectadas, considerando para tal raios de afastamento diferentes. Este modelo considera igualmente a existência de um mapa do mundo, com posições de partida e meta definidas. Perante todos estes imperativos, o algoritmo deve elaborar um caminho que evite colisões com os obstáculos, resultando num conjunto de pontos [*x, y*], equidistantes.

4.3.3.1. Identificação de referências

O algoritmo dos campos potenciais para o sistema de navegação inicia-se com a matriz $\mathbf{MAP}_{[m \times x]}$, que contém a informação da dimensão do mapa do mundo, em metros.

$$\mathbf{MAP} = \begin{bmatrix} tl & tr \\ bl & br \end{bmatrix}, \; TL, TR, BL, BR \in \mathbb{N} \quad (4.34)$$

Cada canto do mapa tem a sua coordenada cartesiana [*x, y*], designando-se o canto superior esquerdo como *tl* (*top left*), o canto superior direito *tr* (*top right*), o canto

inferior esquerdo *bl* (*bottom left*) e o canto inferior direito *br* (*bottom right*). Assume-se o ponto central do mapa como a origem no sistema de eixos cartesianos.
As relações geométricas entre as coordenadas de cada ponto aplicam-se ($tl_x < tr_x,\ tl_y = tr_y, \ldots$), de modo a que o mapa resulte ortogonal.

O vector $\mathbf{map}_{dim}$ define a dimensão do mapa, em metros, em x e em y.

$$x_{dim} = \sqrt{(tl_x^2 + tr_x^2)}$$

$$y_{dim} = \sqrt{(tl_y^2 + bl_y^2)}$$

$$\mathbf{map}_{dim} = [x_{dim}, y_{dim}] \tag{4.35}$$

A informação da posição cartesiana das referências no mundo é fornecida pelo detector de *beacons*, através da matriz **DC***q*, onde *q* define o número de beacons.

$$\mathbf{DC}_q = \begin{bmatrix} dc_{1,1} & dc_{1,2} \\ \vdots & \vdots \\ dc_{q,1} & dc_{q,2} \end{bmatrix}$$

O raio de segurança em torno dos *beacons* é definido por $r_{beacons,} \in \mathbb{R}^+$.

A matriz $\mathbf{BEACONS}_q$ define a posição de cada *beacon* e o respectivo raio de segurança. Assume-se ser o mesmo, no entanto pode definir-se um raio específico por cada *beacon*, se se achar conveniente.

$$\mathbf{BEACONS}_q = \begin{bmatrix} dc_{1,1} & dc_{1,2} & r_{beacons} \\ \vdots & \vdots & \vdots \\ dc_{q,1} & dc_{q,2} & r_{beacons} \end{bmatrix} \tag{4.36}$$

A posição das pessoas no mundo é enviada pelo detector de pessoas, através da matriz $\mathbf{SP}_p$, onde *p* define o número de pessoas detectadas.

$$\mathbf{SP}_p = \begin{bmatrix} Xpk_1 & Ypk_1 \\ \vdots & \vdots \\ Xpk_p & Ypk_p \end{bmatrix}$$

O raio de segurança em torno das pessoas é definido por $r_{people,} \in \mathbb{R}^+$. Este raio está relacionado com a distância de segurança entre a navegação e as pessoas, abordadas na secção 3.2.7.

A matriz $\mathbf{PEOPLE}_p$ define a posição de cada pessoa e o respectivo raio de segurança.

$$\mathbf{PEOPLE}_p = \begin{bmatrix} Xpk_{1,1} & Ypk_{1,2} & r_{people} \\ \vdots & \vdots & \vdots \\ Xpk_{p,1} & Xpk_{p,2} & r_{people} \end{bmatrix} \tag{4.37}$$

A posição do ponto de partida define-se pelo vector **start** = $[x_s, y_s \theta_s]$ e a meta pelo vector **goal** = $[x_g, y_g \theta_g]$

Com as informações das referências exteriores, cria-se a matriz **OBST**, concatenada verticalmente, que identifica os obstáculos no campo de potências. Cada linha da matriz é composta por um vector do obstáculo, nas posições $[x, y, raio_{seg}]$

$$\mathbf{OBST} = beacons_q || people_{p=} = \begin{bmatrix} dc_{1,1} & dc_{1,2} & r_{beacons} \\ \vdots & \vdots & \vdots \\ dc_{q,1} & dc_{q,2} & r_{beacons} \\ Xpk_{1,1} & Ypk_{1,2} & r_{people} \\ \vdots & \vdots & \vdots \\ Xpk_{p,1} & Xpk_{p,2} & r_{people} \end{bmatrix} \tag{4.38}$$

A utilização dos campos de potências requer a utilização de um gradiente para o cálculo do melhor caminho. Em termos de programação, a maneira mais simples de criar este gradiente é converter o sistema métrico em pixéis. Isto permite criar um mapa com uma resolução numa escala bem definida.

Em termos de exemplo, se se considerar um mapa de [10, 10] metros, aplicando uma conversão em [1000, 1000] pixéis obtém-se uma resolução de 0,01m ou seja 1cm de afastamento entre cada pixel.

Como todos os valores de entrada (em metros) têm que ser convertidos, o mais simples é criar uma sub-rotina que faça esta conversão automaticamente. Considere-se N_m o número de metros, N_{pix} o número de pixéis, f a constante entre ambos (duas constantes, f_x e f_y se o mapa for rectangular), $[x, y, \theta]$ a designação de coordenadas cartesianas e $[u, , v , w]$ a respectiva conversão em pixéis.

i) Sub-rotina para converter metros em pixéis. Propõe-se a designação "*map2bitmap*".

$$N_m = [N_{m,x},\ N_{m,y}],\ metros$$
$$N_{pix} = [N_{pix,u},\ N_{pix,v}],\ pixeis$$
$$f_x = \frac{N_{pix,u}}{N_{m,x}} \quad ; \quad f_y = \frac{N_{pix,v}}{N_{m,y}}$$
$$u = \frac{N_{pix,u}}{2} + x.f_x \quad ; \quad v = \frac{N_{pix,v}}{2} + y.f_y \quad ; \tag{4.39}$$
$$[x, y, \theta]_{meters} \xrightarrow{map2bitmap} [u, v, w]_{pixels}$$

Fim da sub-rotina.

A conversão das variáveis que definem o modelo, de metros para pixéis, permite criar um índice em cada pixel. Assuma-se $pix_{i,j} \in [0; 1], i = \{1 \dots u\},\ j = \{1 \dots v\}$ onde o índice *0* identifica uma região livre e *1* uma região obstruída.

Com a sub-rotina criada, define-se a escala *f*, relação entre o número de metros do mapa existente e o número de pixéis desejados.

$$N_m = \mathbf{MAP}_{dim} \quad ; \quad N_{pix} = [i, j], \in \mathbb{N} \quad ; \quad f = {}^{N_{pix}}\!/_{N_m}$$

Aplicando a conversão ao mapa, obtemos o mapa equivalente em pixéis.

$$\mathbf{MAP} \xrightarrow{map2bitmap} \mathbf{MAP}_{pix}$$

$$\mathbf{MAP}_{pix} = \begin{bmatrix} pix_{1,1} & \dots & pix_{1,v} \\ \vdots & \ddots & \vdots \\ pix_{u,1} & \dots & pix_{u,v} \end{bmatrix} \tag{4.40}$$

Os pontos de partida e meta são igualmente convertidos

$$\mathbf{start} \xrightarrow{map2bitmap} \mathbf{start}_{pix}\ ,\ \mathbf{start}_{pix} = [u_S, v_S, w_S]$$
$$\mathbf{goal} \xrightarrow{map2bitmap} \mathbf{goal}_{pix}\ ,\ \mathbf{goal}_{pix} = [u_G, v_G, w_G]$$

Cada obstáculo tem associado um raio de segurança, logo é necessário identificar a área circundante, como área obstruída, antes de se fazer a conversão de metros para pixéis.
A matriz $\mathbf{OBST}_i$ contém o conjunto de vectores com a posição $[x, y, raio_{seg}]_i$ de cada obstrução, distribuídos pelas linhas i.

Com o centro definido, é necessário calcular os pontos cobertos por esta área.

$$O_{x,i} = r_i \times \cos(n) + x_i$$
$$O_{y,i} = r_i \times \sin(n) + y_i \quad (4.41)$$

O número de pontos $[O_x, O_y]_i$ depende da resolução do intervalo entre $n = \{0, \dots, 2\pi\}$. A matriz **Area** pode conter um simples ciclo que calcule *Ox* e *Oy* em função de *n*.

$Area_i(n) = [Ox_n, Oy_n]_i$, $centro\ [x, y]_i$, $raio_{seg,i}$

Convertendo a matriz **Area**, de metros para pixéis, obtém-se a região circular equivalente, com os respectivos centros e raio se segurança, sempre função da linha de referência *i*.

$$\mathbf{Area}_i \xrightarrow{map2bitmap} \mathbf{PixArea}_i,\ centro\ [u, v]_i\ ,\ r_{pix,i}$$

$$\mathbf{PixArea}_i = \begin{bmatrix} u_1 & v_1 \\ \vdots & \vdots \\ u_n & v_n \end{bmatrix}_i$$

Que pode ser simplificada no seguinte vector

$$\mathbf{PixArea_i} = \mathbf{area}_i = \begin{bmatrix} pix_{u1,v1} \\ \vdots \\ pix_{un,vn} \end{bmatrix}_i \quad (4.42)$$

Com uma regra simples pode-se evidenciar os pontos que estão cobertos pelas áreas $i = dimensão\{obst\}$, como zonas interditas de passagem (índice = 1) e as restantes como zonas de circulação livre (índice = 0).

$$pix_{i,j} = \begin{cases} 1, & \text{se } pix_{ij} \in \mathbf{area}_i \rightarrow ocupado \\ 0, & \text{se } pix_{ij} \notin \mathbf{area}_i \rightarrow livre \end{cases}$$

Condensando toda a informação anterior, pode designar-se a matiz *OcupInv*, de dimensão *ij*.

$$\mathbf{OcupInv} = \begin{bmatrix} pix_{1,1} & \dots & pix_{1,j} \\ \vdots & \ddots & \vdots \\ pix_{i,1} & \dots & pix_{i,j} \end{bmatrix},\ pix_{i,j} \in [0, 1] \quad (4.43)$$

Aplicando um gradiente na escala [0, 255] a **OcupInv** obtém-se uma nova matriz **Goalgrad**, que evidencia as posições dos obstáculos.

Os pontos com máximo gradiente actuam como um campo de repulsão. Os pontos com menor gradiente actuam como campos de atracção.

$$\mathbf{Goalgrad} = grad_{[0,255]}\|OcupInv\| \tag{4.44}$$

4.3.3.2. Função de navegação

Para se idealizar uma trajectória, é necessário definir a origem como o ponto de máxima repulsão e a meta como o de máxima atracção. Para tal, cria-se uma função de navegação, com uma sub-rotina que simule um robô e procure o caminho que mais rapidamente tenda para zero.

Essencialmente, a ideia do robô virtual prende-se com um algoritmo, que para uma dada posição de início $pix_{i,j}$, procure de entre os pontos na sua vizinhança, qual é aquele que contém o maior vector (maior diferença de gradientes) entre esse ponto e a meta. A matriz de gradientes define o grau do gradiente na vizinhança de $pix_{i,j}$. Com a posição actual, aplica-se um ganho e introduz-se o resultado no robô virtual. Este calcula o ponto seguinte que maior inclinação produz no vector de gradientes.

Fazendo um ciclo para este processo, obtém-se um modelo que, para cada iteração, calcula o vector preferencial da trajectória e desvia-se igualmente dos obstáculos, contornando-os suavemente.

i) Sub-rotina do robô virtual

Simula-se o movimento através do escalar v, obtido através do vector **u**. O intervalo de tempo durante o qual o robô se desloca é fixo, e definido por $dt \in \mathbb{R}_0^+$. A variável *dq* representa o incremento da velocidade, função do gradiente $\vec{u}$, para as posições *x* e *y* na sua vizinhança. Na prática, o algoritmo recebe a posição actual e em função do gradiente, calcula qual a posição seguinte.

$$dq(x,y) = \mathbf{u} \times dt$$
$$robotX_{next} = robotX + dq(x)$$
$$robotY_{next} = robotY + dq(y)$$
$$robot\theta_{next} = \tan^{-1}\left(\frac{u_y}{u_x}\right) \tag{4.45}$$

Propõe-se o nome de "*moveRobot*" para esta sub-rotina.

$$[u,v,\theta]_{now} \xrightarrow{moveRobot} [u,v,\theta]_{next}$$

Fim da sub-rotina

4.3.3.3. Ciclo de navegação

Pode criar-se um ciclo de 1…n iterações, com a condição de paragem $\|robot_{now} - Goal\| < d,\ \ d \in \mathbb{R}^+$. A posição actual é definida pelo vector **robot** na iteração actual *n* $\mathbf{robot} = [robotX, robotY]_n$. O vector **gradf** contém os índices dos gradientes, para os pontos de cálculo actuais *robotX* e *robotY*. O ganho K ($K \in \mathbb{N}$) permite as acções de controlo do robô virtual.

$$\mathbf{gradf} = [robotX(Goalgrad_i);\ robotY(Goalgrad_j)] \tag{4.46}$$

O vector **u** representa o declive preferencial que o robô deve seguir, nas direcções *ij*.

$$\mathbf{u} = -K \times gradf \tag{4.47}$$

Aplicando esse vector à sub-rotina do robô virtual, obtém-se a nova posição.

$$robot_{now}(u, dt) \xrightarrow{moveRobot} robot_{next}\{u_{next}, v_{next}\} \tag{4.48}$$

A nova posição do robot é actualizada

$$robotX = u_{next}$$
$$robotY = v_{next} \tag{4.49}$$

A matriz **TRAJ**, guarda as actualizações das posições, desde o ponto de partida até à meta, alcançada na iteração *m*.

$$\mathbf{TRAJ} = [robotX, robotY]_m\ ,\ m = \{1, \dots, n\} \tag{4.50}$$

Quando se atingem as condições de paragem anteriores, o ciclo pára, indicando que a trajectória óptima entre o ponto de partida e a meta, contornando todos os obstáculos, está calculada. Com este passo conclui-se o ciclo da função de navegação.

4.3.3.4. Cálculo da trajectória do sistema de navegação

A trajectória calculada contém os pontos em pixéis, pelo que é necessário converte-los para metros. Aplica-se a sub-rotina "*bitmap2map*" idêntica à anterior, apenas inverte o processo de cálculo.

$$[u, v, w]_{pixels} \xrightarrow{bitmap2map} [x, y, \theta]_{meters}$$

A trajectória, em metros, designa-se por

$$\mathbf{TRAJ}_{meters} = [x, y]_m\ ,\ \ m = \{1, \dots, n\} \tag{4.51}$$

É importante referir que o número de iterações *m* é estritamente função da precisão que se deseje para o algoritmo. Aumentar esta precisão implica, que na proximidade da meta, o número de pontos com distâncias quase infinitesimais entre si aumenta consideravelmente. Assim sendo, é necessário estabelecer um parâmetro que garanta um igual número de pontos por metro. Designe-se este parâmetro por *resol*.

$$resol = {}^{nº\ pontos}/_{metro}$$

Calcule-se igualmente a distância total entre todos os pontos.

$$distance = \sum_1^m (\|Traj_{meters}\|_1 \dots \|Traj_{meters}\|_m) \tag{4.52}$$

$$h = \frac{distance}{resol}, \qquad h \in \mathbb{N}_0$$

Falta calcular uma trajectória, que, em função do nº pontos/ metro, estabeleça uma distância equidistante. Este passo permite a redução do número total de pontos, terminando com a influência do número de iterações da função de navegação. Este cálculo é feito através do processo de interpolação de uma sequência de pontos não repetidos numa curva bidimensional. A função *interpac* faz esta interpolação automaticamente. O seu código pode ser encontrado no CD da disseração.

$$\mathbf{TRAJ}_{meters} \xrightarrow{interpolação} \mathbf{TRAJ}_g \ , \ g = \{1, \dots, h\} \tag{4.53}$$

A matriz **TRAJ** guarda as posições, equidistantes, em metros, desde o ponto de partida até à meta, contornado os obstáculos no mapa.

$$\mathbf{TRAJ} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_h & y_h \end{bmatrix}_{meters} \tag{4.54}$$

Com a matriz (4.54), o campo de potências pode enviar para o sistema de controlo do robot o caminho que este deve seguir. A figura 4.5 apresenta o diagrama do modelo proposto.

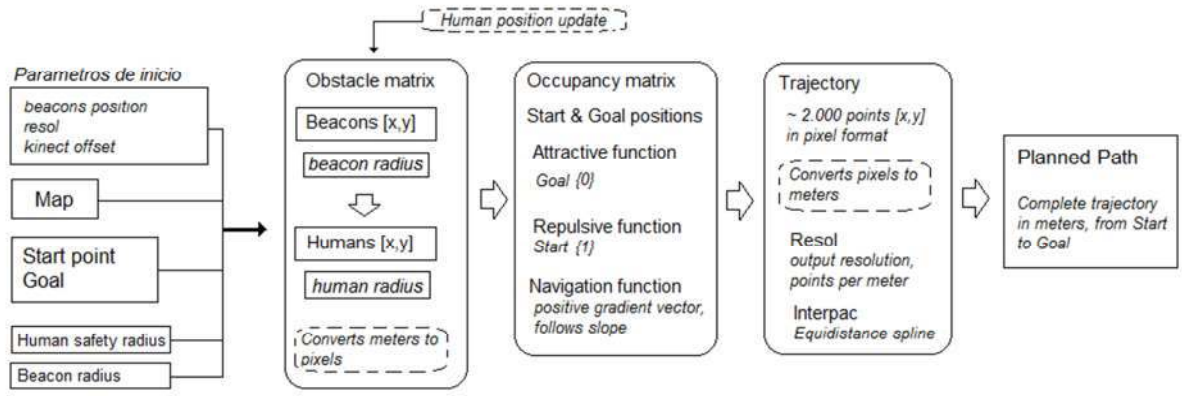


Figura 4.5 – Diagrama do algoritmo do campo de potencias

Resumo do algoritmo

O presente algoritmo faz uma representação simples do mundo baseado numa imagem *bitmap* convertida no formato *png*, que contem a região livre, extremos e obstáculos.

A localização cartesiana dos obstáculos é programada, definido como um círculo de raio discreto com centro nas coordenadas inseridas, $obst_i = [x, y, r]$. Os pontos de origem e meta são definidos na forma de coordenadas cartesianas[10]. O algoritmo calcula a trajectória com menor custo, evitando os obstáculos desde que o ponto inicial e meta estejam contidos na mesma região livre de navegação. O algoritmo vai iterando os pontos seguintes até o resultado estar dentro do intervalo proximo de zero. O conjunto de pontos cria a trajectoria desejada.

4.3.4. Algoritmo de navegação

O código que executa todo o processo proposto na dissertação está esquematizado na figura 4.6. Segue-se a explicação resumida do mesmo.

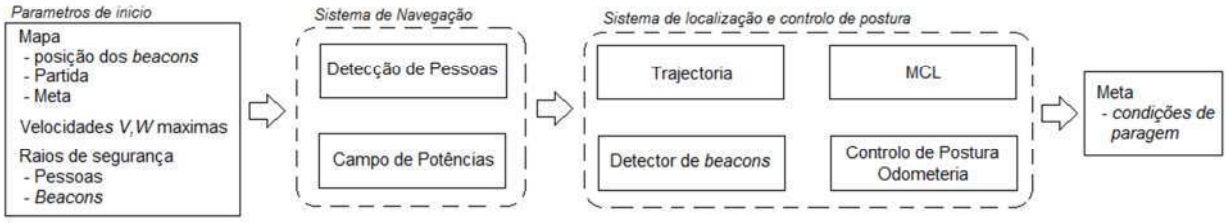


Figura 4.6 – Algoritmo geral do robô

[10] Efectivamente a utilização de coordenadas cartesianas implica a conversão em pixéis na posição [x,y] das mesmas, sendo que a origem deste referencial é o *pixel* [0,0] no canto superior esquerdo da imagem.

Este é o algoritmo principal da dissertação. O algoritmo junta todos os modelos propostos anteriormente e executa o conjunto de ordens necessárias para que o robô estabeleça uma trajectória baseada no ponto de início e meta desejada.

O modelo inicia com os parâmetros iniciais, como a posição da partida e meta, velocidades máximas, raios de segurança, desenho do mapa e posições dos *beacons*. Inclui igualmente todos os parâmetros necessários para cada um dos sistemas específicos.

Segue-se o sistema de navegação, onde se calcula a trajectória com o campo de potências. Se durante o trajecto for detectada uma ou mais pessoas, é neste bloco que se procedem às necessárias alterações ao caminho.

A informação passa para o sistema de localização, que faz uso do módulo já estudado do estimador MCL. Neste bloco, faz-se a detecção de *beacons*, a predição da posição baseada na odometria, a actualização da posição baseada nos *beacons* e o controlo de postura do robô. Este bloco repete-se num ciclo até que: (i) sejam atingidas as condições de meta ou (ii) seja detectada uma (ou mais) pessoa(s).

Se for o caso, repete-se o bloco do sistema de navegação, para cálculo de nova trajectória, da posição actual até à meta. Atingidas as condições de paragem da meta, o ciclo termina e o robô encerra os processos anteriores. Exporta-se para o ecrã o desvio padrão do erro de estimativa de posição ao longo do tempo, visualmente identificável como elipses de incerteza.

Todo o algoritmo, incluindo a descrição detalhada das suas linhas de código, pode ser encontrado no CD que acompanha a presente dissertação.

4.4. Resultados das simulações

Nesta secção apresentam-se os resultados da simulação dos sistemas de controlo e planeamento, apresentados neste capítulo. Inicia-se com as simulações de controlo de postura. Seguem-se as simulações do planeamento de trajectórias, considerando a presença de pessoas e obstáculos.

4.4.1. Simulação de movimento para uma postura

Esta secção pretende simular o controlo de postura, apresentado na secção 4.2.3. Foi desenvolvido um modelo em matlab/simulink para controlo de postura, conforme ilustra o diagrama de blocos da figura 4.7. O modelo de controlo da postura baseia-se no

modelo (Corke, 2011), devidamente alterado para representar um robô com sistema diferencial, conforme equação (4.21).

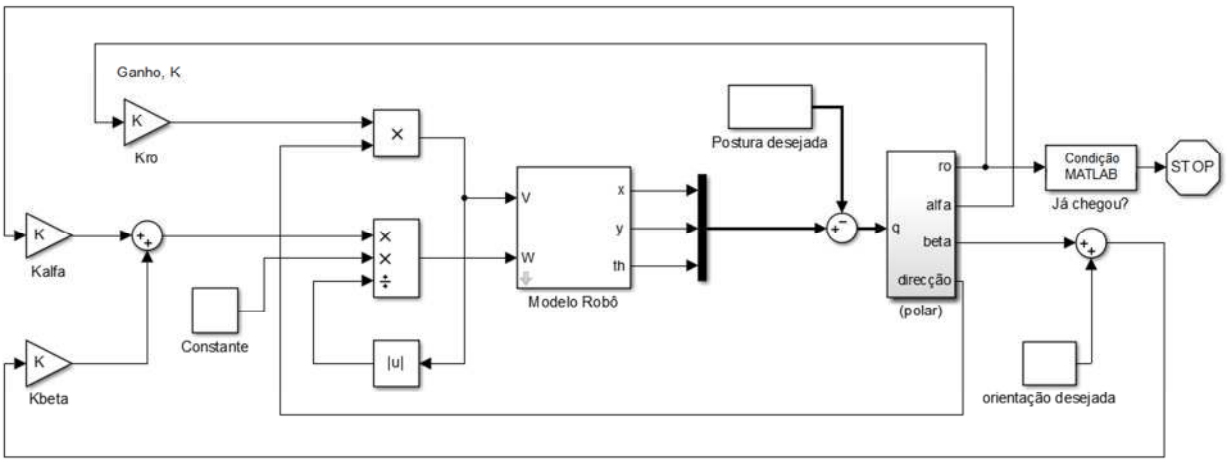


Figura 4.7 – Sistema de controlo da postura do robô

Foram feitos dois tipos diferentes de ensaios. No primeiro tipo, testaram-se os parametros de controlo que representam os ganhos da direção K_α, da distancia K_ρ, e da orientação K_β, não esquecendo as condições de estabilidade (4.23). No segundo ensaio, simulou-se o controlo de postura do robô.

A simulação, no primeiro tipo de ensaio, considerou como ponto de partida $[0, 0, \pi/2]$ e como meta $[2, 2, -\pi/2]$. O objectivo da mesma foi testar os parâmetros K_α, K_ρ e K_β. A figura 4.8 contém a representação dos ensaio destes ganhos. Conclui-se que os valores que apresentam uma trajectória mais suave para o robo percorrer são $K_\alpha = 9$, $K_\beta = -3$ e $K_\rho = 3$.

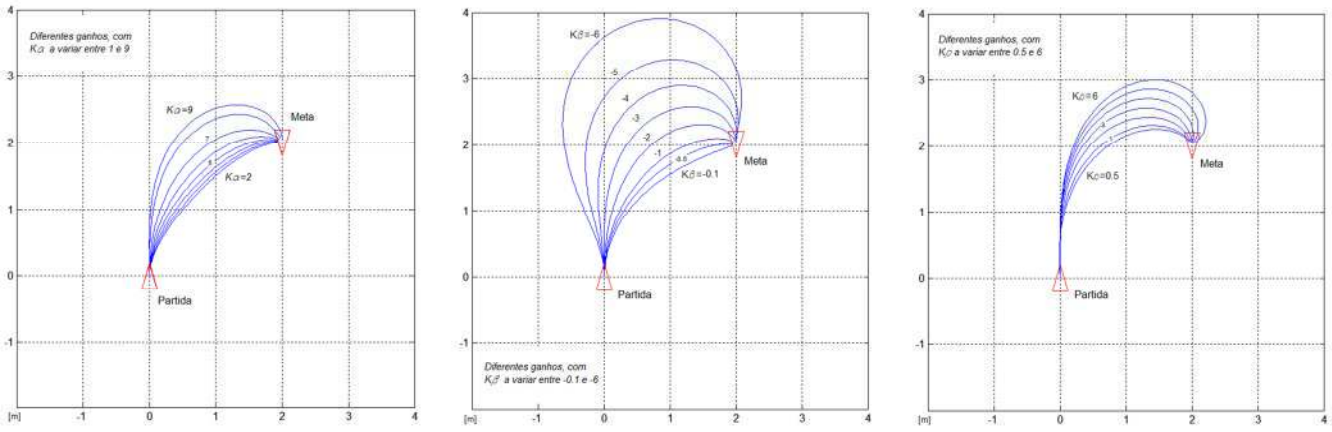


Figura 4.8 – Ensaio dos parâmetros a) K_α, b) K_β e c) K_ρ

O segundo ensaio faz uso dos parâmetros testados anteriormente e simula diversas posições de início do robô, com uma meta em $[2, 2, -\pi/2]$. Os pontos de partida e respectivas posturas ficam definidos na tabela 4.1.

Tabela 4.1 – Posturas iniciais

Início							
1	[0, 0, π/2]	5	[0, 4, π/2]	9	[4, 4, π/2]	13	[4, 0, π/2]
2	[-1, 1, π/2]	6	[1, 5, π/2]	10	[5, 3, π/2]	14	[3, -1, π/2]
3	[-2, 2, π/2]	7	[2, 6, π/2]	11	[6, 2, π/2]	15	[2, -2, π/2]
4	[-1, 3, π/2]	8	[3, 5, π/2]	12	[5, 1, π/2]	16	[1, -1, π/2]

A figura 4.9 ilustra o ensaio.

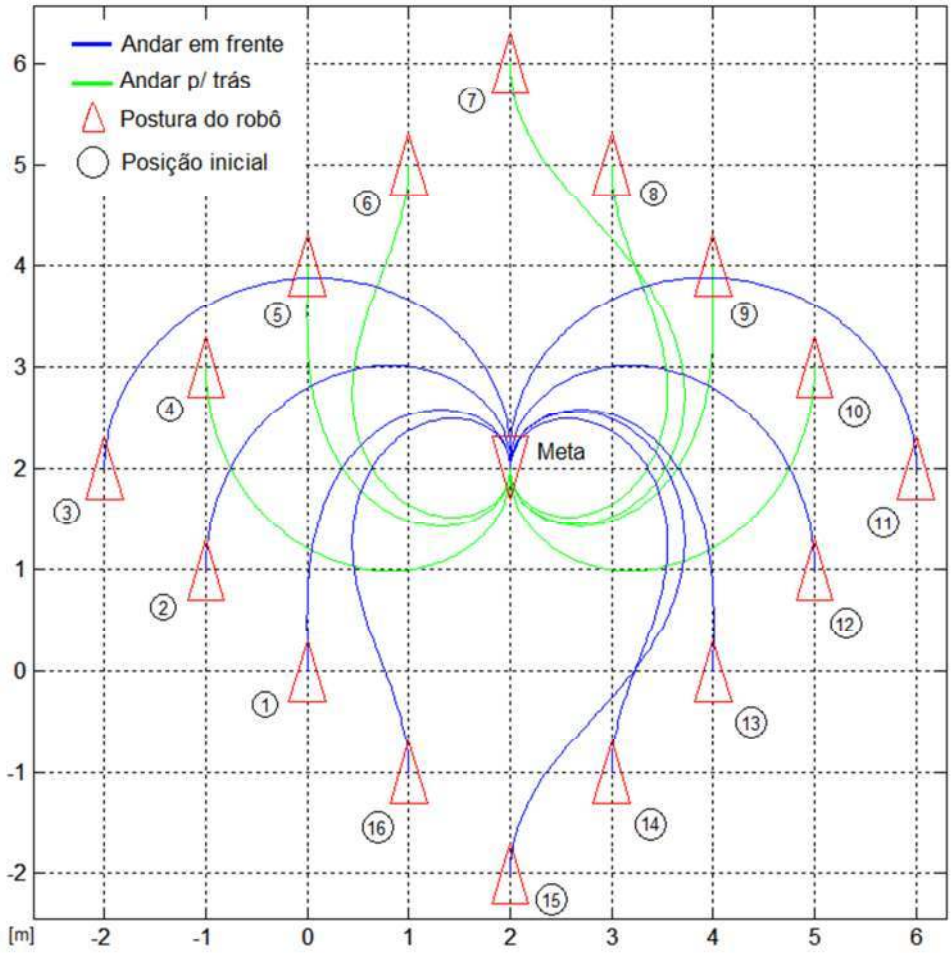


Figura 4.9 – Resultados da simulação do controlo de postura

Verifica-se que o robô tem em conta a orientação inicial e decreve o movimento que, de acordo com a lei de controlo aplicada, apresenta um caminho até à meta, de acordo com

os ganhos *K* escolhidos. É possível obter diferentes trajectórias alterando estes ganhos, não havendo nenhuma condição que garanta a menor distância de movimento até à meta. Pode concluir-se deste ensaio que o robô converge para a postura desejada, a partir de qualquer uma das posturas iniciais testadas.

Nos casos em que $\alpha \notin I_1 = \left\{-\frac{\pi}{2}, \frac{\pi}{2}\right\}$ existe uma inversão da velocidade ($v = -v$), conforme equação (4.19). Também é visível na experiência a imposição de sinal constante na velocidade, desde o ponto de partida até à meta, independetemente de $v \in I_1$ ou $v \in I_2$. Isto evita trajectórias com inversões abruptas da direcção, logo mais suavizadas para o objectivo proposto do robô.

As simulações demonstram um sistema de controlo estável com os ganhos K escolhidos, pelo que pode ser aplicado no robô real.

4.4.2. Simulação do campo de potências

Nesta secção são apresentadas as simulações do planemento de trajectórias desenvolvido na secção 4.3.3. O objectivo da simulação é a produção de um caminho do ponto de partida até à meta, evitando os obstáculos detectados e com um desenho o mais linear e suave possível. Este caminho é calculado através de um conjunto de pontos cartesianos. É fundamental criar um mecanismo que assegure uma distância entre pontos constante, para evitar que o robô seja forçado a calcular velociades lineares e angulares a enviar aos motores em cada milimetro de avanço. O algoritmo proposto resolve este problema, segmentando qualquer curva complexa "*spline*" em pontos cartesianos de igual afastamento (distância euclidiana). Dado que o afastamento é parametrizado, o resultado permite uma resolução ajustavel no caminho, sem afectar o desenho do mesmo.

4.4.2.1. Simulação da trajectória com *beacons*

O ensaio de navegação decorre no mundo centrado em [0,0]m. Os extremos do mundo são definidos com base no mapa predefinido com dimensões laterais 10m, idênticas em x e y. Definem-se 6 obstáculos, conforme tabela 4.2

Tabela 4.2 – Posição dos obstáculos

Obstáculo	Posição [m]
*obst*1	[0.80, 3.30]
*obst*2	[-1.30, 3.70]
*obst*3	[0, 4.05]
*obst*4	[0.50, 4.05]
*obst*5	[-0.50, 2.80]
*obst*6	[1.50, 1.50]

Para cada *beacon* é definido um raio de segurança $r_{beacons}$. Partindo do ponto de origem, definido em [0, 0, pi/2] e meta em [1.3, 4.6, pi/2], o algoritmo procura a trajectória contida na região definida que apresente o menor custo e contorna os obstáculos. A resolução é $[500 \times 500]$ pixel, com $resol = 5\, pontos/metro$ para o caminho calculado.

Para o primeiro ensaio, assume-se $r_{beacons} = 0.50$ m. A figura 4.10 apresenta o resultado em gradiente, com enfâse no campo repulsivo dos obstáculos.

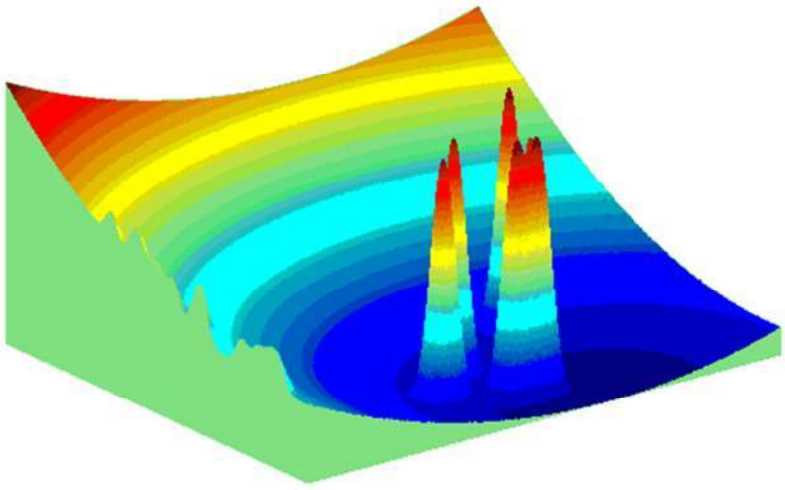


Figura 4.10 – Imagem de gradientes do campo de potências

A figura 4.11 apresenta uma imagem bidimensional do caminho óptimo calculada pelo algoritmo. Nesta fase, os pontos estão calculados em escala de pixéis. Notar que o caminho segue um arco que resulta do balanço entre as forças atractivas e repulsivas, função do raio de segurança centrado no obstáculo.

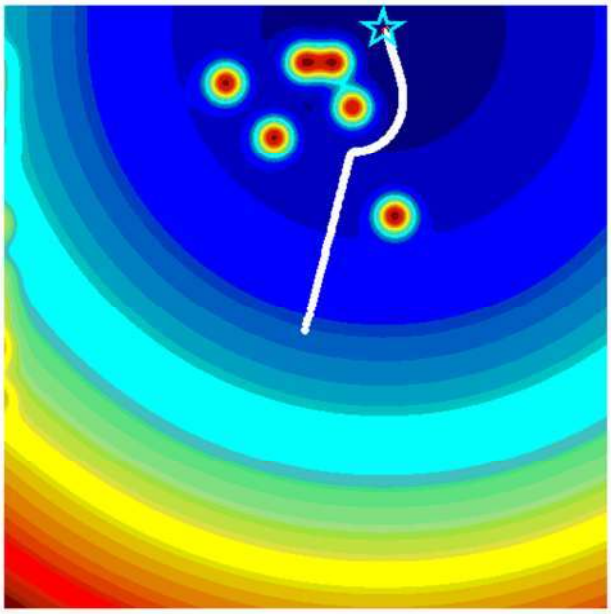


Figura 4.11 – Representação do mundo na simulação do campo de potências

A figura 4.12 faz a representação esquemática do caminho, já convertido em metros, incluindo o raio de segurança $r_{beacons} = 0.50$ m. O caminho segue a menor distância possível da partida até à meta, apenas produzindo alteração quando está na área de influência do *beacon*.

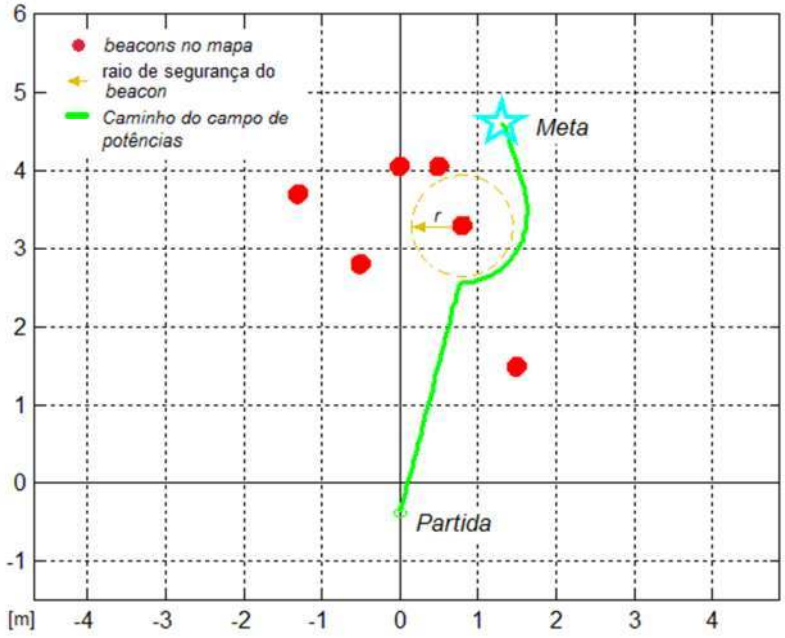


Figura 4.12 – Planemeamento do caminho com campos de potência

Simularam-se diferentes raios de segurança dos *beacons*, conforme é visível na figura 4.13. A simulação a) $r_{beacons} = 0.10$ m contém um caminho rasante ao *beacon*. Em b)

$r_{beacons} = 0.20$ m, o caminho alarga, no entanto convêm não ignorar as dimensões físicas do robô. Em c) $r_{beacons} = 0.80$ m exagera-se propositadamente o raio, o que produz um caminho com um arco consideravelmente maior. Note-se que amentar o raio alarga a influência do campo de repulsão em torno do obstáculo, o que pode fazer com que haja sobreposição, ($r_{beacons} > {}^{l}/_{2}$), $l = distancia\ entre\ beacons$, efectivamente bloqueando um possível caminho, se estes estiverem em linha recta entre o ponto de partida e a meta.

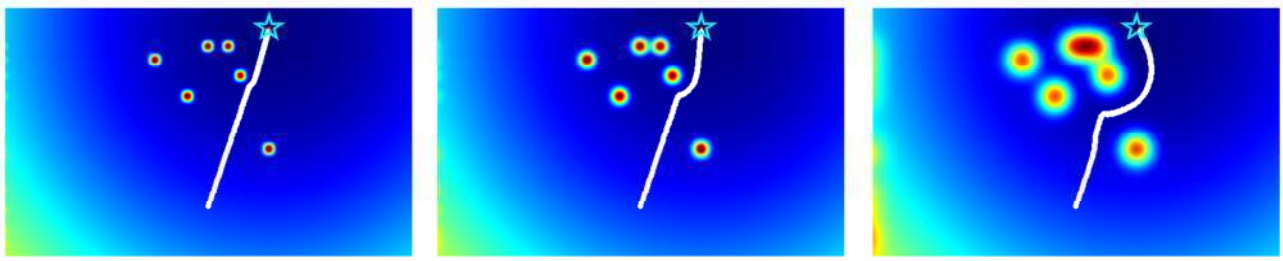


Figura 4.13 – Caminhos com diferentes raios de segurança nos *beacons*

Neste caso, estamos perante um caso de mínimos locais, o que torna o algoritmo incapaz de calcular um caminho até à meta. Para evitar esta situação, deve garantir-se $r_{beacons} < {}^{l}/_{2}$.

4.4.2.2. Simulação do caminho com detecção de pessoas

A presença de humanos é feita com recurso ao algoritmo de deteção de humanos, que exporta os resutlados numa matriz com coordenadas cartesianas da posição das pessoas detectadas. Com base nesta posição define-se um círculo centrado nesse ponto, e cujo raio é definido pela variável r_{people}. A secção 3.2.7 aborda este tema com maior detalhe.

A simulação é feita com as mesmas condições da simulação de *beacons*, com partida em [0, 0, pi/2], meta em [1.3, 4.6, pi/2] e posição dos obstáculos definida na tabela 4.2. Partindo do ensaio anteriror, é feita a deteção de uma pessoa. O algoritmo acrescenta a posição cartesiana desta, incluindo o raio de segurança $r_{people} = 0.50$ m. Resulta um caminho que evita a posição da pessoa detectada, visivel na figura 4.14.
Em qualquer altura o algoritmo pode ser chamado caso haja nova detecção de pessoas, recalculando um novo tcaminho baseado na posição actual do robô e a posição da meta.

Verificou-se nesta fase de ensaios a necessidade de acrescentar no algoritmo uma condição de paragem do robô quando a área de segurança da pessoa cobre a posição do alvo.

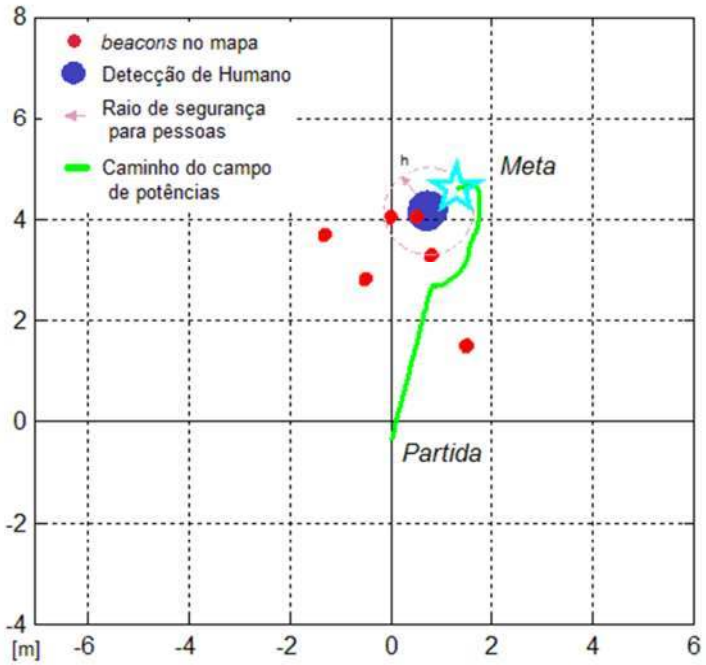


Figura 4.14 – Caminho actualizado perante a detecção de uma pessoa

Conclui-se desta fase de simulações que o algoritmo do campo de potências é uma boa escolha para providenciar um caminho ao sistema de navegação do robô, pois consegue produzir caminhos facilmente actualizáveis na presença de pessoas, com uma resolução de 0.05m entre pontos equidistantes, não repetidos. O factor mais importante é o tempo de cálculo do processo, que nunca ultrapassou $t_{trajecto} = 0.5$ seg na fase de actualização (pessoa detectada após inicio do caminho). O tempo de computação deve ser o menor possível, de modo a se obter um sistema de navegação com um reduzido tempo de amostragem, aumentando a velocidade de reacção no desvio de obstáculos e pessoas.

4.5. Ensaios experimentais do algoritmo geral

Foram realizados 82 ensaios experimentais para testar o sistema de navegação, em conjunto com a detecção de *beacons* e pessoas. Destes, 21 dizem respeito ao algoritmo completo com todas as funções e serviram para afinar os inúmeros parâmetros que compõem os diversos blocos, para que todos funcionem em uníssono. Apresentam-se alguns ensaios experimentais que representam as diversas etapas de teste.

Condições comuns aos ensaios.

O robô tem a distância entre rodas L=0.484 m e o raio da roda r=0.058 m. O mapa tem a dimensão 10×10 m. Os *beacons* têm um raio de segurança $r_{beacons} = 0.20$ m. Utilizaram-se os valores de co-variância:

Co-variância da odometria, $W = [0.1^2,\ 0.5^2]$ $[\mathrm{m}^2, \mathrm{rad}^2]$

A co-variância do sensor visual (V) foi estimada no capítulo 2.4.2.5.

$V = [0.1486,\ 0.0564]$ [m, rad]

Co-variância da incerteza do conjunto de pontos do filtro de partículas, em cada iteração, $Q = [0.05^2,\ 0.05^2,\ 1^2]$ [m, m, rad]

Incerteza da estimativa obtida do filtro de partículas, $L = [0.10,\ 0.10]$

Os valores de Q e L não foram estimados, utilizando-se por isso valores típicos. Estes valores de co-variância foram adoptados para os restantes ensaios, dado conseguirem produzir uma convergência aceitável.

4.5.1. Fase de ensaios inicial

O primeiro ensaio é composto pelo robô, na posição de partida $[0, -0.5,\ {}^{\pi}/_{2}]$ com a meta $[2.5, -0.5,\ {}^{\pi}/_{2}]$. Os *beacons* estão definidos na tabela 4.3.

Tabela 4.3 – Posição dos *beacons*

Beacons	Cor Referência	Posição [m]
*beac*1	Vermelho	[0.80, 3.30]
*beac*2	Laranja	[0, 4.05]
*beac*3	Verde	[-1.30, 3.70]
*beac*4	Azul	[0.50, 4.05]
*beac*5	Castanho	[1.50, 1.50]
*beac*6	Amarelo	[-0.50, 2.80]

A figura 4.15 apresenta um excerto da informação disponível após o robô completar a trajectória e navegar até à meta.

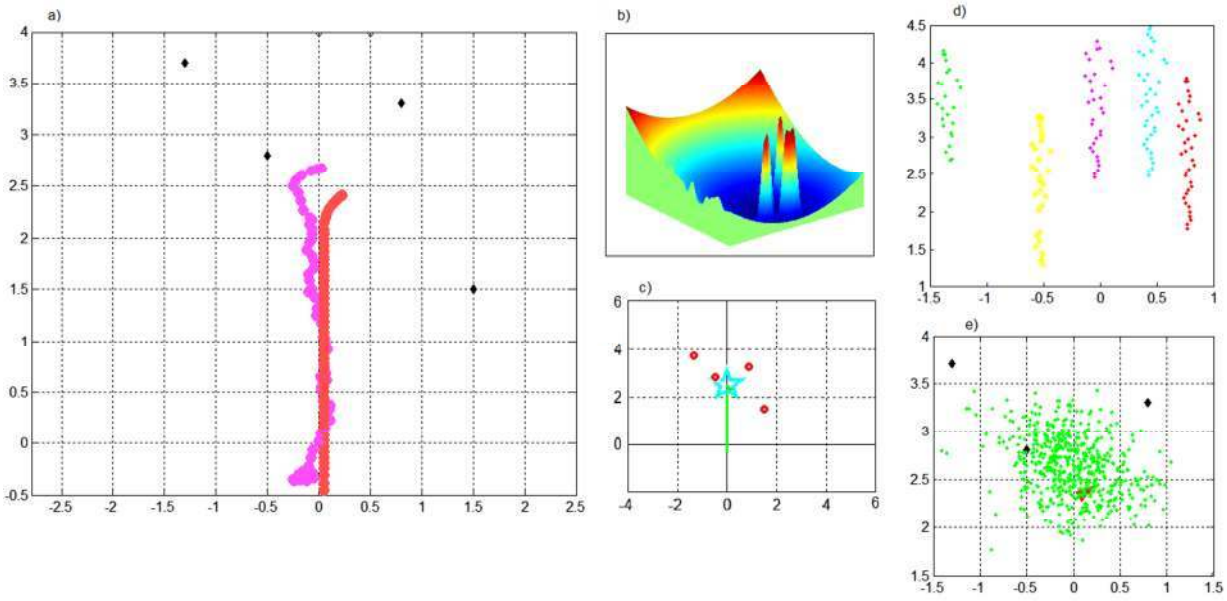


Figura 4.15 – Ensaios do modelo completo do robô

A figura *b)* mostra o cálculo inicial feito pelo campo de potências. A figura *c)* esquematiza a trajectória que o robô deve percorrer, desde o ponto de partida até à meta, incluindo a posição dos *beacons*. A figura *a)* esquematiza a trajectória percorrida pelo robô, calculada a partir do estimador MCL, ao longo do tempo. Os pontos estimados pela odometria estão representados pela côr vermelha e os pontos estimados pelo MCL pela côr rosa. O objectivo do ensaio é que estas duas linhas sejam coincidentes, garantido um caminho óptimo. Neste ensaio verificaram-se algumas discrepâncias, nomeadamente a necessidade de alterar, de entre vários parâmetros, a resolução da trajectória para evitar movimentos bruscos do robô. Verificou-se igualmente um desvio do mesmo junto à posição da meta, o que veio a demonstrar a necessidade de criar condições de paragem adicionais além da posição de meta. A figura *d)* mostra a posição dos *beacons* de cor correspondente, detectados pela Kinect. Notar que estes não se encontram perfeitamente alinhados ao longo do tempo, resultado da trajectória algo brusca do robô neste ensaio. A figura *e)* mostra a nuvem de pontos aleatórios criados pelo estimador MCL. A média dos pesos destes pontos resulta na posição assumida pelo estimador como a posição correcta do robô, com uma percentagem de incerteza associada.

4.5.2. Fase dos ensaios de convergência do MCL

O segundo ensaio é composto pelo robô, na posição de partida $[0, 0, \pi/2]$ com a meta $[-0.50,\ 2.00,\ \pi/2]$. Os *beacons* estão definidos na tabela 4.4.

Tabela 4.4 – Posição dos *beacons*

Beacons	Cor Referência	Posição [m]
*beac*1	Vermelho	[0.80, 3.30]
*beac*2	Laranja	[-1.35, -0.80]
*beac*3	Verde	[-1.30, 3.70]
*beac*4	Azul	[0.00, 4.05]
*beac*5	Castanho	[1.50, 1.50]
*beac*6	Amarelo	[-0.50, 2.80]

Este ensaio foi pontuado pela afinação dos parâmetros que compõem o estimador MCL, nomeadamente as co-variâncias. A figura 4.16 apresenta esquematicamente o ensaio.

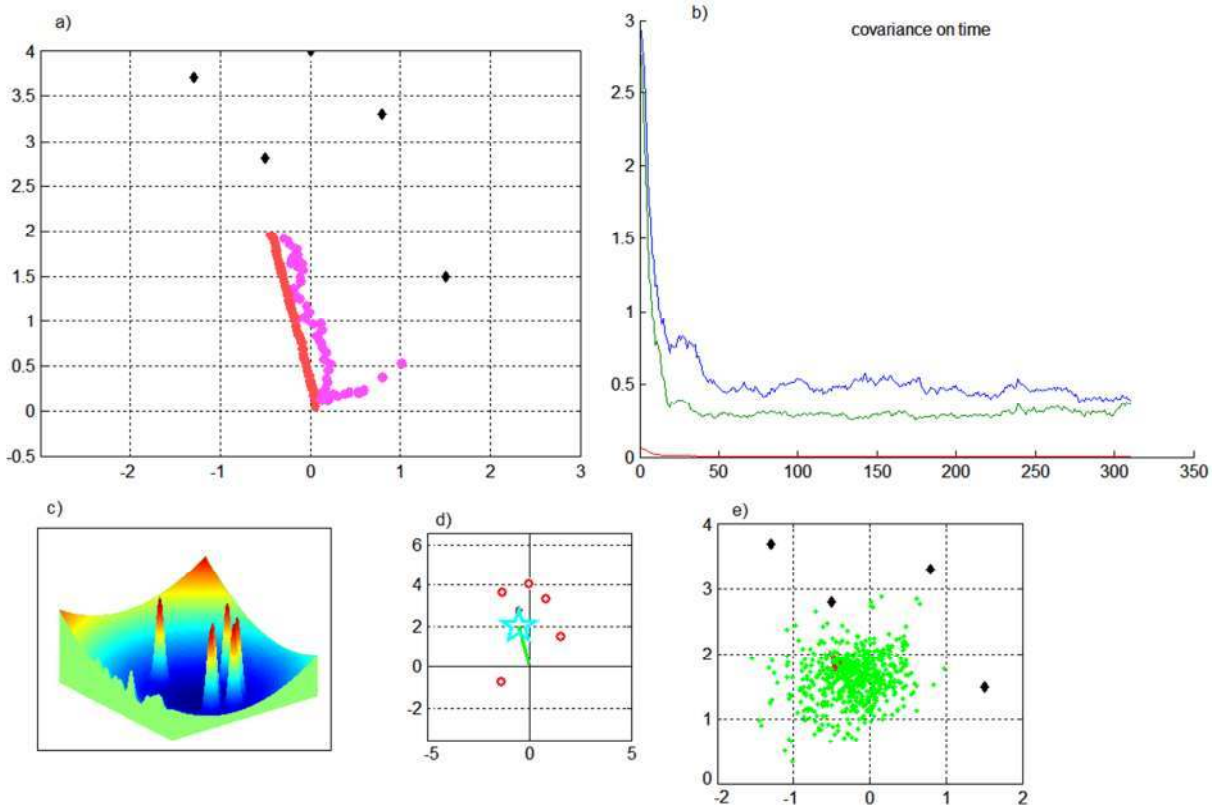


Figura 4.16 – Ensaio do modelo completo do robô

Pode observar-se em *a)* a trajectória percorrida pelo robô, desde o ponto de partida até à meta. A figura *c)* demostra os obstáculos calculados pelo campo de potências, a figura d) a trajectória ideal. A figura *e)* mostra a nuvem de pontos no último instante antes do

robô atingir as condições de meta. Neste ensaio destaca-se a figura *b)* que mostra o gráfico da co-variância do estimador no tempo, para os eixos *XX* (verde),*YY* (azul) e a orientação θ (vermelho). No início do ensaio pede-se ao estimador a posição do robô, durante 25 iterações, nas quais não existe movimento (forçar-se $v = \omega = 0$). Este procedimento foi adoptado, por se ter verificado algumas dificuldades na convergência inicial do estimador. Notar que ambos os eixos *XX* e *YY* mantêm uma convergência aproximadamente constante ao longo do tempo.

4.5.3. Fase de ensaios com todos os blocos, sem presença de pessoas.

O terceiro ensaio é composto pelo robô, na posição de partida $[0, 0.50, {}^{\pi}/_{2}]$ com a meta $[-1.00,\ 3.00,\ \pi]$. Os *beacons* estão definidos na tabela 4.5.

Tabela 4.5 – Posição dos *beacons*

Beacons	Cor Referência	Posição [m]
*beac*1	Vermelho	[0.80, 3.30]
*beac*2	Laranja	[-1.35, -0.80]
*beac*3	Verde	[-1.30, 3.70]
*beac*4	Azul	[0.00, 4.05]
*beac*5	Castanho	[1.50, 1.50]
*beac*6	Amarelo	[-1.50, 1.50]

A figura 4.17 apresenta graficamente o ensaio.

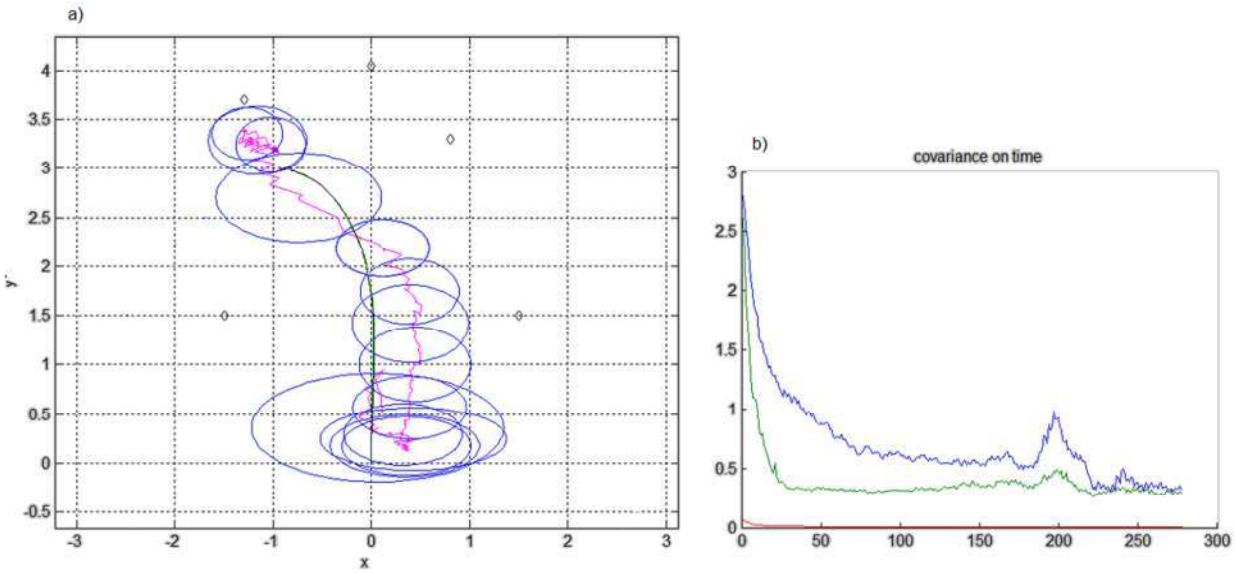


Figura 4.17 – Ensaio do modelo completo do robô

Verifica-se, na figura b) a convergência do estimador MCL, desde o ponto de início até a meta, inclusive a perda momentânea e nova captura da localização dos *beacons* entre os *frames* 180 e 200, com nova convergência para a meta. Estes picos verificaram-se aparecer em muitos dos ensaios. Analisado as imagens da detecção nas frames mencionadas, é possível observar a redução do número de *beacons* observados, o que leva a um aumento instantâneo da incerteza. A elipse de incerteza, visível na figura a) demonstra o desvio padrão da co-variância no tempo do estimador, entre a posição estimada e a real posição do robô, que se reduz à medida que a certeza aumenta.

4.5.4. Fase de ensaios com todos os blocos, com presença de pessoas.

O último ensaio é composto pelo robô, na posição de partida $[-0.20, -0.50, {}^{\pi}/_{2}]$ com a meta $[0.20,\ 2.20,\ {}^{\pi}/_{2}]$. Os *beacons* e a pessoa estão definidos na tabela 4.6. O raio de segurança da pessoa $r_{people} = 0.50$ m

Tabela 4.6 – Posição dos *beacons* e pessoa

Beacons	Cor Referência	Posição [m]
*beac*1	Vermelho	[0.80, 3.30]
*beac*2	Laranja	[-1.30, 3.60]
*beac*3	Verde	[-1.30, -0.80]
*beac*4	Azul	[0.00, 3.70]
*beac*5	Castanho	[1.50, 1.50]
*beac*6	Amarelo	[0.00, -1.10]
Pessoa	-	[0.20, 3.10]

A fase final dos ensaios inclui a navegação com a presença de pessoas. (ver figura 4.18)

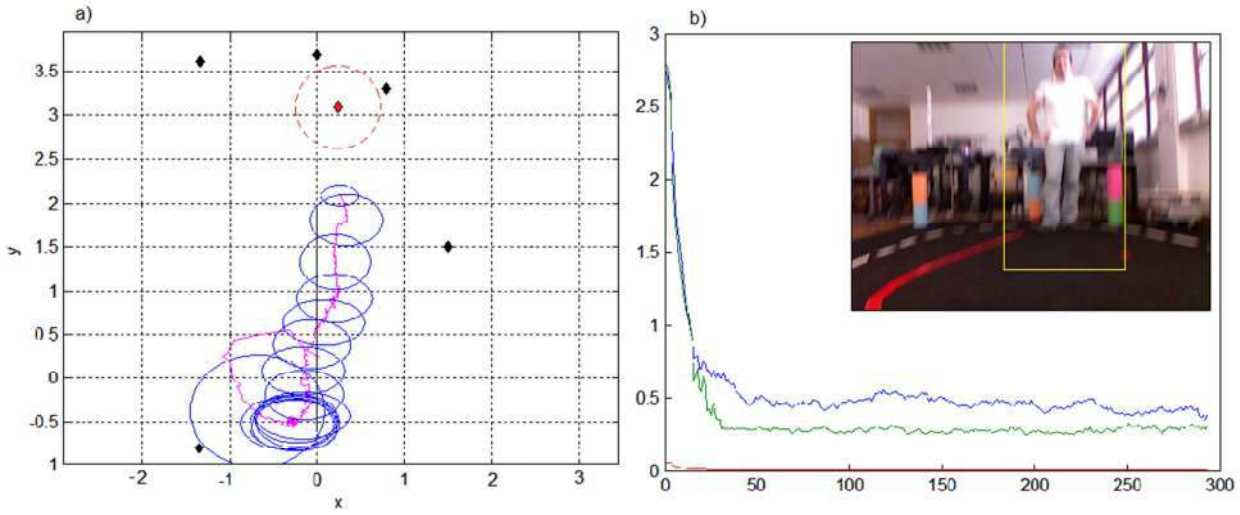


Figura 4.18 – Ensaio do modelo completo do robô

Notar que a trajectória inicial do estimador recebe, por imposição, velocidade nula, o que resulta na estimação errática nos instantes iniciais, com convergência visível a partir da *frame* 20. Destes ensaios verificou-se ser necessária uma condição para evitar movimentos erráticos e o encerrar abrupto do robô, quando este detecta uma pessoa, ou seja: Condição de apenas calcular uma nova trajectória, se se detectar o desvio da pessoa da posição inicial *i*, já processada, para uma nova posição *j*, fora do raio de não-calculo $\|i - j\| \gg r_{dist}$. Essencialmente cria-se um círculo com uma pequena área, centrado na posição inicial da primeira detecção da pessoa, e evita-se o cálculo, desde que as novas detecções da *mesma pessoa* coloquem a sua posição dentro desta área. Esta condição foi aplicada e resultou num movimento fluido do robô, perante detecções constantes de pessoas, conforme figura *a)* demonstra. Igualmente visível na figura é a representação da elipse de incerteza do estimador, calculada em pontos de amostra. Os eixos da elipse são compostos pela incerteza em *xx* e *yy*, conforme gráfico da co-variância no tempo (figura b). A orientação $\theta \approx 0$, está representada a vermelho na figura b, onde se apresenta também uma imagem capturada num dos momentos da detecção de pessoa. Notar que, mesmo perante a oscilação do robô, que produz algumas imagens desfocadas, o detector consegue captar a pessoa.

4.5.5. Conclusões dos ensaios de navegação

O robô foi testado com todos os módulos activados nos últimos 21 ensaios realizados, pelo que se podem concluir os seguintes pontos:

i) A convergência do estimador está directamente ligada com o número de *beacons* visíveis na imagem, o que pode levar a situações de divergência sempre que o robô deixa de captar um ou mais *beacons*.

ii) As condições de iluminação, embora não mencionadas nos ensaios de navegação, influenciam grandemente a capacidade de detecção dos *beacons*, o que torna necessário verificar a capacidade de captura antes de executar o ensaio de navegação.

iii) O campo de potências demostra ser uma boa escolha, com capacidade de actualização da trajectora em $t < 0.50$ s quando se dá a detecção de uma pessoa. Para este resultado é obrigatória uma resolução $N_{pix} = [500 \times 500]$. Resoluções superiores causam tempos de processamento superiores a 1 s, o que torna o movimento do robô errático e pouco fluído.

iv) O sistema de controlo de postura demostrou ser estável para valores de $K_\alpha = 9$, $K_\rho = 3$ e $K_\beta = -3$.

v) O raio de segurança para pessoas, $r_{people} = 0.50$ m, ensaiado demostrou ser o suficiente. Numa aplicação de maior escala, recomenda-se o raio social $r_{social} \geq 0.80$ m, conforme proposto por Trautman *et al* (2013).

Conclusões

A presente dissertação demonstra o desenvolvimento de um sistema de navegação do robô, na presença de pessoas no mundo.

Apresentou-se um sistema visual de detecção de beacons, que permite captar e calcular a posição tridimensional destas referências no mundo. O sistema faz a detecção de padrões de cor na imagem, que representam os alvos. Determina-se de seguida a sua posição relativa ao robô. Conclui-se com a aplicação de uma rede neuronal artificial, para melhoria da informação. Os ensaios feitos validam o modelo proposto.

O sistema de localização foi desenvolvido com base num estimador de Monte Carlo. Este trabalha em conjunto com os sensores visuais e odométricos para estimar a postura do robô móvel. Este método tem como principais vantagens a localização global do robô e a utilização de referências exteriores (*beacons*) para estimar a postura. O sistema de localização proposto foi validado com ensaios de localização.

O sistema proposto para a detecção humanos permite detectar, em tempo real, pessoas no mundo. Este foi desenvolvido com base num sistema visual que detecta a silhueta humana e calcula a sua posição no mundo. Com esta posição conhecida, pode assegurar-se a distância de segurança física e psíquica das pessoas relativamente ao robô móvel. Os ensaios efectuados validam a confiança neste método de detecção visual.

O controlo de postura do robô foi desenvolvido com base num modelo de controlo diferencial. Este apresenta vantagens ao não alterar bruscamente a trajectória calculada. Este modelo foi posteriormente validado com recurso a simulações.

O sistema de navegação do robô móvel proposto funde os sistemas anteriores. Este permite que o robô móvel calcule um caminho inicial, desde a partida até à meta, e altere este caminho em tempo real, caso se detectem pessoas no mundo. O robô respeita as distâncias de segurança das pessoas, produzindo um novo caminho até à meta. O sistema de navegação proposto tem por base os campos de potências. Este apresenta como principais vantagens o cálculo de trajectórias que incluem o desvio de obstáculos no mundo, sejam estes pessoas ou os *beacons*.

A tese concluiu-se com os ensaios do sistema proposto completo do robô móvel, incluíndo ensaios de navegação com e sem a presença de pessoas.

Desenvolvimentos Futuros

Propõe-se continuar os testes dos algoritmos de navegação e detecção de humanos, de modo a que as componentes teóricas e simuladas sejam testadas em tempo real, com todas as variáveis não previstas que um teste desta natureza obriga.

O desenvolvimento contínuo destes algoritmos prende-se com situações pontuais que se verificaram durante os ensaios, nomeadamente a problemática de uma curva onde o sistema de visão deixa de ver temporariamente os *beacons*, quando estes estiverem tapados por pessoas durante a curva do robô ou a variabilidade das condições de iluminação afectarem a captura das cores e influenciarem a detecção dos *beacons*.

Apresenta-se o desafio de utilizar tecnologias alternativas ou complementares para a detecção mais robusta dos *beacons*, como *rangefinders* ou radiofrequência, de modo a ultrapassar algumas das limitações encontradas ao longo do desenvolvimento da presente dissertação.

Referências Bibliográficas

Arambula, Cosío F., Padilla Castañeda. "*Autonomous robot navigation using adaptive potential fields"*, Mathematical and Computer Modelling, Vol. 40, Issues 9–10, pp. 1141–1156, Elsevier, 2004. doi: (Digital Object Identifier) 10.1016/j.mcm.2004.05.001

Biswas, Joydeep, Manuela Veloso. "*Depth Camera Based Indoor Mobile Robot Localization and Navigation"*, 2012 IEEE International Conference on Robotics and Automation, pp. 1697 – 1702, May 14-18, 2012. doi: 978-1-4673-1405-3/12

Burt, Peter and Ted Adelson. "*The Laplacian pyramid as a compact image code"*. IEEE Transactions on Communications 9(4), pp. 532–540, 1983. doi: 10.1109/tcom.1983.1095851.

Bay, Herbert, Tinne Tuytelaars and Luc van Gool. "*SURF: Speeded up robust features*. *Proc. 9th European Conference on Computer Vision (ECCV'06)"*, Springer Lecture Notes in Computer Science, 3951, pp. 404-417, 2006. doi: 10.1007/11744023_32.

Benavidez, P. "*Mobile robot navigation and target tracking system"*, System of Systems Engineering (SoSE), 6th International Conference, IEEE, pp. 299 – 304, June 2011. doi: 10.1109/SYSOSE.2011.5966614

Bay, Herbert, Andreas Ess, Tinn Tuytelaars and Luc van Gool. *SURF: Speeded up robust features*. *Computer Vision and Image Understanding*, 110(3), pp. 346-359, 2008. doi:10.1016/j.cviu.2007.09.014.

Campos, Francisco, L. Correia, and J. M. F. Calado, "*Robot Visual Localization through Local Feature Fusion: an Evaluation of Multiple Classifiers Combination Approaches*," Journal of Intelligent and Robotic Systems, Vol. 77, n. 2, pp. 377-390, 2015. doi: 10.1007/s10846-013-0016-3.

Comaniciu, D., "*An algorithm for data-driven bandwidth selection"*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(2), pp. 281–288, 2003. doi: 10.1109/TPAMI.2003.1177159

Corke, Peter. "*Robotics, Vision and Control: Fundamental Algorithms in MATLAB"*, Volume 73. Queensland University of Technology, Australia, Springer Science & Business Media, pp. 65-78, 2011. doi: 10.1007/978-3-642-20144-8

Chen, S. Y. "*Kalman Filter for Robot Vision: A Survey"*. IEEE Transactions on Industrial Electronics, vol. 59, no. 11, pp. 4409 – 4420, November 2012. doi: 10.1109/TIE.2011.2162714

Carreira, Fernando, J.M.F. Calado, C. Cardeira, *"A Mobile Robot Navigation Planning in a Human Populated Environment"*, in Proceedings of the 11th International Conference on Mobile Robots and Competitions, Robotica 2011.

Carreira, Fernando, C. Christo, D. Valerio, M. Ramalho, C. Cardeira, J.M.F. Calado, P. Oliveira, "*2D PCA-based localization for mobile robots in unstructured environments,*" in Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference, pp.3867-3868, 7-12 Oct. 2012 doi: 10.1109/IROS.2012.6386272

Carreira, Fernando, João M. F. Calado, Carlos Cardeira, Paulo Oliveira. *"Enhanced PCA-Based Localization Using Depth Maps with Missing Data, Experimental Validation"*, Journal of Intelligent & Robotic Systems, Vol. 77, Issue 2, pp. 341-360, Dordrecht, 2014. doi: 10.1007/s10846-013-0013-6

Carreira, Fernando, J.M. Ferreira Calado, C. Cardeira, P. Oliveira. "*Complementary Filter Design with Three Frequency Bands: Robot Attitude Estimation*," in Autonomous Robot Systems and Competitions (ICARSC), 2015 IEEE International Conference, pp.168-173, 8-10 April 2015. doi: 10.1109/ICARSC.2015.33

Crowley, James L. and Richard M. Stern. "*Fast computation of the difference of low pass transform"*. IEEE Transactions on Pattern Analysis and Machine Intelligence 6(2), pp. 212-222, 1984. doi:10.1109/tpami.1984.4767504.

Crowley, James L. and Olivier Riff. *"Fast computation of scale normalised Gaussian receptive fields*. *Proc. Scale-Space'03",* Springer Lecture Notes in Computer Science 2695, pp. 584-598, 2003. doi: 10.1007/3-540-44935-3_41.

Cao, M., L. Yu, and P. Cui. "*Simultaneous localization and map building using constrained state estimate algorithm"*, in Proc. Chin. Control Conf., pp. 315–319, 2008. doi: 10.1109/CHICC.2008.4605126

Dalal, Nadal and Bill Triggs. "*Histograms of oriented gradients for human detection"*. Proc. Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, Vol. I, pp. 886-893, 2005. doi: 10.1109/CVPR.2005.177

Desouza, G.N. "*Vision for mobile robot navigation: a survey"*, Pattern Analysis and Machine Intelligence, IEEE, Vol. 24, Issue 2, pp. 237 – 267, 2002. doi: 10.1109/34.982903

Ganganath, N. and H Leung. "*Mobile Robot Localization Using Odometry And Kinect Sensor"*, Department of Electrical and Computer Engineering Schulich School of Engineering, pp. 91 – 94, University of Calgary, 2012. doi: 978-1-4673-0898-4/12

Gavrila, D. M., J. Giebel, and S. Munder. "*Vision-based pedestrian detection: the protector+ system"*. In Proc. of the IEEE Intelligent Vehicles Symposium, Parma, Italy, 2004. doi: 10.1109/IVS.2004.1336348

Gavrila D. M. and V. Philomin. "*Real-time object detection for smart vehicles"*. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Fort Collins, Colorado, USA, pp. 87–93, 1999. doi: 10.1109/ICCV.1999.791202

Holmes, S., G. Klein, and D. Murray. "*A square root unscented Kalman filter for visual mono SLAM",* in Proc. IEEE Int. Conf. Robot. Autom., pp. 3710–3716, 2008. doi: 10.1109/ROBOT.2008.4543780

Huang F. S. and K. T. Song. "*Vision SLAM using omni-directional visual scan matching",* in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., pp. 1588–1593, 2008. doi: 10.1109/IROS.2008.4650919

Huang, Wen-Tsai, Tsai, Chun-Lung and Lin, Huei-Yung, "*Mobile Robot Localization using Ceiling Landmarks and Images Captured from an RGB-D Camera"*. The 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 855-860, July 11-14, 2012. doi: 978-1-4673-2576-9/12

Horn K.P. and G. Schunck. "*Determining optical flow"*. Artificial Intelligence, 17, pages 185-203, 1981. DOI: 10.1117/12.965761

Huanga, Wesley H., Brett R., Finka Fajenb, R. Jonathan and William H. Warrenc. "*Visual navigation and obstacle avoidance using a steering potential function"*, Robotics and Autonomous Systems, Volume 54, Issue 4, pp. 288–299, Elsevier, 2006. doi: 10.1016/j.robot.2005.11.004

Haykin, Simon, *Neural Networks: "A Comprehensive Foundation"*, International edition, Prentice Hall International Editions Series, 1999. ISBN (International Standard Book Number) 9780139083853

Hyukdoo, C., Y. K. Dong, P. H. Jae, K. Euntai, and K. Young-Ouk. "*CV-SLAM using ceiling boundary"*, in Proc. 5th IEEE Conf. Ind. Electron. Appl., pp. 228–233, 2010. doi: 10.1109/ICIEA.2010.5516788

Hunt, Kenneth J., George R. Irwin, Kevin Warwick. "*Neural Network Engineering in Dynamic Control Systems - Advances in Industrial Control",* Springer Science & Business Media, 2012. ISBN 1447130669, 9781447130666

Jeong-Gwan, K., A. Su-Yong and O. Se-Young. "*Modified neural network aided EKF based SLAM for improving an accuracy of the feature map",* in Proc. Int. Joint Conf. Neural Network, pp. 1–7, 2010. doi: 10.1109/IJCNN.2010.5596656

Kavraki *et al*, "*Probabilistic Roadmap for Path Planning in High Dimensional Configuration Spaces"*, IEEE Transactions on robotics and automation, Vol. 12, Nº 4, August, 1996. doi: 10.1109/70.508439

Khatib, M., R. Chatila. "*An Extended Potential Field Approach for Mobile Robot Sensor-Based Motions",* in Proceedings of the Intelligent Autonomous Systems IAS-4, IOS Press, Karlsruhe, Germany, pp. 490-496, March 1995.

Latombe, J. C., "*Robot Motion Planning*", Kluwer Academic Publishers, Norwood, Massachusetts, 1991.

Li S. and P. Ni. "*Square-root unscented Kalman filter based simultaneous localization and mapping",* in Proc. Int. Conf. Inf. Autom., pp. 2384–2388, 2010. doi: 10.1109/ICINFA.2010.5512187

Li, M. H., B. R. Hong and R. H. Luo. "*Mobile robot simultaneous localization and mapping using novel Rao-Blackwellised particle filter"*, Chinese J. Electron., vol. 16, no. 1, pages 34–39, 2007.

Lowe, David G., "*Object recognition from local scale-invariant features"*. Proc. 7th International Conference on Computer Vision (ICCV'99), Corfu, Greece, pp. 1150-1157, 1999. doi: 10.1109/ICCV.1999.790410.

Lindeberg, Tony, *"Feature detection with automatic scale selection"*. International Journal of Computer Vision 30(2), pages 77-116, 1998. doi: 10.1023/A:1008045108935.

Lindeberg, Tony and Lars Bretzner. *"Real-time scale selection in hybrid multi-scale representations*." Proc. Scale-Space'03, Springer Lecture Notes in Computer Science 2695, pp. 148-163, 2003. doi: 10.1007/3-540-44935-3_11

Lindeberg, Tony, *"Scholarpedia, 7(5):10491",* KTH Royal Institute of Technology, Stockholm, Sweden, 2012. doi:10.4249/scholarpedia.10491

Linde, Oskar and Tony Lindeberg. *"Object recognition using composed receptive field histograms of higher dimensionality*." Proc 17th International Conference on Pattern Recognition (ICPR'04), Cambridge, U.K. Vol. I, pp. 1-6, 2004. doi: 10.1109/ICPR.2004.1333965.

Linde, Oskar and Tony Lindeberg. *"Composed complex-cue histograms: An investigation of the information content in receptive field based image descriptors for object recognition"*. Computer Vision and Image Understanding, 116, pp. 538-560, 2012. doi:10.1016/j.cviu.2011.12.003.

Lai, Kevin., Liefeng Bo, Xiaofeng Ren, and Dieter Fox. *"Sparse Distance Learning for Object Recognition Combining RGB and Depth Information"*, Robotics and Automation (ICRA), IEEE International Conference, Shanghai, China, 2011. doi: 10.1109/ICRA.2011.5980377

Leibe, B., E. Seemann, and B. Schiele. "*Pedestrian detection in crowded scenes*." In Proceedings of the Conference on Computer Vision and Pattern Recognition, San Diego, California, USA, pp. 876–885, June 2005. doi: 10.1109/CVPR.2005.272

Lowe, D. G. "*Distinctive image features from scale-invariant key points"*. International Journal of Computer Vision, 60(2), pp. 91–110, 2004. doi: 10.1023/B:VISI.0000029664.99615.94

Lau, Boris, Christoph Sprunk and Wolfram Burgard. "*Efficient grid-based spatial representations for robot navigation in dynamic environments"*, Selected Papers from the 5th European Conference on Mobile Robots, (ECMR 2011), Vol. 61, Issue 10, pp. 1116–1130, October 2013. doi: 10.1016/j.robot.2012.08.010

Lam, Chi-Pang, Chen-Tun Chou, Kuo-Hung Chiang, Li-Chen Fu. "*Human-Centered Robot Navigation—Towards a Harmoniously Human–Robot Coexisting Environment"*, Robotics, IEEE Transactions on Vol. 27, Issue 1, pp. 99 – 112, 2011. doi: 10.1109/TRO.2010.2076851

Lowen, R., M.R. Roubens. "*Fuzzy Logic: State of the Art"*, Vol. 12 de Theory and Decision Library D, Springer Science & Business Media, 2012. ISBN 9401120145, 9789401120142

Mahon I., S. B. Williams, O. Pizarro, and M. Johnson-Roberson. "*Efficient view-based SLAM using visual loop closures",* IEEE Trans. Robot., vol. 24, no. 5, pp. 1002–1014, October 2008. doi: 10.1109/TRO.2008.2004888

Mallios, A., P. Ridao, D. Ribas, and E. Hernández. "*Probabilistic sonar scan matching SLAM for underwater environment"* in Proc. IEEE OCEANS, pp. 1–8, 2010. doi: 10.1109/OCEANSSYD.2010.5603650

Muja, Marius and David G. Lowe, "*Fast approximate nearest neighbours with automatic algorithm configuration"*. Proc. International Conference on Computer Vision Theory and Applications (VISAPP'09) (Lisbon, Portugal), pp. 331—340, 2009.

Mikolajczyk, Krystian and Cordelia Schmid. "*A performance evaluation of local descriptors*." International IEEE Transactions on Pattern Analysis and Machine Intelligence 27(19), pp. 1615—1630, 2005. doi: 10.1109/tpami.2005.188.

Mikolajczyk, Krystian and Cordelia Schmid, "*A performance evaluation of local descriptors*." International IEEE Transactions on Pattern Analysis and Machine Intelligence*,* 27(19), pages 1615—1630, 2005. doi: 10.1109/tpami.2005.188.

Mikolajczyk, K., C. Schmid, and A. Zisserman. "*Human detection based on a probabilistic assembly of robust part detectors*." In Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic, volume I, pp. 69–81, 2004. doi: 10.1007/978-3-540-24670-1_6

Motlagh O., Tang S.H., Ismail N., Ramli A.R. "*An expert fuzzy cognitive map for reactive navigation of mobile robots"*, Fuzzy Sets and Systems, Volume 201, Elsevier, 2012.doi: 10.1016/j.fss.2011.12.013

Nguyen, Thi Thanh Van, Manh Duong Phung, Thuan Hoang Tran and Quang Vinh Tran. "*Mobile Robot Localization Using Fuzzy Neural Network Based Extended Kalman Filter"*, 2012 IEEE International Conference on Control System, Computing and Engineering, pp. 416-421, November 23 – 25, 2012. doi: 978-1-4673-3143-2/12

Nigrin, Albert. *"Neural Networks for Pattern Recognition*", The MIT Press, Cambridge MA, 1993. ISBN: 9780262140546

Okabe *et al*. "*Spatial Tessellations: Concepts and applications of Voronoi Diagrams"*, John Wiley & Sons, 2009. ISBN: 047031785X, 9780470317853

Proesmans, M., L. Van Gool, E. Pauwels, and A. Oosterlinck. "*Determination of optical flow and its discontinuities using non-linear diffusion*." In Proceedings of the 3rd European Conference on Computer Vision, Stockholm, Sweden, Vol. 2, pp. 295–304, 1994. ISBN: 3540338349, 9783540338345

Raja P. and S. Pugazhenthi. "*Optimal path planning of mobile robots: A review"*, International Journal of Physical Sciences Vol. 7(9), pp. 1314 - 1320, February 2012. doi: 10.5897/IJPS11.1745

Rodrigues, J.; Cardeira, C.; Carreira, F.; Calado, J.M.F.; Oliveira, P., "*A Bayesian grid method PCA-based for mobile robots localization in unstructured environments,*" in Advanced Robotics (ICAR), 2013 16th International Conference, pp.1-6, 25-29 Nov. 2013. doi: 10.1109/ICAR.2013.6766487

Siegwart, Roland and Illah R. Nourbakhsh, **"***Introduction to Autonomous Mobile Robots*." The MIT Press, Cambridge, England, pp. 47-88, 2004. ISBN 0-262-19502-X

Stentz, Anthony. *"The Focussed D* Algorithm for Real-Time Replanning*." in Proceedings of the International Joint Conference on Artificial Intelligence, Robotics Institute, Carnegie Mellon University, U. S. A., August 1995.

Simmons, R. *"The Curvature Velocity Method for Local Obstacle Avoidance*", in Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, April 1996. doi: 10.1109/ROBOT.1996.511023

Swain, Michael J. and Dana H. Ballard. "*Colour indexing*." International Journal of Computer Vision 7(1), pp.11-32, 1991. doi: 10.1007/bf00130487.

Schiele, Bernt and James L. Crowley. "*Recognition without correspondence using multidimensional receptive field histograms*." International Journal of Computer Vision 26(1), pp. 31-50, 2000. doi: 10.1007/bfb0015571.

Schapire, R. E. "*The boosting approach to machine learning, an overview*." in MSRI Workshop on Nonlinear Estimation and Classification, 2002. doi: 10.1007/978-0-387-21579-2_9

Scholkopf, Bernhard and Alex Smola. "*Learning with Kernels*." The MIT Press, Cambridge, MA, USA, 2002. ISBN: 0262194759, 9780262194754

Simmons, Reid and Sven Koenig. "*Probabilistic robot navigation in partially observable environments"*, IJCAI, Vol. 95 pp.1080-1087, 1995. ISBN: 0080429297, 9780080429298

Thrun, Sebastian, Dieter Fox, Wolfram Burgard, Frank Dellaert. "*Robust Monte Carlo localization for mobile robots"*, Artificial Intelligence 128, Elsevier, pp. 99-141, 2001. doi: 0004-3702/01

Thrun, Sebastian, Dieter Fox, Wolfram Burgard. "*Probabilistic Robotics"*, MIT Press, Cambridge, MA, pp. 187 – 391, 2005. ISBN: 0262201623, 9780262201629

Teslic, L., I. Skrjanc, and G. Klancar. "*Using a LRF sensor in the Kalman filtering- based localization of a mobile robot*", ISA Trans, Vol. 49, no. 1, pp. 145–153, January 2010. ISBN: 1464965226, 9781464965227

Thanh, T. N., Y. Sakaguchi, H. Nagahara, and M. Yachida. "*Stereo SLAM using two estimators",* in Proc. IEEE Int. Conf. Robot. Biomimetics, pp. 19–24, 2006. doi: 10.1109/ROBIO.2006.340253

Theodoridis, Theodoros, Huosheng Hu, Klaus McDonald-Maier, and Dongbing Gu. "*Kinect Enabled Monte Carlo Localisation for a Robotic Wheelchair"*, Advances in Intelligent Systems and Computing Vol. 193, pp. 153-163, 2013. doi: 10.1007/978-3-642-33926-4_14

Tsourveloudis, N.C., K.P. Valavanis, T. Hebert. "*Autonomous vehicle navigation utilizing electrostatic potential fields and fuzzy logic"*, Robotics and Automation, IEEE Transactions on Vol. 17, Issue 4, pp. 490 – 497, 2002. doi 10.1109/70.954761

Trautman, P., J. Ma, R.M. Murray, A. Krause, "*Robot navigation in dense human crowds: the case for cooperation".* Robotics and Automation (ICRA), 2013 IEEE International Conference, pp. 2153-2160, 6-10 May 2013. doi: 10.1109 / ICRA.2013.6630866

Vadakkepat, P., Kay Chen Tan and Wang Ming-Liang. "*Evolutionary artificial potential fields and their application in real time robot path planning"*, Evolutionary Computation, Proceedings of the 2000 Congress, Vol. 1, pp. 256 – 263, 2000. doi: 10.1109/CEC.2000.870304

Viola, P., M. J. Jones, and D. Snow. "*Detecting pedestrians using patterns of motion and appearance*." in Proceedings of the 9th International Conference on Computer Vision, Nice, France, Vol. 1, pp. 734–741, 2003. doi: 10.1109/ICCV.2003.1238422

Wu, E., L. Zhao, Y. Guo, W. Zhou, and Q. Wang. *Monocular vision SLAM based on key feature points selection* in Proc. Int. Conf. Inf. Autom., pp. 1741–1745, 2010.
doi: 10.1109/ICINFA.2010.5512217

Xia, Lu, Chia-Chih Chen and J. K. Aggarwal. "*Human Detection Using Depth Information by Kinect*." Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Computer Society Conference, June 2011. doi: 10.1109/CVPRW.2011.5981811

Young, Richard. "*The Gaussian derivative model for spatial vision: I. Retinal mechanisms*", Spatial Vision 2, pp. 273–293, 1987. doi: 10.1163/156856887X00222

Anexos

Ensaio de verificação da rede neuronal artificial para coordenadas **POLARES** fonte: *var_data_polar_nn*

	Alvo fixo - referencia						Alvo variavel					
	Posição Medida		Posição Estimada		P. Estimada NN		Posição Medida		Posição Estimada		P. Estimada NN	
	x,ref	y,ref	X,ref	Y,ref	r,out,ref	b,out,ref	x,var	y,var	XX	ZZ	r,out	b,out
1	1,500	3,000	-0,012	2,977	2,970	-0,008	0,400	4,000	-1,116	3,933	4,036	-0,261
2	1,500	3,000	-0,006	2,978	2,970	-0,006	0,550	4,000	-0,978	3,978	4,057	-0,234
3	1,500	3,000	-0,006	2,978	2,970	-0,006	0,650	4,000	-0,869	3,973	4,044	-0,214
4	1,500	3,000	-0,006	2,951	2,944	-0,006	0,800	4,000	-0,703	3,921	3,984	-0,183
5	1,500	3,000	-0,006	2,978	2,970	-0,006	0,950	4,000	-0,564	3,911	3,976	-0,157
6	1,500	3,000	-0,006	2,978	2,970	-0,006	1,100	4,000	-0,422	3,979	4,072	-0,134
7	1,500	3,000	-0,006	2,951	2,944	-0,006	1,200	4,000	-0,318	3,933	4,020	-0,109
8	1,500	3,000	-0,006	2,951	2,944	-0,006	1,300	4,000	-0,222	3,971	4,071	-0,090
9	1,500	3,000	-0,012	2,951	2,944	-0,009	1,350	4,000	-0,174	3,928	4,021	-0,076
10	1,500	3,000	-0,006	2,951	2,944	-0,006	1,700	4,000	0,192	3,851	3,935	0,023
11	1,500	3,000	-0,012	2,951	2,944	-0,009	1,800	4,000	0,284	3,907	3,994	0,049
12	1,500	3,000	-0,012	2,951	2,944	-0,009	1,900	4,000	0,372	3,871	3,954	0,077
13	1,500	3,000	-0,012	2,951	2,944	-0,009	2,000	4,000	0,476	3,887	3,967	0,110
14	1,500	3,000	-0,006	2,951	2,944	-0,006	2,150	4,000	0,627	3,963	4,030	0,159
15	1,500	3,000	-0,006	2,951	2,944	-0,006	2,250	4,000	0,713	3,978	4,038	0,186
16	1,500	3,000	-0,012	2,951	2,944	-0,009	2,350	4,000	0,810	3,993	4,046	0,214
17	1,500	3,000	-0,006	2,951	2,944	-0,006	2,500	4,000	0,940	3,969	4,022	0,250
18	1,500	3,000	-0,006	2,978	2,970	-0,006	2,650	4,000	1,074	4,009	4,051	0,276
19	1,500	3,000	-0,006	2,951	2,944	-0,006	2,800	4,000	1,173	3,950	4,005	0,300
20	1,500	3,000	-0,012	2,951	2,944	-0,009	2,850	4,000	1,222	3,973	4,026	0,304
21	1,500	3,000	-0,012	2,951	2,944	-0,009	0,100	3,500	-1,413	3,471	3,755	-0,380
22	1,500	3,000	-0,006	2,951	2,944	-0,006	0,200	3,500	-1,325	3,520	3,760	-0,351
23	1,500	3,000	-0,006	2,951	2,944	-0,006	0,300	3,500	-1,224	3,497	3,695	-0,328
24	1,500	3,000	-0,006	2,951	2,944	-0,006	0,450	3,500	-1,066	3,439	3,585	-0,294
25	1,500	3,000	-0,006	2,951	2,944	-0,006	0,600	3,500	-0,914	3,447	3,548	-0,253
26	1,500	3,000	-0,006	2,951	2,944	-0,006	0,750	3,500	-0,780	3,489	3,559	-0,214
27	1,500	3,000	-0,006	2,951	2,944	-0,006	0,900	3,500	-0,616	3,455	3,500	-0,174
28	1,500	3,000	-0,006	2,951	2,944	-0,006	1,000	3,500	-0,516	3,458	3,489	-0,147
29	1,500	3,000	-0,006	2,951	2,944	-0,006	1,100	3,500	-0,425	3,531	3,553	-0,120
30	1,500	3,000	-0,012	2,951	2,944	-0,009	1,200	3,500	-0,327	3,496	3,508	-0,094
31	1,500	3,000	-0,006	2,951	2,944	-0,006	1,300	3,500	-0,230	3,529	3,535	-0,067
32	1,500	3,000	-0,006	2,951	2,944	-0,006	1,700	3,500	0,193	3,478	3,478	0,052
33	1,500	3,000	-0,006	2,951	2,944	-0,006	1,800	3,500	0,282	3,474	3,480	0,076
34	1,500	3,000	-0,006	2,951	2,944	-0,006	1,950	3,500	0,418	3,472	3,494	0,114
35	1,500	3,000	-0,012	2,951	2,944	-0,009	2,100	3,500	0,565	3,514	3,565	0,154
36	1,500	3,000	-0,006	2,951	2,944	-0,006	2,250	3,500	0,709	3,533	3,612	0,198
37	1,500	3,000	-0,006	2,951	2,944	-0,006	2,400	3,500	0,858	3,567	3,674	0,245
38	1,500	3,000	-0,006	2,951	2,944	-0,006	2,550	3,500	0,980	3,565	3,695	0,284
39	1,500	3,000	-0,006	2,951	2,944	-0,006	2,700	3,500	1,104	3,613	3,755	0,316
40	1,500	3,000	-0,006	2,951	2,944	-0,006	2,850	3,500	1,214	3,660	3,802	0,339
41	1,500	3,000	-0,006	2,951	2,944	-0,006	0,150	3,000	-1,366	2,962	3,279	-0,444
42	1,500	3,000	-0,006	2,925	2,919	-0,007	0,300	3,000	-1,211	2,955	3,209	-0,405
43	1,500	3,000	-0,006	2,925	2,919	-0,007	0,450	3,000	-1,076	2,990	3,184	-0,361
44	1,500	3,000	-0,006	2,925	2,919	-0,007	0,600	3,000	-0,927	2,967	3,105	-0,316
45	1,500	3,000	-0,006	2,925	2,919	-0,007	0,750	3,000	-0,762	2,939	3,021	-0,264
46	1,500	3,000	-0,006	2,925	2,919	-0,007	0,900	3,000	-0,610	2,961	3,001	-0,210
47	1,500	3,000	-0,006	2,951	2,944	-0,006	1,050	3,000	-0,472	2,962	2,975	-0,165
48	1,500	3,000	-0,012	2,925	2,918	-0,009	1,200	3,000	-0,329	2,965	2,963	-0,117
49	1,500	3,000	-0,006	2,925	2,919	-0,007	1,350	3,000	-0,188	2,971	2,962	-0,069
50	1,500	3,000	-0,023	2,924	2,918	-0,013	1,700	3,000	0,176	2,956	2,957	0,059
51	1,500	3,000	-0,018	2,924	2,918	-0,011	1,800	3,000	0,268	2,957	2,964	0,092
52	1,500	3,000	-0,012	2,951	2,944	-0,009	1,900	3,000	0,373	2,957	2,973	0,128
53	1,500	3,000	-0,006	2,951	2,944	-0,006	2,000	3,000	0,493	2,983	3,008	0,167
54	1,500	3,000	-0,006	2,925	2,919	-0,007	2,200	3,000	0,665	2,958	3,011	0,224
55	1,500	3,000	-0,006	2,951	2,944	-0,006	2,350	3,000	0,827	2,989	3,075	0,273
56	1,500	3,000	-0,006	2,951	2,944	-0,006	2,500	3,000	0,960	3,001	3,132	0,316
57	1,500	3,000	-0,006	2,951	2,944	-0,006	2,650	3,000	1,086	3,024	3,210	0,358
58	1,500	3,000	-0,006	2,925	2,919	-0,007	2,800	3,000	1,218	3,095	3,346	0,395
59	1,500	3,000	-0,006	2,951	2,944	-0,006	2,900	3,000	1,290	3,130	3,413	0,411
60	1,500	3,000	-0,006	2,951	2,944	-0,006	3,000	3,000	1,379	3,187	3,498	0,425
61	1,500	3,000	-0,006	2,951	2,944	-0,006	0,150	2,500	-1,375	2,421	2,793	-0,527
62	1,500	3,000	-0,006	2,951	2,944	-0,006	0,400	2,500	-1,127	2,480	2,708	-0,435

Ensaio de verificação da rede neuronal artificial para coordenadas **POLARES** fonte: *var_data_polar_nn*

	Alvo fixo - referencia						Alvo variavel					
	Posição Medida		Posição Estimada		P. Estimada NN		Posição Medida		Posição Estimada		P. Estimada NN	
	x,ref	y,ref	X,ref	Y,ref	r,out,ref	b,out,ref	x,var	y,var	XX	ZZ	r,out	b,out
63	1,500	3,000	-0,006	2,951	2,944	-0,006	0,650	2,500	-0,873	2,501	2,642	-0,351
64	1,500	3,000	-0,006	2,951	2,944	-0,006	0,800	2,500	-0,715	2,487	2,581	-0,295
65	1,500	3,000	-0,006	2,951	2,944	-0,006	0,900	2,500	-0,616	2,488	2,553	-0,256
66	1,500	3,000	-0,006	2,925	2,919	-0,007	1,000	2,500	-0,513	2,487	2,528	-0,216
67	1,500	3,000	-0,006	2,951	2,944	-0,006	1,150	2,500	-0,362	2,468	2,485	-0,158
68	1,500	3,000	-0,006	2,951	2,944	-0,006	1,300	2,500	-0,218	2,498	2,505	-0,101
69	1,500	3,000	-0,006	2,951	2,944	-0,006	1,700	2,500	0,169	2,456	2,476	0,055
70	1,500	3,000	-0,006	2,951	2,944	-0,006	1,800	2,500	0,282	2,473	2,505	0,103
71	1,500	3,000	-0,006	2,951	2,944	-0,006	1,900	2,500	0,380	2,509	2,555	0,144
72	1,500	3,000	-0,006	2,951	2,944	-0,006	2,000	2,500	0,491	2,492	2,559	0,190
73	1,500	3,000	-0,006	2,925	2,919	-0,007	2,150	2,500	0,633	2,496	2,594	0,245
74	1,500	3,000	-0,006	2,951	2,944	-0,006	2,300	2,500	0,773	2,484	2,620	0,296
75	1,500	3,000	-0,006	2,951	2,944	-0,006	2,450	2,500	0,916	2,475	2,657	0,347
76	1,500	3,000	-0,006	2,951	2,944	-0,006	2,600	2,500	1,066	2,530	2,759	0,395
77	1,500	3,000	-0,006	2,951	2,944	-0,006	2,750	2,500	1,196	2,533	2,828	0,446
78	1,500	3,000	-0,006	2,951	2,944	-0,006	0,400	2,000	-1,124	1,969	2,272	-0,522
79	1,500	3,000	-0,006	2,951	2,944	-0,006	0,550	2,000	-0,984	1,991	2,177	-0,450
80	1,500	3,000	-0,006	2,925	2,919	-0,007	0,700	2,000	-0,809	1,978	2,112	-0,388
81	1,500	3,000	-0,006	2,925	2,919	-0,007	0,850	2,000	-0,676	1,978	2,087	-0,338
82	1,500	3,000	-0,006	2,951	2,944	-0,006	1,000	2,000	-0,521	1,959	2,030	-0,272
83	1,500	3,000	-0,006	2,951	2,944	-0,006	1,150	2,000	-0,374	1,977	2,008	-0,196
84	1,500	3,000	-0,006	2,925	2,919	-0,007	1,300	2,000	-0,220	1,997	2,011	-0,117
85	1,500	3,000	-0,006	2,951	2,944	-0,006	1,750	2,000	0,205	2,012	2,041	0,094
86	1,500	3,000	-0,012	2,951	2,944	-0,009	1,900	2,000	0,365	2,026	2,063	0,173
87	1,500	3,000	-0,006	2,951	2,944	-0,006	2,000	2,000	0,472	2,009	2,062	0,227
88	1,500	3,000	-0,006	2,951	2,944	-0,006	2,200	2,000	0,672	2,020	2,139	0,312
89	1,500	3,000	-0,006	2,951	2,944	-0,006	2,400	2,000	0,894	2,030	2,263	0,394
90	1,500	3,000	-0,006	2,951	2,944	-0,006	2,600	2,000	1,081	2,053	2,394	0,458
91	1,500	3,000	-0,001	2,978	2,970	-0,004	0,800	1,500	-0,710	1,489	1,608	-0,432
92	1,500	3,000	-0,006	2,951	2,944	-0,006	1,000	1,500	-0,512	1,487	1,597	-0,329
93	1,500	3,000	-0,006	2,951	2,944	-0,006	1,150	1,500	-0,361	1,485	1,557	-0,239
94	1,500	3,000	-0,006	2,951	2,944	-0,006	1,300	1,500	-0,230	1,466	1,483	-0,149
95	1,500	3,000	-0,006	2,925	2,919	-0,007	1,350	1,500	-0,174	1,473	1,474	-0,109
96	1,500	3,000	-0,006	2,951	2,944	-0,006	1,400	1,500	-0,125	1,461	1,456	-0,075
97	1,500	3,000	-0,006	2,951	2,944	-0,006	1,700	1,500	0,167	1,483	1,527	0,125
98	1,500	3,000	-0,006	2,951	2,944	-0,006	1,800	1,500	0,255	1,486	1,538	0,187
99	1,500	3,000	-0,006	2,951	2,944	-0,006	1,900	1,500	0,369	1,496	1,549	0,263
100	1,500	3,000	-0,006	2,951	2,944	-0,006	2,000	1,500	0,467	1,505	1,558	0,322
101	1,500	3,000	-0,006	2,951	2,944	-0,006	2,150	1,500	0,618	1,498	1,579	0,402
102	1,500	3,000	-0,006	2,951	2,944	-0,006	2,300	1,500	0,776	1,500	1,661	0,465
103	1,500	3,000	-0,006	2,951	2,944	-0,006	1,000	1,000	-0,507	0,957	1,057	-0,479
104	1,500	3,000	-0,006	2,951	2,944	-0,006	1,100	1,000	-0,386	0,980	1,003	-0,364
105	1,500	3,000	-0,006	2,951	2,944	-0,006	1,200	1,000	-0,293	0,976	1,034	-0,289
106	1,500	3,000	-0,006	2,951	2,944	-0,006	1,300	1,000	-0,184	0,975	1,033	-0,179
107	1,500	3,000	-0,006	2,951	2,944	-0,006	1,400	1,000	-0,088	0,966	0,981	-0,072
108	1,500	3,000	0,108	0,077	NaN	NaN	1,600	1,000	0,069	0,959	0,950	0,094
109	1,500	3,000	-0,006	2,951	2,944	-0,006	1,700	1,000	0,161	0,972	0,983	0,192
110	1,500	3,000	-0,006	2,951	2,944	-0,006	1,800	1,000	0,254	0,980	1,021	0,290
111	1,500	3,000	-0,006	2,951	2,944	-0,006	1,900	1,000	0,361	0,978	1,049	0,393
112	1,500	3,000	-0,006	2,951	2,944	-0,006	2,000	1,000	0,476	0,974	1,074	0,480
113	1,500	3,000	-0,006	2,951	2,944	-0,006	1,200	0,500	0,108	0,077	NaN	NaN
114	1,500	3,000	-0,006	2,951	2,944	-0,006	1,300	0,500	-0,146	0,459	0,449	-0,322
115	1,500	3,000	-0,006	2,951	2,944	-0,006	1,400	0,500	-0,042	0,433	0,407	-0,152
116	1,500	3,000	-0,006	2,951	2,944	-0,006	1,700	0,500	0,203	0,422	0,438	0,378
117	1,500	3,000	0,108	0,077	2,944	-0,006	1,600	0,500	0,108	0,077	NaN	NaN
118	1,500	3,000	-0,006	2,951	2,944	-0,006	1,800	0,500	0,312	0,429	0,519	0,551
119	0,000	3,000	-0,019	2,834	2,834	-0,007	1,350	4,800	1,310	4,804	4,979	0,266
120	0,000	3,000	-0,024	2,810	2,810	-0,008	1,200	4,800	1,197	4,872	5,017	0,241
121	0,000	3,000	-0,024	2,810	2,810	-0,008	1,000	4,800	1,197	4,872	5,017	0,241
122	0,000	3,000	-0,014	2,810	2,810	-0,005	0,850	4,800	0,865	4,942	5,017	0,173
123	0,000	3,000	-0,014	2,834	2,834	-0,005	0,700	4,800	0,865	4,942	5,017	0,173
124	0,000	3,000	-0,019	2,810	2,810	-0,007	0,500	4,800	0,508	4,872	4,898	0,104

Ensaio de verificação da rede neuronal artificial para coordenadas **POLARES** fonte: *var_data_polar_nn*

	Alvo fixo - referencia						Alvo variavel					
	Posição Medida		Posição Estimada		P. Estimada NN		Posição Medida		Posição Estimada		P. Estimada NN	
	x,ref	y,ref	X,ref	Y,ref	r,out,ref	b,out,ref	x,var	y,var	XX	ZZ	r,out	b,out
125	0,000	3,000	-0,024	2,810	2,810	-0,008	0,350	4,800	0,353	4,872	4,885	0,072
126	0,000	3,000	-0,024	2,810	2,810	-0,008	0,200	4,800	0,225	4,942	4,947	0,045
127	0,000	3,000	-0,019	2,810	2,810	-0,007	0,100	4,800	0,233	4,942	4,947	0,047
128	0,000	3,000	-0,019	2,834	2,834	-0,007	-0,250	4,800	-0,205	4,872	4,876	-0,042
129	0,000	3,000	-0,019	2,834	2,834	-0,007	-0,400	4,800	-0,363	5,013	5,026	-0,072
130	0,000	3,000	-0,014	2,834	2,834	-0,005	-0,500	4,800	-0,449	4,942	4,962	-0,091
131	0,000	3,000	-0,014	2,810	2,810	-0,005	-0,600	4,800	-0,557	5,013	5,044	-0,111
132	0,000	3,000	-0,014	2,810	2,810	-0,005	-0,700	4,800	-0,650	5,013	5,055	-0,129
133	0,000	3,000	-0,019	2,810	2,810	-0,007	-0,900	4,800	-0,828	4,872	4,942	-0,168
134	0,000	3,000	-0,019	2,810	2,810	-0,007	1,400	4,500	1,368	4,492	4,696	0,296
135	0,000	3,000	-0,019	2,810	2,810	-0,007	1,250	4,500	1,232	4,492	4,658	0,268
136	0,000	3,000	-0,019	2,810	2,810	-0,007	1,100	4,500	1,096	4,492	4,624	0,239
137	0,000	3,000	-0,014	2,810	2,810	-0,005	1,000	4,500	0,988	4,551	4,657	0,214
138	0,000	3,000	-0,038	2,834	2,834	-0,013	0,800	4,500	0,779	4,492	4,559	0,172
139	0,000	3,000	-0,009	2,810	2,810	-0,003	0,600	4,500	0,613	4,551	4,592	0,134
140	0,000	3,000	-0,009	2,810	2,810	-0,003	0,400	4,500	0,416	4,492	4,511	0,092
141	0,000	3,000	-0,009	2,810	2,810	-0,003	0,250	4,500	0,276	4,551	4,559	0,061
142	0,000	3,000	-0,019	2,810	2,810	-0,007	0,150	4,500	0,176	4,551	4,554	0,039
143	0,000	3,000	-0,014	2,810	2,810	-0,005	-0,300	4,500	-0,248	4,612	4,619	-0,054
144	0,000	3,000	-0,014	2,810	2,810	-0,005	-0,500	4,500	-0,437	4,551	4,572	-0,096
145	0,000	3,000	0,005	2,834	2,834	0,002	-0,700	4,500	-0,636	4,551	4,595	-0,139
146	0,000	3,000	0,005	2,834	2,834	0,002	-0,900	4,500	-0,846	4,612	4,689	-0,181
147	0,000	3,000	0,005	2,834	2,834	0,002	-1,050	4,500	-1,490	-0,978	1,782	0,990
148	0,000	3,000	0,005	2,834	2,834	0,002	-1,150	4,500	-1,065	-0,978	1,446	0,828