INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Mecânica

ISEL

Robô Móvel para Máquinas-ferramenta

RÚBEN CARDOSO BERNARDINO

(Licenciado em Engenharia Mecânica)

Trabalho Final de Mestrado para obtenção do grau de

Mestre em Engenharia Mecânica

Orientadores:

Professor Doutor Francisco M. de Oliveira Campos

Professor Doutor Pedro Miguel Abreu Silva

Júri:

Presidente: Doutor João Manuel Ferreira Calado

Vogais: Doutora Alexandra Bento Moutinho

Doutor Francisco M. de Oliveira Campos

Dezembro de 2016

Dedicado aos meus pais.

Este documento não foi escrito
segundo o novo acordo ortográfico.

“Whenever we proceed from the known into the unknown we may hope to understand, but we may have to learn at the same time a new meaning of the word ’understanding’."

Werner Heisenberg, Physics and Philosophy: The Revolution in Modern Science (1958)

Agradecimentos

Ao Professor Francisco Campos e ao Professor Pedro Silva por terem aceite trabalhar comigo e por estarem sempre disponíveis para me ajudar, de forma exemplar, a progredir no meu trabalho que conclui uma fase tão importante da minha vida académica.

Ao Professor Fernando Carreira por todas as dicas e trocas de opiniões que me deu ao longo do desenvolvimento do meu trabalho.

Aos meus pais, Luís e Cristina, que sempre me proporcionaram as ferramentas necessárias à minha passagem pelo ensino superior e sempre me apoiaram em todas as decisões que tomei.

Ao meu irmão, David, por todo o apoio e motivação.

Aos meus amigos Ana Balixa, Francisco Zdanowski, Frederico Benito, José Simões, Tiago Machado, Sérgio Costa e Mariana Martins pela amizade, companheirismo e por todas as horas passadas a enfrentar o desafio que foi estes últimos dois anos.

A todos os meus restantes amigos que me acompanharam e ajudaram neste percurso que tanto me fez crescer.

Por fim e não menos importante à Teresa Sousa, que sempre me incentivou para atingir o sucesso e sempre me ajudou em tudo o que precisei a nível pessoal e ao longo do meu percurso académico.

Simbologia

Variável	Descrição	Propriedades
${}^{A}\xi_{B}$	Postura relativa de um sistema de coordenadas B a um outro inercial A	[3×3]
${}^{A}p$	Posição de um ponto P relativamente a um sistema de coordenadas A	[2×1]
${}^{A}\tilde{p}$	Posição de um ponto P expresso em coordenadas homogéneas	[3×1]
(x, y)	Coordenadas de um ponto representado num sistema de coordenadas	[2×1]
$\hat{x}, \hat{y}$	Versor de um sistema de coordenadas; Variável estimada	
t	Tempo	Escalar $[s]$
$\vec{t},\ {}^{0}P$	Translação	[2×1]
$\theta,\ \varphi$	Orientação do robô relativamente a um sistema de coordenadas de referência	Escalar $[rad]$
${}^{V}R_{B}$	Matriz de rotação de um sistema de coordenadas B relativamente a um outro V	[2×2]
u_n	Vector pertencente a um conjunto de vectores	
δ_{ij}	Delta de Kronecker	Escalar
$O(n)$	Grupo ortogonal de ordem n	
$SO(n)$	Grupo ortogonal especial de ordem n	
v	Velocidade	Escalar $[m.s^{-1}]$
$\dot{\Psi},\ \omega$	Velocidade angular	Escalar $[rad.s^{-1}]$
R	Distância do centro de massa do robô à origem de um referencial	Escalar $[m]$
W, L	Distância entre rodas do robô	Escalar $[m]$
$\dot{x}$	Velocidade segundo a direcção do eixo x	Escalar $[m.s^{-1}]$
$\Delta t,\ T$	Intervalo de tempo	Escalar $[s]$
$k,\ a,\ b$	Constante genérica	
s	Variável complexa genérica	
$G(s), H(s)$	Função de transferência	
$Y(s), C(s), B(s)$	Função de sinal de saída	
$X(s)$	Função de sinal de entrada	
$R(s)$	Função de sinal de entrada de referência	
$E(s), e(\cdot)$	Desvio	
K_{P}	Ganho proporcional	Escalar
K_{I}	Ganho integral	Escalar
K_{D}	Ganho derivativo	Escalar
$u(t)$	Função de acção de controlo genérica	
$f(\cdot), g(\cdot), y(\cdot), h(\cdot)$	Função genérica	
C_{ε}	Ponto representado na esfera unitária	[3×1]

d	Distância entre pontos focais	Escalar $[m]$
l	*Latus rectum*; Distância da caneta ao eixo entre rodas	Escalar $[m]$
$\tilde{m}$	Ponto representado num plano normalizado	[3×1]
A	Matriz de parâmetros intrínsecos da câmara	
ε	Distância de um ponto ao centro da esfera unitária	Escalar
D	Distância genérica	
ρ	Distância euclidiana	Escalar $[m]$
ω_k	Acção de controlo incerta	
$p(\cdot)$	Função de densidade de probabilidade genérica	
υ	Ruído de um processo	
x_0	Condição inicial	
μ	Média	Escalar
σ^2	Variância	Escalar
$\mathcal{N}(\mu, \sigma^2)$	Distribuição normal	
$P,\ Q,\ R,\ Z$	Matriz de covariância	
$\hat{x}^+$	Variável estimada no instante seguinte	
z	Inovação	
K	Ganho de Kalman	
F	Matriz de transição	
H	Matriz de medição	
G	Matriz de controlo	
r	Raio da roda	Escalar $[m]$
n	Número de pontos; Vector perturbação	
c_i	Condição inicial genérica	
s	Trajectória	
p	Vector de posição	
q	Vector de velocidade	
J	Matriz Jacobiana	
$traj$	Trajectória discretizada	
M	Matriz de pontos do espaço	
m	Matriz de píxeis de uma imagem	
L	Distância relativa de um *beacon* ao sistema de coordenadas de referência	Escalar $[m]$
Z	Distância relativa de um *beacon* ao robô	Escalar $[m]$

Subscrito

v_{left}	Velocidade da roda esquerda
v_{right}	Velocidade da roda direita
v_t	Velocidade tangencial
v_n	Velocidade normal
v_{lim}	Velocidade limite do robô

Abreviaturas

2D	Duas dimensões
3D	Três dimensões
SO	**S**pecial **O**rthogonal Group
O	**O**rthogonal Group
WMR	**W**heeled **M**obile **R**obot
DDOF	**D**ynamic **D**egrees **O**f **F**reedom
DOF	**D**regrees **O**f **F**reedom
PDF	**P**robability **D**ensity **F**unction
RPM	**R**otações **P**or **M**inuto
PID	Controlador **P**roporcional *I*ntegrador e **D**erivativo
LED	**L**ight **E**mitting **D**iode
RGB	**R**ed **G**reen **B**lue
RMSD	**R**oot **M**ean **S**quade **D**eviation
KF	**K**alman **F**ilter
EKF	**E**xtended **K**alman **F**ilter
IFR	**I**nternational **F**ederation Of **R**obotics
CNC	**C**omputerized **N**umerical **C**ontrol
USB	**U**niversal **S**erial **B**us
GPS	**G**lobal **P**ositioning **S**ystem
UAV	**U**nmanned **A**ir **V**ehicle
EMD	**E**lementary **M**otion **D**etector
AGV	**A**utomatic **G**uided **V**ehicle
LTI	**L**inear **T**ime **I**nvariant

Resumo

A presente dissertação apresenta uma solução para um robô móvel com navegação baseada em visão para ser aplicado em máquinas-ferramenta. Para tal, foi utilizado um protótipo de um robô de tracção diferencial de pequena escala onde um instrumento de escrita simula a actuação da máquina-ferramenta. O protótipo dispõe ainda de uma câmara equipada com uma lente olho de peixe controlada por uma placa *Raspberry Pi*. Com o objectivo de imprimir uma determinada trajectória desejada com rigor, começou-se por desenvolver um algoritmo de planeamento onde se discretizou a dita trajectória. Seguidamente, foi definido o modelo cinemático do protótipo o qual, juntamente com o planeamento, permitiram desenvolver o anel de controlo para os motores onde se incorporou também um controlador *PID*. Com as leituras dos *encoders* das rodas, foi possível efectuar a localização baseada em odometria, a qual não apresentou rigor suficiente devido ao deslize das rodas e erros de modelação. Recorrendo à calibração da câmara através de uma *toolbox* implementada em ambiente *MATLAB*, obteve-se os parâmetros intrínsecos da câmara. Com estes foi então possível transformar as imagens obtidas pela câmara para que fosse retirada a deformação imposta pela lente de olho de peixe. Assim, foi possível estabelecer uma relação entre píxel e metro. Para a localização baseada em visão, foram utilizadas três referências (*beacons*) colocadas em posições conhecidas e a partir das quais foi calculada a postura do robô relativamente às mesmas. Esta avaliação utiliza apenas duas referências em vez das três que normalmente são utilizadas na triangulação, utilizando distâncias para o cálculo da localização em vez de ângulos. Esta informação juntamente com os dados da odometria são introduzidos num filtro de Kalman estendido com o objectivo de reduzir o desvio da estimativa baseada apenas na odometria. Por fim, a viabilidade da conjugação dos dois métodos de localização, quando aplicados em ambientes industriais, é discutida.

Palavras Chave: Robótica móvel; Visão computacional; Robô diferencial; Calibração; Odometria; Controlo; Controlador PID; Filtro de Kalman estendido.

Abstract

The present dissertation presents a vision based mobile robot system to be applied in machine tools. The prototype under study is a small scale differential drive robot where an incorporated writing tool simulates the action of the machine tool. Besides that, the robot has a fish-eye camera controlled by a *Raspberry Pi* board. The first step in devising a system for printing with enough precision was the development of a planning algorithm that produces a discretized trajectory. The next step was the definition of the kinematic model of the prototype which, when combined with the planning, allowed the development of a PID controller loop for the motors. Using the *encoders* readings a robot position estimate is computed, although the precision is not sufficient due to wheel slippage and modelling errors. The camera calibration was achieved using an open-source *MATLAB* toolbox, which produced the camera's intrinsic parameters. With these parameters in hand, it was possible to transform the images obtained by the camera so that the deformation caused by the lens was removed. Once undistorted images are available, it is possible to take reliable measures on them. For the purpose of vision based localization, three active beacons were placed in known positions in the world, inside the robot's viewing area. This computation relies only on two beacons instead of three, as is commonly used in triangulation, since the robot's pose is computed using distances instead of bearings. This information is combined with the odometry estimate in an exteded Kalman filter with the goal of reducing the error that the odometry is unable to handle. To conclude, the viability of the conjugation between the two approaches to localization, when applied in industrial scenarios, is discussed.

Keywords: Mobile robotics; Computer vision; Differential drive robot; Calibration; Odometry; Control; PID controller; Extended Kalman filter.

Índice

Lista de Figuras

Capítulo 1

Introdução

1.1 Estrutura do documento

Capítulo 1

Este capítulo dedica-se ao enquadramento do tema da robótica na indústria, avaliando a sua importância e a tendência da sua aplicação. É ainda apresentada uma perspectiva pessoal da motivação para o presente trabalho, juntamente com a relevância do tema proposto. Seguidamente apresentam-se os objectivos do trabalho assim como os métodos utilizados para os abordar. Por fim é apresentado um estado da arte da navegação baseada em visão e da calibração de câmaras digitais.

Capítulo 2

Este capítulo contém o fundamento teórico que suporta o trabalho desenvolvido. Começa-se por descrever o importante conceito de posturas relativas, seguindo-se os conceitos intrínsecos à navegação, como a locomoção, a cinemática, o controlo e o planeamento. Posteriormente é introduzida a componente sensorial do robô na secção dedicada à percepção. Por fim, é apresentada a secção da localização no qual se apresentam as abordagens para a determinação da postura do robô.

Capítulo 3

Este capítulo apresenta a abordagem utilizada para desenvolver o caso de estudo. O capítulo começa por descrever a estrutura e o modelo cinemático do robô, e de seguida aborda os algoritmos de localização por odometria e planeamento. A lei de controlo surge após estes conceitos, e por fim, é apresentada a componente de localização, constituída pelo algoritmo de identificação de *beacons*, calibração, transformação de imagem e filtro de Kalman estendido.

Capítulo 4

Neste capítulo são apresentados e discutidos os resultados obtidos através dos métodos propostos no capítulo anterior. É apresentado o resultado da discretização utilizada para o planeamento, a acção do controlador *PID*, a localização baseada em odometria, a transformação de imagem, a identificação dos *beacons*, a localização baseada em visão e por fim o resultado produzido após a aplicação do filtro de Kalman estendido.

Capítulo 5

O capítulo que encerra o presente trabalho é onde se discute a viabilidade do modelo tendo em conta os objectivos definidos e os resultados obtidos. São ainda apresentadas propostas para trabalho futuro.

1.2 Enquadramento e motivação

A robótica é uma ciência suportada em várias áreas de engenharia, como a mecânica, a electrotecnia e a electrónica. Algumas ciências também se apresentam como influência como as ciências sociais, cognitivas e de computadores. De acordo com o Departamento de Estatística da Federação Internacional da Robótica (Department), as vendas de robôs industriais atingiram um valor de 240,000 unidades no ano 2015. Este valor, nunca antes registado, representa um aumento de 12% relativamente ao ano anterior, onde se registaram 221,000 unidades. Esta tendência crescente tem-se vindo a verificar desde a crise financeira de 2009. De acordo com os estudos da *FIR*, pelo ano de 2018 são esperados 2.3 milhões de unidades a funcionar em fábricas, mais do dobro do valor de 2009 (1.0 milhão). À expectativa de uma crescente aposta na robótica a nível industrial, está associado um maior investimento económico no ramo, o que por sua vez promove mais investimento em investigação e novas tecnologias.

Normalmente, a nível industrial, uma máquina-ferramenta está limitada a operar na sua mesa de trabalho, o que implica, para peças de grandes dimensões, que a máquina-ferramenta seja de uma dimensão proporcional. A robótica tem, cada vez mais, vindo a ser introduzida nas máquinas-ferramenta, desde as máquinas mais antigas operadas por *NC* (numerical control) até ao mais recente conceito denominado *CNC* (*computerized numerical control*). As máquinas mais actuais *CNC* permitem através de um único comando efectuar múltiplas operações de maquinação, mesmo que entre elas implique uma mudança de ferramenta.

Numa perspectiva pessoal, senti a necessidade e o interesse de aprofundar o meu conhecimento na área da robótica, que se tem vindo a revelar um novo grande aliado da engenharia mecânica no

desenvolvimento de novas tecnologias. Vi também uma porta aberta a nível profissional ao aceitar este desafio visto que a aplicação desta ciência na indústria tem vindo a apresentar uma tendência exponencial de ano para ano. Para além disso, as competências intrínsecas à robótica a abordar no desenvolvimento deste trabalho eram do meu interesse, e uma vez mais, apresentam grande envolvência com a engenharia mecânica - entre elas destaco a área do controlo e a programação.

Com a limitação espacial das máquinas-ferramenta e a aplicação da robótica na indústria em mente, surgiu a ideia de se desenvolver um robô móvel modelo que permitisse remover a dita limitação. Resumidamente, a máquina estaria associada a um robô móvel, que permitiria que esta se deslocasse em torno de toda a peça a ser maquinada, passando a apresentar uma mesa de trabalho hipoteticamente infinita. A aplicação industrial tornar-se-á clara se se pensar numa quantidade relativamente grande de peças a ser maquinadas e as quais apresentam dimensões demasiado grandes para uma máquina-ferramenta convencional. Em vez de se produzir peças mais pequenas, enviar as mesmas para a oficina para serem maquinadas e posteriormente enviadas de volta para serem acopladas, enviar-se-ia a máquina-ferramenta uma única vez até à fábrica, onde trabalharia as peças originais de grandes dimensões sem ser necessário que estas fossem repartidas. Em termos de transporte, as centenas de peças a serem enviadas e devolvidas, certamente sairão mais dispendiosas do que enviar e devolver uma máquina-ferramenta de dimensão relativamente pequena uma única vez.

1.3 Objectivos do trabalho e metodologia utilizada

O presente trabalho apresenta como objectivo o desenvolvimento de um robô móvel modelo capaz de simular a operação de uma máquina ferramenta. Para se simular o percurso e a actuação da ferramenta, foi utilizado um instrumento de escrita, como por exemplo uma caneta. Desta forma, pretende-se que o robô móvel seja capaz de reproduzir uma determinada geometria ou, se se preferir, um determinado desenho, com a maior exactidão possível. Para tal, será necessário desenvolver um algoritmo de planeamento que produza uma descrição da trajectória que o robô terá que percorrer.

Com o planeamento definido, o passo seguinte passará por executar a operação nele estipulada. Para isso, será necessário conhecer a cinemática do robô, desenvolver um controlador para a actuação e, uma vez que a locomoção será à custa de rodas, estabelecer um algoritmo de localização com base nos sensores internos do robô.

Pretende-se ainda utilizar um algoritmo de localização probabilístico que tenha em conta as leituras dos sensores internos e externos do robô, visto que este irá dispor de uma câmara com lente de olho

de peixe. A dita câmara será utilizada para a identificação de referências (*beacons*) para que se possa calcular, em termos relativos, a postura do robô.

Para abordar os objectivos em questão, começou-se por definir o planeamento através de trajectórias discretizadas as quais, através do modelo cinemático do robô, foram traduzidas em velocidades nas rodas do robô. Seguidamente, procedeu-se ao desenvolvimento do algoritmo de localização do robô por odometria a partir das leituras dos encoders das rodas, juntamente com a implementação do controlador *PID*.

Com a odometria definida estabeleceu-se a comunicação com uma placa *Raspberry Pi* e uma câmara com uma lente de olho de peixe para que as imagens captadas fossem introduzidas num algoritmo de processamento de imagem. Este algoritmo consistiu na remoção da deformação na imagem e na detecção e localização dos *beacons* por análise de cor. Com os *beacons* identificados, procedeu-se ao cálculo da postura do robô relativamente a estes, para que esta informação fosse introduzida, juntamente com os dados obtidos na odometria, num filtro de Kalman estendido.

1.4 Estado da Arte

A robótica é uma área muito vasta onde várias ciências se cruzam e cooperam entre si. Cada vez mais, no estudo de sistemas autónomos se tenta atribuir capacidades sensoriais mais próximas das do ser humano. Muitas vezes, esta aproximação é até superada ao tentar reproduzir mecanismos biológicos observados na natureza, com capacidades superiores às do Humano. Estas abordagens acopladas a mecanismos desenvolvidos pelo próprio Homem, tornam o número de configurações de sistemas autónomos quase infinito.

1.4.1 Estado da Arte em navegação baseada em visão

A mais recente tecnologia de mecanismos sensoriais é a tentativa de reprodução da visão do ser Humano, a qual permite obter uma noção espacial e uma sensibilidade de cor que quando aliados ao processamento por parte do cérebro permitem que se consiga desempenhar a função de localização.

À utilização deste conceito denomina-se visão computacional, e o sensor responsável por adquirir a informação espacial *2D* e a sensibilidade de cor através da aquisição de imagens é denominado câmara. Com esta capacidade, foram introduzidos modelos de robôs móveis com navegação baseada em visão, onde a postura é obtida com base numa avaliação relativa a uma determinada referência observável.

Uma aplicação desta abordagem, na altura ainda muito pouco explorada, foi apresentada em Moravec [1983] num robô denominado *Stanford Cart*. Este sistema dispunha de uma câmara de vídeo que transmitia a informação para um computador que produzia a actuação do robô através de sinal rádio após executar um determinado algoritmo de planeamento. Contudo, e por a tecnologia se encontrar limitada ao desenvolvimento da altura, o robô demorava cerca de 10 a 15 minutos a percorrer 1 metro. Após esse metro percorrido, o robô efectuava uma paragem, adquiria imagens e processava-as durante um longo período de tempo, planeando posteriormente uma nova trajectória. De seguida o ciclo repetia-se. Isto resultava na necessidade de esperar 5 horas até que um percurso de 20 m fosse completo. Esta longa espera até que o planeamento fosse completo trazia alguns problemas que induziam erro, pois o espaço observável poderia entretanto mudar, como por exemplo as sombras não apresentavam as mesmas orientações e dimensões. A titulo de curiosidade, nesta publicação é evidenciado o interesse na contribuição para o desenvolvimento de uma inteligência artificial geral.

Anos mais tarde surge uma nova referência no desenvolvimento de um robô móvel com navegação baseada em visão denominado *POLLY* e apresentado em Horswill [1994] e Horswill [1995]. Uma das menções nestas publicações é precisamente o *Stanford Cart*. Neste modelo, era utilizada uma visão monocromática que restringia o espaço onde o robô operava a apresentar uma única cor. Resumidamente, uma imagem era obturada, identificava-se cada píxel correspondente à cor definida, e evitava-se todos os restantes píxeis. Em termos de planeamento o *POLLY* recorria a um mapa topológico no qual incluía certas referências que utilizava para calcular a sua postura. Estas referências eram imagens obturadas em pontos específicos no espaço de trabalho. À medida que o robô se deslocava, a câmara de vídeo procurava uma correspondência entre a imagem que estava a ser obturada no momento e as referências previamente obtidas, conseguindo assim calcular a sua localização. O principal problema desta abordagem era uma vez mais a identificação das sombras como obstáculos.

Em Aider et al. [2005] propôs-se uma correspondência para linhas rectas que definiam regiões, as quais não variavam com a mudança de posição da câmara. Estas abordagens eram, de certa forma, apenas variações dos modelos propostos mais antigos e onde se tentava melhorar, e abordar de formas diferentes, a componente da visão. Com a visão computacional em foco principal no desenvolvimento de novos modelos, foram publicadas novas abordagens para o processamento de imagem, numa tentativa de estimar, com uma precisão cada vez melhor, a postura do robô. Uma destas abordagens é apresentada em Gartshore et al. [2002] onde se declara que o problema de correspondência de referências era resolvido com a aplicação do método de grelha de ocupação. De uma forma resumida, o espaço era repartido por uma grelha, onde cada célula apresentava um determinado valor de probabilidade de estar ocupada. Esta abordagem era utilizada para a identificação de referências, obtendo-se

um rasto de uma potencial localização de cada referência. Com este método, era construido o mapa utilizado para o planeamento, sendo também apresentado na mesma obra um algoritmo de identificação de referências através do processamento da matriz *RGB* da imagem.

A grande parte das obras publicadas para modelos de uma única câmara a operar em espaços interiores, foca-se em métodos de seguimento de referências, como por exemplo paredes, ou de seguimento de centros, como por exemplo o centro de um corredor. Um destes casos é a abordagem apresentada em Murillo et al. [2008], onde se propõe um método para detecção de portas no espaço onde o robô opera através de um algoritmo probabilístico e de uma base de dados de características comuns a portas, conseguida à custa de uma aprendizagem experimental. Um exemplo para uma abordagem de seguimento de centros é proposta em Saitoh et al. [2008], onde um robô móvel construído a partir de uma cadeira de rodas, percorria o centro de um corredor com o auxílio de uma câmara USB. Nesta obra, quando um obstáculo era detectado era tomada uma decisão entre um comando de paragem ou um desvio de trajectória. A tomada de decisão dependeria da dimensão do objecto, da distância a que o robô se encontrava do mesmo e da dimensão do corredor. Outra abordagem sugerida em Chen and Birchfield [2006] consiste em executar uma operação de repetição. Resumidamente, o robô é manualmente conduzido ao longo da trajectória desejada enquanto regista todos os pontos de referência necessários para que, numa segunda operação e após a aprendizagem, consiga reproduzir a mesma trajectória efectuada manualmente. As vantagens desta solução apresentam-se como a dispensa da calibração da câmara, a possibilidade de operar em espaços interiores ou exteriores de qualquer configuração (não necessitando que o espaço de operação seja plano, restrição muito comum nas outras abordagens) e a dispensa de um mapa. Estas propriedades conferem a este algoritmo uma natureza totalmente qualitativa. O mesmo método de aprendizagem manual e posterior reprodução foi também apresentada em Kidono et al. [2000] com a variante de que em vez de apenas se utilizar uma câmara única, eram utilizadas duas, produzindo o conceito denominado visão estéreo.

O que a visão estéreo desbloqueia é a capacidade de avaliar a profundidade de uma imagem obturada, à custa de um processamento mais complexo e algoritmos mais robustos. Um conhecimento prévio necessário para que seja possível esta avaliação de profundidade são os parâmetros intrínsecos das câmaras, como o desvio do centro de imagem causado pela lente ou a distância focal. Com os parâmetros determinados, é necessário que as linhas de visão para cada câmara se intersectem no ponto de que se deseja obter a informação de profundidade. Garantidas estas condições, a visão estéreo irá permitir identificar, rastrear e calcular a distância de referências em tempo real e numa perspectiva *3D*. Alguns exemplos de navegação baseada em visão estéreo são apresentados em Murray and Little [2000], Olson et al. [2003] e Konolige et al. [2006]. A navegação de um robô móvel baseada em visão

só é eficaz se as referências utilizadas forem constante e correctamente identificadas à medida que o robô se move, podendo uma referência mal considerada induzir erros na operação e desvios na trajectória. Foi com esta preocupação que foram propostos critérios para a identificação de boas referências em Shi and Tomasi [1994]. A importância de se conseguir obter as coordenadas tridimensionais de certas referências de uma imagem é evidenciada ao ser possível calcular a distância entre a referência identificada e o ponto focal da câmara, algo que revela uma grande importância na localização do robô e na interpretação do espaço. Uma abordagem para mapeamento baseado em visão estéreo é apresentada em Wooden [2006]. O método foi proposto para um ambiente exterior e começava por adquirir imagens através das câmaras calibradas (capazes de distinguir cor) passando estas, posteriormente, por uma biblioteca que contém toda a informação relativa à geometria das câmaras e ao seus parâmetros intrínsecos de onde resultava um mapa de profundidade. Um mapa de profundidade não é mais do que uma representação *3D* da elevação do terreno num referencial de coordenadas locais. Posto isto, cada coordenada era então transformada, resultando a rotação instantânea em torno dos eixos que definem o referencial de coordenadas. Estas rotações são denominadas de *roll* (em torno do eixo x) e *pitch* (em torno do eixo y) e eram estimadas por um sistema de navegação inercial. Para o caso do *yaw* (rotação em torno do eixo z) este era obtido através da transformação do mapa de profundidade do referencial local para o referencial global (baseado em *GPS*). Após as transformações era então aplicada uma operação derivativa no mapa do terreno com o objectivo de identificar mudanças abruptas. Por fim, era então actualizado o mapa global com as novas medições estimadas do terreno e derivadas. Um exemplo famoso de um robô móvel com navegação baseada em visão estéreo é o *Mars Exploration Rover* desenvolvido pela *NASA* e apresentado em Goldberg et al. [2002].

Com o avanço tecnológico surgem novas câmaras que permitem uma abertura de 360°, garantindo assim uma visão radial total do espaço, independentemente da orientação da câmara. A este conceito denomina-se visão omnidireccional e foi uma tecnologia que imediatamente captou a atenção do mundo da robótica. Tipicamente, estes sistemas de visão utilizam câmaras com uma lente de olho de peixe ou câmaras panorâmicas. Apresentando a vantagem óbvia do campo de visão, a desvantagem que esta abordagem apresenta é o custo associado e a complexidade dos algoritmos de visão que dependem da geometria da câmara utilizada. Algumas abordagens que recorrem a esta tecnologia para a navegação são apresentadas em Fiala and Basu [2004], Gaspar et al. [2000] e Winters and Santos-Victor [1999]. A maior parte dos algoritmos usados naqueles trabalhos foi baseado num conceito denominado fluxo óptico que consiste na representação do movimento aparente de tudo o que constitui uma determinada cena a ser observada, causado pelo deslocamento relativo entre a câmara e a cena. Em termos computacionais, o fluxo óptico é normalmente representado através de vectores *2D* onde cada vector descreve o deslocamento da posição inicial de um determinado ponto numa

imagem, para a sua posição seguinte. Num robô móvel com navegação baseada em visão, o fluxo óptico é induzido pelo movimento da câmara à medida que o robô se desloca. Uma aplicação que tira grande proveito do fluxo óptico são os veículos aéreos não tripulados (*UAV*), como é demonstrado no trabalho de Netter and Francheschini [2002]. Neste caso, foi utilizado um sistema que se assemelha a um sistema nervoso, onde cada fotorreceptor constituinte de uma grelha está ligado a um detector de movimento elementar electrónico (*EMD*), modelo explicado em Frye [2015], que realiza o cálculo do fluxo óptico local numa posição particular.

Como se pode verificar até a este ponto, as abordagens para a navegação baseada em visão são vastas e algumas delas inspiradas em sistemas biológicos observados na natureza. Contudo há algumas que se destacam devido à sua simplicidade, eficiência e necessidade de investimento relativamente pequeno, como é o caso da navegação por rastreamento de linha. Esta abordagem é amplamente utilizada e pode ser idealizada como um seguimento contínuo de uma referência (linha) que, apesar de ser simples, requer que o sensor utilizado esteja muito próximo da linha, limitando assim a área de operação do robô. O modelo de navegação por rastreamento de linha poderá ser encontrado em veículos industriais de condução automática (*AGV*), utilizados onde as operações por estes desempenhadas são monótonas como por exemplo numa fábrica ou num hospital. Alguns exemplos são propostos em Zhang et al. [2004], Ishikawa et al. [1988], Beccari et al. [1997] e Ismail et al. [2009].

Outro conceito que não poderia deixar de ser mencionado devido à grande atenção que recebeu por parte dos investigadores, é a localização e mapeamento simultâneos ou *SLAM* (Simultaneous Localization And Mapping). Um robô móvel que aplique este método constrói um mapa do ambiente onde está inserido e, ao mesmo tempo, utiliza o mesmo para calcular a sua postura. Inicialmente a postura do robô e o mapa não são conhecidos, contudo o seu modelo cinemático está definido e o robô desloca-se num ambiente desconhecido onde se encontram referências artificiais ou, dependendo da abordagem definida, naturais. Identificadas as referências, uma estimação simultânea da postura do robô e das referências é efectuada com base na observação das referências, tal como, por exemplo, está descrito em Dissanayake et al. [2001]. A maior parte das aplicações de algoritmos de *SLAM* são feitas em ambientes interiores, bem estruturados e relativamente estáticos, como por exemplo é apresentado em Zunino and Christensen [2001]. Contudo algumas publicações apresentam aplicações em ambientes exteriores mais dinâmicos, como por exemplo em Liu and Thrun [2003]. No que toca à componente probabilística do algoritmo, diferentes filtros podem ser aplicados, sendo o mais comum o filtro de Kalman estendido (*EKF*). A aplicação do filtro surge como solução do problema do erro crescente induzido na odometria devido à derrapagem das rodas, a superfícies irregulares ou a folgas mecânicas. Uma vez que as referências identificadas eram introduzidas no mapa relativo à

posição do robô, as posições destas referências vinham elas também desviadas da sua posição real. O filtro *EKF* parte do principio que a incerteza do processo e dos sensores podem ser modelados como distribuições Gaussianas. Esta consideração no entanto nem sempre é viável pois o sistema poderá apresentar respostas que não conseguem ser modeladas através de tais distribuições. Uma das desvantagens do filtro de Kalman (*KF*) e da sua variante para sistemas não lineares, o filtro de Kalman estendido (*EKF*), é o facto de este não ser viável para operações de longa duração. Ao aumentar o número de referências, o computador responsável pelo cálculo chegará a um ponto que não será capaz de responder em tempo real devido à grande quantidade de informação a ser processada. Este problema surge por existir uma correlação entre cada referência. Numa tentativa de solucionar este problema é proposto em Guivant and Nebot [2001] um novo filtro comprimido que tem como base o filtro de Kalman estendido. Outras abordagens são apresentadas em Sim et al. [2005] e Montemerlo et al. [2003] onde se evita o filtro de Kalman, recorrendo a um filtro de partículas que, apesar de não apresentar as desvantagens do primeiro, é normalmente um processo lento em termos de cálculo.

1.4.2 Estado da Arte em calibração

A calibração da câmara, ou câmaras, dependendo da abordagem pode, ou não, ser necessária. O processo de calibração permite corrigir a distorção induzida pela lente quando associada a um novo mapeamento. Para alem disso, é ainda possível determinar a relação entre cada píxel da câmara e o espaço real. Quando se fala de calibração, poderão estar envolvidos dois conceitos, a calibração geométrica e a calibração de cor. Contudo apenas será importante para o presente trabalho a calibração geométrica.

De uma forma resumida, a calibração irá quantificar o quanto a câmara altera a imagem obturada. A dita alteração depende dos seguintes parâmetros:

1. Centro da imagem
2. Distância focal
3. Factor de escala
4. Factor de inclinação
5. Distorção da lente

Todos estes, são obtidos no processo de calibração e são únicos para cada conjunto câmara e lente.

A abordagem utilizada no presente trabalho para a calibração passou pela utilização de uma *toolbox* desenvolvida para o ambiente *MATLAB*. A grande referência que inspirou vários autores foi a *toolbox* desenvolvida e descrita por Jean-Yves Bouguet apresentada em Bouguet. Nesta referência o autor descreve toda a abordagem usada na *toolbox*, incluindo a informação sobre a forma como são obtidos os parâmetros intrínsecos da câmara acima numerados.

Um dos autores inspirados por esta obra, foi Christopher Mei que apresenta em Mei a *toolbox* por si desenvolvida com base em funções propostas por Jean-Yves Bouguet, para câmaras omnidireccionais.

Por fim, Davide Scaramuzza, também inspirado por Jean-yves Bouguet, propõe a utilização de um polinómio de Taylor para a função de imagem, desenvolvendo uma *toolbox* para câmaras omnidireccionais onde aplica a sua proposta. A *toolbox* é apresentada em Scaramuzza enquanto que os métodos propostos são descritos em Scaramuzza et al. [2006a], Scaramuzza et al. [2006b] e Scaramuzza et al. [2008].

Capítulo 2

Robótica

2.1 Posição e Orientação

A representação da orientação e posição de objectos inseridos num determinado ambiente, no mundo da robótica e da visão computacional, é provavelmente o ponto mais importante no desenvolvimento de qualquer modelo. Os ditos objectos poderão ser robôs, câmaras, peças, obstáculos ou trajectórias.

Um ponto representado no espaço é um conceito conhecido da matemática, e pode ser descrito através de um vector que representa o deslocamento desse ponto relativamente à intersecção dos eixos ortogonais de um sistema de coordenadas de referência. A intersecção referida é conhecida como a origem desse sistema. Um sistema de coordenadas de referência poderá também ser denominado por referencial.

É possível definir um objecto através de um certo conjunto de pontos. Assumir que o mesmo é rígido significa que os pontos que o constituem encontram-se sempre à mesma distância relativamente à origem do seu referencial. Desta forma, fará sentido utilizar o referencial do objecto para representar a sua posição e orientação relativamente a um outro referencial de referência, em vez de se utilizar um ponto, ou pontos existentes no objecto. Considere-se a figura 2.1, o referencial *{B}* poderá ser associado a um objecto enquanto o referencial *{A}* poderá ser o referencial de referência, por exemplo. Normalmente associa-se como índice dos eixos de cada referencial o nome do mesmo, por exemplo x_B e y_B. Denomina-se postura à conjugação de uma determinada posição e orientação, sendo esta representada graficamente através de um referencial. Desta forma, tendo em conta o exemplo da figura 2.1, é possível associar o referencial *{B}* à postura de um determinado objecto. A postura de um referencial relativamente a um referencial de referencia é representado através da letra ξ. Ainda na mesma imagem é representada a postura relativa ${}^{A}\xi_{B}$ que descreve *{B}* relativamente a *{A}*. A postura relativa ${}^{A}\xi_{B}$ pode ainda ser associada a um movimento, isto é, se um determinado objecto

assume como referencial *{A}*, ao sofrer uma translação e uma rotação poderá ser transformado em *{B}*. A letra que se encontra em índice superior está associada ao referencial de referência enquanto que a letra que se encontra em índice inferior está associada ao referencial que está a ser descrito. Caso o índice superior não seja representado, pode-se assumir que a variação de postura é relativa ao referencial do mundo, denominado *O*.

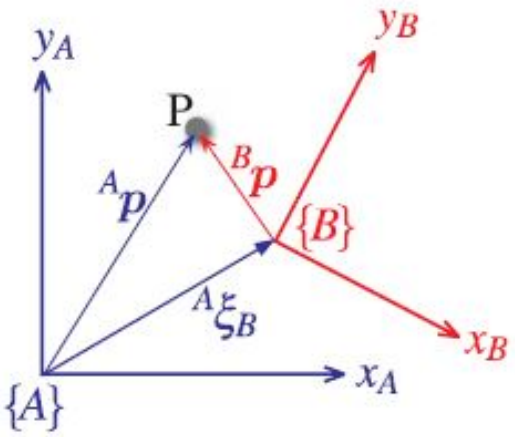


Figura 2.1: O ponto **P** pode ser descrito através de um vector relativo ao referencial *{A}* ou *{B}*. A postura de *{B}* relativamente a *{A}* é representada por ${}^A\xi_B$, adaptada de Corke [2011].

O ponto *P* representado na figura 2.1 pode ser descrito relativamente a qualquer um dos referenciais

$$ {}^Ap = {}^A\xi_B \cdot {}^Bp \tag{2.1}$$

sendo que o lado direito da equação define um movimento de *{A}* para *{B}* e posteriormente para *P*. O produto interno é responsável por transformar o vector ${}^Bp$, resultando um novo vector que define o mesmo ponto relativamente a um referencial diferente, no caso do exemplo apresentado ${}^Ap$. É importante entender que a postura relativa é descrita através de uma matriz de transformação, portanto todas as manipulações algébricas entre posturas relativas são válidas.

2.1.1 Representação de posturas em 2D

Um determinado espaço bidimensional, ou plano, é normalmente representado através de um referencial onde se define o eixo *x* para a discretização horizontal e o eixo *y* para a vertical. Os versores de cada eixo são denominados $\hat{x}$ e $\hat{y}$. Um determinado ponto representado no plano poderá ser definido através das suas coordenadas (*x,y*) ou através de um vector

$$ p = x\hat{x} + y\hat{y} \tag{2.2}$$

A figura 2.2 apresenta um referencial *{B}* que se pretende descrever relativamente ao referencial de referência *{A}*. A translação neste caso será representada pelo vector $t = (x, y)$ e a rotação no

sentido anti-horário pelo ângulo θ. Tendo em conta o conceito de postura relativa, será claro declarar que ${}^{A}\xi_{B} \sim (x, y, \theta)$, sendo que o símbolo $\sim$ denota que ambas as representações são equivalentes. Contudo a representação proposta não é conveniente na concretização de manipulações algébricas, forçando a utilização de funções trigonométricas complexas para o fazer. Para simplificar o problema, a abordagem a utilizar passará por considerar um ponto *P* arbitrário no espaço que será descrito relativamente a ambos os referenciais, posteriormente será determinada a relação entre ${}^{A}p$ e ${}^{B}p$. A solução sugerida será abordada em duas partes, a primeira tratará da rotação e a segunda da translação.

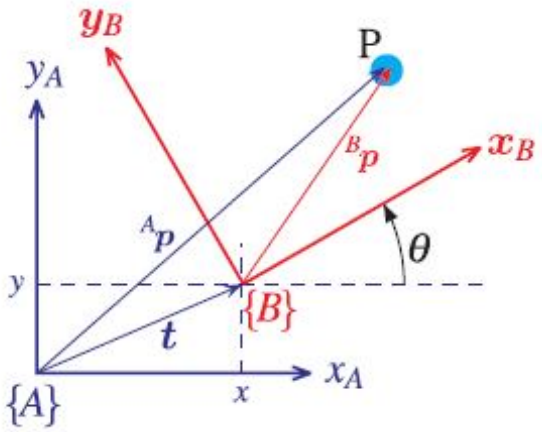


Figura 2.2: Dois referenciais 2D *{A}*, *{B}* e um ponto **P**. *{B}* sofre uma translação e uma rotação relativamente a *{A}*, adaptada de Corke [2011].

Para se abordar a rotação considera-se um novo referencial *{V}* no qual os eixos são paralelos aos de *{A}* mas onde a origem é partilhada com o referencial *{B}*, considere-se a figura 2.3 para uma melhor percepção. Tendo em conta a equação 2.2 é possível representar o ponto *P* em relação a *{V}* através de um produto entre um vector linha e um vector coluna.

$$\begin{aligned} {}^{V}p &= {}^{V}x\ \hat{x}_{V} + {}^{V}y\ \hat{y}_{V} \\ &= \begin{pmatrix} \hat{x}_{V} & \hat{y}_{V} \end{pmatrix} \begin{pmatrix} {}^{V}x \\ {}^{V}y \end{pmatrix} \end{aligned} \tag{2.3}$$

O referencial *{B}* é definido através dos versores que representam os seus eixos ortogonais, sendo que estes sofrem o efeito de uma determinada rotação definida por θ

$$\hat{x}_{B} = cos\theta\ \hat{x}_{V} + sin\theta\ \hat{y}_{V}$$

$$\hat{y}_{B} = -sin\theta\ \hat{x}_{V} + cos\theta\ \hat{y}_{V}$$

que podem ser reescritos em notação matricial

$$\begin{pmatrix} \hat{x}_{B} & \hat{y}_{B} \end{pmatrix} = \begin{pmatrix} \hat{x}_{V} & \hat{y}_{V} \end{pmatrix} \begin{pmatrix} cos\,\theta & -sin\,\theta \\ sin\,\theta & cos\,\theta \end{pmatrix} \tag{2.4}$$

Utilizando a equação 2.2 para representar o ponto *P* em relação ao referencial *{B}* e introduzindo a equação 2.4, obtém-se

$$ {}^{B}p = \begin{pmatrix} \hat{x}_V & \hat{y}_V \end{pmatrix} \begin{pmatrix} cos\,\theta & -sin\,\theta \\ sin\,\theta & cos\,\theta \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \end{pmatrix} \tag{2.5} $$

Igualando os coeficientes do lado direito da equação 2.3 e da equação 2.5 obtém-se

$$ \begin{pmatrix} {}^{V}x \\ {}^{V}y \end{pmatrix} = \begin{pmatrix} cos\,\theta & -sin\,\theta \\ sin\,\theta & cos\,\theta \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \end{pmatrix} $$

A igualdade apresentada verifica-se uma vez que o vector que define o ponto *P* tem o mesmo módulo quer para o referencial *{V}* quer para o *{B}*, uma vez que apenas se está a ter em conta a rotação e nenhuma translação. Desta forma, descreve-se a transformação de um ponto representado no referencial *{B}* sujeito a uma rotação, num ponto representado no referencial *{V}*. A matriz responsável pela dita transformação é denominada matriz de rotação e é representada por ${}^{V}R_{B}$

$$ \begin{pmatrix} {}^{V}x \\ {}^{V}y \end{pmatrix} = {}^{V}R_{B} \begin{pmatrix} {}^{B}x \\ {}^{B}y \end{pmatrix} \tag{2.6} $$

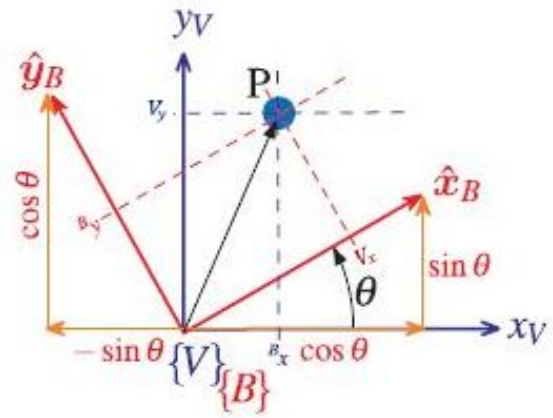


Figura 2.3: Representação do efeito da rotação no referencial em 2D. O ponto **P** pode ser descrito relativamente ao referencial *{V}* ou ao *{B}*, adaptada de Corke [2011].

A matriz de rotação tem características que são importantes de mencionar. Primeiramente, cada coluna da matriz corresponde a um vector unitário, o que define a matriz como ortonormal. Segundo Kuttler [2016] um conjunto de vectores $\{u_1, ..., u_n\}$ denomina-se ortonormal quando, se verifica a condição

$$ u_i \cdot u_j = \delta_{ij} $$

sendo que a letra δ_{ij} é chamada delta de Kronecker e define-se como

$$\delta_{ij} \equiv \begin{cases} 1 \; if \; i = j \\ 0 \; if \; i \neq j \end{cases}$$

sendo i e j qualquer inteiro em $\{1, 2, ..., n\}$ para qualquer $n \in \mathbb{N}$. Se o conjunto é ortonormal então, os vectores que o constituem são linearmente independentes. Para além disso, a matriz de rotação é ortogonal uma vez que verifica a condição de ortogonalidade

$$Uma \; matriz \; U \; de \; dimensão \; real \; n \times n \; é \; denominada \; \textbf{Ortogonal} \; se \; UU^T = U^T U = I.$$

Outra propriedade importante da matriz de rotação é o facto de pertencer ao grupo especial ortogonal de dimensão 2, ou seja $R \in SO(2) \subset \mathbb{R}^{2\times 2}$. De acordo com a Encyclopedia of Mathematics e com Gutiérrez [2005] matematicamente, um grupo ortogonal de dimensão n, denominado $O(n)$, é o grupo de matrizes ortogonais de dimensão $n \times n$ onde a operação entre matrizes desse grupo é dada pela multiplicação de matrizes. Quando o determinante de uma matriz ortogonal tem valor 1 ou -1 pertence ao grupo especial ortogonal denominado $SO(n)$, que não é mais do que um importante subgrupo de $O(n)$. O grupo especial ortogonal pode também ser denominado grupo de rotação uma vez que nas dimensões 2 e 3, os seus elementos são normalmente rotações em torno de um ponto (dimensão 2) ou em torno de uma linha (dimensão 3). Ao pertencer a este subgrupo, o módulo de um vector permanece inalterado após a transformação, isto é $|^B p| = |^V p|, \forall\theta$. Tendo em conta as propriedades da matriz de rotação, é possível reescrever a equação 2.6

$$\begin{pmatrix} {}^B x \\ {}^B y \end{pmatrix} = ({}^V R_B)^{-1} \begin{pmatrix} {}^V x \\ {}^V y \end{pmatrix} = ({}^V R_B)^T \begin{pmatrix} {}^V x \\ {}^V y \end{pmatrix} = ({}^B R_V) \begin{pmatrix} {}^V x \\ {}^V y \end{pmatrix}$$

É interessante observar que em vez de se recorrer a um ângulo, que é um escalar, utiliza-se uma matriz 2×2 onde os seus elementos não são independentes (não confundir com dependência linear). Cada coluna tem módulo unitário, o que define duas restrições, sendo estas ortogonais, o que introduz uma terceira restrição. Com as restrições definidas e verificadas, o que se obtém é efectivamente um ponto. Se se imaginar uma representação gráfica, não será mais do que dois vectores com módulo unitário, que se intersectam com um ângulo recto, formando uma intersecção (ponto). Definida a rotação, primeira parte da abordagem proposta para a transformação de postura, será agora necessário considerar a componente de translação. Tendo em conta a figura 2.2, para se considerar a translação entre as origens dos referenciais *{A}* para *{V}*, uma vez que os eixos do primeiro são paralelos aos do

segundo e vice-versa, bastará apenas concretizar uma soma vectorial

$$\begin{aligned}\begin{pmatrix} {}^{A}x \\ {}^{A}y \end{pmatrix} &= \begin{pmatrix} {}^{V}x \\ {}^{V}y \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix} \\ &= \begin{pmatrix} cos\,\theta & -sin\,\theta \\ sin\,\theta & cos\,\theta \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix} \\ &= \begin{pmatrix} cos\,\theta & -sin\,\theta & x \\ sin\,\theta & cos\,\theta & y \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ 1 \end{pmatrix}\end{aligned}$$

que, simplificando, resulta

$$\begin{pmatrix} {}^{A}x \\ {}^{A}y \\ 1 \end{pmatrix} = \begin{pmatrix} {}^{A}R_{B} & t \\ 0_{1\times 2} & 1 \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ 1 \end{pmatrix} \tag{2.7}$$

onde $t = (x, y)$ define a translação e ${}^{A}R_{B}$ define a rotação. De notar que ${}^{A}R_{B} = {}^{V}R_{B}$ uma vez que os eixos de *{A}* e de *{V}* são paralelos. As coordenadas dos vectores para o ponto *P* expressas em coordenadas homogéneas permitem reescrever a equação

$$\begin{aligned}{}^{A}\tilde{p} &= \begin{pmatrix} {}^{A}R_{B} & t \\ 0_{1\times 2} & 1 \end{pmatrix} {}^{B}\tilde{p} \\ &= {}^{A}T_{B}\,{}^{B}\tilde{p}\end{aligned} \tag{2.8}$$

onde ${}^{A}T_{B}$ é denominada matriz de transformação homogénea. Por comparação com a equação 2.1, evidencia-se que a matriz de transformação homogénea representa a postura relativa

$$\xi(x, y, \theta) \sim \begin{pmatrix} cos\,\theta & -sin\,\theta & x \\ sin\,\theta & cos\,\theta & y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.9}$$

De acordo com Vince [2008], as coordenadas homogéneas permitem efectuar certas operações em pontos num espaço Euclidiano através de multiplicações de matrizes. Para se definir um ponto em coordenadas cartesianas utilizam-se n valores, onde n corresponde à dimensão do espaço. Para o caso das coordenadas homogéneas para o mesmo ponto, será necessária uma dimensão $n+1$, ou seja, $n+1$ valores para que o mesmo fique definido. De uma forma geral, recorre-se ao valor 1 e introduz-se o mesmo na ultima coordenada do vector, desta forma, o ponto definido em duas dimensões (x, y) passará a ser definido em coordenadas homogéneas por $(x, y, 1)$. O mesmo é válido para um ponto definido em três dimensões (x, y, z) que assumirá a forma $(x, y, z, 1)$. O motivo de se adicionar a

unidade à nova dimensão pode ser justificada através da definição de coordenada homogénea segundo Corke [2011] para um vector (x, y) escrito na forma homogénea $\tilde{p}$

$$\tilde{p} \in \mathbb{P}^2, \tilde{p} = (x_1, x_2, x_3)$$

$$onde\ x = x_1/x_3,\ y = x_2/x_3\ e\ x_3 \neq 0$$

sendo que o til indica que um determinado vector é homogéneo.

2.2 Navegação

Segundo Corke [2011], navegação pode ser definida como a resposta para o problema de levar um robô de um determinado ponto de partida até a um destino definido. Numa primeira abordagem ao problema de navegação poderá pensar-se que uma lista de comandos, ou mapa, seja essencial para a sua resolução. Contudo, uma grande parte das operações efectuadas por robôs não necessitam de qualquer tipo de mapa ou instrução, baseando-se apenas numa abordagem reactiva conseguida à custa da informação recebida pelos sensores do robô. Exemplos da navegação reactiva são as operações de deslocamento até um ponto de luz, seguir uma linha de cor delineada no chão, percorrer um labirinto seguindo uma parede ou aspirar uma sala percorrendo um percurso aleatório. O robô neste caso não tem informação acerca de onde está e não necessita da mesma para concretizar a sua operação.

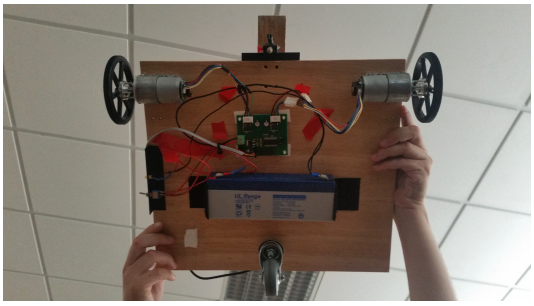
Contrastando com a anterior, a abordagem da navegação baseada em mapa é desempenhada por robôs menos simples. Apesar de o robô ser capaz de responder a operações mais exigentes, a informação necessária para resolver o problema é também mais exigente. Esta complexidade provém da necessidade da posição do robô ser sempre conhecida e do mapa do espaço de trabalho estar bem definido. Isto induz uma maior quantidade de variáveis no modelo, torando a abordagem mais robusta em termos computacionais.

Para o presente trabalho a abordagem utilizada foi a navegação baseada em mapa.

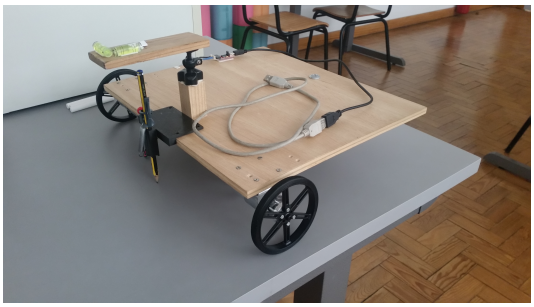
2.2.1 Locomoção

Um robô móvel necessita de mecanismos de locomoção que possibilitem o seu deslocamento. Devido à grande variedade de configurações existentes para os mecanismos, a selecção do mais apropriado para cada caso torna-se um factor importante no desenvolvimento de um robô móvel. Existem robôs que conseguem caminhar, saltar, correr, deslizar, patinar, nadar, voar ou rolar. A grande parte dos mecanismos referidos são inspirados pelos seus homólogos biológicos. O robô utilizado no presente trabalho é um robô móvel com rodas, denominado *Wheeled Mobile Robot* ou *WMR*. Existem várias configurações para um *WMR*, cada uma com as suas vantagens e desvantagens, sendo a configuração

normalmente escolhida de acordo com o espaço e ambiente onde o robô irá operar. Uma tabela comparativa entre as diferentes configurações é apresentada em Roland Siegwart [2011]. Contudo para o caso do presente trabalho, não foi necessária uma construção complexa. Desta forma o robô é constituído por duas rodas motoras independentes e uma roda livre, sendo que a este modelo se dá o nome de *differential drive robot*, evidenciando o facto de ser um robô de tracção diferencial. O protótipo descrito é apresentado na figura 2.4. O motivo de não se considerar outras configurações deve-se ao facto de o ambiente onde o robô se move ser um laboratório onde o piso é relativamente regular, não causando praticamente nenhuma instabilidade enquanto o robô opera. Ainda assim, é de salientar a importância do mecanismo de locomoção, pois a aplicação de um robô móvel fora de um ambiente académico é raramente feita em ambientes controlados. Desta forma, aconselha-se a consultar a tabela 2.1 da página 39 da referência Roland Siegwart [2011] onde se discute as diferentes abordagens para o mecanismo de locomoção.



(a) Parte inferior.



(b) Parte superior.

Figura 2.4: Configuração do robô.

2.2.2 Cinemática

Como já foi introduzido no ponto dedicado à locomoção (2.2.1), uma roda motora de um robô diferencial é controlada independentemente da outra e vice-versa. Ao conjugar as velocidades em cada roda, é possível ao robô concretizar operações como rodar sob um ponto, andar numa linha recta, percorrer uma trajectória curvilínea ou seguir um caminho previamente delineado. De acordo com a figura 2.5, R representa o raio de curvatura instantâneo da trajectória do robô, enquanto que W representa a distância entre as rodas do robô.

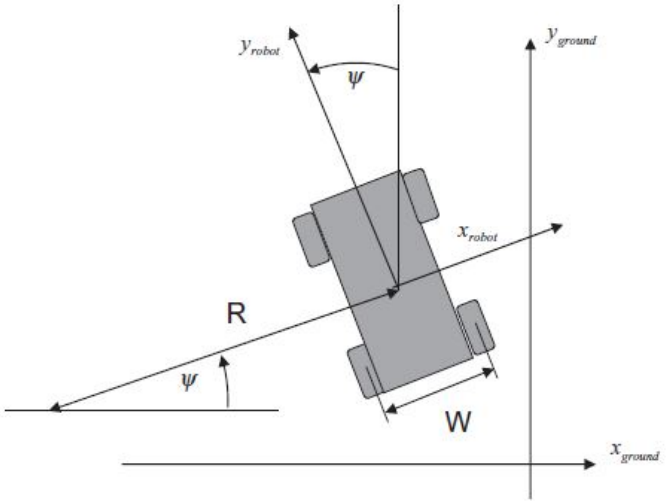


Figura 2.5: Diagrama esquemático de um robô diferencial, adaptada de Cook [2011].

Segundo Cook [2011], a partir de considerações geométricas, as velocidades da roda esquerda e da roda direita são dadas por

$$v_{left} = \dot{\Psi}(R - W/2) \tag{2.10a}$$

$$v_{right} = \dot{\Psi}(R + W/2) \tag{2.10b}$$

Subtraindo as equações descritas acima

$$v_{right} - v_{left} = \dot{\Psi}W$$

permitindo resolver em ordem à velocidade angular

$$\dot{\Psi} = \frac{v_{right} - v_{left}}{W} \tag{2.11}$$

Resolvendo em ordem ao raio de curvatura instantâneo à custa da velocidade da roda esquerda

$$R = \frac{v_{left}}{\dot{\Psi}} + \frac{W}{2}$$

que substituindo fica

$$R = \frac{v_{left}}{\frac{v_{right} - v_{left}}{W}} + \frac{W}{2}$$

ou finalmente

$$R = \frac{W}{2}\frac{v_{right} + v_{left}}{v_{right} - v_{left}} \tag{2.12}$$

que resulta na expressão para a velocidade ao longo do eixo longitudinal do robô

$$v_{y_{robot}} = \dot{\Psi}R = \frac{v_{right} - v_{left}}{W}\frac{W}{2}\frac{v_{right} + v_{left}}{v_{right} - v_{left}} = \frac{v_{right} + v_{left}}{2} \tag{2.13}$$

Resumindo, as equações do movimento para o referencial do robô descrevem-se como

$$v_{left} = \dot{\Psi}(R - W/2) \tag{2.14a}$$

$$v_{right} = \dot{\Psi}(R + W/2) \tag{2.14b}$$

$$\dot{\Psi} = \frac{v_{right} - v_{left}}{W} \tag{2.14c}$$

As equações do movimento podem ainda ser descritas relativamente ao referencial do mundo

$$\dot{x} = -\frac{v_{right} + v_{left}}{2} sin\Psi \tag{2.15a}$$

$$\dot{y} = \frac{v_{right} + v_{left}}{2} cos\Psi \tag{2.15b}$$

$$\dot{\Psi} = \frac{v_{right} - v_{left}}{W} \tag{2.15c}$$

É ainda importante considerar que as velocidades não são alteradas instantaneamente, desta forma introduz-se as variáveis de controlo para as velocidades nas rodas no sistema de equações já definido (2.15a,2.15b e 2.15c). Definindo-se as equações a introduzir como

$$\dot{v}_{_{right}} = u_1 \tag{2.16a}$$

$$\dot{v}_{_{left}} = u_2 \tag{2.16b}$$

o sistema de equações, que é agora de quinta ordem, define o modelo cinemático. Poderá ser conveniente reformular as equações para um modelo discreto que permitirá realizar simulações de uma forma mais simples. Claramente as equações não são lineares e portanto os métodos de conversão de sistemas lineares representados em continuo para uma representação em discreto não são aplicáveis. Uma forma de contornar o problema passa pela utilização o método de integração de Euler descrito por Richard Khoury [2016]. O método é uma aproximação de primeira ordem de séries de Taylor para a integração e declara que a derivação pode ser aproximada à diferença finita

$$\dot{x}(t) \approx \frac{x(t + \Delta t) - x(t)}{\Delta t} \tag{2.17}$$

que pode ser manipulada resultando

$$x(t + \Delta t) \approx x(t) + \dot{x}(t)\Delta t \tag{2.18}$$

Definindo $t = kT$ e o tempo de amostra $\Delta t = T$ e aplicando o conceito de diferença finita no sistema

de equações definido para o modelo cinemático, é possível obter um modelo discreto

$$x((k+1)T) = x(kT) - T\frac{v_{right}(kT) + v_{left}(kT)}{2} sin\Psi(kT) \tag{2.19a}$$

$$y((k+1)T) = y(kT) + T\frac{v_{right}(kT) + v_{left}(kT)}{2} cos\Psi(kT) \tag{2.19b}$$

$$\Psi((k+1)T) = \Psi(kT) + T\frac{v_{right}(kT) - v_{left}(kT)}{W} \tag{2.19c}$$

$$v_{right}((k+1)T) = v_{right}(kT) + Tu_1(kT) \tag{2.19d}$$

$$v_{left}((k+1)T) = v_{left}(kT) + Tu_2(kT) \tag{2.19e}$$

Existem métodos mais sofisticados e mais precisos para modelos discretos, contudo o modelo de Euler pode ser bastante útil quando o tempo de amostragem é suficientemente pequeno. Os modelos discretos podem ser utilizados para analise de sistemas, desenho de controladores, desenho de estimadores e para simulação de sistemas.

2.2.2.1 Robôs holonómicos

Segundo Roland Siegwart [2011] o conceito de holonomia é aplicado em diferentes áreas da matemática incluindo equações diferenciais, funções e na definição de constrangimentos. No campo da robótica a definição de holonomia pode ser enunciada com base na relação entre os graus de liberdade do robô e os graus de liberdade do seu espaço de trabalho. Assim, um robô é holonómico se e só se os seus graus de liberdade diferenciais forem iguais aos seus graus de liberdade (*DDOF=DOF*). Para melhor compreender a diferença entre os dois conceitos pode-se definir *DDOF* como a capacidade de um robô atingir certas trajectórias (chegar a um determinado destino), enquanto que *DOF* será a capacidade de o robô atingir certas posturas (ser capaz de chegar a um destino com uma determinada orientação definida). A configuração do robô utilizado no presente trabalho é um exemplo de um robô não holonómico, pois não permite o deslocamento na direcção paralela ao eixo das rodas, ou seja *DOF* > *DDOF*. Um exemplo de um robô holonómico seria um robô omnidireccional.

2.2.3 Controlo

2.2.3.1 Sistemas de malha aberta e malha fechada

Muitos dos sistemas aplicados nas mais diversas áreas, com destaque para a engenharia e para a ciência, necessitam de responder a alterações impostas pelo ambiente onde estão inseridos. A forma como estes sistemas se adaptam e respondem às perturbações é definida pelo seu sistema de controlo.

Para o caso de um sistema de malha fechada, o erro de actuação, que não é mais do que a diferença entre o sinal de entrada *(input)* e o sinal de saída *(output)*, é alimentado ao controlador com o objectivo

de reduzir o erro, trazendo o sinal de saída do sistema para um valor desejado. O termo malha fechada sugere a utilização de uma acção de controlo baseada em *feedback*.

Já para o caso de um sistema de malha aberta, o sinal de saída não tem qualquer influência na acção de controlo. Quando comparado com o caso de malha fechada, o sistema de malha aberta não quantifica e não compara o sinal de saída com o sinal de entrada. Um exemplo prático deste sistema seria uma máquina de lavar roupa. Embeber, lavar e enxaguar são operações que acontecem em intervalos de tempo definidos. Contudo, a máquina não mede o sinal de saída que neste caso seria o quão lavada está a roupa. Desta forma, para cada sinal de entrada de referencia corresponde uma condição de operação fixa que faz com que a precisão do sistema dependa de uma calibração. Na presença de uma perturbação, um sistema de malha aberta não irá realizar a tarefa pretendida. Isto faz com que na prática, apenas se possa aplicar este sistema quando a relação entre o sinal de entrada e o sinal de saída for conhecida e não existirem perturbações internas nem externas. Desta forma, a acção de controlo dos sistemas de malha aberta, não são baseadas em *feedback*.

As descrições apresentadas para os sistemas de malha aberta e malha fechada, serão melhor compreendidas após a introdução do diagrama de blocos.

2.2.3.2 Diagrama de blocos

Um diagrama de blocos, segundo Ogata [2010], não é mais do que uma representação gráfica de cada componente constituinte do sistema e da forma como o sinal se processa ao longo do mesmo. Permite também visualizar de uma forma clara a interligação existente entre cada componente do sistema. Um bloco é um símbolo utilizado para definir uma função matemática, denominada função de transferência, aplicada no sinal de entrada produzindo um determinado sinal de saída. É importante mencionar que cada bloco é unilateral, ou seja, permite um sinal de entrada e um sinal de saída, mas não permite efectuar a operação inversa que torna a saída na entrada e a entrada na saída. Através da figura 2.6 torna-se claro a unilateralidade dos blocos, sendo que as setas representam o fluxo do sinal.

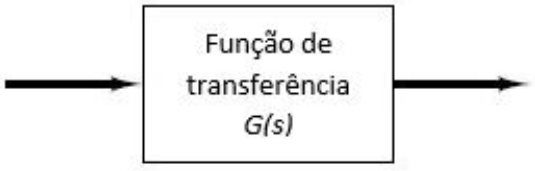


Figura 2.6: Elemento de um diagrama de blocos, adaptada de Ogata [2010].

Ainda de acordo com o mesmo autor, uma função de transferência de um sistema de equações dife-

renciais linear e invariável no tempo, pode ser definida como o rácio entre a transformada de Laplace do sinal de saída *Y(s)* e a transformada de Laplace do sinal de entrada *X(s)*, assumindo que todas as condições iniciais são zero. De uma forma resumida, uma transformada de Laplace é um artificio matemático que permite converter uma resolução de equações diferenciais numa resolução de equações polinomiais. Para uma explicação mais detalhada sobre as transformadas de Laplace, aconselha-se a consulta da referência Dyke [2014].

Assim, para um sistema linear e invariável no tempo, a função de transferência *G(s)* define-se como

$$\begin{aligned} G(s) &= \frac{Y(s)}{X(s)} \\ &= \frac{b_0 s^m + b_1 s^{m-1} + ... + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + ... + a_{n-1} s + a_n} \end{aligned}$$

Ao utilizar o conceito de função de transferência, é possível representar a dinâmica de um sistema através de equações algébricas dependentes da variável complexa *s*.

Um diagrama de blocos contém a informação relativa ao comportamento dinâmico mas não inclui qualquer informação relativa à construção física do sistema. Consequentemente, muitos sistemas distintos e sem qualquer tipo de relação podem ser representados pelo mesmo diagrama de blocos.

Para além do elemento bloco existe ainda o ponto de adição, como se pode observar na figura 2.7 que indica, tal como o próprio nome sugere, a operação de adição ou subtracção. O símbolo positivo ou negativo indica se o sinal está a ser adicionado ou subtraído. É importante manter a coerência em termos dimensionais e em termos de unidades ao efectuar a adição ou subtracção.

Pode ainda também ser considerado como elemento, o ponto de ramificação, onde o sinal se separa seguindo para outros blocos ou pontos de adição.

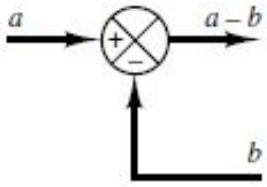


Figura 2.7: Ponto de adição, adaptada de Ogata [2010].

Através da figura 2.8 é possível observar um exemplo de um diagrama de blocos de um sistema de malha fechada, conceito já previamente introduzido. O sinal de saída *C(s)* é reintroduzido no sistema (*feedback*) através do ponto de adição, onde é comparado com o sinal de entrada de referência *R(s)* gerando o desvio *E(s)* que quando sujeito à função de transferência *G(s)* gera um novo valor para *C(s)*.

Quando o sinal de saída é enviado de volta para ser comparado com o sinal de entrada, é necessário que este seja convertido para que a comparação seja possível. Tome-se como exemplo um sistema de controlo de temperatura onde o sinal de saída é normalmente a temperatura controlada. Tendo unidades de temperatura, o sinal de saída terá que ser convertido para uma força, uma posição ou uma voltagem antes de poder ser comparado com o sinal de entrada. A dita conversão é conseguida através do elemento de *feedback* cuja função de transferência se representa por *H(s)*. A implementação deste elemento no diagrama de blocos fará com que o sinal a ser reintroduzido para ser comparado (*feedback*) passe a ser o sinal $B(s) = H(s)C(s)$.

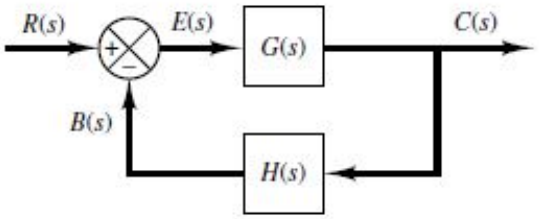


Figura 2.8: Diagrama de blocos de um sistema de malha fechada com função de transferência de conversão de *feedback*, adaptada de Ogata [2010].

A configuração de um diagrama de blocos de um sistema de malha aberta é apresentado na figura 2.9. Após a descrição de um sistema de malha fechada, o sistema de malha aberta é simples de entender, tendo apenas um sinal de entrada *R(s)* sujeito a funções de transferência que produzem o sinal de saída *C(s)*. Para este caso, não existe qualquer tipo de variação do sinal de saída por mais que se execute o sistema.

Figura 2.9: Diagrama de blocos de um sistema de malha aberta. *feedback*, adaptada de Ogata [2010].

Ainda de acordo com Ogata [2010], a função de transferência da malha fechada é dada por

$$C(s) = \frac{G(s)}{1 + G(s)H(s)}R(s) \tag{2.20}$$

após uma determinada dedução que poderá ser consultada na mesma referência (pág.19) e que não se considerou relevante para o presente trabalho.

2.2.3.3 Controladores

Um controlador automático, ou apenas controlador, é responsável por comparar a medição obtida na saída do sistema com a medição de referência, calcular o desvio associado e aplicar um sinal de controlo que reduza, idealmente, o desvio para valor nulo.

Considerando o exemplo apresentado na figura 2.10, é possível concluir que o desvio é calculado através do ponto de adição e ajustado por uma determinada função ou aparelho que para o caso do exemplo se trata de um amplificador.

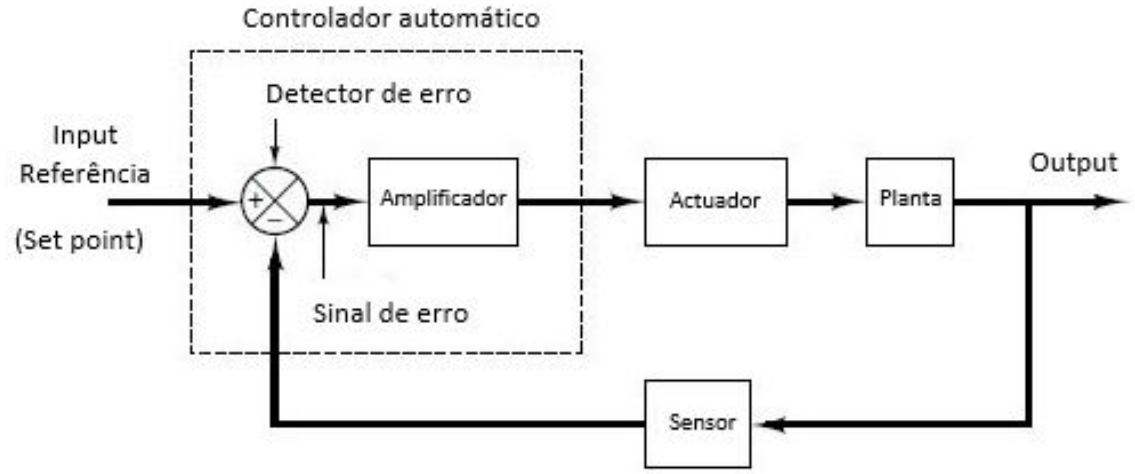


Figura 2.10: Diagrama de blocos de um sistema de controlo industrial, adaptada de Ogata [2010].

De uma forma geral, os controladores industriais podem ser controladores de duas posições (on-off) ou então controladores *PID*. Estes últimos são os mais importantes e os mais utilizados, pois o caso dos controladores de duas posições não permite o ajuste do desvio. A sigla *PID* traduz-se em *Proporcional*, *Integral* e *Derivativo*, sendo que cada componente poderá ser utilizada independentemente. Assim, é possivel ter-se um controlador com uma configuração *P*, *PI*, *PD* ou por fim *PID*. O caso do amplificador é um exemplo de um controlador proporcional. Considere-se que *e* representa o desvio calculado entre o sinal e o valor de referência, e *u* o valor de saída do controlador. Por analogia com exemplo utilizado, *e* será o sinal que entra no amplificador e *u* o sinal que sai. Matematicamente, a acção de controlo do controlador *PID* é dada por

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt} \tag{2.21}$$

onde K_p representa o ganho proporcional, K_I o ganho integral e K_D o ganho derivativo. Os parâmetros ajustáveis de um controlador *PID* são apenas os ganhos de cada uma das suas partes.

Tendo em conta as funções de transferência para as diferentes configurações do controlador *PID* descritas em Ogata [2010] e avaliando a sua influência no sistema através dos diagramas de Bode descri-

tos em Nise [2015] é possível descrever a influência de cada parte do controlador no comportamento de um sistema. O controlador proporcional (*P*) contribui para a estabilidade do sistema permitindo uma velocidade de resposta mediana contudo, não garante que o valor de referência seja atingido. Já para o caso do controlador integral (*I*) a velocidade de resposta é lenta, contudo este controlador filtra perturbações e garante que se atinge o valor de referência. É necessário ter atenção ao aplicar um controlador *I* pois este poderá induzir oscilações. Para o caso do controlador derivativo (*D*) tem-se uma velocidade de resposta elevada ao custo de sensibilidade a ruído.

É importante ter em conta que nem sempre a estabilidade do sistema é garantida na aplicação de um controlador *PID*.

2.2.3.4 Representação espaço de estados

Segundo Nise [2015] existem duas abordagens possíveis no desenvolvimento e análise de sistemas de controlo em malha fechada. A primeira é denominada clássica ou abordagem no domínio da frequência. Esta é baseada na conversão das equações diferenciais do sistema numa função de transferência, gerando um modelo matemático do sistema que relaciona algebricamente a representação do sinal de saída com o sinal de saída. A substituição da equação diferencial pela equação algébrica não só simplifica a representação de subsistemas individuais como simplifica a modelação de subsistemas interligados.

A primeira desvantagem da abordagem clássica deve-se à sua aplicabilidade, uma vez que pode ser aplicada apenas em sistemas lineares e que não variam no tempo conhecidos por *LTI* (linear time invariant) ou em sistemas que possam ser aproximados aos mesmos.

Uma grande vantagem da abordagem no domínio de frequência é o facto de permitir realizar a análise da estabilidade e da resposta transitória de uma forma rápida. Desta forma, é possível avaliar o efeito de uma variação de parâmetros do sistema até que um determinado ponto de funcionamento desejado seja atingido.

Com o aumento da necessidade de sistemas de controlo, muito incentivada devido ao inicio da exploração espacial, a modelação de sistemas através de equações diferenciais lineares e que não variam no tempo e subsequentes funções de transferência, tornou-se inadequada.

A representação em espaço de estados (também denominada abordagem moderna ou abordagem no domínio do tempo) é um método unificado que permite modelar, analisar e projectar uma grande variedade de sistemas. Esta representação permite, por exemplo, representar sistemas não lineares que tenham folgas, saturação ou zonas mortas e para além disso consegue lidar com sistemas com

condição inicial diferente de zero.

Para o caso do presente trabalho, a representação em espaço de estados apenas será utilizada para o caso de sistemas lineares e que não variam no tempo ou para sistemas linearizados, tal como foi apresentado no ponto 2.2.2 por exemplo.

A representação espaço de estado pode variar consoante se pretenda que o sistema seja modelado em continuo ou em discreto. Segundo Panos J Antsaklis [2010], a representação em espaço de estados de um sistema de dimensão finita dinâmico e discreto é dado por equações com a forma

$$x_i(k+1) = f_i(k, x_1(k), ..., x_n(k), u_1(k), ..., u_m(k)) \quad i = 1, ..., n \tag{2.22a}$$

$$y_i(k) = g_i(k, x_1(k), ..., x_n(k), u_1(k), ..., u_m(k)) \quad i = 1, ..., p \tag{2.22b}$$

para $k = k_0, k_0 + 1, ...$, sendo k_0 um número inteiro. Assumindo $x(k)^\top = (x_1(k), ..., x_n(k))$, $f(\cdot)^\top = (f_1(\cdot), ..., f_n(\cdot))$, $u(k)^\top = (u_1(k), ..., u_m(k))$, $y(k)^\top = (y_1(k), ..., y_p(k))$ e $g(\cdot)^\top = (g_1(\cdot), ..., g_m(\cdot))$ é possível reescrever as equações 2.22

$$x(k+1) = f(k, x(k), u(k)) \tag{2.23a}$$

$$y(k) = g(k, x(k), u(k)) \tag{2.23b}$$

assumindo que $f : Z \times R^n \times R^m \to R^n$ e que $g : Z \times R^n \times R^m \to R^p$ e que Z representa um número inteiro.

Sendo f uma função, para um dado $k_0, x(k_0)$ e um dado $u(k), k = k_0, k_0 + 1, ...$, a equação 2.23a apresenta uma solução única $x(k)$ que existe para todo $k = k_0, k_0 + 1, ...$ Para alem disso, para as mesmas condições, $y(k)$ é unicamente definida para $k = k_0, k_0 + 1, ...$

Para a formulação apresentada, k_0 denota o tempo inicial, k denota o tempo, $u(k)$ denota o *input* do sistema (no instante k), $y(k)$ denota o *output* do sistema, ou a resposta do sistema (no instante k), $x(k)$ define o estado (no instante k) e $x_i(k), i = 1, ..., n$ denota as variáveis de estado. A equação 2.23a é denominada equação de estado enquanto que a 2.23b é denominada equação de resposta ou equação de *output*. É importante mencionar que no caso de sistemas descritos em continuo através de equações diferenciais ordinárias, é possível evoluir o tempo progressivamente ou regressivamente. Contudo para o caso de sistemas discretos descritos por 2.23, é necessário restringir a evolução do tempo k para o caso progressivo para que sejam garantidas soluções únicas.

Para consulta de outras abordagens para a representação em espaços de estados, aconselha-se a consulta da referência Panos J Antsaklis [2010].

2.2.4 Planeamento

Relembrando a informação apresentada no inicio do presente capitulo, distinguem-se dois tipos de navegação, a reactiva e a baseada em mapa sendo que para o presente trabalho apenas se destaca a relevância do segundo tipo. Segundo Roland Siegwart [2011], dado um determinado mapa e um determinado objectivo, o planeamento envolve identificar a trajectória que despois de executada, levará o robô a atingir o seu objectivo. O planeamento é uma competência estratégica do robô para resolver o problema de decisão sobre o que fazer ao longo do tempo para que este possa atingir os seus objectivos.

Existe ainda uma segunda competência importante que um robô pode apresentar durante a navegação que é a capacidade de este se adaptar a eventuais perturbações. É importante mencionar que ao ser apresentada a distinção entre a navegação baseada em mapa e a navegação reactiva, não implica que a navegação baseada em mapa não possua uma componente reactiva, não dependendo no entanto exclusivamente dessa componente (ao contrário dos robôs baseados em navegação reactiva).

Desta forma, o planeamento será acompanhado por um algoritmo de localização que será a componente reactiva responsável pela reformulação do planeamento e que irá tentar garantir que o robô atinja o seu objectivo. O problema de localização será discutido num ponto mais adiante no presente trabalho, dedicando-se o presente ponto apenas ao problema de planeamento.

Para robôs móveis que operam numa superfície regular, geralmente representa-se a postura do robô através das variáveis (x, y, θ). Contudo os robôs são não holonómicos quando apresentam uma configuração de tracção diferencial. Para tais robôs, os constrangimentos não holonómicos limitam a velocidade do robô $(\dot{x}, \dot{y}, \dot{\theta})$ em cada configuração (x, y, θ). Segundo Roland Siegwart [2011], a abordagem mais comum a utilizar passa por assumir apenas para o planeamento que o robô é, por suposição, holonómico, o que irá simplificar bastante o processo. Esta suposição é especialmente comum em robôs de tracção diferencial uma vez que conseguem rodar sobre um ponto e portanto, uma trajectória holonómica pode ser facilmente imitada caso o ângulo do robô, em termos de orientação, não for importante.

Para além da simplificação sugerida, ainda é costume assumir que o robô é simplesmente um ponto. Deste modo é possível simplificar ainda mais o planeamento para uma representação em 2D utilizando apenas os eixos x e y do espaço.

A representação do espaço de operação do robô pode variar desde uma descrição geométrica continua

a uma decomposição geométrica de vários elementos ou até mesmo um mapa topológico. O primeiro passo em qualquer sistema de planeamento é transformar o modelo espacial, eventualmente continuo, num mapa discreto adequado ao algoritmo de planeamento desejado. Múltiplas abordagens para algoritmos de planeamento podem ser consultadas na referência LaValle [2006].

2.3 Percepção

Uma componente importante no desenvolvimento de um robô, é a sua capacidade de avaliar o ambiente e o espaço em que está inserido. Esta capacidade é adquirida ao incorporar na estrutura do robô sensores capazes de extrair a informação necessária para que o robô possa responder à influência do seu exterior.

2.3.1 Sensores

Na robótica é usada uma grande variedade de sensores, cuja finalidade pode ir da simples medição da temperatura interna dos componentes electrónicos do robô ou da velocidade de rotação dos motores até medições mais complexas como a postura global do robô. Segundo Roland Siegwart [2011] os sensores podem ser distinguidos por serem internos ou externos e por serem activos ou passivos. Tal como a própria definição indica, os sensores internos são aqueles que medem parâmetros internos do sistema (robô) como por exemplo a velocidade dos motores, os ângulos das juntas de um braço robótico ou a voltagem da bateria. Para o caso dos sensores externos estes medem tudo o que é exterior ao sistema, como por exemplo, medições de distância, intensidade de luminosidade ou intensidade sonora. Os sensores passivos são aqueles que medem uma determinada energia proveniente do ambiente, como por exemplo sondas de temperatura ou microfones. Os sensores activos, ao contrário do que acontece para os passivos, necessitam de emitir energia para o ambiente medindo posteriormente a reacção do ambiente a essa emissão.

Será relevante sublinhar a importância dos sensores de visão. Estes sensores recriam a função da visão humana, permitindo obter uma enorme quantidade de informação relativamente ao ambiente onde o robô está inserido assim como uma interacção rica e inteligente em ambientes dinâmicos. A função do sensor passa por capturar feixes de luz e converter os mesmos numa imagem digital. Seguidamente e exteriormente ao sensor, a imagem digital será sujeita a um processamento que permite salientar informação como o cálculo de profundidade, detecção de movimento, identificação de cores, detecção de características, reconhecimento de cenas, entre outras.

2.3.2 Câmaras omnidireccionais

A análise de imagens e o seu processamento são duas grandes áreas conhecidas como visão computacional e processamento de imagem respectivamente. Um sensor de visão já foi descrito no capitulo 2.3.1, assim como o valor acrescentado que traz a um sistema autónomo.

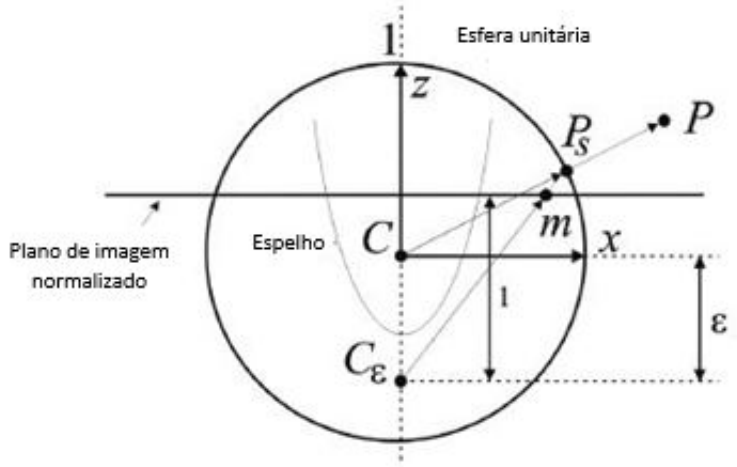
De acordo com Roland Siegwart [2011], uma câmara omnidireccional é uma câmara que permite um campo de visão alargado, apresentando pelo menos uma abertura de 180 graus. Existem diferentes formas de construir uma câmara omnidireccional. Câmaras do tipo *dioptic* utilizam uma combinação de lentes com configurações especificas (por exemplo uma lente olho de peixe) e normalmente permitem um campo de visão ligeiramente superior do que 180 graus. As câmaras do tipo *catadioptric* combinam uma câmara normal com um espelho com uma configuração específica, como por exemplo parabólica, hiperbólica ou elíptica e permitem um campo visão largamente superior a 180 graus em elevação e 360 de azimute. Por fim, as câmaras do tipo *polydioptric* utilizam múltiplas câmaras que geram um campo de visão sobreposto, sendo a única configuração que permite uma visão verdadeiramente omnidireccional. A figura 2.11 ilustra as diferentes configurações descritas.

Figura 2.11: (a) Câmara *Dioptric*; (b) Câmara *catadioptric*; (c) um exemplo de uma câmara *polydioptric* produzida por Immersive Media, adaptada de Roland Siegwart [2011].

Um sistema de visão é denominado central quando os raios ópticos relativos a um objecto a ser observados, se intersectam num único ponto em 3D denominado centro de projecção. O centro de projecção de uma câmara é denominado centro óptico da câmara. Todas as câmaras olho de peixe são centrais.

Com a publicação Geyer and Daniilidis [2001], demonstrou-se que todas as projecções numa câmara do tipo *catadioptric* assim como as projecções de perspectiva, podem ser equivalentes a uma projec-

ção de uma esfera, centrada no centro de projecção, num plano perpendicular ao centro de projecção, e a uma distância ε do centro da esfera. A descrição apresentada é ilustrada na imagem 2.12.



Espelho	ε
Parábola	1
Hipérbole	$\frac{d}{\sqrt{d^2+4l^2}}$
Elipse	$\frac{d}{\sqrt{d^2+4l^2}}$
Perspectiva	0

Figura 2.12: Modelo de projecção unificado para câmaras do tipo *catadioptric* centrais proposto por Geyer and Daniilids, adaptada de Roland Siegwart [2011].

O objectivo do modelo é encontrar a relação entre a direcção do ponto da cena a ser observado e a coordenada do píxel correspondente na imagem captada. O modelo proposto em Geyer and Daniilidis [2001] segue um processo de quatro passos. Seja $P = (x, y, z)$ um ponto da cena projectado no espelho centrado em C (ver Fig. 2.12).

1. O primeiro passo consiste em projectar o ponto da cena na esfera unitária

$$P_s = \frac{P}{\|P\|} = (x_s, y_s, z_s) \tag{2.24}$$

2. As coordenadas do ponto são transformadas para um novo referencial de referência centrado em $C_\varepsilon = (0, 0, -\varepsilon)$

$$P_\varepsilon = (x_s, y_s, z_s + \varepsilon) \tag{2.25}$$

Observe-se que ε varia entre 0 (espelho plano) e 1 (espelho parabólico) servindo o caso do espelho plano apenas para referência, uma vez que não se pratica na realidade. O valor de ε pode ser obtido à custa da distância *d* entre os pontos focais (*foci*) da secção cónica e o *latus rectum l*, tal como apresentado na tabela da figura 2.12. O *latus rectum* de uma secção cónica é a corda passante pelo ponto focal e que é paralela à directriz da secção cónica.

3. P_ε é então projectado no plano de imagem normalizado distanciado de 1 relativamente a C_ε

$$\tilde{m} = (x_m, y_m, 1) = (\frac{x_s}{z_s + \varepsilon}, \frac{y_s}{z_s + \varepsilon}, 1) = g^{-1}(P_s) \tag{2.26}$$

4. Por fim, o ponto $\tilde{m}$ é representado na imagem da câmara através do ponto $\tilde{p} = (u, v, 1)$ sendo transformado através da matriz de parâmetros intrínsecos da câmara *A*

$$\tilde{p} = A \cdot \tilde{m} \tag{2.27}$$

A função g^{-1} é bijectiva e sua inversa g é dada por

$$P_s = g(m) \sim \begin{bmatrix} x_m \\ y_m \\ 1 - \varepsilon \frac{x_m^2 + y_m^2 + 1}{\varepsilon + \sqrt{1 + (1 - \varepsilon^2)(x_m^2 + y_m^2)}} \end{bmatrix} \tag{2.28}$$

onde $\sim$ indica que g é proporcional ao vector apresentado.

A equação 2.28 pode ser obtida invertendo a equação 2.26 e impondo a restrição que obriga P_s a estar na esfera unitária, ou seja $x_s^2 + y_s^2 + z_s^2 = 1$. A partir deste constrangimento é possível obter uma expressão para z_s em função de ε, x_m e y_m. Contudo esta transformação não é relevante para o presente trabalho, podendo ser consultada a dedução em Barreto and Araujo [2001].

Para se obter o factor de escala, será suficiente normalizar $g(m)$ na esfera unitária. Note-se que a equação 2.28 é o modelo principal de projecção para câmaras centrais do tipo *catadioptric*. A mesma equação expressa a relação entre o ponto *m* no plano da imagem normalizado e o vector unitário P_s no referencial de referência do espelho.

Apesar de o modelo descrever todas as câmaras centrais do tipo *catadioptric* e as câmaras de perspectiva, não permite descrever câmaras de olho de peixe (*dioptric*). Uma extensão para o modelo apresentado foi proposto por Ying and Hu [2004] onde se aproximava uma câmara de olho de peixe a uma do tipo *catadioptric*. Apesar de tudo a abordagem proposta apenas produz resultados com uma precisão limitada que podem ser justificados pelo facto de as três variantes das câmaras do tipo *catadioptric* (parábola, hipérbole e elipse) poderem ser representadas através da mesma função paramétrica, enquanto os modelos de projecção para câmaras de olho de peixe variam consoante a amplitude da lente. Para solucionar o problema, foi proposto por Scaramuzza et al. [2006a] uma nova escolha para a função g que passa pela utilização de um polinómio de Taylor cujos coeficientes são obtidos através do processo de calibração, ultrapassando assim a falta de conhecimento acerca dos parâmetros intrínsecos da câmara. Desta forma, a relação entre o ponto na imagem normalizada $\tilde{m} = (x_m, y_m, 1)$ e o vector unitário P_s no referencial do olho de peixe pode ser descrita por

$$P_s = g(m) \sim \begin{bmatrix} x_m \\ y_m \\ a_0 + a_2\rho^2 + ... + a_N\rho^N \end{bmatrix} \tag{2.29}$$

sendo $\rho = \sqrt{x_m^2 + y_m^2}$ a distância euclideana relativamente ao centro do plano de imagem. É de notar que o termo de primeira ordem ($a_1 \rho$) do polinómio não é considerado. Isto deve-se ao facto de a primeira derivada do polinómio, calculada em $\rho = 0$ ser nula tanto para as câmaras de olho de peixe como para as câmaras do tipo *catadioptric* (isto poderá ser confirmado derivando a equação 2.28). Devido ao facto de se tratar de um polinómio, a expressão permite abranger câmaras do tipo *catadioptric*, olho de peixe e perspectiva, sendo apenas necessário variar o grau do polinómio.

Para obter os parâmetros intrínsecos de uma câmara de olho de peixe será necessário recorrer ao processo de calibração. Os métodos mais comuns tiram partido de padrões de xadrez planares que são observados em diferentes posições e orientações. Para o caso das câmaras omnidireccionais é importante que as imagens para calibração apresentem o padrão de xadrez em posições em torno da câmara, abrangendo o máximo possível do seu campo de visão.

Para se concretizar a calibração existem *toolboxes* de licença gratuita para o ambiente *Matlab* que podem diferir no modelo de projecção adoptado ou no padrão de calibração utilizado. Destaca-se a *toolbox* apresentada em Scaramuzza et al. [2006b] e em Scaramuzza et al. [2008] que tira proveito do modelo por polinómios de Taylor (abordagem proposta pelo mesmo autor).

2.4 Localização

Como foi discutido no ponto 2.2, para um robô navegar é necessário um mapa ou uma condição que o faça reagir. O ponto 2.3 mostrou que o robô possui normalmente meios de avaliar o ambiente que o rodeia bem como o seu próprio movimento. Estes dispositivos, os sensores, serão essenciais para a sua auto-localização. Neste ponto assume-se que o robô dispõe de informação sobre a trajectória a realizar e sobre como realizá-la (através de uma lei de controlo), na condição de conhecer a sua posição no ambiente. Será por isso necessário inferir a sua posição, com base na informação fornecida pelos sensores. A esse processo designa-se localização.

2.4.1 Dead reckoning

Para se obter a informação relativa à postura do robô existem duas vias tal como foi apresentado no ponto 2.3.1: a que recorre a sensores externos e a que recorre a sensores internos. De acordo com Corke [2011], à estimativa da postura do robô baseada numa estimativa prévia e na informação sobre o espaço percorrido, dá-se o nome de *Dead Reckoning*.

O conceito de *Dead Reckoning* surge em diversos contextos de localização, com maior predominância na navegação de navios. Quando aplicado ao solo, normalmente utiliza-se o termo odometria,

que evidencia o facto de a informação relativa à velocidade ser proveniente das rodas. É fácil de associar o termo odometria ao aparelho de medida denominado odómetro, uma vez que o princípio de funcionamento é o mesmo.

Segundo Sebastian Thrun [2005] a odometria é normalmente obtida através da integração da informação proveniente dos *encoders* das rodas. Apesar de aparentar ser um método robusto, a odometria sofre de um problema que a inviabiliza quando se procura uma boa precisão, o do arrastamento. É ainda importante mencionar que a odometria é retrospectiva, ou seja, apenas se terá informação após o robô se ter deslocado. Ainda de acordo com Sebastian Thrun [2005], este efeito não traz qualquer tipo de impacto na implementação de filtros, mas torna a informação inútil para um planeamento preciso e para controlo.

Seguindo a formulação apresentada em Roland Siegwart [2011], assuma-se que um determinado robô diferencial descreve um arco numa escala temporal pequena. A distância percorrida pela roda esquerda e pela roda direita são dadas por D_l e D_r respectivamente. As leituras dos *encoders* são obtidas em picos de *encoder*, ou seja, se um *encoder* apresentar 360 picos por volta, cada pico de *encoder* corresponde a um grau, perfazendo a volta completa os 360 graus. A relação não será tão directa se o número de picos de *encoder* por volta *N*, for diferente de 360.

Assim, será conveniente formular uma variação de picos de *encoder* por cada leitura, dada por

$$\Delta tick = tick' - tick \tag{2.30}$$

onde o apóstrofo representa a nova leitura. Com isto torna-se possível definir as distâncias percorridas por cada roda,

$$D = 2\pi r \frac{\Delta tick}{N} \tag{2.31}$$

onde D tanto pode ser a roda esquerda como a roda direita. Será lógico avaliar a distância percorrida média em vez de avaliar a distância percorrida de cada roda independentemente

$$D_c = \frac{D_l + D_r}{2} \tag{2.32}$$

Introduzindo o modelo cinemático do robô diferencial apresentado na secção 2.2.2 e substituindo Tv_{left} e Tv_{right} pelos deslocamentos das rodas medidos pelos *encoders* é definida a odometria:

$$x' = x + D_c cos\phi \tag{2.33a}$$

$$y' = x + D_c cos\phi \tag{2.33b}$$

$$\phi' = \phi + \frac{D_r - D_l}{L} \tag{2.33c}$$

sendo L a distância entre rodas.

Como já foi mencionado, a odometria oferece uma estimativa da posição com precisão limitada, sendo que o seu erro cresce com o tempo. É no entanto possível implementar modelos de erro para tentar minimizar o efeito do arrastamento como é por exemplo sugerido em Roland Siegwart [2011]. O arrastamento é causado por deformações das rodas, escorregamentos, irregularidades no solo, erros nos *encoders* e outras folgas mecânicas que são praticamente impossíveis de prever, controlar ou contornar.

2.4.2 Filtros

Segundo Solà [2007] um filtro tem em conta informação prévia e a informação instantânea para inferir as características de um fenómeno de interesse. A grande vantagem desta técnica é o facto de toda a informação prévia contribuir para uma aprendizagem que irá permitir tornar a operação cada vez mais precisa. No presente trabalho será usado um filtro como método de localização probabilístico que refina o conhecimento acerca do estado do robô a partir de medições fornecidas pelo mesmo.

2.4.2.1 Formulação do problema de filtragem

Seguindo a formulação de Solà [2007], considere-se um sistema dinâmico $\sum$ sujeito a perturbações aleatórias. Considere-se também sensores que fornecem ao sistema informação sujeita a ruído. Para além disso assuma-se que os engenheiros responsáveis pelo desenvolvimento do robô possuem algum conhecimento acerca da dinâmica do sistema e de como as leituras dos sensores se relacionam com o estado do sistema. Considere-se ainda que se conhece o tipo de perturbações a que o sistema está sujeito e a forma como estas afectam a dinâmica do sistema, o mesmo para o ruído na medição dos sensores. Por fim, considere-se que se sabe o estado inicial do sistema.

A equação determinística

$$x_k = f_k(x_{k-1}, \omega_k) \tag{2.34}$$

é denominada equação de evolução e exprime, em discreto e na notação espaço de estado, a evolução Markoviana do estado do sistema x_k do instante $k-1$ até k quando sujeito a uma acção de controlo incerta (visto que contempla o ruído) ω_k.

A segunda equação determinística

$$y_k = h_k(x_k, \upsilon_k) \tag{2.35}$$

é denominada equação de medida e descreve as leituras com ruído (υ_k) dos sensores no intervalo k.

O conhecimento estocástico é descrito por uma função densidade de probabilidade (*fdp* ou *pdf* - *probability density function*)

$$p(\omega_k) \tag{2.36}$$

denominada controlo incerto e que exprime a natureza das perturbações a entrar no sistema.

a *pdf* denominada ruído da medição representa-se por

$$p(\upsilon_k) \tag{2.37}$$

enquanto que a *pdf*

$$p(x_0) \tag{2.38}$$

descreve o conhecimento prévio acerca da condição inicial, sendo esta denominada inicial prévia.

Toda esta formulação está resumida na figura 2.13.

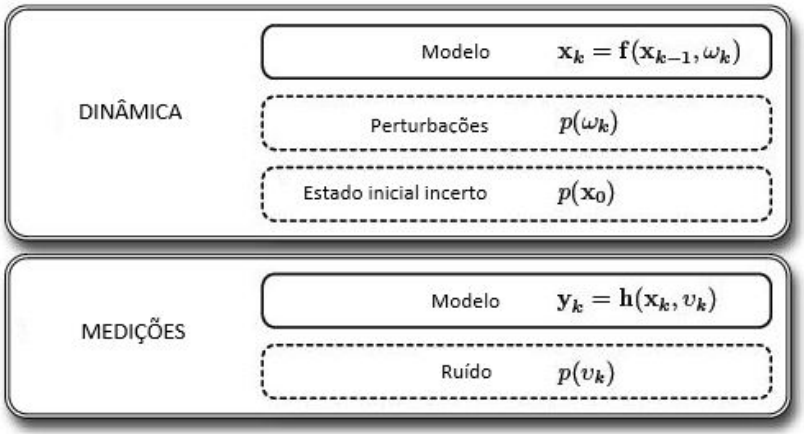


Figura 2.13: Formulação do problema de filtragem, adaptada de Solà [2007].

2.4.2.2 Filtro de Kalman

De acordo com Roland Siegwart [2011], a teoria do filtro de Kalman assume que o sistema é linear e que a configuração geral do robô, o modelo de erro de odometria e o modelo de erro para a medida são afectados por ruído branco Gaussiano. Resumidamente, toda a incerteza é considerada Gaussiana e tem-se o constrangimento de que a equação de evolução e a equação de medida têm que ser lineares.

De acordo com a formulação apresentada em Solà [2007], considere-se o sistema Gaussiano linear

$$x_k = Fx_{k-1} + G\omega_k \tag{2.39a}$$

$$y_k = Hx_k + \upsilon_k \tag{2.39b}$$

$$p(x_0) = \mathcal{N}(x_0 - \bar{x_0}; P_0) \tag{2.39c}$$

$$p(\omega_k) = \mathcal{N}(\omega_k - 0; Q) \tag{2.39d}$$

$$p(\upsilon_k) = \mathcal{N}(\upsilon_k - 0; R) \tag{2.39e}$$

$$\tag{2.39f}$$

onde

$$\mathcal{N}(x - \bar{x}; P) = \frac{1}{\sqrt{(2\pi)^n \, |P|}} exp(-\frac{1}{2}(x - \bar{x})^\top P^{-1}(x - \bar{x})) \tag{2.40}$$

é a *pdf* de uma variável x Gaussiana de dimensão n com média $\bar{x}$ e matriz de covariância P, e onde x_{k-1}, ω_k e υ_k são mutuamente independentes.

As propriedades das densidades Gaussianas podem ser consultadas na referência Solà [2007], assim como algumas noções da teoria das probabilidades e elementos para uma solução geral para filtros.

O filtro de Kalman é descrito por um conjunto de equações que estimam o valor médio e a matriz de covariância da *pdf* do estado do sistema. O processo de estimação ocorre em dois passos designados, por ordem, por *previsão* e *correcção*. A dedução destes passos poderá ser consultada nas obras Solà [2007] e Roland Siegwart [2011], não sendo contudo relevante para o presente trabalho, apenas importando a sua formulação final.

Filto de Kalman: previsão Definindo $\hat{x}$ para a estimação de x, o passo da previsão é dado por

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} \tag{2.41a}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^\top + G_k Q_k G_k^\top \tag{2.41b}$$

Onde $k|k-1$ indica o instante seguinte, ou seja, nesta previsão, está a ser calculado o valor estimado de x para o instante seguinte e a matriz de covariância P também para o instante seguinte, tendo como base os valores do instante actual.

Filtro de Kalman: correcção As equações para o passo da correcção são dadas por

$$Z_k = H_k P_{k|k-1} H_k^\top + R_k \tag{2.42a}$$

$$K_k = P_{k|k-1} H_k^\top \cdot Z_k^{-1} \tag{2.42b}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - H_k \hat{x}_{k|k-1}) \tag{2.42c}$$

$$P_{k|k} = P_{k|k-1} - K_k Z_k K_k^\top \tag{2.42d}$$

onde Z representa a matriz de covariância da variável denominada inovação $z = y - H\hat{x}$ que define o quanto a medida difere da medida esperada $H\hat{x}$. A inovação é uma variável Gaussiana com valor médio não nulo $\mathcal{N}(z; Z)$. A matriz K é denominada ganho de Kalman e corrige, de uma forma optimizada, a estimativa saída da previsão $\mathcal{N}(x - \hat{x}; P)$ proporcionalmente à inovação.

2.4.2.3 Filtro de Kalman estendido

O filtro de Kalman estendido não é mais do que o filtro de Kalman aplicado a sistemas não lineares, que é uma das restrições na aplicação do segundo. Desta forma, o filtro de Kalman estendido, a cada instante, lineariza as funções não lineares em torno da melhor e mais recente estimativa seguindo-se a aplicação das equações do filtro de Kalman do modelo linear. A linearização está descrita no apêndice *A* da obra de Solà [2007], não sendo relevante a dedução da linearização para o presente trabalho.

Considere-se o sistema Gaussiano não linear

$$x_k = f(x_{k-1}, \omega_k) \tag{2.43a}$$

$$y_k = h(x_k) + \upsilon_k \tag{2.43b}$$

$$p(x_0) = \mathcal{N}(x_0 - \bar{x_0}; P_0) \tag{2.43c}$$

$$p(\omega_k) = \mathcal{N}(\omega_k - \bar{\omega_k}; Q) \tag{2.43d}$$

$$p(\upsilon_k) = \mathcal{N}(\upsilon_k; R) \tag{2.43e}$$

onde o facto de $\bar{\omega}_k \neq 0$ surge por ser ter em conta acções de controlo conhecidas, ou seja, são conhecidos os valores médios das perturbações.

Filtro de Kalman estendido: previsão A equação de evolução é linearizada em torno da melhor estimativa prévia e da acção de controlo conhecida relativamente ao estado do sistema e à perturbação através das matrizes Jacobianas

$$F_x = \left.\frac{\partial f}{\partial x^\top}\right|_{\hat{x}_{k-1|k-1}, \bar{\omega}_k} \tag{2.44a}$$

$$F_\omega = \left.\frac{\partial f}{\partial \omega^\top}\right|_{\hat{x}_{k-1|k-1}, \bar{\omega}_k} \tag{2.44b}$$

necessárias na aplicação das equações de previsão

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, \bar{\omega}_k) \tag{2.45a}$$

$$P_{k|k-1} = F_x P_{k-1|k-1} F_x^\top + F_\omega Q_k F_\omega^\top \tag{2.45b}$$

Filtro de Kalman estendido: correcção A equação de medida é linearizada em torno da melhor estimativa prévia relativamente ao estado do sistema através da matriz Jacobiana

$$H = \left.\frac{\partial h}{\partial x^\top}\right|_{\hat{x}_{k|k-1}} \tag{2.46}$$

necessária nas equações de correcção

$$Z_k = H P_{k|k-1} H^\top + R_k \tag{2.47a}$$

$$K_k = P_{k|k-1} H^\top \cdot Z_k^{-1} \tag{2.47b}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - h(\hat{x_{k|k-1}})_k) \tag{2.47c}$$

$$P_{k|k} = P_{k|k-1} - K_k Z_k K_k^\top \tag{2.47d}$$

onde a inovação é agora definida por $z_k = y_k - h(\hat{x_{k|k-1}})_k$ com matriz de covariância Z_k.

Quando o filtro de Kalman estendido apresenta um bom desempenho, a inovação apresenta-se como uma variável Gaussiana com valor médio muito próximo de zero $\mathcal{N}(z; Z)$.

Capítulo 3

Caso de estudo

Com os conceitos introduzidos no capítulo 2 dedicado ao fundamento teórico serão, no presente capítulo, apresentadas as abordagens utilizadas no caso de estudo. Será importante referir que não se fará a distinção, de uma forma explícita, dos conceitos de navegação, percepção ou localização uma vez que esta será suficientemente clara aquando da aplicação destes conceitos.

Para o presente trabalho utilizou-se o ambiente *MATLAB* para desenvolver toda a programação e onde também se utilizou o pacote que permite a comunicação e operação com um *Raspberry Pi*. Utilizou-se também o *software SOLIDWORKS* para o desenvolvimento de peças a serem impressas em $3D$. A escrita do presente documento desenvolveu-se em linguagem *LaTeX* através do interpretador *TeXstudio*.

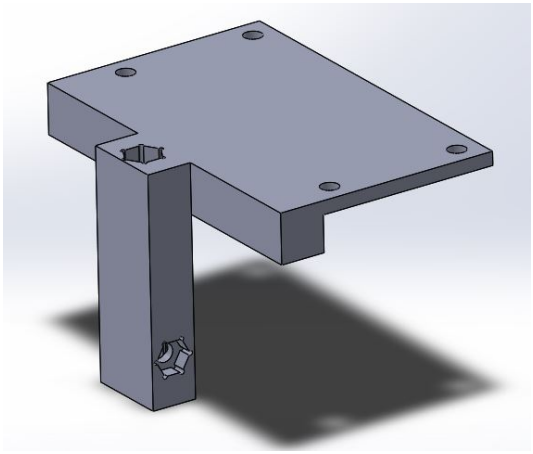
3.1 Estrutura do robô

O protótipo construído apresenta-se com uma configuração de tracção diferencial. O eixo entre rodas das rodas motoras está deslocado do centro de massa do robô que dispõe ainda de uma roda livre. A placa controladora de motores pertence ao conjunto *RD02* comercializado pela *ROBOT ELECTRONICS*, não tendo contudo sido utilizados os motores do mesmo conjunto. Os motores seleccionados são distribuídos pelo fabricante *Pololu* e são motores *DC* de 12 Volts com uma relação de transmissão de $131:1$. Estes motores têm ainda incorporados *encoders* de quadratura que apresentam uma resolução de 64 contagens por revolução do lado do veio do motor, o que corresponde a 8400 contagens por revolução no veio de transmissão. A 12 Volts, os motores produzem 80 *RPM*. Apesar de a placa controladora não ter sido desenhada para os motores propostos, uma vez que a alimentação é a mesma, e os *encoders* apresentam as mesmas características, os equipamentos são compatíveis.

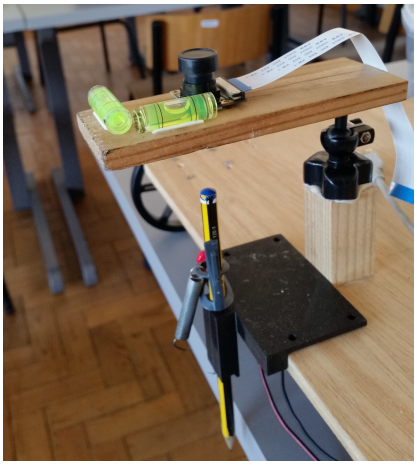
Foi também utilizado um *Raspbery Pi 2 model B*, juntamente com uma câmara com um sensor de 5

megapíxel com uma resolução máxima de *1080p*. Foi ainda acoplada à câmara uma lente de olho de peixe que aumentou o ângulo de abertura de cerca 72 graus para 160 graus. O conjunto da câmara e lente é comercializado pelo fabricante *Waveshare*.

Foi ainda desenvolvido um apoio para o instrumento de impressão através de *SOLIDWORKS* que posteriormente foi impresso em *3D* e incorporado no robô tal como apresentado na figura 3.1.



(a) Modelo produzido em *SOLIDWORKS*.



(b) Resultado da impressão *3D*.

Figura 3.1: Apoio para o instrumento de escrita.

3.2 Cinemática

Tendo em conta a informação apresentada no capítulo 2.2.2, a velocidade do robô, sendo este de tracção diferencial, pode ser calculada a partir das velocidades independentes das rodas

$$v_A = \frac{v_l + v_r}{2} \tag{3.1}$$

que irão definir a velocidade absoluta do robô. Segundo Beer et al. [2009], qualquer movimento plano de um corpo pode ser substituído por uma translação, definida pelo movimento de um ponto de referência arbitrário e por uma rotação em redor do mesmo ponto. Para o caso do presente trabalho o dito ponto de referência será a conhecida intersecção entre o eixo entre rodas e a linha perpendicular, imaginária, que intersecta o ponto de contacto do instrumento de escrita (ponto *A*). A velocidade absoluta de um ponto material, que neste caso será o ponto de contacto do instrumento de escrita (ponto *B*), pode ser obtida da equação da velocidade relativa

$$\vec{v_B} = \vec{v_A} + \vec{v_{B/A}} \tag{3.2}$$

onde o membro à direita da equação é uma soma vectorial. A velocidade v_a corresponde à translação do ponto A, enquanto a velocidade relativa $v_{B/A}$ descreve a influência da rotação do corpo em relação a A, avaliada relativamente a um sistema de coordenadas com centro A e de orientação fixa.

A velocidade relativa $v_{B/A}$ pode ser definida através do produto externo entre o vector de posição B relativo a A ($r_{B/A}$) e a velocidade angular do corpo relativamente ao sistema de coordenadas de referência (ωk), permitindo reescrever a equação 3.2

$$\vec{v_B} = \vec{v_A} + \vec{\omega k} \times \vec{r_{B/A}} \tag{3.3}$$

Para o caso do modelo utilizado, a distância $r_{B/A}$ entre o ponto já referido de entre rodas e o ponto de contacto do instrumento de escrita, foi atribuída à variável l, que desta forma permite definir a velocidade relativa $v_{B/A}$ à qual se irá atribuir a notação v_n

$$\vec{v_n} = \vec{\omega k}\, l \tag{3.4}$$

sendo que a velocidade angular é obtida à custa das velocidades das rodas

$$\omega = \frac{-v_l + v_r}{L} \tag{3.5}$$

onde L representa a distância entre rodas.

Assim, ao ter em conta a figura 3.2, é possível visualizar a influência da translação descrita pela velocidade normal v_n e a influência da rotação pela velocidade tangencial v_t, perpendicular à anterior visto que esta é colinear com o vector de posição entre os pontos avaliados. Apesar de exageradamente afastados para uma fácil visualização, a figura apresenta as velocidades avaliadas entre o ponto do eixo entre rodas e o ponto de contacto da caneta. Ao decompor as velocidades mencionadas, é possível descrever as equações do movimento para o ponto do eixo entre rodas,

$$\dot{x} = v_n cos\theta - v_t sin\theta \tag{3.6a}$$

$$\dot{y} = v_n sin\theta + v_t cos\theta \tag{3.6b}$$

$$\dot{\theta} = \frac{-v_l + v_r}{L}. \tag{3.6c}$$

Introduzindo as equações 3.1, 3.4 e 3.5, é possível reescrever todas as equações em ordem às velocidades das rodas

$$\dot{x}' = \frac{v_l + v_r}{2}\, cos\theta - \frac{v_l + v_r}{L}\, l\, sin\theta \tag{3.7a}$$

$$\dot{y}' = \frac{v_l + v_r}{2}\, sin\theta + \frac{v_l + v_r}{L}\, l\, cos\theta \tag{3.7b}$$

$$\dot{\theta}' = \frac{-v_l + v_r}{L}. \tag{3.7c}$$

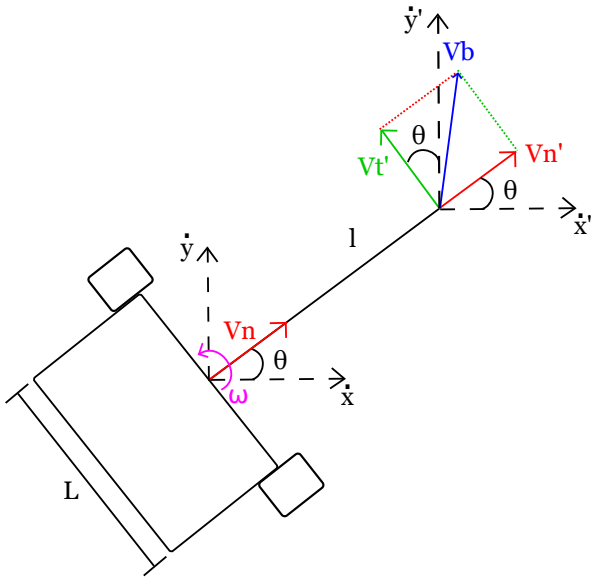


Figura 3.2: Esboço das velocidades relativas consideradas para o modelo cinemático.

É então possível formular a cinemática do robô através da notação matricial relativa ao sistema de coordenadas cartesiano de referência

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{cos\theta}{2} + \frac{l\ sin\theta}{L} & \frac{cos\theta}{2} - \frac{l\ sin\theta}{L} \\ \frac{sin\theta}{2} - \frac{l\ cos\theta}{L} & \frac{sin\theta}{2} + \frac{l\ cos\theta}{L} \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} v_l \\ v_r \end{bmatrix} \tag{3.8}$$

3.3 Odometria

Com a cinemática do robô definida, com a informação apresentada no capítulo 2.2.2, é possível definir as equações que descrevem a evolução da postura do robô em cada instante *k*,

$$x_{k+1} = x_k + v_t T_s cos\theta_k - v_n T_s sin\theta_k \tag{3.9a}$$

$$y_{k+1} = y_k + v_t T_s sin\theta_k + v_n T_s sin\theta_k \tag{3.9b}$$

$$\theta_{k+1} = \theta_k + \omega_k T_s \tag{3.9c}$$

É possível simplificar a notação criando variáveis para o incremento,

$$\Delta d_l = v_l T_s \tag{3.10a}$$

$$\Delta d_r = v_r T_s \tag{3.10b}$$

Introduzindo as equações 3.1, 3.4, 3.5 e 3.10 nas equações 3.9 e aplicando a notação matricial, é possível reescrever as equações

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{cos\theta}{2} + \frac{l\ sin\theta}{L} & \frac{cos\theta}{2} - \frac{l\ sin\theta}{L} \\ \frac{sin\theta}{2} - \frac{l\ cos\theta}{L} & \frac{sin\theta}{2} + \frac{l\ cos\theta}{L} \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} \Delta d_l \\ \Delta d_r \end{bmatrix} \tag{3.11}$$

Por comparação com a notação apresentada no capítulo 2.2.3.4, é possível concluir que a notação da equação 3.11 não é mais do que uma representação em espaço de estados discreta matricial. Esta equação ao prever a postura do robô e sendo esta previsão baseada nas leituras dos *encoders* que definem Δd_l e Δd_r, apresenta as características da localização apresentada no capítulo 2.4.1. Desta forma, a equação 3.11 possibilita a localização *dead reckoning* ou, uma vez que se tratam de *encoders*, odometria.

3.4 Planeamento

Sabe-se, de acordo com a informação apresentada no capítulo 2.2.4, que a navegação de um robô pode ser reactiva ou baseada em mapa. No caso em estudo, a abordagem utilizada foi a segunda opção das referidas. É importante entender o tipo de comportamento que se pretende conseguir durante a operação do robô. Para o caso do presente trabalho o robô opera num espaço livre, o que dispensa a preocupação com eventuais obstáculos. Assim, o planeamento apenas necessita de definir a geometria a produzir. Eventuais perturbações como irregularidades no plano serão muito pequenas e espera-se que o controlador seja capaz de as compensar. Desta forma, tendo como base a informação apresentada em LaValle [2006], implementou-se um planeamento discreto. O algoritmo de planeamento desenvolvido define a trajectória do instrumento de escrita e permite apenas realizar linhas rectas. O algoritmo contudo não é ideal uma vez que assume que o movimento é feito sempre à velocidade limite, não contemplando as fases de aceleração e desaceleração.

Começou-se por definir o passo Δt e a velocidade limite (máxima) do robô v_{lim}. Seguidamente, definiu-se a coordenada (x, y) objectivo relativamente ao referencial de origem, através da introdução das variáveis $x_{destino}$ e $y_{destino}$. É importante mencionar que não será necessário definir o ponto de partida, justificando-se esta dispensa no final da apresentação do método utilizado. Contudo, utilize-se a variável c_i para a condição inicial.

Com as variáveis controláveis definidas, o método segue a seguinte estrutura:

1. Criação dos vectores onde estão contidas a coordenada anterior (sendo na primeira iteração a condição inicial) e a coordenada objectivo,

$$x_{plan} = \begin{bmatrix} c_{ix} & x_{destino} \end{bmatrix}^\top \tag{3.12a}$$

$$y_{plan} = \begin{bmatrix} c_{iy} & y_{destino} \end{bmatrix}^\top \tag{3.12b}$$

   Estes vectores apesar de não serem directamente utilizados no presente modelo teórico, irão simplificar bastante a implementação computacional.

2. Criação de um vector norma, obtido à custa do diferencial entre o ponto de partida e o ponto de chegada para cada coordenada,

$$\Delta x_{plan} = x_{destino} - c_{ix} \tag{3.13a}$$

$$\Delta y_{plan} = y_{destino} - c_{iy} \tag{3.13b}$$

$$s = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix}^\top \tag{3.13c}$$

$$s_{plan} = ||s|| \tag{3.13d}$$

3. Cálculo da duração máxima de operação dur, o que permitirá também obter o número de pontos a utilizar na discretização n assim como a discretização do tempo t_{plan},

$$dur = \frac{s_{plan}}{v_{lim}} \tag{3.14a}$$

$$n = \frac{dur}{\Delta t} \tag{3.14b}$$

$$t_{plan} = \begin{bmatrix} 0 & 1\Delta t & 2\Delta t & ... & n\Delta t \end{bmatrix}^\top \tag{3.14c}$$

4. Discretização da trajectória definida através da criação de incrementos dx e dy para as coordenadas,

$$dx = \frac{\Delta x_{plan}}{n} \tag{3.15a}$$

$$dy = \frac{\Delta y_{plan}}{n} \tag{3.15b}$$

$$traj_{xy} = \begin{bmatrix} c_{ix} & c_{iy} \\ c_{ix} + 1\,dx & c_{iy} 1\,dy \\ c_{ix} + 2\,dx & c_{iy} 2\,dy \\ \vdots & \vdots \\ c_{ix} + n\,dx & c_{iy} + n\,dy \end{bmatrix} \tag{3.15c}$$

Desta forma, cada linha da matriz $traj_{xy}$ corresponde a um par de coordenadas (x, y). O motivo de não se considerar a orientação θ do robô deve-se ao facto de este não ser holonómico, não se podendo impor os valores para (x, y, θ) para cada instante. A evolução da orientação do robô fica então ao critério da cinemática do mesmo após a imposição das velocidades nas rodas. Assim, a restrição imposta pela não holonomia deixa de ser obstáculo visto que o grau de liberdade restrito (neste caso θ) fica livre.

5. Considerando a equação 3.11, passando o vector de postura do instante $k-1$ para o lado esquerdo da equação, definindo o vector dp como a subtracção da postura no instante k pela postura no instante $k-1$ (incrementos de postura), definindo o vector dq como o vector dos

incrementos de velocidade nas rodas e por fim definindo a matriz onde se descreve as relações cinemáticas como a Jacobiana J, é possível representar a mesma equação através da seguinte notação,

$$\Delta p = J\,\Delta q \tag{3.16}$$

Tendo em conta a discretização criada no passo anterior do método, define-se a matriz auxiliar dp para toda a trajectória, que irá conter os incrementos de postura,

$$dp = \begin{bmatrix} traj_{xy}(2,1) - traj_{xy}(1,1) & traj_{xy}(2,2) - traj_{xy}(1,2) \\ traj_{xy}(3,1) - traj_{xy}(2,1) & traj_{xy}(3,2) - traj_{xy}(2,2) \\ \vdots & \\ traj_{xy}(n,1) - traj_{xy}(n-1,1) & traj_{xy}(n,2) - traj_{xy}(n-1,2) \end{bmatrix} = \begin{bmatrix} dx_1 & dy_1 \\ dx_2 & dy_2 \\ \vdots & \vdots \\ dx_m & dy_m \end{bmatrix} \tag{3.17}$$

Será importante referir que o vector dp, por uma questão de simplicidade, não especifica a orientação do robô pelos motivos mencionados no ponto anterior. Para a operação em causa isto não é problema visto que o resultado produzido pelo instrumento de escrita para uma determinada trajectória é sempre o mesmo independentemente da orientação.

É agora possível definir o vector Δp para cada incremento. No entanto, será definido Δp não como um vector mas sim como uma matriz onde estão contidos os incrementos de postura planeados,

$$\Delta p = \begin{bmatrix} dx_1 & dy_1 \\ dx_2 & dy_2 \\ \vdots & \vdots \\ dx_{n-1} & dy_{n-1} \end{bmatrix}^\top \tag{3.18}$$

Sabendo que Δq é definido por,

$$\Delta q = \begin{bmatrix} \Delta d_{l\,1} & \Delta d_{r\,1} \\ \Delta d_{l\,2} & \Delta d_{r\,2} \\ \vdots & \vdots \\ \Delta d_{l\,n-1} & \Delta d_{r\,n-1} \end{bmatrix}^\top \tag{3.19}$$

é então possível determinar para cada passo, definido por t_{plan} (equação 3.14c), a relação entre a postura e a velocidade tal como já foi apresentada na equação 3.16. Considere-se uma variável auxiliar $i = 1, 2, ..., n$ para a contagem de incrementos $i\Delta t$,

$$\begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix} = \begin{bmatrix} \frac{cos\Delta\theta_i}{2} + \frac{l\ sin\Delta\theta_i}{L} & \frac{cos\Delta\theta_i}{2} - \frac{l\ sin\Delta\theta_i}{L} \\ \frac{sin\Delta\theta_i}{2} - \frac{l\ cos\Delta\theta_i}{L} & \frac{sin\Delta\theta_i}{2} + \frac{l\ cos\Delta\theta_i}{L} \end{bmatrix} \begin{bmatrix} \Delta d_{l\,i} \\ \Delta d_{r\,i} \end{bmatrix} \tag{3.20}$$

Será agora importante referir que as variáveis desconhecidas são os incrementos de velocidades definidos pelo vector Δq.

Uma vez que a postura θ não é contemplada, a matriz Jacobiana J é invertível, dispensando o cálculo da matriz pseudoinversa Jacobiana como apresentado em Tzafestas [2014],

$$\Delta q = J^{\dagger}\, \Delta p \tag{3.21}$$

sendo apenas necessário implementar uma função de resolução para sistemas de equações lineares do tipo $A * x = B$ existente no ambiente *MATLAB*, formulando-se

$$\Delta q = \Delta p \setminus J \tag{3.22}$$

onde J representa a Jacobiana simplificada tal como apresentada em 3.20 e $\setminus$ representa a função de resolução denominada *mldivide*.

6. Com os incrementos definidos para as velocidades das rodas, bastará agora efectuar um somatório de cada uma das variáveis de incremento para que se possa obter progressão no tempo, ou seja, formulando para a equação 3.22,

$$\Delta q_{_{i+1}} = \sum_{1}^{i} (\Delta q_{_i}) + (\Delta p_{_i} \setminus J) \tag{3.23}$$

Concluindo este passo, obtém-se os incrementos a enviar para as rodas a cada passo $i\Delta t$ e encerrando assim o algoritmo de planeamento onde se definiu todos os pontos da discretização quer em termos de posição da ferramenta, quer em termos de velocidades nas rodas.

3.5 Controlo

3.5.1 Comunicação com a placa controladora

Para operar a placa controladora dos motores *MD25* é necessário enviar e receber uma determinada quantidade de *bytes*. Um dos comandos mais relevantes é a aquisição da informação dos *encoders*. De acordo com a documentação da placa, quando um comando de leitura de *encoders* é enviado, serão enviados de volta 4 *bytes* que devem ser acoplados para formarem um número com sinal de 32 *bit*,

$$enc = b_{_4} + b_{_3} \times 256 + b_{_2} \times 256^2 + b_{_1} \times 256^3 \tag{3.24}$$

onde a variável b representa o *byte* juntamente com a sua indexação (visto serem recebidos 4). A título de curiosidade, esta equação resulta num valor decimal de 1070642 ou, no sistema hexadecimal, num valor de 0x105632.

Com a conversão apresentada na equação 3.24, o valor obtido para a variável enc, atribuída para a leitura do *encoder*, representará os picos de *encoder* lidos, tendo este 360 leituras por revolução, é possível estipular que cada leitura corresponde a um grau. Multiplicando as leituras pela relação de transmissão é possível estabelecer a relação entre a revolução do motor e a revolução do veio de potência, medido em graus. Desta forma, define-se que enc apresenta como unidade o grau.

Todos os restantes comandos a enviar à placa, como por exemplo a atribuição de velocidade ou a leitura da voltagem apenas necessitam de uma conversão do sistema hexadecimal para o sistema decimal e vice-versa. A documentação da placa *MD25* apresenta a correspondência dos valores no sistema hexadecimal a cada comando, assim como a quantidade de bytes enviados e recebidos para cada um. Será ainda importante referir que a placa dispõe de um controlador próprio.

3.5.2 Lei de controlo

A lei de controlo aplicada para o sistema em estudo passa por controlar a velocidade de cada roda de forma independente. O anel de controlo para cada roda é descrito através do diagrama de blocos, conceito introduzido no capítulo 2.2.3.2, apresentado na figura 3.3. Analisando o diagrama de blocos, ou a lei de controlo, é notório que o sinal de entrada é a velocidade planeada e o sinal de saída são os picos de *encoder*. Esta leitura de picos de *encoder*, após sofrer uma filtragem para anular o efeito de *overflow*, que será descrito mais adiante, e uma conversão para velocidade linear, será comparada com a velocidade planeada, gerando um valor de erro. Este erro será a variável necessária para que se possa concretizar a operação de controlo através do controlador PID, conceito introduzido no capítulo 2.2.3.3, resultando, após este processo, numa variável de actuação que irá ditar qual a velocidade a impôr à roda. Contudo esta velocidade terá que ser limitada para casos em que haja instabilidades no controlo, evitando que sejam enviados sinais que variam entre grandes valores de velocidade e valores muito pequenos num curto espaço de tempo, algo que poderia danificar o equipamento. Imposto este limite, a informação da velocidade a ser imposta nas rodas é enviada para a placa e esta irá enviar de volta a leitura dos picos de *encoder*. O ciclo repete-se até que a operação definida no planeamento esteja concluída.

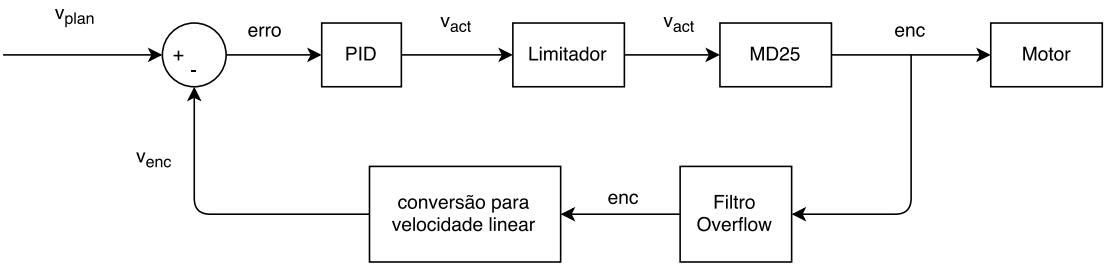


Figura 3.3: Diagrama de blocos da lei de controlo considerada para cada roda.

Segundo Graf [1999], *overflow* é definido como a condição que ocorre sempre que o resultado de uma operação aritmética excede a capacidade de representação do número no domínio digital. Para o caso da leitura dos *encoders* pela placa *MD25*, ao chegar ao limite máximo da representação estes saltam a sua leitura para o limite mínimo e vice-versa.

A introdução do filtro de Kalman estendido no sistema leva a uma expansão do anel de controlo tal como apresentado na figura 3.4. Por uma questão de simplicidade de representação, considere-se que os blocos de actuação de cada roda contêm o anel de controlo representado na figura 3.3. Contudo o ponto de adição do novo anel de controlo, para além de calcular o erro entre a postura planeada e a estimada, é também uma tomada de decisão. Caso o erro seja superior a um determinado valor estipulado, será definido um novo planeamento corrigido a utilizar para o resto da operação, caso contrário o planeamento inicialmente definido continuará a servir de referência. Outro aspecto importante de mencionar é o facto de o filtro de Kalman não ser implementado em cada ciclo. Será definido no início de cada operação uma determinada quantidade de vezes que o filtro será aplicado ao longo da trajectória definida. Nos pontos de decisão de aplicação do filtro de Kalman estendido será considerado o *feedback* representado no novo anel de controlo, sendo que para os restantes pontos o sistema irá funcionar em anel aberto. Assim, não se pode considerar o diagrama apresentado na figura 3.4 como realista, servindo este apenas de ilustração auxiliar na descrição do funcionamento do sistema.

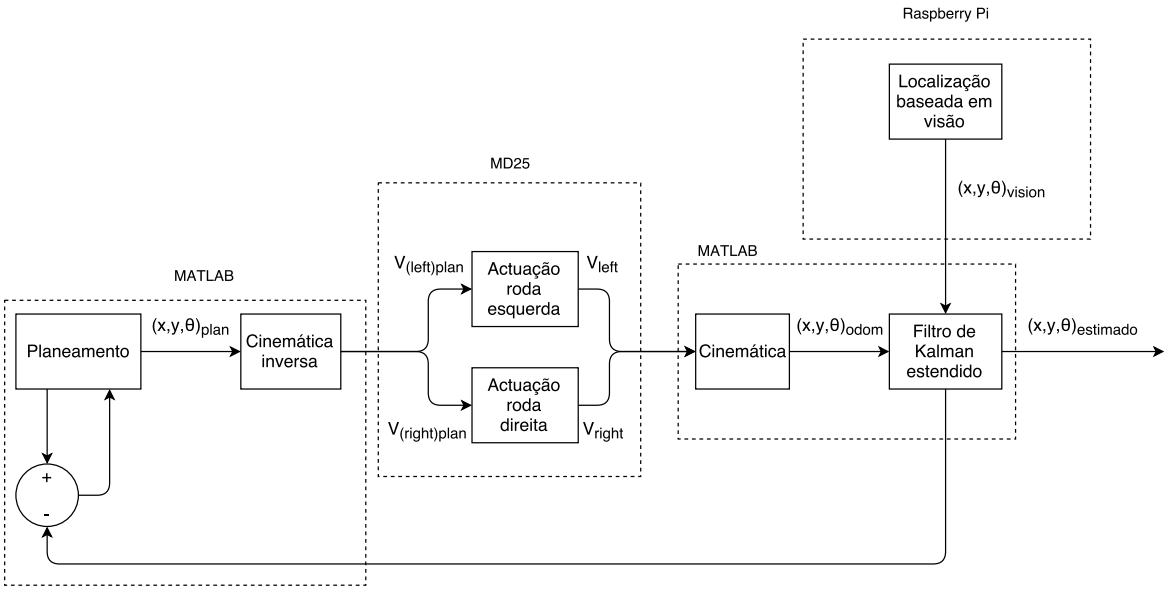


Figura 3.4: Diagrama de blocos da lei de controlo considerada para o presente trabalho aquando da utilização do filtro de Kalman estendido.

Considera-se importante definir a forma que um controlador PID assume no domínio digital. Assim, de acordo com Tzafestas [2014], as componentes proporcional, integrativa e derivativa da equação

2.21 podem ser reescritas de acordo com as notações seguintes

$$\dot{e} = \frac{\mathrm{d}\, e(t)}{\mathrm{d}t} \approx \frac{e_{new} - e_{old}}{\Delta t} \tag{3.25a}$$

$$\int_0^t e(\tau)\, \mathrm{d}\tau \approx \sum_{i=0}^{n} e(i\Delta t)\Delta t = \Delta t\, E \tag{3.25b}$$

$$\Delta t\, E_{new} = \Delta t \sum_{i=1}^{n+1} e(i\Delta t) = \Delta t\, e((n+1)\Delta t) + \Delta t\, E_{old} \tag{3.25c}$$

Desta forma, veja-se a implementação em termos computacionais do controlador PID ao longo do tempo $k\Delta t$. Para a componente proporcional apenas bastará multiplicar o desvio pelo ganho proporcional,

$$u_k = K_P e_k \tag{3.26}$$

Apesar de o controlador utilizado para o presente trabalho não recorrer à componente derivativa, para se incluir a mesma, tendo em conta a equação 3.25a, seria:

$$u_k = K_P e_k + K_D\, \frac{e_k - e_{k-1}}{\Delta t_{iter}} \tag{3.27a}$$

sendo Δt_{iter} o tempo da iteração k.

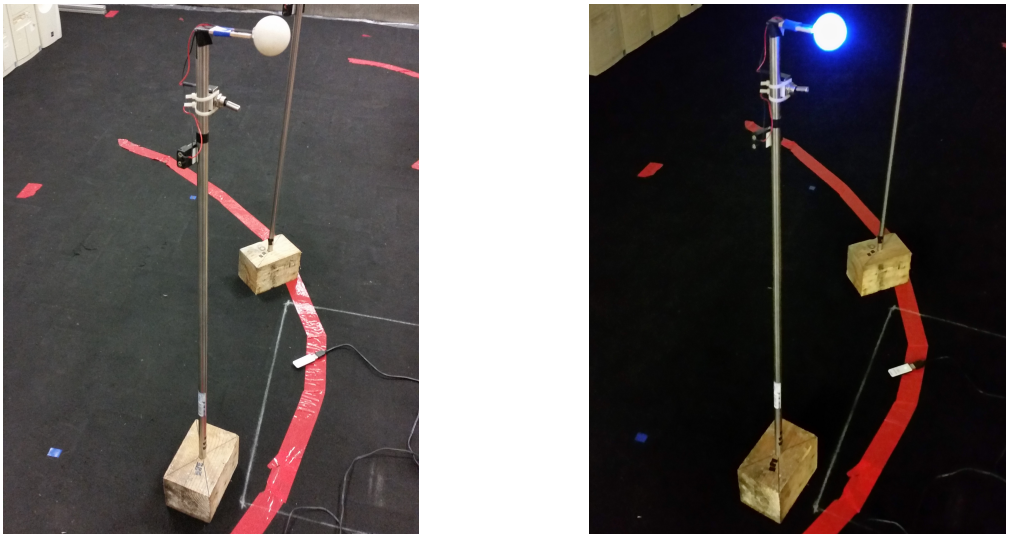
Por fim, tendo em conta as equações 3.25b e 3.25c, a componente integral é introduzida

$$E_k = E_{k-1} + e\Delta t_{iter} \tag{3.28a}$$

$$u_k = K_P e_k + K_D\, \dot{e}_k + K_I\, E_k \tag{3.28b}$$

3.6 Identificação de beacons

A identificação de *beacons* é feita através de imagens obtidas a partir da câmara com as propriedades já apresentadas no capítulo 3.1. A resolução adoptada para as imagens obtidas foi de 800×600 píxeis. Não se optou por uma resolução superior uma vez que quanto maior a informação a ser processada, mais lento se torna o cálculo computacional. Os *beacons* foram construídos com base numa bola de ténis de mesa com uma lâmpada *LED* de alto brilho de $10\ cm$ de diâmetro embutida, tal como é apresentado na figura 3.5.



(a) *LED* desligado. (b) *LED* ligado.

Figura 3.5: *Beacon*.

O código de cores utilizado nos algoritmos computacionais foi o *RGB*. Para uma melhor clarificação acerca de conceitos relacionados com códigos de cores aconselha-se as referências Hunt [2004], onde se descreve o principio da adição e o código *RGB*, Levkowitz [1997], onde se descreve o código *RGB* de uma forma mais simples e Poynton [2003] como uma alternativa às duas anteriores.

Com base nas referências mencionadas para a teoria das cores, os valores de intensidade para cada cor primária do código *RGB* variam entre 0 e 255. Observando a figura 3.6, será fácil de entender que da combinação $R = 0$, $G = 0$ e $B = 0$ resulta a cor preta, representada na imagem por *Bk*, enquanto que da combinação $R = 255$, $G = 255$ e $B = 255$ resulta a cor branca, representada na imagem por *W*.

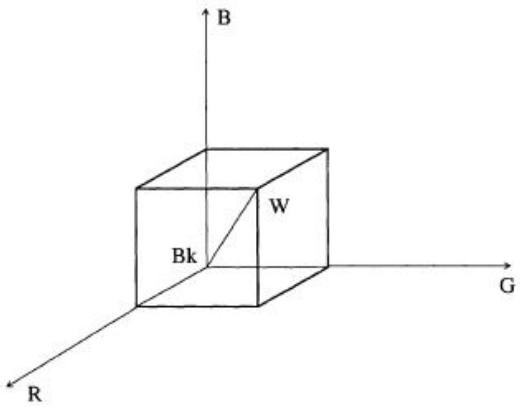


Figura 3.6: Representação do espaço de cor *RGB*, adaptada de Levkowitz [1997].

Computacionalmente, em ambiente *MATLAB* e de acordo com a documentação do *software*, o código

de cores *RGB* é tratado como uma matriz de dimensão $m \times n \times 3$ onde m e n representam a dimensão da imagem em píxel e a profundidade 3 representa as cores *R*, *G* e *B*. Resumindo, uma imagem apresentada em *MATLAB* através do código *RGB*, terá uma matriz apenas dedicada aos vermelhos, uma sobreposta dedicada aos azuis e ainda uma terceira sobreposta pelas duas primeiras dedicada aos verdes que adicionadas elemento a elemento (relembrando o principio da adição) apresentam uma determinada cor. Cada elemento de cada matriz, varia o seu valor desde 0 a 255.

Antes de se implementar o algoritmo de identificação de *beacons* foi necessário realizar uma operação experimental de determinação do código *RGB* de cada *beacon* através de imagens captadas a partir da câmara e posteriormente introduzidas no *MATLAB*. É importante ter em conta que a luminosidade presente no espaço de trabalho pode variar ao longo do dia devido à iluminação natural. Por essa razão, é importante operar num espaço que mantenha os seus níveis de luminosidade relativamente constantes como por exemplo uma sala com as janelas cobertas e sempre com o mesmo nível de iluminação artificial.

Definido o código de cada *beacon*, procedeu-se à implementação do algoritmo de identificação aplicado a cada nova imagem captada. Primeiramente foi calculada a raiz do desvio médio quadrático (*Root-mean-square deviation* - *RMSD*) entre a referência *RGB* do *beacon* e a cor de cada píxel da nova imagem captada. Segundo Hyndman and Koehler [2006], o *RMSD* pode ser calculado através da equação,

$$RMSD = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y_t} - y_t)^2}{n}} \tag{3.29}$$

onde o t varia de 1 a 3 e $y1$, $y2$ e $y3$ correspondem aos valores dos canais *R*, *G* e *B* respectivamente.

Seguidamente foi feita uma binarização da imagem, tendo como condição o *RMSD* ser inferior a uma determinada tolerância que se determine apropriada. Esta tolerância irá contemplar as variações de luminosidade do espaço e deverá ser testada experimentalmente, pois depende do espaço onde o *beacon* está inserido. Se o ambiente observado for muito rico de cores, poderá significar que a cor do *beacon* a identificar se encontra em muitos pontos da imagem que não correspondem ao *beacon*. É então necessário fazer um balanço sobre o valor de tolerância a utilizar para que não se polua a identificação do *beacon* mas também para que não se despreze pontos pertencentes ao mesmo. Da binarização irá resultar uma matriz binária que irá detectar todos os píxeis da nova imagem que satisfazem a condição definida para o *RMSD*. Com estes índices, calcula-se o valor médio de todos estes pontos que resultará, aproximadamente, no índice da matriz da imagem onde se encontra o centro de massa do *beacon*.

3.7 Calibração

Foi utilizada uma *toolbox* desenvolvida por Davide Scaramuzza e apresentada em Scaramuzza et al. [2006b] e Scaramuzza et al. [2008] para se efectuar a calibração da câmara sob o efeito da lente de olho de peixe. Uma explicação detalhada acerca do funcionamento da *toolbox* é apresentada no *website* Scaramuzza assim como uma explicação de como se proceder à calibração. O menu apresentado pela *toolbox* está representado na figura 3.7.

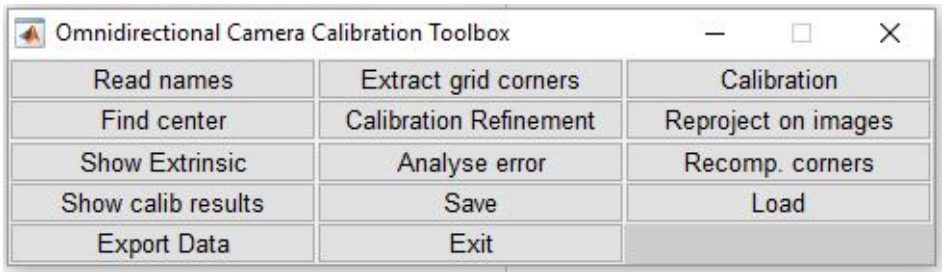


Figura 3.7: Menu da *toolbox* de calibração.

Para o processo de calibração, foi necessário imprimir um padrão de xadrez numa superfície rígida para que o padrão mantivesse as mesmas características ao serem captadas diferentes imagens. Assim, este padrão disponibilizado em Scaramuzza foi impresso e acoplado a uma placa de madeira branca como é apresentado na figura 3.8 onde está descrita uma imagem captada através da câmara.

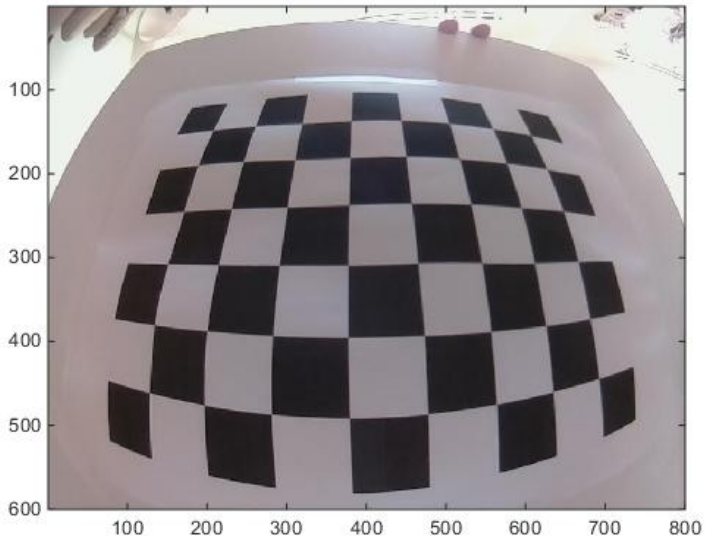


Figura 3.8: Placa com padrão xadrez utilizado na calibração.

Com este padrão, foram captadas 9 imagens onde se apresenta o padrão em diferentes ângulos e onde se procura cobrir toda a área visível da imagem, ou seja, preencher toda a abertura da lente.

Seguidamente foi solicitado à *toolbox* que carregasse as imagens captadas e que identificasse os cantos de cada quadrado do padrão de xadrez automaticamente. Para se atribuir um sistema de coordenadas ao padrão, a *toolbox* pede ao utilizador a quantidade de quadrados que se encontram na direcção do novo eixo x e no eixo y. Esta distinção é possível visto que o padrão de xadrez apresenta uma quantidade de quadrados a cheio ao longo de uma dimensão diferente da quantidade ao longo da outra dimensão. Para além disso é ainda necessário definir a dimensão da aresta de cada quadrado do padrão.

Procedeu-se então à operação automática de calibração como já mencionada no capítulo 2.3.2 e onde se utilizou uma expansão polinomial de Taylor de grau 4 por sugestão do autor da *toolbox*. Com os parâmetros intrínsecos da câmara obtidos através da calibração foi então possível realizar a operação de determinação do centro da imagem. É importante que a calibração seja realizada antes desta operação pois a lente utilizada induz um erro que desvia o centro real da imagem captada, diferindo este do ponto que se encontra a metade da altura e metade da largura da imagem.

A partir deste ponto considerou-se a calibração concluída. Contudo foi ainda possível refinar os resultados obtidos através de um novo cálculo dos cantos do padrão de xadrez tendo como base a informação obtida no primeiro processo de calibração ou através de uma refinação dos parâmetros de calibração que utiliza o algoritmo de Levenberg-Marquadt, um algoritmo de optimização para resolver problemas de mínimos quadrados não lineares (O. Tingleff [2004]).

3.8 Transformação de imagem

A *toolbox* utilizada para calibração dispõe de duas funções, uma denominada *world2cam* e outra *cam2world*.

Considerando que *M* representa uma matriz de dimensão $3 \times N$ que contém as coordenadas dos pontos 3D $M = [X; Y; Z]$ e que *m* representa uma matriz de dimensão $2 \times N$ que contém as coordenadas dos píxeis da imagem captada, estas relacionam-se através das funções mencionadas da *toolbox*. Assim, aplicando a função *world2cam* é possível, tendo em conta os parâmetros intrínsecos da câmara, transformar as coordenadas *M* em coordenadas *m* enquanto que a função *cam2world* permite transformar as coordenadas *m* em coordenadas *M* sendo que para esta ultima função, as coordenadas *M* estarão representadas numa esfera unitária.

Será importante mencionar que as coordenadas $M = [X; Y; Z]$ ao estarem representadas na esfera unitária, satisfazem a condição $x^2 + Y^2 + Z^2 = 1$.

Além destas duas funções a *toolbox* dispõe de outras que não se justificam mencionar, contudo, o

algoritmo desenvolvido para a transformação de imagem foi baseado nessas funções.

Cada imagem captada pela câmara virá sob o efeito de transformação da lente de olho de peixe. Esta deformação será problemática uma vez que os pontos representados na imagem não se encontram todos no mesmo plano, algo que complica bastante a determinação da profundidade destes pontos. Desta forma, a transformação aplicada às imagens captadas transpõe os pontos sob o efeito da lente de olho de peixe para um plano comum.

O algoritmo desenvolvido começa por definir a dimensão final desejada para a imagem, que no caso do presente trabalho foi 800×600. Para a determinação da profundidade foi utilizada uma razão entre o comprimento da moldura e um factor de escala representado por *fc*. O valor a atribuir a *fc* é obtido experimentalmente até que se obtenham bons resultados de imagem. De certa forma o *fc* funciona como uma ampliação ou redução em termos de profundidade.

Seguidamente, criou-se uma malha, ou seja, uma discretização no espaço, com as dimensões finais pretendidas (800×600) e onde o incremento é unitário. Desta forma, o que se está a criar não é mais do que uma indexação *(i,j)* onde $i = 1, 2, ..., 600$ e $j = 1, 2, ..., 800$. A esta matriz de indexação, será subtraído metade do valor de cada dimensão para se ter um recentramento da matriz, obtendo-se assim $i = -299, ..., 0, ..., 300$ e $-399, ..., 0, ..., 400$ onde a coordenada $(i, j) = (0, 0)$ representa o centro da malha. Para além desta indexação, foi criada uma matriz unitária de dimensão 800×600 pela qual se multiplicou o valor obtido na razão entre o comprimento da moldura e o factor de escala *fc* criando-se desta forma a indexação da coordenada z, que é um valor constante. A figura 3.9 demonstra um exemplo de uma malha discretizada

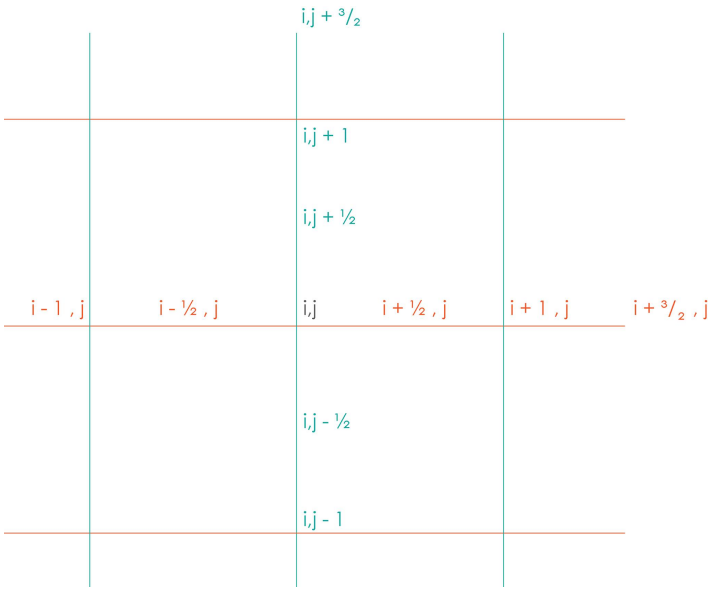


Figura 3.9: Exemplo de uma malha discretizada.

Com a indexação definida, procedeu-se a uma concatenação das três dimensões, resultando numa matriz de dimensão 3×480000 onde estão incluídas as coordenadas de cada píxel de uma imagem de dimensão 800×600 esta, será a matriz *M*.

Será importante relembrar que a matriz *M* neste ponto será, hipoteticamente, uma imagem simulada de uma captura da câmara.

O passo seguinte passou por aplicar a função *world2cam* à matriz dos pontos do espaço *M* resultando na matriz dos pontos na imagem *m* representados sob a forma (x, y).

Com a matriz dos pontos na imagem *m* criada, foi necessário excluir os pontos que apresentavam valor negativo, os pontos que excediam as dimensões da imagem, ou seja quando $(x > 800)$ e $y > 600$, e os resultados que não fossem números.

Ao concluir estes passos, o que se obteve foi uma matriz *m* que não é mais do que uma matriz guia ou, se se preferir, uma moldura onde se irá introduzir a informação das imagens capturadas pela câmara.

Ao ser captada uma nova imagem, de dimensões $800 \times 600 \times 3$ descrita em *RGB*, cada píxel desta nova imagem foi realocado para o índice definido pela matriz *m*. Após a realocação, a imagem resultante é a imagem deformada, onde cada píxel está representado num plano, representada num único plano comum, ou seja, todos os píxeis estão à mesma profundidade em termos espaciais (dimensão z do espaço).

3.9 Localização baseada em visão

A localização baseada em visão foi abordada como um problema *2D* e não como *3D*. O motivo desta consideração deve-se à posição dos *beacons* ser conhecida e a altura ser constante, ou seja, estão todos situados no mesmo plano horizontal. Para além disso, a câmara é montada vidrada para cima, estando sempre num plano abaixo do plano dos *beacons*. Os referidos planos por serem paralelos, as dimensões das distâncias dos *beacons* não são deformadas pela perspectiva quando projectadas numa imagem, considerando que os *beacons* se encontram sempre no campo de visão da câmara. Isto permite fazer medições sobre as distâncias e ângulos entre os *beacons* no plano da câmara desde que a correspondência píxel-metro seja conhecida. Esta relação assumirá um valor diferente caso a cota entre os planos seja alterada, o que não acontece no presente trabalho. A formulação para a representação de posturas em *3D* é apresentada em Corke [2011] e permite melhor entender a simplificação considerada. A incorporação da câmara no protótipo é apresentada na figura 3.10. Os *beacons* foram construídos com base numa bola de ténis de mesa com uma lâmpada *LED* de alto

brilho de 10 cm de diâmetro embutida, tal como é apresentado na figura 3.5.

Figura 3.10: Câmara com lente de olho de peixe.

Para o algoritmo de localização baseada em visão foi utilizada a abordagem proposta em Calabrese and Indiveri [2005]. Tendo em conta a simplificação descrita, para se representar a posição dos *beacons* no plano da câmara bastará definir um factor de escala responsável pela conversão de píxel da imagem para metro no espaço. Para se encontrar o dito factor foi utilizado um objecto de dimensão conhecida situado no plano dos *beacons*, posteriormente foi medida a quantidade de píxeis correspondente à representação do objecto e definiu-se a relação

$$factor = \frac{dimensão\ real}{pixeis\ correspondentes}$$

que permite estabelecer a conversão da medida em píxel para distância. Após esta abordagem experimental, o factor de conversão permanece o mesmo para qualquer caso visto que o plano dos *beacons* e o plano da câmara se encontram sempre à mesma altura. A descrição do problema é apresentada na figura 3.11 onde $< 0 >$ é o sistema de coordenadas de referência (espaço), e $< 1 >$ o sistema de coordenadas do robô. Como foi visto no capítulo 2.1, a matriz de rotação que relaciona dois sistemas de coordenadas pode ser obtida através de um ponto avaliado relativamente a cada um deles. O problema é que a orientação do robô é uma incógnita, tornando necessário um segundo ponto de referência para que se possa calcular a orientação. Desta forma, $L_1 \in R^{2\times1}$ e $L_2 \in R^{2\times1}$ são as distâncias dos *beacons* relativamente ao sistema de coordenadas de referência cujo o valor é conhecido, $Z_1 \in R^{2\times1}$ e $Z_1 \in R^{2\times1}$ são as mesmas distâncias mas relativamente ao sistema de coordenadas do robô. A resolução do problema de localização para este caso passou por encontrar o valor de P (translação) e de φ (rotação). Sendo que o índice superior identifica o sistema de coordenadas a que a variável é relativa, a matriz de rotação entre o sistema de coordenadas $< 0 >$ e $< 1 >$ é dada por

$$^0R_1 = \begin{bmatrix} cos(\varphi) & -sin(\varphi) \\ sin(\varphi) & cos(\phi) \end{bmatrix} \tag{3.30}$$

O sistema de equações que definem o problema da localização é dado então por

$$^{0}L_{2} =^{0} P +^{0} R_{1}\ ^{1}Z_{2} \tag{3.31a}$$

$$^{0}L_{1} =^{0} P +^{0} R_{1}\ ^{1}Z_{1} \tag{3.31b}$$

$$^{0}(L_{2} - L_{1}) =^{0} R_{1}\ ^{1}(Z_{2} - Z_{1}). \tag{3.31c}$$

Uma vez que a única incógnita na equação 3.31c é φ, resolve-se a mesma em ordem a $cos(\varphi)$ e $sin(\varphi)$

$$cos(\varphi) = \frac{\Delta L_x\, \Delta Z_x + \Delta L_y\, \Delta Z_y}{\Delta L_x^2 + \Delta L_y^2} \tag{3.32a}$$

$$sin(\varphi) = \frac{\Delta L_y\, \Delta Z_x + \Delta L_x\, \Delta Z_y}{\Delta L_x^2 + \Delta L_y^2} \tag{3.32b}$$

onde as distâncias $\Delta L_x \equiv^{0} (L_2 - L_1)_x$, $\Delta L_y \equiv^{0} (L_2 - L_1)_y$, $\Delta Z_x \equiv^{1} (Z_2 - Z_1)_x$ e $\Delta Z_y \equiv^{1} (Z_2 - Z_1)_y$ são conhecidas. É importante referir que enquanto L_1 e L_2 identificarem dois *beacons* distintos, então as equações 3.32a e 3.32b estão sempre definidas. A orientação do robô $\varphi \in [-\pi, \pi]$ pode ser obtida pela relação trigonométrica $\varphi = atan2(cos(\varphi), cos(\varphi))$. Com esta definida, a posição do robô ^{0}P poderá ser obtida à custa da equação 3.31a ou 3.31b. Uma vez que se apresentam três *beacons* disponíveis, esta abordagem foi aplicada em conjuntos distintos de dois *beacons* calculando-se a postura final do robô através da média das posturas obtidas para cada par.

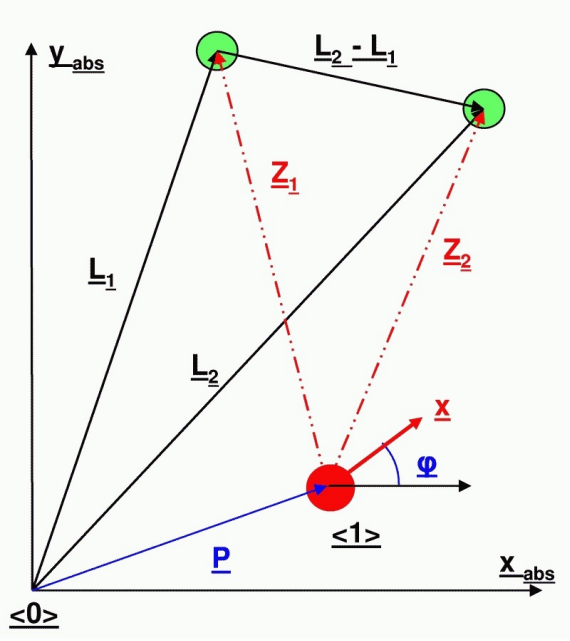


Figura 3.11: Representação do problema de localização 2D, adaptada de Calabrese and Indiveri [2005].

3.10 Filtro de Kalman estendido

Como descrito no capítulo 2.4.2.3 o filtro de Kalman estendido, ou *EKF*, é aplicado no caso de sistemas não lineares, que é o caso do modelo utilizado no presente trabalho.

Para a aplicação do *EKF*, começou-se por definir o sistema sob a forma,

$$x_k = f(x_{k-1}, u_{k-1}, n_{k-1}) \tag{3.33a}$$

$$y_k = h(x_k) + v_k \tag{3.33b}$$

onde *x* representa o vector de estado, *u* o vector de controlo, *n* o vector de perturbação, *y* o vector de medição e *v* o ruído da medição. A função $f(x, u, n)$ à qual se irá denominar função de transição, foi associada à função utilizada para a odometria, que depende da posição prévia do robô e da actuação do controlador *PID* para calcular a posição no instante seguinte (3.11). Já a função de observação $h(x)$ calcula a previsão da localização visual a partir do vector de estados. A variável *y* representa a localização obtida a partir da identificação dos *beacons*, ou seja, das leituras da câmara.

Com as funções definidas, definiu-se as matrizes de covariância *Q* associada ao controlo, *R* associada à medição e *P* associada ao estado do robô. Estas matrizes dependem de estimações experimentais onde se tenta determinar a dispersão dos dados em torno do valor médio de cada variável. Com estas variáveis definidas definiu-se o ciclo temporal, onde a cada ciclo se aplicam os dois passos do filtro, a previsão e a correcção.

Para a previsão, definiu-se as Jacobianas F_x relativa ao estado e F_n relativa à perturbação. As Jacobianas, tal como apresentado nas equações 2.44, são obtidas através da derivada da função de transição em ordem a cada variável de cada parâmetro (estado, controlo e perturbação). Com as matrizes Jacobianas definidas, foi aplicado o passo da previsão de acordo com as equações 2.45 e onde se considerou que não haviam perturbações,

$$x_k = f(x_{k-1}, u_{k-1}, 0) \tag{3.34a}$$

$$P_k = F_{x\,k-1} \times P_{k-1} \times F_{x\,k-1}^{\top} + F_{n\,k-1} \times Q_{k-1} \times F_{n\,k-1}^{\top}. \tag{3.34b}$$

Seguidamente foi calculada a matriz Jacobiana H_x relativa ao estado obtida através da equação 2.46, permitindo obter a matriz de covariância da função de observação,

$$E_k = H_k \times P_k \times H_k^{\top}. \tag{3.35}$$

Concluído este cálculo, foi então possível aplicar o passo da correcção de acordo com as equações 2.47, começando por se obter a inovação z, a covariância da inovação Z e o ganho de Kalman K,

$$z_k = y_k - h(x_k) \tag{3.36a}$$

$$Z_k = R_k + E_k \tag{3.36b}$$

$$K_k = P_k \times H_k^\top \times Z_k^{-1} \tag{3.36c}$$

aplicando por fim a correcção,

$$x_k^{corrected} = x_k + K_k \times z_k \tag{3.37a}$$

$$P_k^{corrected} = P_k - K_k \times Z_k \times K_k^\top. \tag{3.37b}$$

Para o caso do algoritmo de localização baseada em visão utilizado, o sensor irá fornecer o valor da postura (x, y, θ) do robô. Desta forma, são produzidas precisamente as variáveis de estado, o que fará com que $h(x) = x$, definindo a matriz Jacobiana H_x como uma matriz identidade. As matrizes Jacobianas obtidas através das derivadas parciais, como apresentado nas equações 2.44 e 2.46, apresentam-se como:

$$F_x = \begin{bmatrix} 1 & 0 & v_{left}\, t(-\frac{1}{2}sin(\theta) + \frac{l\, cos(\theta)}{L}) + v_{right}\, t(-\frac{1}{2}sin(\theta) - \frac{l\, cos(\theta)}{L}) \\ 0 & 1 & v_{left}\, t(\frac{1}{2}cos(\theta) + \frac{l\, sin(\theta)}{L}) + v_{right}\, t(\frac{1}{2}cos(\theta) - \frac{l\, sin(\theta)}{L}) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.38a}$$

$$F_u = \begin{bmatrix} t(\frac{1}{2}cos(\theta) + \frac{l\, sin(\theta)}{L}) & t(\frac{1}{2}cos(\theta) - \frac{l\, sin(\theta)}{L}) \\ t(\frac{1}{2}sin(\theta) - \frac{l\, cos(\theta)}{L}) & t(\frac{1}{2}sin(\theta) + \frac{l\, cos(\theta)}{L}) \\ -\frac{t}{L} & \frac{t}{L} \end{bmatrix} \tag{3.38b}$$

$$F_n = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{3.38c}$$

$$H_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.38d}$$

Capítulo 4

Análise de resultados

Os resultados descritos no presente capítulo foram obtidos através de diferentes experiências. Cada uma foi parametrizada para que fosse visível o comportamento do método sujeito a avaliação e para que um resultado considerado satisfatório fosse obtido. Para uma avaliação global, foi desenhada uma experiência onde se pretende imprimir um quadrado com $10\ cm$ de lado para que fosse possível avaliar a integração dos métodos no modelo.

4.1 Planeamento

Tendo em conta o método de planeamento descrito em 3.4 foi ajustado experimentalmente um Δt de 0.1 segundos e uma velocidade limite de $0.002\ m/s$ (3.14). O primeiro destes parâmetros foi escolhido por forma a que as aproximações usadas no planeamento tenham um efeito negligenciável. Este efeito será discutido mais à frente nesta secção. A velocidade limite foi definida suficientemente baixa para que os efeitos da dinâmica do robô tenham um impacto mínimo no seguimento da trajectória.

A postura inicial do robô encontra-se descrita na figura 4.1 onde a seta vermelha representa a posição do instrumento de escrita no sistema de coordenadas de referência do espaço. Para as experiências realizadas, o instrumento de escrita utilizado foi um lápis, a posição inicial no espaço corresponde à coordenada $(x, y) = (0, 0)$ e o robô encontra-se com uma orientação $\theta = 0$ relativamente ao eixo x do referencial.

Para apresentar os resultados do algoritmo de planeamento optou-se por descrever uma linha vertical, paralela ao eixo das ordenadas do referencial de referência, e com $10\ cm$ de comprimento. O motivo para tal, é para que seja evidenciada a diferença na actuação das rodas, visto que o robô terá que rodar sobre si próprio, o que não aconteceria caso a trajectória fosse uma linha horizontal.

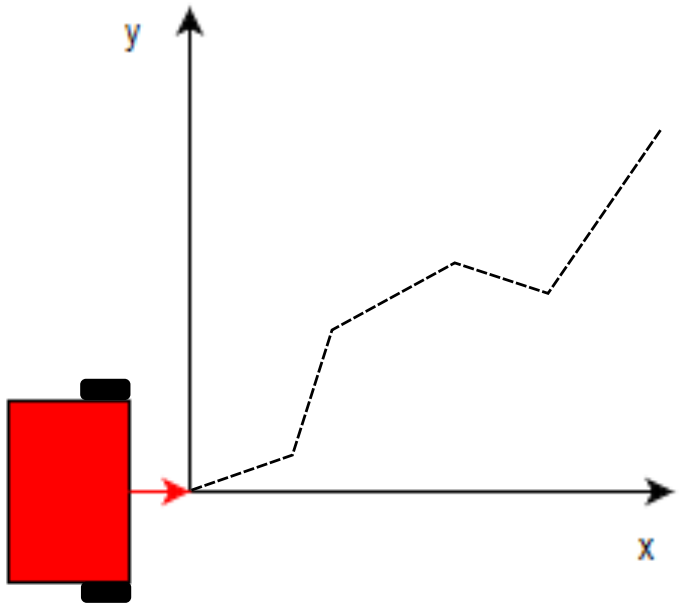


Figura 4.1: Postura inicial do robô.

Através da figura 4.2 é possível observar parte do resultado da discretização da trajectória que terá inicio na coordenada $(0, 0)$ terminando em $(0, 0.1)$. As trajectórias dos ângulos das rodas, correspondentes à discretização de postura definida, são apresentadas na figura 4.3. É óbvio, tendo em conta a condição inicial do robô, que este terá que rodar para se deslocar na direcção do eixo das ordenadas do referencial de referência, portanto, um deslocamento negativo para a roda esquerda e outro positivo para a roda direita, indicam que a rotação será realizada, como esperado.

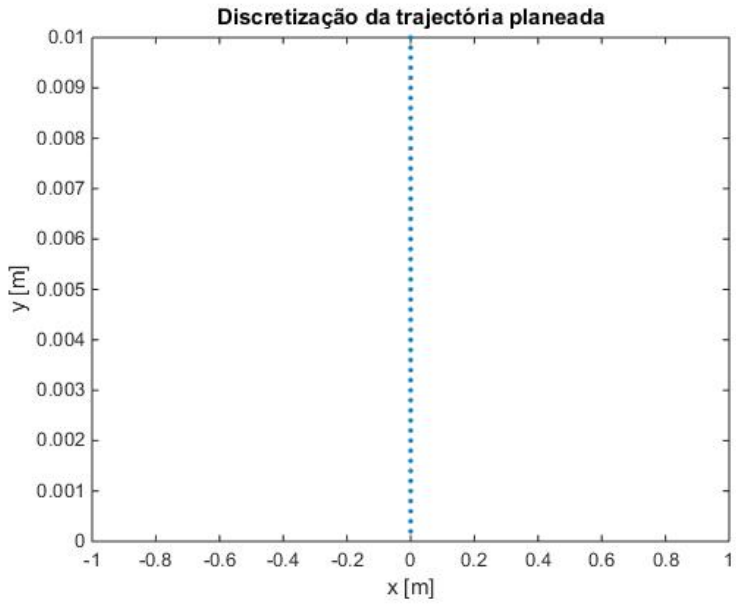


Figura 4.2: Resultado da discretização para uma trajectória vertical de 10 cm de comprimento.

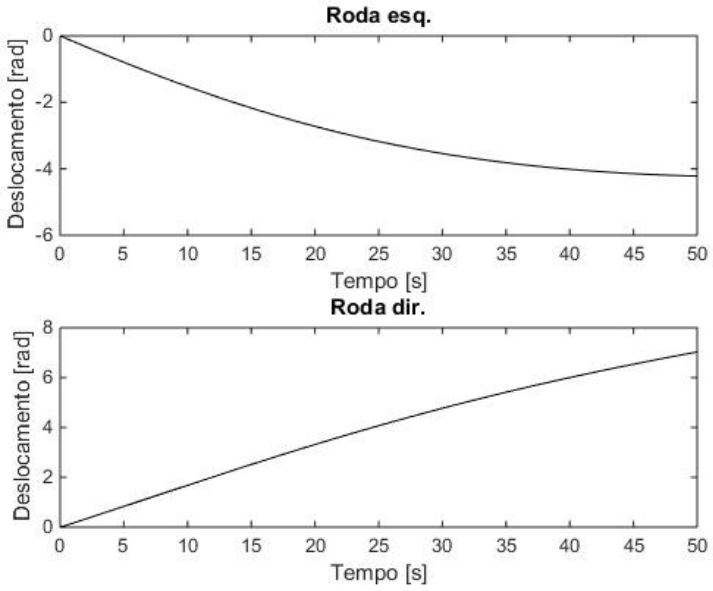


Figura 4.3: Deslocamento para cada roda para uma trajectória vertical de 10 cm de comprimento.

A discretização da trajectória (a azul) é comparada com a trajectória estimada (onde se considera a cinemática do robô, a vermelho) na figura 4.4. É notório o erro induzido pela diferença finita, contudo a ordem de grandeza é tão pequena (10^{-19}) que não se considera relevante, assumindo-se que é de facto uma linha recta. Foi avaliado o erro para $\Delta t = 1s$ na figura 4.5. Confirma-se que o incremento definido para o tempo é suficientemente pequeno para que sejam garantidos bons resultados e que o erro não é relevante mesmo para valores de incremento elevados.

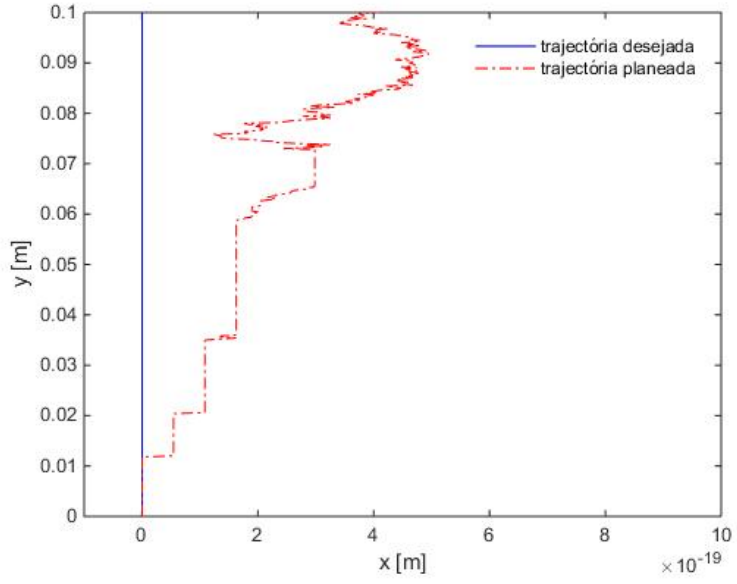


Figura 4.4: Trajectória discretizada e trajectória estimada para uma linha vertical de 10 cm de comprimento ($\Delta t = 0.1s$).

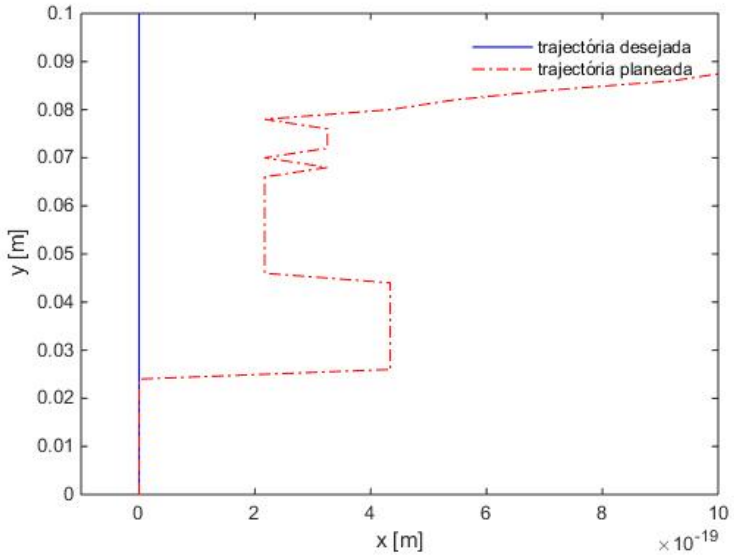


Figura 4.5: Trajectória discretizada e trajectória estimada para uma linha vertical de 10 cm de comprimento $\Delta t = 1s$.

A figura 4.6 apresenta a postura do robô ao longo da trajectória, sendo cada linha correspondente à postura para cada ponto da discretização. As linhas representadas unem o ponto de contacto do instrumento de escrita ao centro do eixo entre rodas. Desta forma, o facto das primeiras linhas começarem em abcissas negativas é clarificado ao considerar a figura 4.1. É importante destacar que o final de as linhas mencionadas formam a linha vertical definida como trajectória.

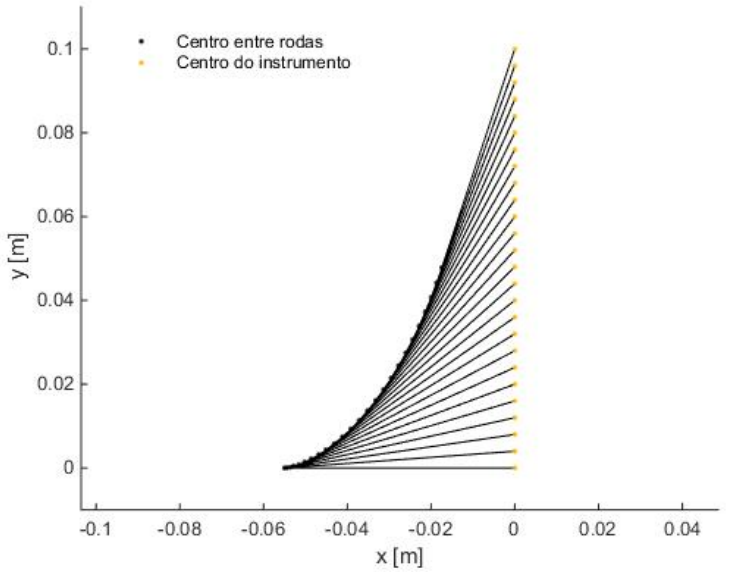


Figura 4.6: Postura do robô ao longo da trajectória para uma linha vertical de 10 cm de comprimento.

Para uma análise de múltiplas operações, planeou-se as trajectórias necessárias para formar um quadrado com 10 cm de lado, apresentadas na figura 4.7. Destaca-se que o ponto entre rodas do robô apresenta uma trajectória bastante diferente de um quadrado. Segundo o modelo cinemático do robô, aquela é a trajectória do ponto entre rodas necessária para que a ferramenta descreva de facto um quadrado. A figura 4.8 apresenta a posição estimada para o instrumento de escrita, comparativamente à figura 4.7, não será mais do que o ultimo ponto de cada linha representada na mesma. É notável a irrelevância do erro induzido pela diferença finita.

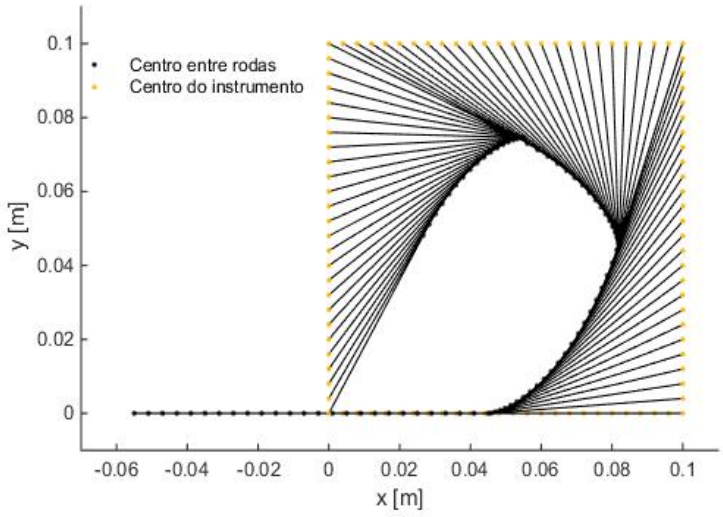


Figura 4.7: Postura do robô estimada para trajectórias que formam um quadrado de 10 cm de lado.

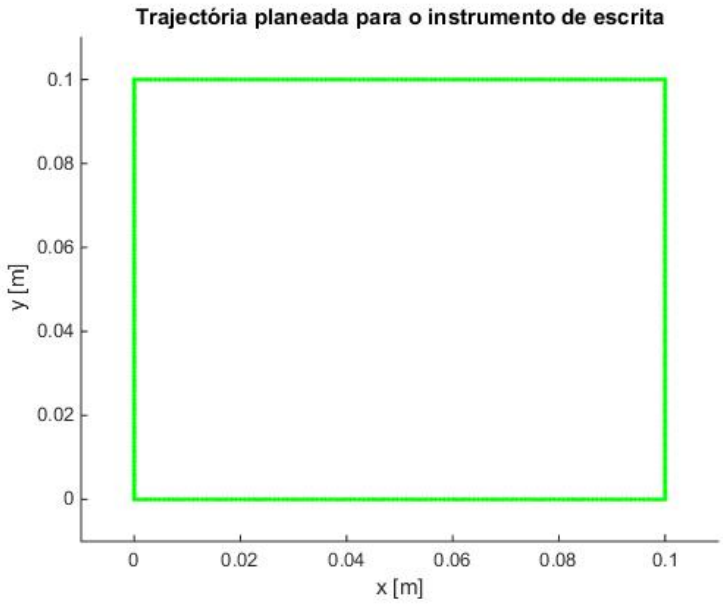


Figura 4.8: Posição do instrumento de escrita estimada para trajectórias que formam um quadrado de 10 cm de lado.

4.2 Controlo

Após vários testes para avaliar a resposta por parte do sistema na implementação do controlador *PID*, chegou-se à decisão de utilizar um controlador *PI* com um ganho proporcional de $K_P = 500$ e um ganho integrador de $K_I = 500$. Para uma comparação entre os controladores para cada motor, comandou-se o robô para que este descrevesse uma trajectória que desenharia uma linha horizontal. Desta forma, os motores realizaram a mesma operação, permitindo sobrepor a influência do controlador *PI* de cada motor. Esta sobreposição pode ser observada através da figura 4.9 e 4.10 onde se apresenta o incremento de velocidade para cada roda face ao tempo. Numa primeira abordagem, a acção de controlo para o motor esquerdo (a verde) e direito (a vermelho), quando comparados com a linha de referência (a azul) aparenta uma resposta estável e sem oscilações. Ao ampliar o gráfico, é notório no inicio da operação a convergência para o valor de referência e a pequena oscilação que tenta manter o sistema na referência, não apresentando contudo oscilações abruptas nem desvios evidentes do valor de referência.

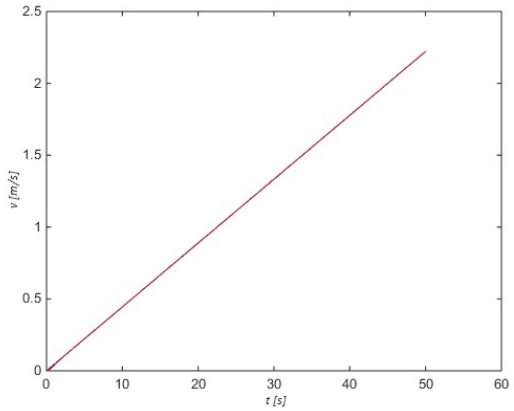
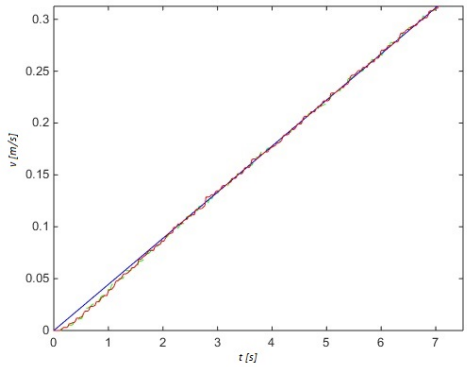
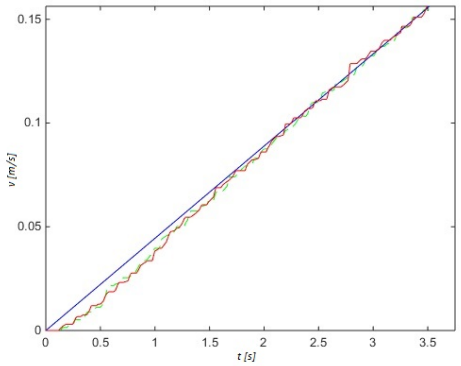


Figura 4.9: Acção de controlo para o motor da roda esquerda e direita e valores de referência para uma trajectória que descreve uma linha horizontal com 10 cm de comprimento.



(a) Primeira ampliação.



(b) Segunda ampliação.

Figura 4.10: Acção de controlo para o motor da roda esquerda e direita e valores de referência para uma trajectória que descreve uma linha horizontal com 10 cm de comprimento.

Para se avaliar a acção de controlo de operações sucessivas, considerou-se a experiência efectuada para o quadrado de 10 cm de lado apresentado na figura 4.7. As conclusões da análise para este caso são semelhantes ao anterior: existe uma primeira convergência até ao valor de referência, seguindo-se de uma resposta estável, como pode ser visto na figura 4.11, ampliada na figura 4.12. Nestas figuras são apresentadas quatro linhas para cada motor uma vez que cada uma corresponde a um dos segmentos que perfazem o quadrado. A roda esquerda está representada a verde, a direita a vermelho e valores de referencia a azul.

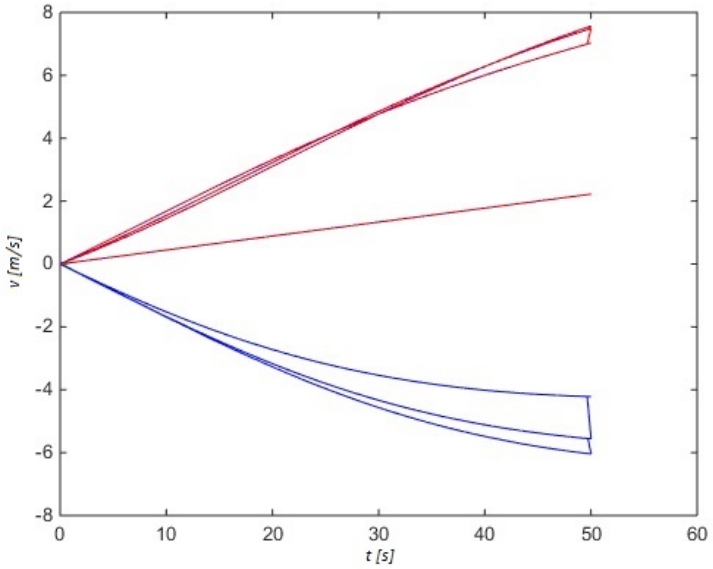
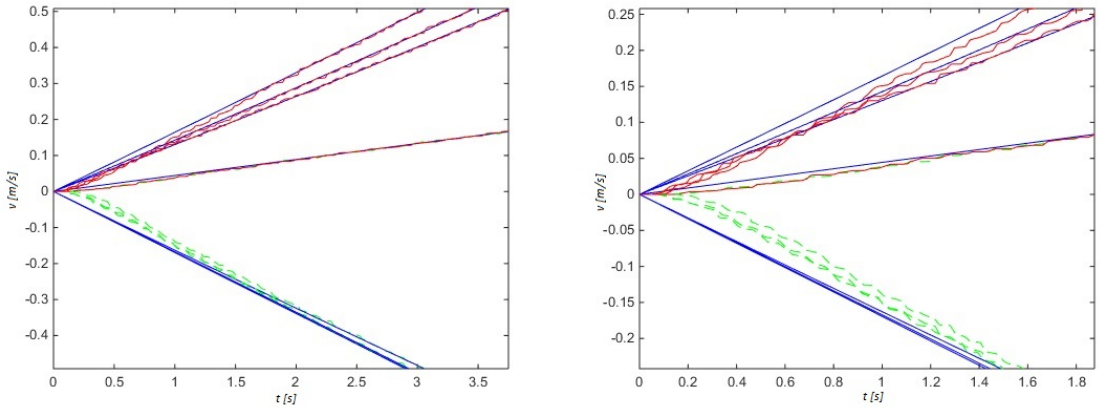


Figura 4.11: Acção de controlo para o motor da roda esquerda e direita e valores de referência para um conjunto de trajectórias que descrevem um quadrado com 10 cm de lado.



(a) Primeira ampliação. (b) Segunda ampliação.

Figura 4.12: Acção de controlo para o motor da roda esquerda e direita e valores de referência para um conjunto de trajectórias que descrevem um quadrado com 10 cm de lado.

4.3 Odometria

Ainda para o caso do quadrado de 10 cm de lado, a figura 4.13 apresenta a trajectória do robô com base na informação recebida pelos encoders. São notórios ligeiros desvios na mudança de direcção ao comparar com a figura 4.8 que descreve a trajectória planeada. A figura 4.14, ampliada na figura 4.15, apresenta a trajectória do ponto entre rodas com base na leitura dos encoders, permitindo compreender a forma como o robô se desloca para que se consiga produzir o quadrado. A figura 4.16 apresenta o resultado produzido pelo robô ao realizar o quadrado conforme planeado na figura 4.7.

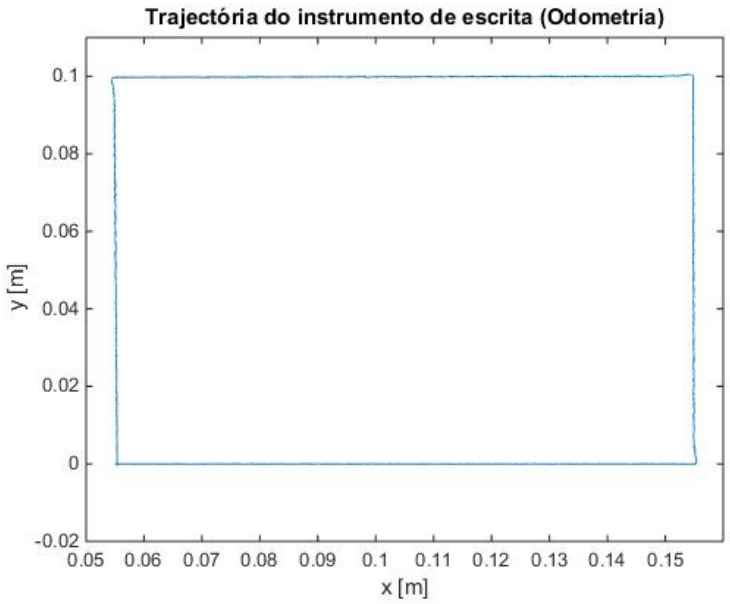


Figura 4.13: Leituras dos encoders para uma trajectória que descreve um quadrado com 10 cm de lado.

Apesar de a trajectória estimada pelas leituras dos *encoders* ser aproximadamente um quadrado (fig. 4.13), o quadrado real é bastante diferente, sendo evidente que o ângulo interior é superior a 90° na segunda mudança de direcção. Este facto contribui para que o quadrado não seja fechado, faltando cerca de 6 mm para que isso aconteça. Para além do desvio induzido pelo deslize das rodas, já expectável na localização baseada em odometria como visto no capítulo 2.4.1, existem algumas questões dimensionais que se consideram relevantes e que contribuem para o mesmo desvio. Tais questões são o facto de as rodas não estarem totalmente perpendiculares ao corpo do robô, como pode ver ser visto em duas perspectivas distintas apresentadas na figura 4.17.

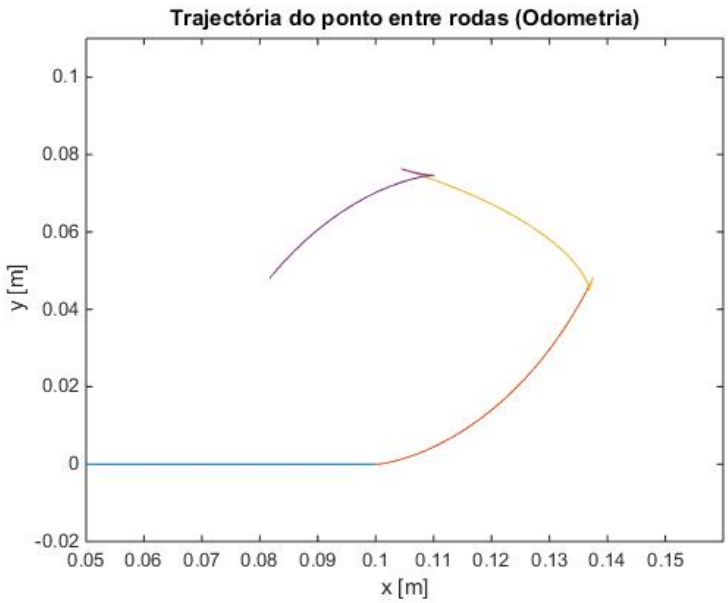
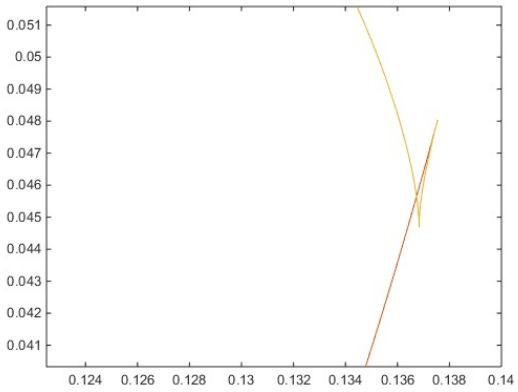
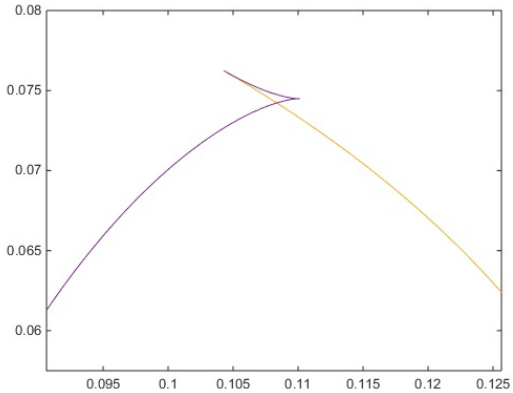


Figura 4.14: Trajectória do ponto entre rodas na descrição de um quadrado com 10 cm de lado com base nas leituras dos encoders.



(a) Primeira ampliação.



(b) Segunda ampliação.

Figura 4.15: Trajectória do ponto entre rodas na descrição de um quadrado com 10 cm de lado com base nas leituras dos encoders.

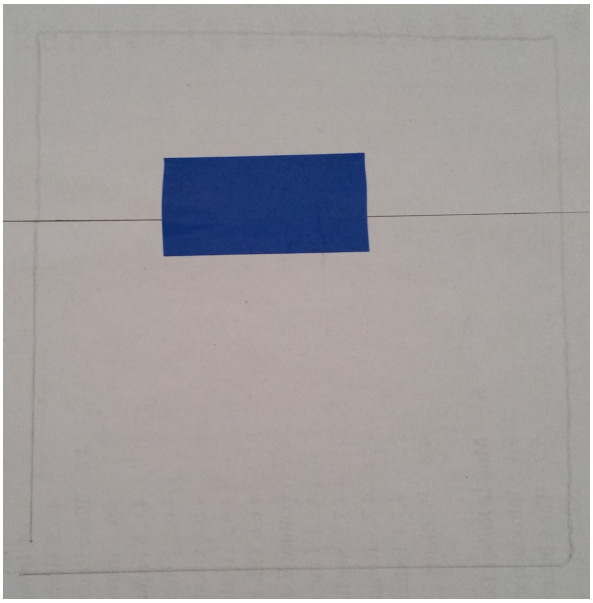
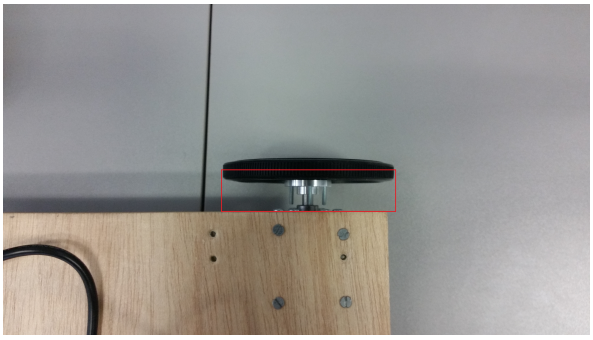
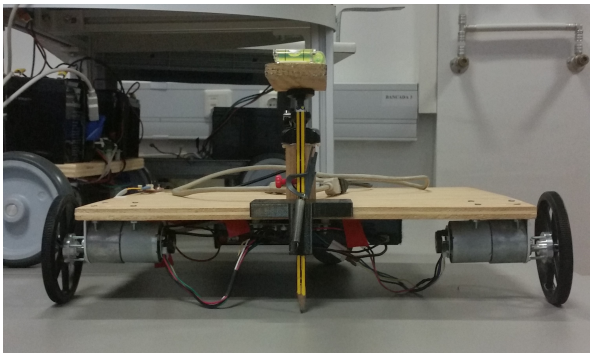


Figura 4.16: Resultado produzido pelo robô com localização baseada em odometria para uma trajectória que define um quadrado com 10 cm de lado.



(a) Perspectiva superior.



(b) Perspectiva frontal.

Figura 4.17: Imperfeição dimensional.

4.4 Calibração

Sabendo que $f(\rho) = a_0 + a_2\rho^2 + ... + a_N\rho^N$, tal como apresentado na equação 2.29, é apresentado na figura 4.18 o gráfico da função $f(\rho)$ para o caso da câmara utilizada. É ainda apresentado o ângulo do raio óptico em função da distância euclidiana ao centro da imagem ρ. Estes gráficos, apesar de serem únicos para cada conjunto câmara e lente, apenas permitem ter uma noção de como se comporta a função $f(\rho)$ e de que forma é que os raios ópticos incidem na lente, desta forma, não são mais do que características intrínsecas à câmara. O facto de os valores das ordenadas apresentarem um valor negativo é devido à escolha do referencial pelo autor da *toolbox*, tal como pode ser observado na figura 4.19. Nesta, são apresentados os planos para cada configuração do tabuleiro de xadrez utilizadas na calibração da câmara. Exemplos destas configurações são apresentadas na imagem 4.20, onde se encontram também a identificação dos cantos de cada quadrado, o centro de imagem calibrado e os eixos do referencial do tabuleiro, sendo que todos estes parâmetros foram obtidos através do cálculo da calibração. Na figura 4.21 é apresentada a distribuição do erro na estimação da posição de cada canto do tabuleiro de xadrez para as diferentes configurações, cada uma correspondente a uma cor. Com estes resultados é visível que a identificação dos cantos e a definição dos eixos que definem o plano do tabuleiro são satisfatórios e não se apresentam desvios. A dispersão do erro para cada ponto do tabuleiro apresenta-se bastante concentrado, tendo em conta que a figura é apresentada em píxel.

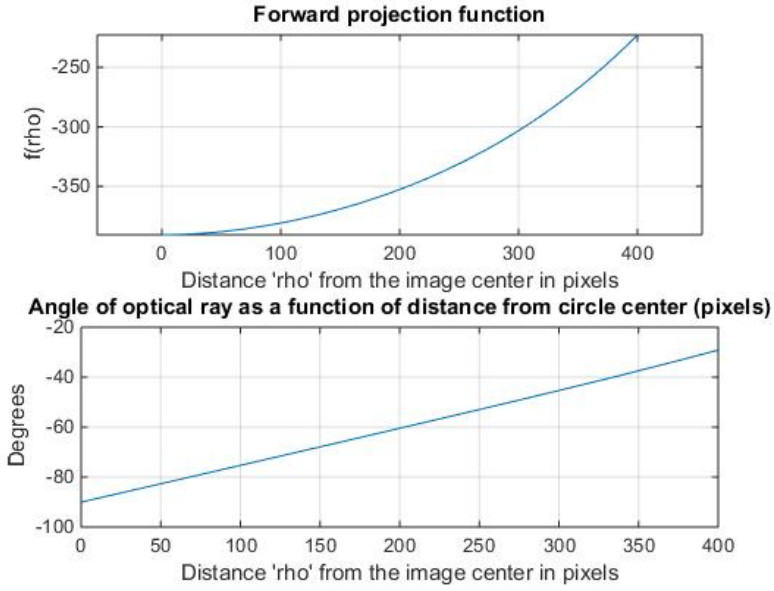


Figura 4.18: Função $f(\rho)$ e ângulo do raio óptico em função da distância euclidiana ao centro da imagem.

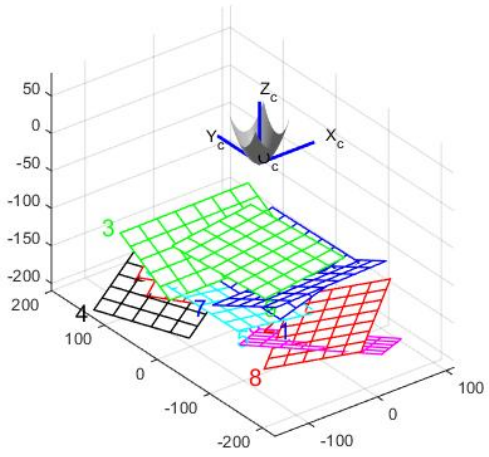
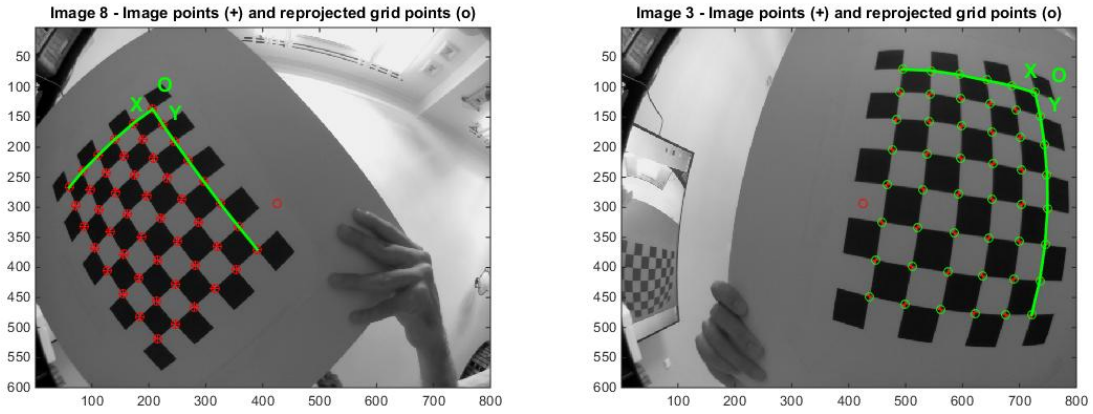


Figura 4.19: Representação dos planos de cada tabuleiro de xadrez utilizado para a calibração.



(a) Primeiro caso.

(b) Segundo caso.

Figura 4.20: Exemplos de imagens do tabuleiro de xadrez utilizadas na calibração com a representação dos cantos identificados, o centro da imagem calibrada e os eixos cartesianos definidos.

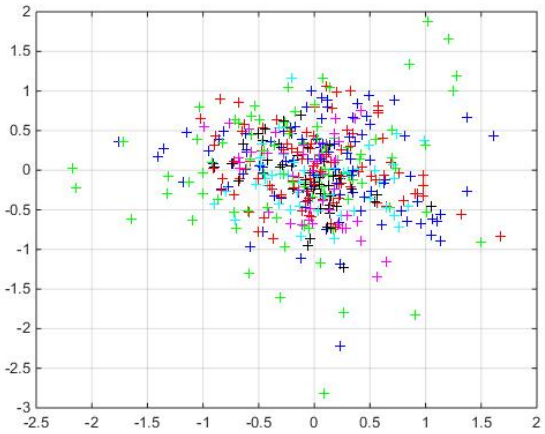
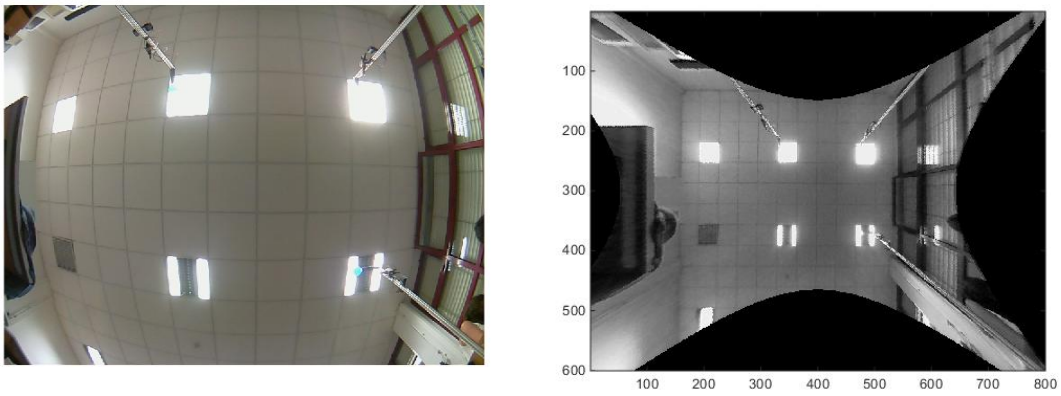


Figura 4.21: Distribuição do erro na projecção para cada ponto do tabuleiro de xadrez para as diferentes configurações do mesmo (em píxel).

4.5 Transformação de imagem

Como mencionado em 3.6, a resolução utilizada para as imagens foi 800×600 para que não houvesse demasiada informação a ser processada e consequente aumento do tempo de computação do modelo. O parâmetro experimental atribuído foi $fc = 5$. A imagem original é apresentada em 4.22a onde se tem a câmara apontada para o tecto e é clara a deformação induzida pela lente de olho de peixe. A figura 4.22b apresenta a mesma imagem após sofrer a alteração imposta pelo algoritmo de transformação de imagem proposto em 3.8. O motivo de se apresentarem regiões a preto com o formato de hipérboles está relacionado com o facto de a câmara utilizar uma perspectiva cónica.

De uma forma simples, a imagem original não é mais do que uma imagem circular ou elipsoidal. Para a imagem estar representada no mesmo plano, será necessário excluir os pontos que se encontram fora do mesmo quando representados na nova imagem transformada. Como se pode verificar, a imagem transformada apresenta o tecto todo com a mesma profundidade. Ao observar o canto da sala, é notório o resultado satisfatório, pois as linhas apresentam-se perpendiculares entre si.



(a) Não transformada. (b) Após o algoritmo de transformação de imagem.

Figura 4.22: Imagem captada pela câmara.

4.6 Identificação de beacons

Como é visível na figura 4.22a, a imagem é demasiado clara para que seja possível distinguir as cores de cada *beacon*, sendo que neste caso o código de cor *RGB* que o *beacon* apresenta é perto da cor branca. Para resolver este problema, conseguindo produzir a imagem apresentada em 4.23, utilizou-se um tempo de exposição bastante pequeno. Por comparação a imagem 4.22a, é conseguida com um tempo de exposição superior. Esta solução não dá tempo suficiente ao sensor para ler os feixes de luz necessários para definir a luminosidade real do espaço, evidenciando as cores dos *beacons* visto que a imagem, apesar de o ambiente apresentar bastante contra-luz, está altamente escurecida.

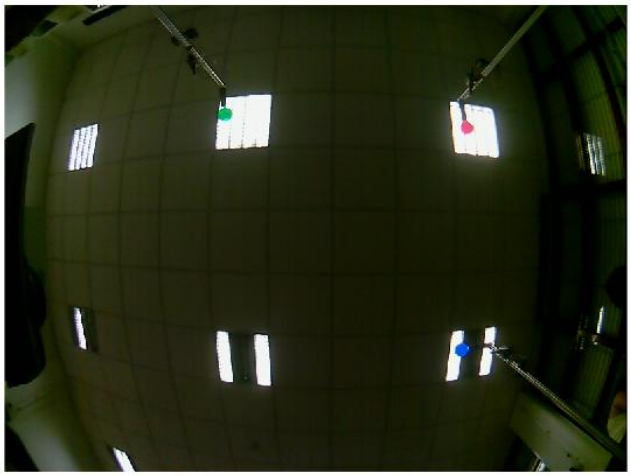


Figura 4.23: Imagem captada pela câmara com um baixo tempo de exposição.

A binarização tendo em conta a tolerância, tal como descrito em 3.6 é apresentada na figura 4.24 onde se mostra a identificação do *beacon* da cor azul (cor definida para o presente exemplo) apresentada pela mancha amarela, juntamente com o ponto médio desta identificação apresentado pela cruz preta. Identificada a coordenada do píxel de interesse, este é identificado na imagem transformada, tal como pode ser visto na figura 4.25, onde para além do *beacon* está representado o ponto correspondente ao centro da imagem calibrada.

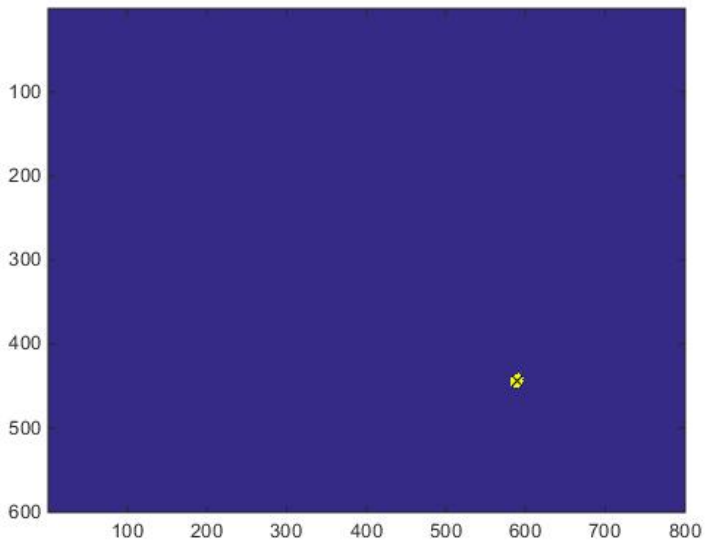


Figura 4.24: Binarização destacando a mancha de píxeis identificados e o ponto médio da nuvem.

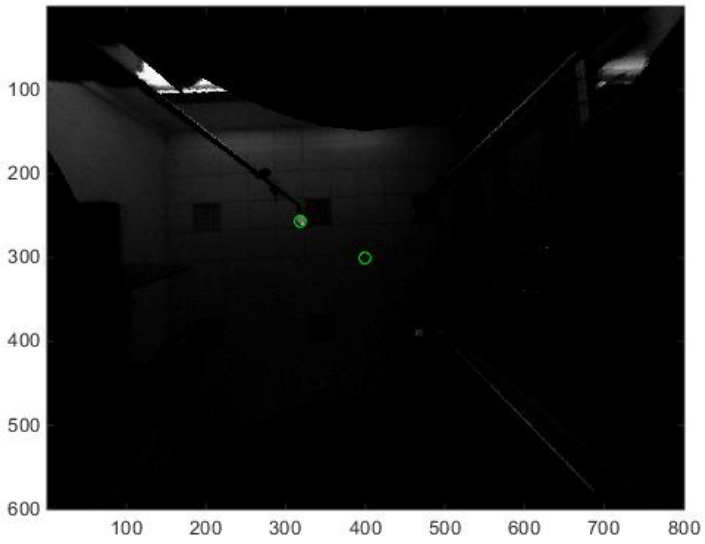


Figura 4.25: Imagem captada pela câmara com um baixo tempo de exposição após o algoritmo de transformação de imagem.

4.7 Filtro de Kalman estendido

Para aplicar o método apresentado na secção 3.10 para o filtro de Kalman estendido foram definidas, através de experimentação, as matrizes de covariância associadas ao estado inicial (*P*), à perturbação (*Q*) e à medição (*R*):

$$P = \begin{bmatrix} 0.005^2 & 0 & 0 \\ 0 & 0.005^2 & 0 \\ 0 & 0 & 0.0175^2 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0.001^2 & 0 \\ 0 & 0.001^2 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.005^2 & 0 & 0 \\ 0 & 0.005^2 & 0 \\ 0 & 0 & 0.0175^2 \end{bmatrix}$$

Na implementação do método definiu-se uma tolerância de $5cm$ entre a estimação dada pelo filtro de Kalman estendido e a odometria. Caso essa tolerância seja ultrapassada, um novo planeamento de trajectória é definido que assume como condição inicial a estimação dada pelo filtro de Kalman estendido. A observação introduzida no filtro é dada pelo algoritmo de localização baseada em visão descrito na secção 3.9.

Com as matrizes definidas, foi realizada uma experiência onde se definiu como trajectória uma linha recta com $10cm$ e onde as leituras da odometria, da localização baseada em visão e do filtro de Kalman estendido são apresentadas na figura 4.26. A dispersão das leituras por parte do algoritmo da visão são notáveis, contudo, tendo em conta a ordem de grandeza da variação dos valores, considera-se que a estimação por parte do filtro de Kalman estendido apresenta resultados coerentes. Os pontos mais distantes medidos por parte da visão, quando avaliados perpendicularmente à trajectória encontram-se a sensivelmente $4mm$, permitindo definir a dispersão das leituras como aceitável para o estudo em questão.

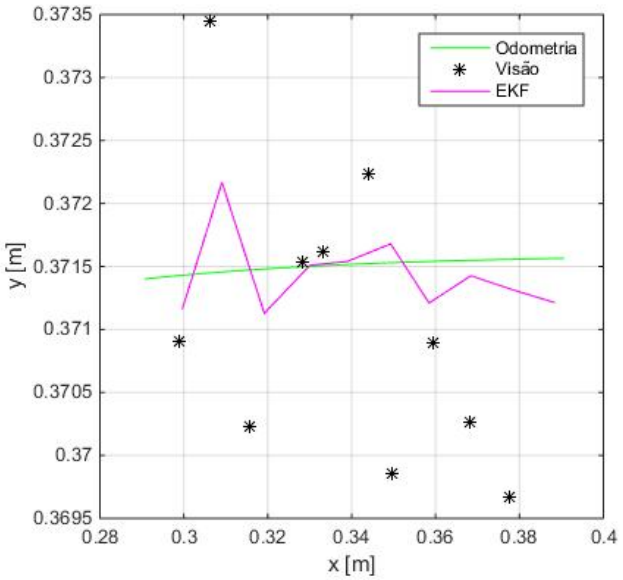


Figura 4.26: Leituras de odometria, visão e estimação do filtro de Kalman estendido para uma trajectória de $10cm$.

Uma vez que os resultados por parte da visão e do filtro de Kalman estendido para a primeira experiência não apresentaram desvios consideráveis nem leituras incoerentes, procedeu-se a uma tentativa de obter um quadrado com $10cm$ de lado, caso utilizado para o algoritmo de planeamento e odometria. O resultado para a segunda experiência é apresentado na figura 4.27. As interrupções na linha da odometria (verde) significam que a tolerância definida foi atingida, e que portanto uma nova trajectória foi planeada com base na estimação utilizada para comparação (linha magenta). Por observação do gráfico é visível que o resultado pretendido não é conseguido, sendo produzida uma linha curva e não uma linha recta. Ao analisar os dados utilizados para a estimação, é notório que a odometria força o sistema a afastar-se da desejada perpendicularidade de trajectórias enquanto que a visão actua no sentido contrário. A estimação está mais próxima da trajectória definida pela visão, o que se considera positivo uma vez que a odometria apresenta uma tendência não desejável. Esta poderá ser

explicada pelas folgas mecânicas, irregularidades no solo onde o robô opera e deslizes das rodas que apresentam maior influência quando uma rotação é induzida no robô. Uma vez que foi definido um comprimento de $10cm$ para a segunda trajectória é claro por análise da figura 4.27 que essa dimensão é ultrapassada. Isto poderá ser explicado pelo facto de a estimação ter passado "ao lado"da coordenada alvo, e fora da tolerância definida de $5cm$. Desta forma, o robô irá apresentar uma trajectória em espiral onde procura tentar alcançar o ponto definido como alvo.

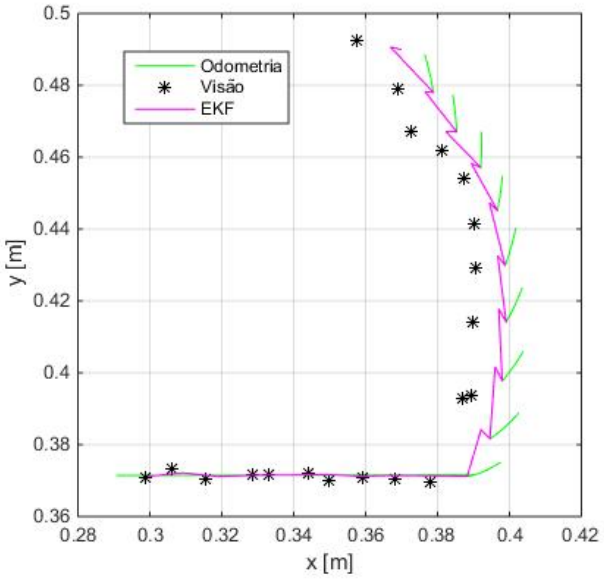
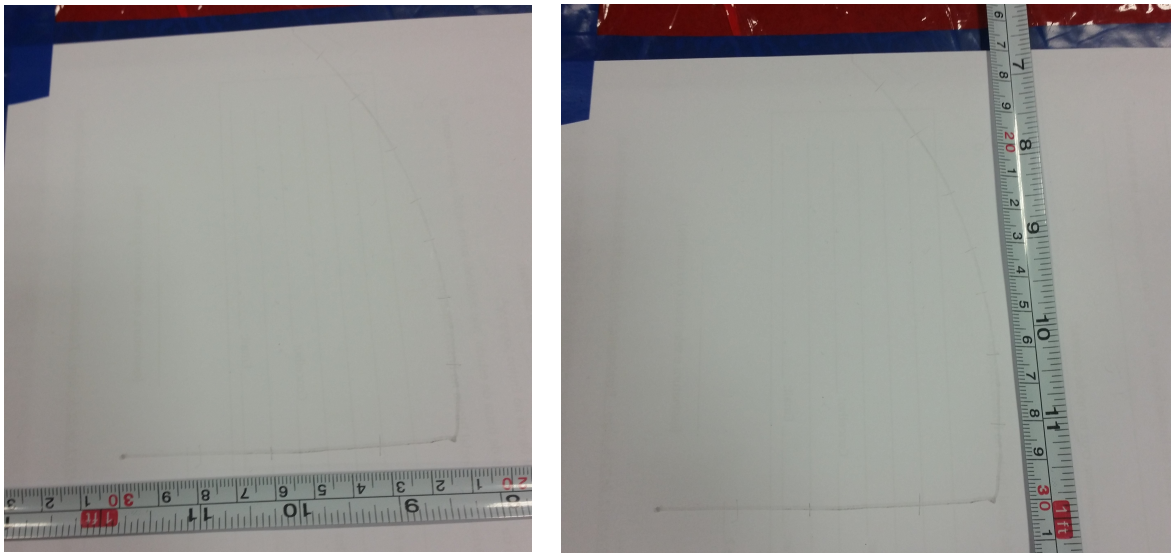


Figura 4.27: Leituras de odometria, visão e estimação do filtro de Kalman estendido para duas trajectórias perpendiculares de $10cm$.

O resultado experimental é apresentado na figura 4.28, onde é descrita a trajectória curvilínea no segundo troço da operação. No primeiro troço, não são atingidos os $10cm$ inicialmente planeados, apresentado cerca de $1cm$ de erro. É difícil quantificar o erro associado ao segundo troço visto que o algoritmo não reconhece que a coordenada alvo foi atingida, contudo é notório que existe um desvio da trajectória mais acentuado a cerca de $5cm$ da intersecção das trajectórias. Ao acentuar cada vez mais esta curva, é visível o inicio da trajectória espiral já descrita.



(a) Primeira trajectória. (b) Segunda trajectória.

Figura 4.28: Resultado experimental para duas trajectórias perpendiculares de $10cm$.

Na figura 4.29 é introduzida uma medição do resultado real, denominado *ground truth*. Um exemplo de medição é apresentado na figura 4.30. Contudo, é difícil de transpor o espaço de trabalho definido pelos *beacons* para o espaço onde o robô opera. Desta forma, considera-se que as medições de *ground truth* são pouco fiáveis, uma vez que deveriam estar relativamente próximas da estimação do filtro.

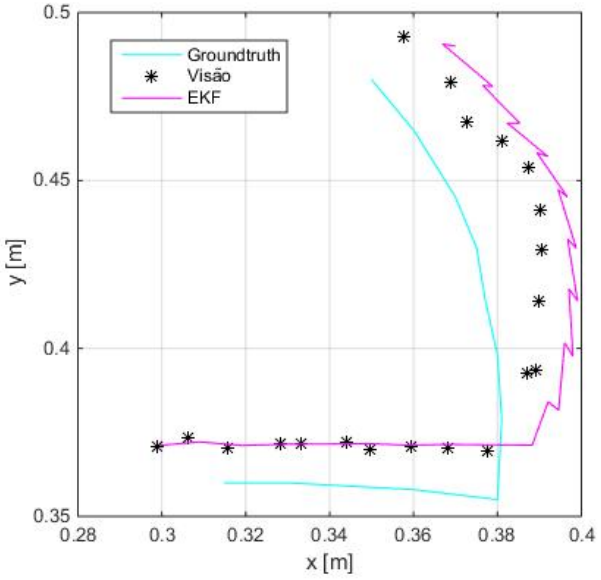


Figura 4.29: Leituras de *ground truth*, visão e estimação do filtro de Kalman estendido para duas trajectórias perpendiculares de $10cm$.

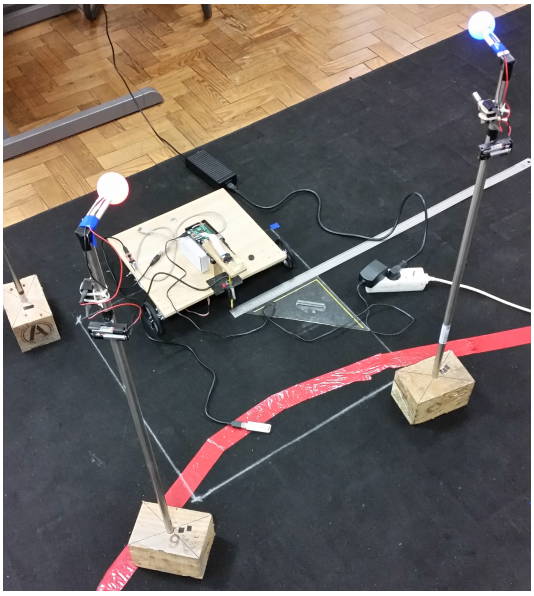


Figura 4.30: Exemplo de uma medição efectuada através dos instrumentos de medida.

Tendo em conta os problemas impostos pelas questões dimensionais do robô, pela construção dos *beacons* e pela correcta definição do espaço de trabalho a partir do qual se determina o *ground truth*, considera-se que a abordagem para o filtro de Kalman estendido produz resultados coerentes, apesar de não atingir os valores desejados. Outra observação a ter em conta é o facto de os resultados obtidos a partir da visão serem os mais aproximados ao valores de *ground truth* apesar de estes não serem fiáveis.

Capítulo 5

Conclusões e desenvolvimentos futuros

Na presente dissertação foi apresentada uma abordagem para um robô móvel de tracção diferencial com localização baseada em visão com o intuito de ser aplicado em máquinas-ferramenta. Para desenvolver o protótipo e os *beacons* foram utilizados componentes pouco dispendiosos, que permitem fazer uma primeira avaliação ao problema proposto. Procurando obter uma boa precisão durante a operação do robô, um primeiro desafio foi apresentado pela odometria onde existe um deslize por parte das rodas e folgas mecânicas que é necessário contornar. Outro problema apresentado pela odometria são as irregularidades do espaço de trabalho, que potenciam fortemente o fenómeno de deslize das rodas. Na tentativa de minimizar esse erro, surge a localização baseada em visão. Nesta abordagem revelou-se importante conseguir uma boa calibração do conjunto câmara-lente. Através dos resultados apresentados, é visível o quão desviado o centro da câmara está quando comparado com o centro da imagem captada. Este é um parâmetro importante quando se pretende avaliar distâncias relativamente à câmara. Outra mais valia do processo de calibração é a determinação dos parâmetros intrínsecos da câmara, que permitem retirar, através do algoritmo de transformação de imagem proposto, a deformação imposta pela lente. A transformação implica que a imagem final fique com uma resolução inferior à original, evidenciando a importância de uma câmara com capacidade de capturar imagens de elevada resolução. Desta forma, considera-se um correcto processo de calibração fundamental para que a localização baseada em visão seja fiável. Uma vez que o protótipo apenas dispõe de uma única câmara, não se tem informação de profundidade sendo a sua quantificação um parâmetro experimental, pois é necessário saber a que distância se encontram determinados pontos observados e conhecidos para que seja possível definir um plano a essa profundidade. As avaliações de distâncias no dito plano relacionam-se com o plano da câmara através de um factor de escala. Uma vez que os *beacons* estão implementados no espaço de trabalho em posições conhecidas e todos à mesma altura (também esta conhecida), é possível utilizar uma única câmara, simplificando a abordagem da localização baseada em visão.

Na abordagem proposta para a identificação de *beacons*, recorre-se a um rastreamento de referências *RGB*. Esta solução requer algum cuidado, pois os níveis de iluminação no espaço tendem a variar consoante a altura do dia, não permitindo que os valores para as referências dos *beacons* sejam sempre constantes. O sensor da câmara revelou-se altamente sensível às ditas variações de luminosidade, reagindo a situações ténues como a passagem de uma nuvem, uma sombra por parte de uma pessoa no espaço de trabalho ou uma fonte luminosa distante que influencia muito pouco o espaço de trabalho do robô.

O algoritmo de localização baseado em visão, assumindo que os *beacons* são devidamente identificados, é de simples implementação e apresenta um custo computacional reduzido, calculando a posição relativa do robô relativamente a pares distintos de dois *beacons*. Para o caso em estudo foram utilizados três pares distintos que produziram resultados satisfatórios que poderiam ser melhorados com sucessivas adições de novos *beacons*.

Na implementação do filtro de Kalman estendido é evidente a limitação do hardware: a incapacidade de obter um *ground truth* fiável, o facto de os suportes para os *beacons* não permitirem definir posições e alturas rigorosas, o facto de a imagem transformada apresentar uma resolução relativamente baixa, a variação das referências *RGB* dos *beacons* e as irregularidades no espaço de trabalho que potenciam os problemas inerentes à odometria, todas estas contribuem para que haja uma incerteza associada à conjugação de todas as medições. Para ser possível ao filtro compensar todos estes erros, seriam necessários múltiplos testes para tentar determinar o erro associado a cada leitura efectuada por cada um destes sistemas, em particular o da localização baseada em visão que é a informação que é introduzida como a referência para correcção. Outro aspecto importante seria uma definição rigorosa do *ground truth*, que permitiria avaliar se a trajectória real produzida pelo robô se encontrava maioritariamente dentro do raio de covariância da estimação produzida pelo filtro. Esta avaliação seria conseguida com, por exemplo, múltiplas câmaras estéreo exteriores ao sistema, capazes de avaliar a posição real do robô.

Apesar dos resultados apresentados não reflectirem precisão suficiente para a aplicação pretendida, onde se pratica operações à escala do milímetro, considera-se que controlando os erros descritos através das sugestões propostas poder-se-á obter resultados mais adequados.

Desenvolvimentos futuros

Para desenvolvimento futuro, propõe-se reformular o algoritmo de segmentação para melhor identificar as regiões identificadas de cada *beacon*. Para tal seria necessário adquirir valores para as referências *RGB* para cara *beacon* ao longo de diferentes horas do dia e em diferentes posições no espaço de trabalho. Posteriormente, a partir da resposta da localização baseada em visão e do erro (calculado relativamente ao *ground truth*) criar-se-ia um algoritmo de ajuste automático dos *thresholds*. Como outra alternativa para a melhoria do resultado por parte da visão propõe-se a consideração de mais *beacons* no espaço de trabalho.

Propõe-se também a implementação de um servo responsável pela actuação do instrumento de escrita. Para além de aumentar a proximidade da realidade da actuação de uma máquina ferramenta, permitiria corrigir pequenos desvios sem que fosse necessário alterar a trajectória do robô.

Outra abordagem a considerar seria a implementação de uma segunda câmara para que a informação acerca da profundidade seja conseguida. Desta forma, a preocupação com o conhecimento da altura dos *beacons* deixaria de existir.

Referências

O. A. Aider, P. Hoppenot, and E. Colle. A model-based method for indoor mobile robot localization using monocular vision and straight-line correspondences. *Robotics and Autonomous Systems*, 52 (2?3):229 – 246, 2005. ISSN 0921-8890. (Cit. em 5.)

J. P. Barreto and H. Araujo. Issues on the geometry of central catadioptric image formation. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–422–II–427 vol.2, 2001. (Cit. em 32.)

G. Beccari, S. Caselli, F. Zanichelli, and A. Calafiore. Vision-based line tracking and navigation in structured environments. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 406–411, Jul 1997. (Cit. em 8.)

F. P. Beer, F. Beer, E. R. Johnston, P. J. Cornwell, and E. Eisenberg. *Vector Mechanics for Engineers: Dynamics*. McGraw-Hill Higher Education, 9th edition, 2009. ISBN 0390994162,9780390994165. (Cit. em 42.)

J.-Y. Bouguet. Camera calibration toolbox for matlab. Dept. of Electrical Engineering, California Institute of Technology. URL `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html`. Acedido em 16-10-2016. (Cit. em 10.)

F. Calabrese and G. Indiveri. An omni-vision triangulation-like approach to mobile robot localization. In *Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.*, pages 604–609, June 2005. (Cit. em 58 e 59.)

Z. Chen and S. T. Birchfield. Qualitative vision-based mobile robot navigation. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2686–2692, May 2006. (Cit. em 6.)

G. Cook. *Mobile Robots: Navigation, Control and Remote Sensing*. John Wiley & Sons, 1 edition, 2011. ISBN 0470630213,9780470630211. (Cit. em 19.)

P. Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer Tracts in Advanced Robotics 73. Springer-Verlag Berlin Heidelberg, 1 edition, 2011. ISBN 3642201431,9783642201431. (Cit. em 12, 13, 14, 17, 33 e 57.)

S. Department. Statistical Department, International Federation of Robotics. URL `http://www.ifr.org/industrial-robots/statistics/`. Acedido em 16-10-2016. (Cit. em 2.)

M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17:229–241, 2001. (Cit. em 8.)

P. Dyke. *An Introduction to Laplace Transforms and Fourier Series*. Springer Undergraduate Mathematics Series. Springer-Verlag London, 2 edition, 2014. ISBN 978-1-4471-6394-7,978-1-4471-6395-4. (Cit. em 23.)

M. Fiala and A. Basu. Robot navigation using panoramic tracking. *Pattern Recognition*, 37(11):2195 – 2215, 2004. ISSN 0031-3203. (Cit. em 7.)

M. Frye. Elementary motion detectors. *Current Biology*, 25(6):R215 – R217, 2015. ISSN 0960-9822. (Cit. em 8.)

R. Gartshore, A. Aguado, and C. Galambos. Incremental map building using an occupancy grid for an autonomous monocular robot. In *Control, Automation, Robotics and Vision, 2002. ICARCV 2002. 7th International Conference on*, volume 2, pages 613–618 vol.2, Dec 2002. (Cit. em 5.)

J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE Transactions on Robotics and Automation*, 16(6): 890–898, Dec 2000. ISSN 1042-296X. (Cit. em 7.)

C. Geyer and K. Daniilidis. Catadioptric projective geometry. *International Journal of Computer Vision*, 45(3):223–243, 2001. ISSN 1573-1405. (Cit. em 30 e 31.)

S. B. Goldberg, M. W. Maimone, and L. Matthies. Stereo vision and rover navigation software for planetary exploration. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 5, pages 5–2025–5–2036 vol.5, 2002. (Cit. em 7.)

R. F. Graf. *Modern Dictionary of Electronics*. Newnes, 7 edition, 1999. ISBN 0750698667,9780750698665. (Cit. em 50.)

J. Guivant and E. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation*, 17:242–257, 2001. (Cit. em 9.)

M. Gutiérrez. El grupo ortogonal. Departamento de Álgebra, Geometría y Topología de la universidade de Málaga, Nov. 2005. (Cit. em 15.)

I. Horswill. Visual collision avoidance by segmentation. In *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, volume 2, pages 902–909 vol.2, Sep 1994. (Cit. em 5.)

I. D. Horswill. *Specialization of Perceptual Processes*. PhD thesis, Massachusetts Institute of Technology, 1995. (Cit. em 5.)

R. W. G. Hunt. *The Reproduction of Colour - 6th Edition (The Wiley-IS&T Series in Imaging Science and Technology)*. 6 edition, 2004. ISBN 0470024259,9780470024256,9780470024263. (Cit. em 52.)

R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679 – 688, 2006. ISSN 0169-2070. (Cit. em 53.)

S. Ishikawa, H. Kuwamoto, and S. Ozawa. Visual navigation of an autonomous vehicle using white line recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):743–749, Sept 1988. ISSN 0162-8828. (Cit. em 8.)

A. H. Ismail, H. R. Ramli, M. H. Ahmad, and M. H. Marhaban. Vision-based system for line following mobile robot. In *Industrial Electronics Applications, 2009. ISIEA 2009. IEEE Symposium on*, volume 2, pages 642–645, Oct 2009. (Cit. em 8.)

K. Kidono, J. Miura, and Y. Shirai. Autonomous visual navigation of a mobile robot using a human-guided experience. *ROBOTICS AND AUTONOMOUS SYSTEMS*, 40:2–3, 2000. (Cit. em 6.)

K. Konolige, M. Agrawal, R. C. Bolles, C. Cowan, M. Fischler, and B. Gerkey. Outdoor mapping and navigation using stereo vision. In *Proceedings of the International Symposium on Experimental Robotics*, 2006. (Cit. em 6.)

K. L. Kuttler. *Elementary Linear Algebra*. Lecture notes. version 9 feb 2016 edition, 2016. (Cit. em 14.)

S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. ISBN 9780521862059,0521862051. (Cit. em 29 e 45.)

H. Levkowitz. *Color theory and modeling for computer graphics, visualization, and multimedia applications*. The Kluwer international series in engineering and computer science SECS 402. Kluwer Academic Publishers, 1 edition, 1997. ISBN 0792399285,9780792399285,9780585284286. (Cit. em 52.)

Y. Liu and S. Thrun. Results for outdoor-slam using sparse extended information filters. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 1, pages 1227–1233 vol.1, Sept 2003. (Cit. em 8.)

E. O. Mathematics. Orthogonal group. URL `http://www.encyclopediaofmath.org/index.php?title=Orthogonal_group&oldid=34320`. Acedido em 16-10-2016. (Cit. em 15.)

C. Mei. Omnidirectional calibration toolbox. Department of Engineering Science, University of Oxford. URL `http://www.robots.ox.ac.uk/~cmei/Toolbox.html`. Acedido em 16-10-2016. (Cit. em 10.)

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI. (Cit. em 9.)

H. P. Moravec. The stanford cart and the cmu rover. *Proceedings of the IEEE*, 71(7):872–884, July 1983. ISSN 0018-9219. (Cit. em 5.)

A. Murillo, J. Ko?ecká, J. Guerrero, and C. Sagüés. Visual door detection integrating appearance and shape cues. *Robotics and Autonomous Systems*, 56(6):512 – 521, 2008. ISSN 0921-8890. From Sensors to Human Spatial Concepts. (Cit. em 6.)

D. Murray and J. J. Little. Using real-time stereo vision for mobile robot navigation. *Autonomous Robots*, 8(2):161–171, 2000. ISSN 1573-7527. (Cit. em 6.)

T. Netter and N. Francheschini. A robotic aircraft that follows terrain using a neuromorphic eye. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 129–134 vol.1, 2002. (Cit. em 8.)

N. S. Nise. *Control Systems Engineering*. Wiley, 7 edition, 2015. ISBN 1118170512,9781118170519. (Cit. em 26.)

H. N. O. Tingleff, K. Madsen. Metods for non-linear least squares problems. Apr. 2004. Technical University of Denmark. (Cit. em 55.)

K. Ogata. *Modern Control Engineering*. Prentice Hal, 5 edition, 2010. ISBN 978-0136156734. (Cit. em 22, 23, 24 e 25.)

C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone. Rover navigation using stereo ego-motion. *Robotics and Autonomous Systems*, 43(4):215 – 229, 2003. ISSN 0921-8890. (Cit. em 6.)

A. N. M. Panos J Antsaklis. *A Linear Systems Primer*. Birkhauser, 2007 edition, 2010. ISBN 0817644601,9780817644604. (Cit. em 27.)

C. Poynton. *Digital Video and HDTV. Algorithms and Interfaces*. The Morgan Kaufmann Series in Computer Graphics. 2003. ISBN 9781558607927,1-55860-792-7. (Cit. em 52.)

D. W. H. Richard Khoury. *Numerical Methods and Modelling for Engineering*. Springer International Publishing, 1 edition, 2016. ISBN 978-3-319-21175-6, 978-3-319-21176-3. (Cit. em 20.)

D. S. Roland Siegwart, Illah Reza Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Intelligent Robotics and Autonomous Agents series. The MIT Press, 2nd edition, 2011. ISBN 0262015358,9780262015356. (Cit. em 18, 21, 28, 29, 30, 31, 34, 35, 36 e 37.)

T. Saitoh, N. Tada, and R. Konishi. *Computer Vision*, chapter Indoor Mobile Robot Navigation by Center Following Based on Monocular Vision, pages 352–366. InTech, Nov. 2008. (Cit. em 6.)

D. Scaramuzza. Ocamcalib: Omnidirectional camera calibration toolbox for matlab. Department of Informatics - Robotics and Perception Group, University of Zurich. URL `http://rpg.ifi.uzh.ch/people_scaramuzza.html`. Acedido em 16-10-2016. (Cit. em 10 e 54.)

D. Scaramuzza, A. Martinelli, and R. Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, pages 45–45, Jan 2006a. (Cit. em 10 e 32.)

D. Scaramuzza, A. Martinelli, and R. Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701, Oct 2006b. (Cit. em 10, 33 e 54.)

D. Scaramuzza, M. Rufli, and R. Siegwart. Automatic detection of checkerboards on blurred and distorted images. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3121–3126, Sept 2008. (Cit. em 10, 33 e 54.)

D. F. Sebastian Thrun, Wolfram Burgard. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. 2005. ISBN 0262201623,9780262201629. (Cit. em 34.)

J. Shi and C. Tomasi. Good features to track. pages 593–600, 1994. (Cit. em 7.)

R. Sim, P. Elinas, and M. Griffin. Vision-based slam using the rao-blackwellised particle filter. In *In IJCAI Workshop on Reasoning with Uncertainty in Robotics*, 2005. (Cit. em 9.)

J. Solà. *Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot: a Geometric and Probabilistic Approach*. PhD thesis, Institut National Politechnique de Toulouse, Feb. 2007. (Cit. em 35, 36, 37 e 38.)

S. Tzafestas. *Introduction to Mobile Robot Control*. Elsevier, 1 edition, 2014. ISBN 978-0-12-417049-0. (Cit. em 48 e 50.)

J. Vince. *Geometric algebra for computer graphics*. Springer-Verlag London, 1 edition, 2008. ISBN 1846289963,9781846289965. (Cit. em 16.)

N. Winters and J. Santos-Victor. Omni-directional visual navigation. In *In Proceedings of the 7th International Symposium on Intelligent Robotics Systems*, pages 109–118, 1999. (Cit. em 7.)

D. Wooden. A guide to vision-based map building. *IEEE Robotics Automation Magazine*, 13(2): 94–98, June 2006. ISSN 1070-9932. (Cit. em 7.)

X. Ying and Z. Hu. *Can We Consider Central Catadioptric Cameras and Fisheye Cameras within a Unified Imaging Model*, pages 442–455. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-24670-1. (Cit. em 32.)

H.-B. Zhang, K. Yuan, Shu-Qimei, and Q.-R. Zhou. Visual navigation of an automated guided vehicle based on path recognition. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 6, pages 3877–3881 vol.6, Aug 2004. (Cit. em 8.)

G. Zunino and H. I. Christensen. Simultaneous localization and mapping in domestic environments. In *Multisensor Fusion and Integration for Intelligent Systems, 2001. MFI 2001. International Conference on*, pages 67–72, 2001. (Cit. em 8.)