Sistema de Navegação para Robot Móvel com Filtros de Kalman

David José Ferreira Vaz

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientadores: Prof. José António Beltran Gerald
Prof. António Joaquim dos Santos Romão Serralheiro

Júri

Presidente: Prof. Gonçalo Nuno Gomes Tavares
Orientador: Prof. José António Beltran Gerald
Vogais: Prof. Fernando José Vieira do Coito

Novembro 2016

Agradecimentos

A elaboração desta Tese de Mestrado contou com imensos apoios e incentivos sem os quais nunca se teria tornado realidade e aos quais dirijo os meus sinceros agradecimentos.

Aos Professores orientadores António Serralheiro e José Gerald pelo seu apoio, disponibilidade e paciência, sugestões e críticas, transmissão do saber e colaboração na resolução dos problemas que foram surgindo no desenvolvimento deste trabalho.

Aos meus camaradas e amigos pelas palavras de incentivo e sugestões que tanta importância tiveram no ultrapassar dos pequenos obstáculos que sempre foram aparecendo.

À Academia Militar e Instituto Superior Técnico pela disponibilização do robot, material e condições necessárias para o desenvolvimento do trabalho.

A todos os professores que ao longo deste curso me foram transmitindo todo o conhecimento indispensável e me indicaram todo um conjunto de ferramentas necessárias para a compreensão e desenvolvimento do conhecimento relacionado com a Engenharia Electrotécnica. Incluo também todos os funcionários e pessoas que trabalham todos os dias para que este sistema funcione e que, apesar de não terem estudado todas estas fórmulas, nunca desistem de ajudar e acabam por se mostrar tão sábios em questões que não se encontram nos livros.

E por último mas não menos importante, agradeço aos meus pais, irmão e familiares pelo constante apoio, ajuda e motivação. Por acreditarem sempre em mim e por todos os seus ensinamentos de vida a eles dedico este trabalho esperando um dia poder retribuir tamanha consideração.

Resumo

O estudo realizado no âmbito desta Tese de Mestrado centra-se no desenvolvimento de um sistema de navegação para um robot móvel terrestre para que este seja capaz de seguir de forma autónoma um percurso predefinido.

Este trabalho reúne informação relativa ao Estado da Arte na temática dos filtros de Kalman e sistemas de navegação, são descritos os algoritmos desenvolvidos e métodos de implementação e são apresentados os resultados relevantes da aplicação deste sistema. Este controlo utiliza os sensores deste robot (*IMU*, GPS e odometria) e processa a informação em *Matlab* e C# num PC, utilizando redes *Wireless* para comunicação.

A informação dos sensores odométricos é combinada para estimativa da posição instantânea do robot e é determinado o erro relativamente à posição obtida pelo GPS. Este erro é filtrado com filtros de Kalman lineares e reintroduzido na estimativa de posição. É assim possível estabelecer um equilíbrio entre os dados de várias fontes (com os erros e ruído associados a cada uma) e aproveitar as potencialidades de previsão dos filtros de Kalman. O objetivo é realizar uma estimativa discreta ótima da posição instantânea do robot. Esta posição é depois utilizada na navegação automática do robot, que se pretende o mais certa possível. Esta navegação baseia-se num programa de demonstração fornecido com o robot mas é descrita e ajustada para melhorar o processo.

Testes de avaliação que ilustram a capacidade do robot percorrer automaticamente um percurso predefinido são apresentados confirmando a validade do trabalho e o funcionamento dos algoritmos.

Palavras-chave: navegação autónoma; filtro de Kalman; estimativa de posição; filtragem de ruído; sensores de movimento.

Abstract

The study in this Master's Thesis focuses on the development of a navigation system for a terrestrial mobile robot for it to be able to move autonomously through a default route.

This work gathers information about the State of the Art of Kalman filters and navigation systems. The developed algorithms and implementation methods are described and the relevant application results of this system are presented. This control uses the sensors in the robot (*IMU*, GPS and odometry) and performs information processing in *Matlab* and C# on a PC using *Wireless* networks for communication.

The information from the odometric sensors is combined to estimate the actual position of the robot and it is determined the error between this position and the GPS position. This error is filtered with linear Kalman filters and is returned to the estimated position. It is thus possible to balance the data from various sources (with the errors and noises associated to each one) and take advantage of the predictive capabilities of the Kalman filter. The objective is to obtain an optimal discreet estimation of the robot's actual position. This position is then used for the automatic navigation of the robot, which is intended to be as correct as possible. This navigation is based on a demonstration program provided with the robot but is described and adjusted to improve the process.

Evaluation tests which illustrate the robot's ability to automatically travel through a predefined path are presented confirming the work's validity and the operation of the algorithms.

Keywords - autonomous navigation; Kalman filters; estimated position; noise filtering; motion sensors.

Conteúdo

Lista de Figuras

Lista de Acrónimos

API, Application Programming Interface
ASCII, American Standard Code for Information Interchange
ASIMO, Advanced Step in Innovative Mobility
DLL, Dynamic-link library
GPS, Global Positioning System
IMU, Inertial Measurement Unit
INS, Inertial Navigation System
IP, Internet Protocol
LAN, Local Area Network
NMEA, National Marine Electronics Association
TCP, Transmission Control Protocol

Capítulo 1

Introdução

A importância do controlo autónomo prende-se com a crescente necessidade de se criarem sistemas capazes de realizar, de forma não intervencionada, tarefas que, de outra forma, seriam de realização inviável ou impossível para o ser humano.

Podem enumerar-se algumas dessas tarefas, sendo de referir, por exemplo, tarefas muito repetitivas, desgastantes ou perigosas, como a montagem de componentes mecânicos, a exploração de locais remotos ou perigosos (grutas, solo oceânico, interior de vulcões, etc), desativação de engenhos explosivos ou exploração de locais militarmente hostis.

Estes sistemas podem estar ainda dependentes de um controlo humano relativamente frequente, no entanto, pretende-se que esse controlo seja cada vez menor, ou seja, que um sistema consiga realizar mais atividades com menor intervenção humana. Assim, as ordens que lhes são atribuídas devem ser cada vez mais gerais e os sistemas devem realizar mais tarefas automaticamente para cumprirem o objetivo pretendido.

1.1 Motivação e definição do problema

Os sistemas de navegação autónoma são um assunto importante para a comunidade cientifica, e o seu desenvolvimento é facilmente percetível com a divulgação das novas tecnologias.

A principal questão desta temática consiste no desenvolvimento de sistemas de controlo capazes de corresponder às necessidades das atividades para as quais estão destinados. Por vezes esta dificuldade passa por problemas de integração do *software* com o *hardware*, a complexidade das atividades a que se destinam ou a dificuldade em adquirir e processar os dados que estão ao dispor do sistema e que são importantes para avaliar o seu estado. Estes dados poderão ser de difícil aquisição ou poderão não ter qualidade suficiente para serem utilizáveis, sobretudo devido a limitações dos sensores, inadequada capacidade de processamento e presença de ruído.

No âmbito do desenvolvimento da robótica autónoma, podem enumerar-se alguns exemplos de sistemas que espelham o sucesso da ciência nesta área. Assim, refere-se como exemplo o robot militar *BigDog* da *Boston Dynamics* (BostonDynamics, 2013), o *Motobot* da *Yamaha* (Yamaha, 2015) e o *ASIMO* da *Honda* (Honda, 2015), ilustrados na Figura 1.1.

Figura 1.1: Exemplos de robots autónomos: BigDog(BostonDynamics, 2013); Motobot(Yamaha, 2015); ASIMO(Honda, 2015).

1.2 Objetivos

Este trabalho é realizado no âmbito da Dissertação para obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores, sob a orientação dos professores José António Beltran Gerald e António Joaquim dos Santos Romão Serralheiro e pretende o desenvolvimento de um sistema de navegação automática para um robot móvel terrestre. Este sistema deve utilizar as capacidades de orientação e aquisição de dados do robot para determinar a sua posição e fazer que se desloque ao longo de uma trajetória previamente definida com o menor erro/desvio possível.

Desta forma, pretende-se a aquisição da informação disponibilizada pelos vários sensores do robot (acelerómetro, giroscópio, recetor GPS, bússola magnética, etc), a sua filtragem e processamento e o envio dos novos dados ao robot para que possa desenvolver a sua atividade de forma autónoma.

Uma vez que à informação dos sensores estará associada uma inevitável quantidade de ruído, este sistema de controlo tem por base a filtragem de informação com recurso a filtros de Kalman, de forma a reduzir os erros na navegação do robot. O desenvolvimento desta filtragem constitui o principal foco da dissertação a realizar.

1.3 Organização do relatório

No Capítulo 2 é feita uma definição inicial do problema referindo-se o enquadramento do tema na área científica e a revisão do Estado da Arte. Como o algoritmo de navegação desenvolvido se relaciona significativamente com a utilização de filtros de Kalman, é sumariamente descrito este filtro, bem como algumas aplicações em algoritmos de navegação. Para isto, referencia-se a aplicação dos filtros em diversos projetos académicos. Seguidamente, no Capítulo 3 procede-se à explicação do trabalho realizado para o alcance dos objetivos propostos e, no Capítulo 4 são apresentados e descritos alguns resultados obtidos com a aplicação do trabalho desenvolvido. Finalmente fazem-se as Conclusões da dissertação (Capítulo 5) seguidas da Bibliografia e Apêndices.

Capítulo 2

Enquadramento do tema na área científica e revisão do Estado da Arte

Tipicamente, um sistema físico é guiado por informação que lhe é atribuída na forma de parâmetros de entrada ou comandos. Após o processamento dessa informação o sistema desenvolve determinada ação, que é avaliada por sensores ou aparelhos de medida. Estas observações acumulam incertezas durante o processo, nomeadamente ruído e erros.

Naturalmente, surge a necessidade de, dados determinados critérios e com base na informação disponível, obter uma estimativa do estado do sistema que se aproxime, tanto quanto possível, do seu estado verdadeiro. Este objetivo é geralmente conseguido com recurso a filtros, cuja ideia é retirar ruído da informação de entrada, para que à saída se obtenha informação passível de ser corretamente utilizada e compreendida.

O conceito de navegação automática consiste na capacidade de um sistema se deslocar de forma não comandada através do espaço, sendo que, essa deslocação deve efetuar-se de forma a cumprir objetivos previamente definidos, e não de forma errática. Para isto, é necessário que se conheça o estado (posição, velocidade e direção) do sistema em dados instantes de tempo com o menor erro possível. Este conhecimento depende apenas da capacidade de perceção dos sensores do sistema do meio envolvente.

Após o conhecimento da posição instantânea, o sistema autónomo deve tentar cumprir uma trajetória previamente definida, desenvolvendo a sua atividade de forma a atingir o próximo ponto da trajetória. Para que esse deslocamento seja feito, é necessário um controlo que defina a forma como o sistema deve evoluir.

A sua evolução deve ser controlada tendo em conta os desvios que existem nos dados conhecidos. Assim, pretende-se neste trabalho, o tratamento destes dados recorrendo a filtros de Kalman, para se obter uma estimativa ótima do verdadeiro estado do sistema. A utilização deste tipo de filtragem justifica-se, sobretudo, pela sua simplicidade de implementação e reconhecido bom desempenho na previsão de estados e redução de ruído. A sua capacidade de determinar valores futuros faz dos filtros de Kalman a ferramenta ideal para utilização em sistemas de controlo em tempo real, evitando os atrasos característicos introduzidos por outros métodos de filtragem que são prejudiciais ao desempenho da navegação.

Na próxima secção abordam-se os pormenores mais relevantes destes filtros.

2.1 Filtros de Kalman

2.1.1 Generalidades

Importa sublinhar que a principal dificuldade no conhecimento do real estado do sistema deve-se ao erro associado à informação que é recebida, ou seja o estado medido. Essa informação apresenta um desvio relativamente ao seu estado verdadeiro. Este desvio pode estar relacionado com erros de sistema ou erros de medida, sendo os primeiros associados aos desvios do sistema relativamente aos comandos introduzidos, e os segundos relacionados com os desvios dos aparelhos de medida.

Na Figura 2.1 expõe-se a aplicação genérica do filtro de Kalman indicando graficamente o processo de controlo de um sistema com recurso a este filtro.

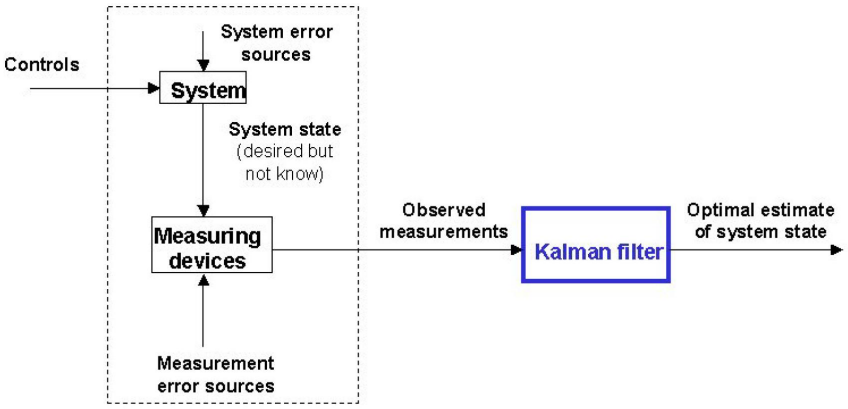


Figura 2.1: Aplicação genérica do filtro de Kalman (Ribeiro, 2004)

O Filtro de Kalman é uma ferramenta que permite estimar as variáveis de um conjunto de processos, ou seja, estima os estados de um sistema linear.

Este filtro destaca-se dos demais pelo facto de ser aquele que minimiza a variância do erro de estimativa e ser de implementação relativamente simples (Ribeiro, 2004) . É francamente utilizado em aplicações como o rastreio de objetos, economia, navegação ou aplicações computacionais como a fusão de dados de radares, *scanner* de laser e câmaras para medidas de velocidade e profundidade (Cornell, 2008).

A dinâmica dos filtros de Kalman resulta de consecutivos ciclos de previsão de estados e filtragem de informação. A dinâmica destes ciclos é baseada na estrutura da função densidade de probabilidade gaussiana.

2.1.2 Formulação

Considere-se, um sistema linear variável no tempo dado pelas seguintes equações (Ribeiro, 2004) (Maybeck, 1982)

$$x_{k+1} = A_k x_k + B_k u_k + G w_k \quad k \geqslant 0 \tag{2.1}$$

$$y_k = C_k x_k + v_k \tag{2.2}$$

onde, (2.1) e (2.2) são, respetivamente, as equações do estado verdadeiro e medido do sistema. Nestas equações, A, B, C e G são matrizes, x é o estado do sistema, u é um *input* conhecido, y é a medida obtida e k o índice de tempo. $x(k) \in \mathbb{R}^n$, $u(k) \in \mathbb{R}^m$, $w(k) \in \mathbb{R}^n$, $v(k) \in \mathbb{R}^r$, $y(k) \in \mathbb{R}^r$. w_k e v_k são o ruído de processo e de medida, respetivamente, e são ruídos gaussianos de média nula.

O vetor x contém toda a informação necessária para conhecer o presente estado do sistema, no entanto, não é possível medi-lo diretamente. Em vez disso, mede-se y que é uma função de x corrompida

pelo ruído v. É possível, então, utilizar y para medir x, tendo sempre presente que esta medida poderá não ter a precisão necessária devido ao ruído.

Destaca-se que o filtro de Kalman, em vez de propagar toda a função densidade de probabilidade condicional, considera apenas o estado previsto e a incerteza da estimativa, aproximando a verdadeira função a uma função densidade de probabilidade gaussiana, como indicado na Figura 2.2.

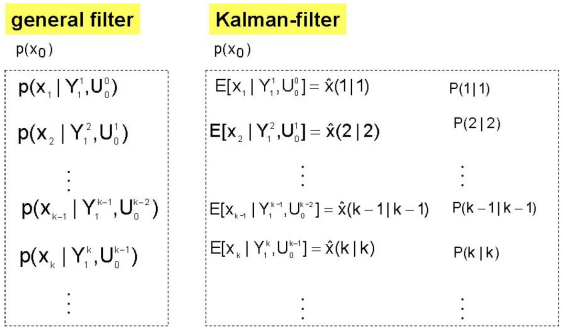


Figura 2.2: Propagação da função densidade de probabilidade no filtro genérico (esquerda) e no filtro de Kalman (direita) (Ribeiro, 2004)

A transição entre estados no filtro de Kalman passa por um ciclo de previsão e outro de filtragem, onde $p(x_{k+1}|Y_1^k, U_0^k)$ representa a informação relacionada com x_{k+1} antes de se conhecer a observação y_{k+1}. O ciclo de filtragem permite melhorar a informação em x_{k+1} após se conhecer y_{k+1}. Este ciclo pode observar-se na Figura 2.3

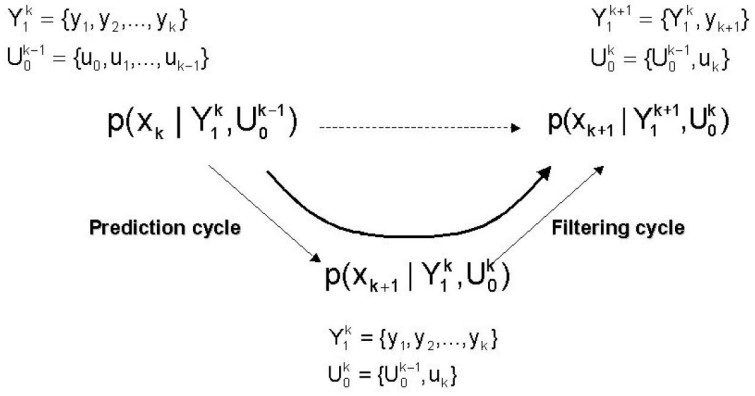


Figura 2.3: Ciclos de previsão e filtragem no filtro de Kalman (Ribeiro, 2004)

O Filtro de Kalman resume-se nas seguintes equações (Ribeiro, 2004):
- Previsão

$$\hat{x}(k+1|k) = A_k\hat{x}(k|k) + B_k u_k \tag{2.3}$$

$$P(k+1|k) = A_k P(k|k) A_k^T + G_k Q G_k^T \tag{2.4}$$

- Filtragem

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K(k)[y(k) - C_k\hat{x}_{k|k-1}] \tag{2.5}$$

$$K(k) = P(k|k-1)C_k^t[C_k P(k|k-1)C_k^T + R]^{-1} \tag{2.6}$$

$$P(k|k) = [I - K(k)C_k]P(k|k-1) \tag{2.7}$$

Nas equações tem-se,
- $\hat{x}$ é o estado estimado (contém as variáveis de estado)
- A é a matriz de transição de estados
- u são as variáveis de controlo (*inputs*)
- B é a matriz de controlo (controlo do efeito das variáveis de controlo nas variáveis de estado)
- P é a matriz de covariância que quantifica o erro da estimativa
- Q é a matriz de covariância que quantifica o erro de processo
- y é a medida obtida (contém variáveis de medida)
- C é a matriz de medida (controla o efeito das medidas na determinação do estado)
- K é o ganho de Kalman
- R é a matriz de variância das medidas.

Pode considerar-se que o filtro de Kalman estima um processo recorrendo a um controlo por *feedback*. O filtro estima o estado de um processo num momento e obtém informação na forma de medidas com ruído.

As equações para o filtro de Kalman dividem-se em dois grupos, sendo equações de atualização temporal e equações de atualização de medida (Welch & Bishop, 2006).

As equações de atualização temporal são responsáveis por fazer avançar no tempo o estado corrente do sistema e as estimativas da covariância do erro, para obter, estimativas *a priori* para o passo seguinte no tempo.

As equações de atualização de medida são responsáveis pelo *feedback*, isto é, por incluir a nova medida na estimativa *a priori* para obter uma estimativa *a posteriori* melhorada. As equações de atualização temporal podem ser vistas como equações de previsão, enquanto que as equações de atualização de medida podem ser vistas como equações de correção (Welch & Bishop, 2006).

Desta forma, o algoritmo final tem um caráter de algoritmo de previsão-correção para resolução de problemas numéricos, como ilustrado na Figura 2.4.

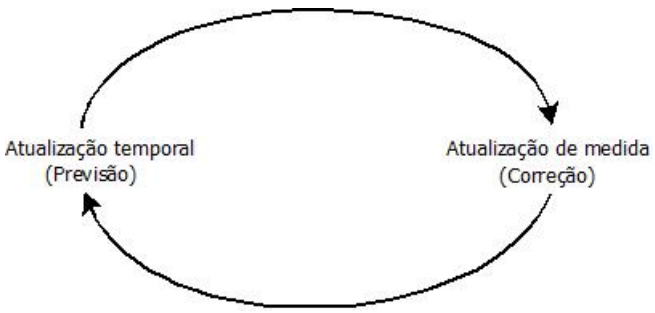


Figura 2.4: Processo de desenvolvimento dos ciclos do filtro de Kalman. A atualização temporal faz avançar no tempo a estimativa do atual estado. A atualização de medida ajusta a estimativa realizada recorrendo à medida obtida no instante seguinte. (Welch & Bishop, 2006)

2.2 Filtro de Kalman estendido

O filtro de Kalman abordado até ao momento deve ser utilizado em sistemas lineares, no entanto, pode ser necessário modelar sistemas não lineares, como é o caso do robot *Jaguar*. Estes sistemas descrevem-se em termos das equações de estado verdadeiro e medido da seguinte maneira:

$$x_{k+1} = f_k(x_k) + w_k \quad k \geqslant 0 \tag{2.8}$$

$$y_k = h_k(x_k) + v_k \tag{2.9}$$

O filtro de Kalman estendido é uma variação do filtro de Kalman descrito para sistemas não lineares. Este funciona através da linearização da dinâmica não linear dos estados e dos modelos de medida.

Neste filtro, as estimativas do erro da trajetória são utilizadas para atualizar a trajetória de referência com a evolução do tempo. Assim, podem filtrar-se atividades de maior duração temporal e complexidade. A distinção na filtragem de informação baseia-se na função de medida h_k, isto é, se é atualizada com base na trajetória correta (filtro de Kalman estendido) ou nominal (filtro de Kalman linear).

2.3 Utilização dos filtros de Kalman na navegação e localização

Para melhor enquadrar a temática dos filtros de Kalman, nesta secção são apresentados e descritos alguns exemplos de utilização destes filtros na prática, com o objetivo de solucionar problemas em sistemas reais.

Como referido, os filtros de Kalman têm o objetivo de tratar informação para que determinado sistema funcione da forma mais correta possível. O problema da utilização da informação tal como é disponibilizada pela sua fonte (sensores) é que a presença de ruído e erros pode levar a que esses sistemas não funcionem corretamente e pode impossibilitar a realização de qualquer tarefa. Eventualmente, a utilização de informação ruidosa e errática pode mesmo levar à destruição do *hardware* em utilização.

O erro de uma trajetória baseada numa navegação sem pontos de referência (*dead-reckoning*), que assenta apenas nos sensores internos de um sistema, como odómetros, aumenta com o tempo e distância. Desta forma, é conveniente o recurso a sensores externos para reconhecimento da posição de um robot num determinado espaço, compensando o erro da navegação sem pontos de referência. Bússolas, e sensores ultrassónicos (sensores de distância, rádio, *GPS - Global Positioning System*, etc) são exemplos de sensores externos que podem ser utilizados.

Os métodos de auto-localização usando sensores externos podem dividir-se em dois grupos (Yi & Choi, 2007): métodos locais e métodos globais. Nos métodos locais, o robot cria um mapa local com base na distância relativa aos objetos existentes no meio e faz corresponder este mapa local com um mapa global previamente guardado. Desta forma, o robot descobre a sua posição no espaço. Relativamente aos métodos globais, o robot calcula a sua posição diretamente nas coordenadas do espaço em que se insere, utilizando a distância a determinados pontos de referência.

Os métodos locais têm a vantagem de permitirem movimento anti-colisão e reconstrução de mapas para ambientes transformados através da utilização de sensores de distância e do sistema de auto-localização. No entanto, estes métodos exigem grande processamento devido à criação de mapas locais e aos processos de comparação com os mapas globais guardados no sistema. Por vezes, é também necessário interromper o movimento dos robots para obtenção da informação local, de forma a obter uma observação correta do ambiente. (Leonard & Durrant-Whyte, 1992) (Ko *et al.* , 1996) .

Relativamente aos métodos globais podem evitar-se a criação de mapas locais e os processos de comparação com o mapa global, para além de que a auto-localização é computacionalmente eficiente e rápida (Leonard & Durrant-Whyte, 1991) (Hernández *et al.* , 2003) .

Um exemplo onde a filtragem da informação é importante são os satélites artificiais, que requerem um sistema de navegação fiável, seguro e completamente autónomo, sobretudo em caso de emergência no espaço.

A navegação celestial é um método de navegação totalmente autónomo para utilização em satélites. Para satélites em órbita próxima à terra (*low Earth orbit - LEO*), a direção da terra é a medida mais importante e a precisão da deteção do horizonte é determinante para a precisão da navegação celestial.

A aquisição da medida do horizonte pode ser efetuada diretamente através de sensores ou indiretamente através da observação de luz refratada pela atmosfera proveniente dos corpos celestes. Uma vez que existe complementaridade entre estes métodos, podem ser utilizados filtros de Kalman para fundir informação, o que permite uma melhoria do desempenho e da fiabilidade do sistema de navegação, como proposto no trabalho de Xiaolin Ning e Jiancheng Fang (Ning & Fang, 2008). Neste caso, utilizam-se, maioritariamente, métodos de localização com recurso a sensores externos, em vez dos dados odométricos.

Sistema de navegação autónoma para robots móveis usando reconhecimento baseado na visão (Siegwart & Nourbakhsh, 2004)

Outro exemplo que utiliza os filtros de Kalman para navegação é o robot *Pygmalion* da *École Polytechnique Fédérale de Lausanne*. Este robot, representado na Figura 2.5, tem tração diferencial sendo impulsionado por motores separados na rodas da frente e usa um laser telemétrico como sensor primário. A representação ambiental realizada pelo robot é contínua e abstrata: o mapa consiste num conjunto de linhas infinitas que descrevem o ambiente. O seu estado estimado é representado por uma distribuição gaussiana, uma vez que utiliza o filtro de Kalman no algoritmo de localização. O valor da sua posição média é representado com um elevado nível de precisão, possibilitando obter a sua localização com um erro reduzido. Este sistema utiliza um método de auto-localização local e pode definir-se como um sistema de navegação autónoma para robots móveis usando reconhecimento baseado na visão.(Siegwart & Nourbakhsh, 2004)

Figura 2.5: Robot Pygmalion(Brega et al. *, 2000)*

De seguida referem-se sumariamente os 5 passos do funcionamento do algoritmo de localização do robot.(Siegwart & Nourbakhsh, 2004)

1. Estimativa da posição do robot: Conhecendo a posição inicial e o "*input*"dado aos motores do robot, é possível estimar a próxima posição e orientação.

2. Observação: O robot utiliza o laser telemétrico para obter informação acerca das superfícies que deteta. Essa informação consiste num conjunto de pontos que o robot utiliza para construir linhas que definem o espaço envolvente. É criada, assim, uma representação da sua localização no espaço.

3. Previsão da medida: Com base num mapa do espaço envolvente guardado previamente e a posição estimada, é gerada uma previsão da atual localização e orientação do robot nesse mapa.

4. Correspondência: Neste passo o algoritmo estabelece uma relação entre a representação observada e prevista.

5. Estimativa: Com aplicação do filtro de Kalman à informação do passo anterior e à posição estimada inicialmente é possível obter uma estimativa melhorada da posição atual e da sua orientação.

Desta forma, pode então resumir-se que o filtro de Kalman promove a determinação dos estados do sistema com precisão cruzando os dados obtidos pelos sensores do robot. Este método permite cruzar os dados resultantes do sensor telemétrico e compará-los com a posição prevista a partir dos dados odométricos. É, no entanto, necessário que o sistema conheça previamente o mapa do espaço.

Sistema de navegação autónoma para robots móveis com estimativa de posição recorrendo a triangulação de sinais rádio (Melo *et al.* , 2013)

Pode considerar-se outro caso de implementação de filtros de Kalman, agora como sistema de navegação autónoma para robots com estimativa de posição recorrendo a triangulação de sinais rádio.

Este método consiste na utilização de sinais de radiofrequência, para além dos restantes sensores como bússola e sensores de odometria, de forma a que o robot percecione a sua posição no espaço. Para isso é realizada triangulação com várias balizas distribuídas pelo espaço que têm emissores-recetores rádio com assinaturas de sinal diferentes. Será necessário, no entanto que seja conhecido o número de balizas e a sua localização no espaço. Pode observar-se na Figura 2.6 um esquema de funcionamento deste sistema. Esta configuração é um exemplo de um método de auto-localização global, em que são utilizados pontos de referência (balizas) para que o robot se situe no espaço.

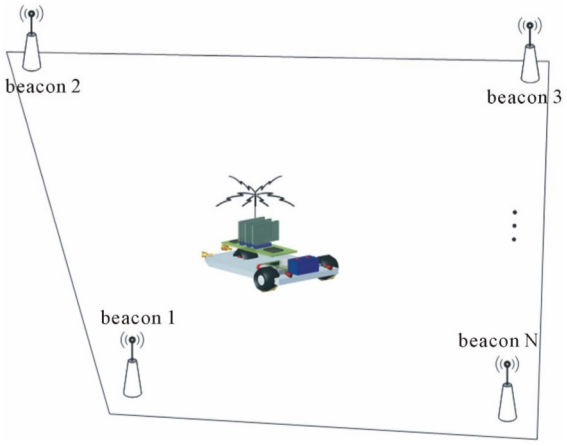


Figura 2.6: Exemplo da distribuição de balizas (beacon) e sistema de comunicação.(Melo et al. *, 2013)*

A sequência das ações realizadas pelo algoritmo descreve-se de seguida (Melo *et al.* , 2013):

1. O robot emite um sinal codificado à primeira baliza e inicia, simultaneamente, uma contagem temporal. Fica depois à espera uma resposta do emissor rádio

2. Se receber um sinal de volta com a mesma codificação, percebe-se que a baliza reconheceu o sinal e respondeu. Neste instante, o robot para o contador temporal e calcula o atraso da resposta.

3. Se não for recebido nenhum sinal considera-se que ocorreu algum erro na transmissão ou que a baliza está fora de alcance.

4. Repetição do ciclo para a baliza seguinte.

A distância entre o robot e a baliza é calculada com base no tempo de atraso entre os sinais com a mesma codificação, tendo em conta que o sinal percorre 1 metro em 3.3 ns.

Na Figura 2.7 pode verificar-se um esquema com a planta de um espaço com a localização da balizas com os emissores-recetores rádio, a trajetória do robot e a sua estimativa da posição.

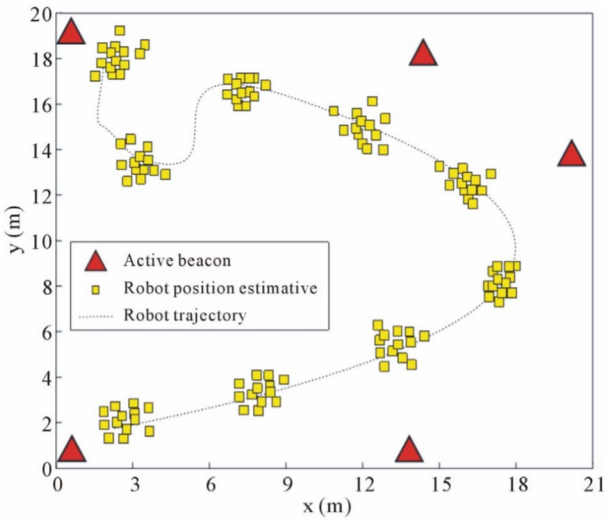


Figura 2.7: Estimativa da posição do robot pela triangulação com as balizas. (Melo et al. *, 2013)*

Utilizando a planta do espaço e calculando as estimativas dos estados do sistema (recurso ao filtro de Kalman) no instante $k+1$, com base na atual posição do robot, é possível ter conhecimento acerca da evolução do sistema com os diferentes comandos que lhe são enviados. Na Figura 2.8 ilustra-se um esquema onde se observa o funcionamento geral do sistema de navegação para este robot. Pode verificar-se que são recolhidos dados pelos sensores relativos à odometria, bússola e triangulação rádio (*perception*), que são depois comparados (*matching*) com as previsões de observação efetuadas (*predicted observations*). Com esta informação é realizada estimativa da posição seguinte (*estimation/pose estimative*) que é usada para estimar as próxima observação que será comparada com a observação no instante de tempo seguinte.

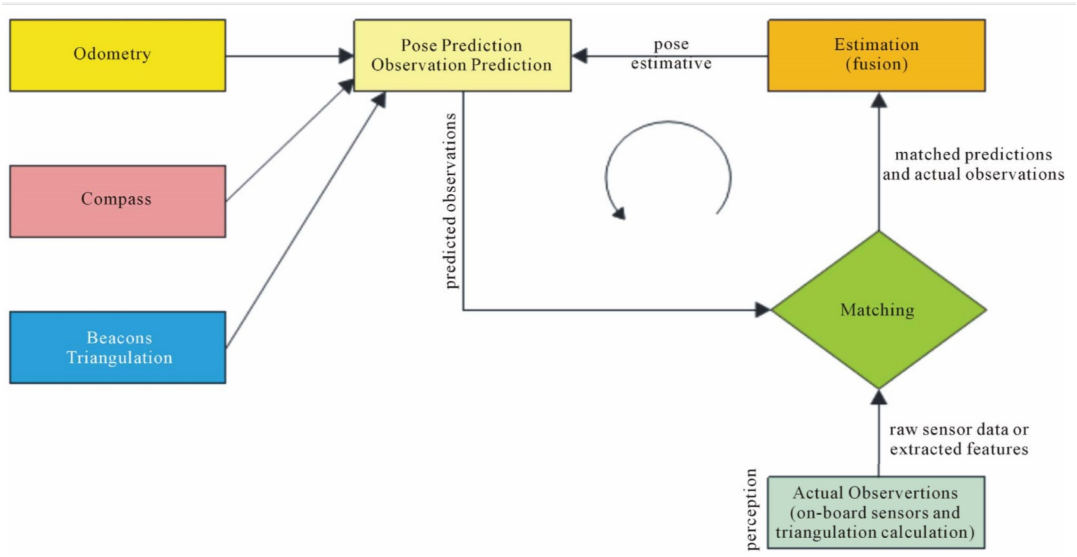


Figura 2.8: Esquema do algoritmo para a localização do robot com recurso ao filtro de Kalman.(Melo et al. *, 2013)*

Sistema de navegação autónoma para robots móveis com recurso a um sistema ultrassónico global (Yi & Choi, 2007)

Este exemplo apresenta semelhanças com o anterior e aproxima-se da forma de utilização do robot usado para a tese de mestrado a desenvolver, uma vez que utiliza um conjunto de quatro ou mais sensores ultrassónicos fixos em posições de referência que representam coordenadas *pseudo*-globais (semelhança com sistema *GPS*), como se pode visualizar na Figura 2.9. Neste caso as coordenadas são relativas a um espaço definido no interior de um edifício, mas o sistema a utilizar na tese de mestrado utiliza (entre outros) o sistema *GPS* como forma de obtenção de informação de localização exterior.

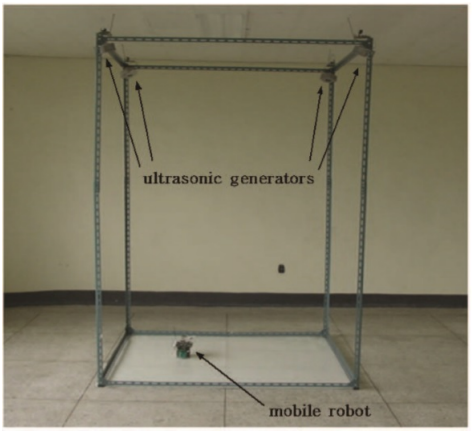
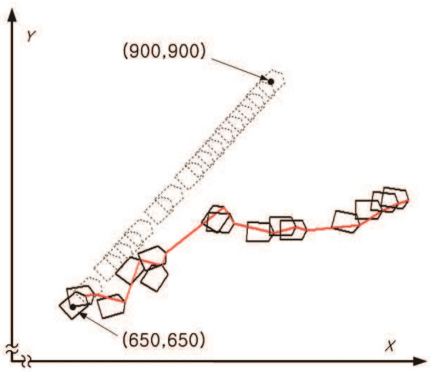


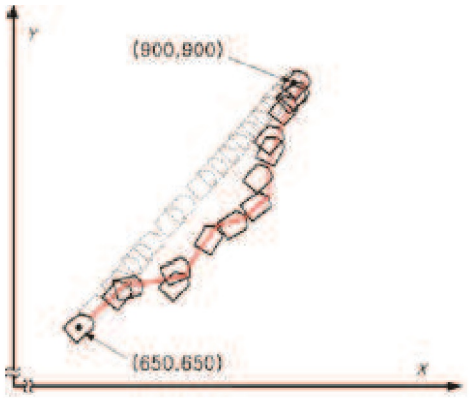
Figura 2.9: Esquema do sistema de navegação autónoma com recurso a um sistema ultrassónico global.(Yi & Choi, 2007)

Desta forma, no trabalho de Soo-Yeong Yi e Byoung-Wook Choi (Yi & Choi, 2007), com base nas medidas de distância entre os emissores ultrassónicos e os recetores no robot e nos algoritmos de fusão de dados de distâncias, é possível determinar a posição do robot no sistema de coordenadas *pseudo*-globais. Tal como no exemplo anterior, a distância entre o robot e os emissores é medida com base no tempo de atraso entre o sinal emitido e recebido entre o robot e o emissor.

A aplicação do filtro de Kalman estendido aos resultados obtidos pela odometria e pelos sensores ultrassónicos permite o aumento da precisão da localização determinada. Na Figura 2.10 apresentam-se alguns resultados deste trabalho que evidenciam a importância da fusão de dados de um sistema de localização externo e interno através do filtro de Kalman.



(a) Auto-localização com recurso aos dados odométricos



(b) Auto-localização com recurso à fusão dados odométricos e dos sensores ultrassónicos através do filtro de Kalman

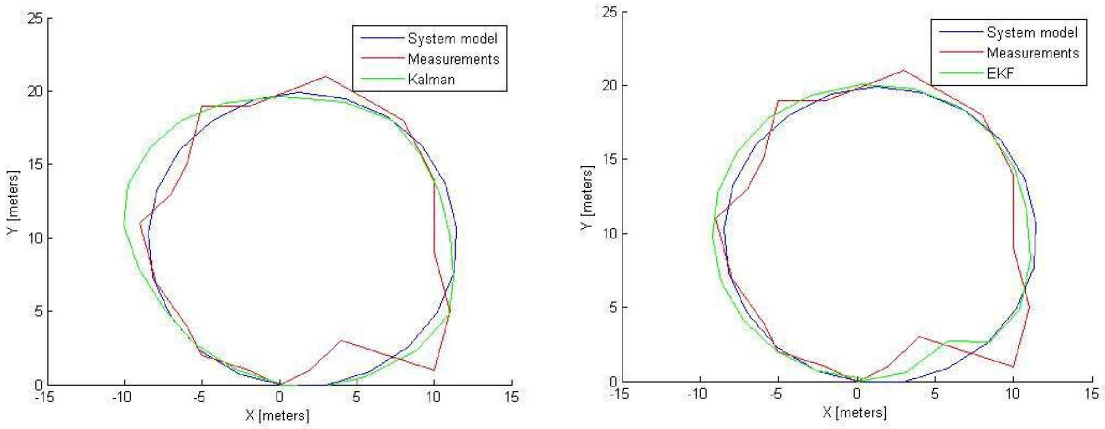
Figura 2.10: Trajetórias seguidas pelo robot considerando diferentes formas de auto-localização (Yi & Choi, 2007)

Foram, até este ponto, apresentados alguns exemplos de aplicação do filtro de Kalman na navegação autónoma de robots. No entanto, para ser possível realizar uma boa navegação, é necessário que o robot conheça a sua localização.

Como referido no subcapítulo 2.2, o filtro de Kalman estendido é uma variação do filtro de Kalman linear para sistemas não lineares que funciona com base na linearização da dinâmica não linear dos estados e dos modelos de medida. Uma das aplicações mais importantes do filtro de Kalman estendido é determinação da localização e a construção de mapas em simultâneo (*SLAM - simultaneous localization and map building*). De seguida apresentam-se alguns exemplos de trabalhos e resultados obtidos na área da localização com recurso a filtros de Kalman.

No trabalho de Suliman, Cruceru e Moldoveanu (Suliman *et al.* , 2009), é descrito um método de localização para robots móveis utilizando filtro da Kalman. Neste trabalho são aplicados o filtro de Kalman linear e estendido e comparam-se os resultados. O objetivo é a obtenção dos valores das coordenadas x e y da posição do robot e a sua orientação.

Os filtros são implementados em *Matlab* e é considerado um caso simples de um robot que segue um percurso definido. Na Figura 2.11 está representado o percurso estimado (Kalman/EFK) comparado com o percurso real (*System model*) para o filtro de Kalman linear à esquerda e estendido à direita. Pode verificar-se que ambos os filtros têm um desempenho muito bom, sendo que, o filtro estendido se aproximou mais do percurso real. Verifica-se também que o filtro de Kalman linear tem tendência para perder precisão com o decorrer do tempo relativamente ao filtro estendido.



(a) Filtro de Kalman linear *(b) Filtro de Kalman estendido*

Figura 2.11: Trajetórias reais, estimadas e medidas obtidas com os dois tipos de filtro de Kalman (Suliman et al. *, 2009)*

No trabalho de Stergios I. Roumeliotis e George A. Bekey (Roumeliotis & Bekey, 1997), é comparado um sistema de localização sem pontos de referência (*dead-reckoning*) sem filtragem de dados com outro em que esses dados são processados com o filtro de Kalman estendido. Desta forma, no primeiro sistema, os dados dos sensores são aplicados ao modelo cinemático do robot e são diretamente utilizados para determinação da localização. Já no modelo com filtragem, essa informação obtém-se após processamento pelo filtro de Kalman estendido.

É utilizado um robot experimental de duas rodas dianteiras motrizes e uma traseira direcional, usado num programa de exploração espacial, e refere-se no artigo um aumento de performance até 40% relativo ao erro de posição com utilização do filtro de Kalman. Os sensores utilizados são sensores de movimento nas duas rodas dianteiras, potenciómetro para o ângulo de direção na roda traseira e giroscópio que mede a velocidade angular do veículo no eixo perpendicular ao plano do movimento.

Num terceiro teste, é incluído um sensor solar como sensor de localização global, que permite o aumento considerável da precisão da localização determinada. A escolha do sensor solar deve-se ao

facto de no espaço, não ser possível a utilização de sistemas como *GPS*, não existir campo magnético impedindo a utilização de bússolas nem existirem mapas suficientemente detalhados que possibilitem a deteção de pontos de referência para comparação com dados telemétricos ou pela visão. Tendo em conta estas condicionantes utiliza-se o sol como ponto de referência.

As equações 2.10, 2.11 e 2.12 definem o modelo cinemático utilizado para definir o movimento do robot,

$$x_{k+1} = x_k - V\Delta T sin(\phi_k + \theta) \tag{2.10}$$

$$y_{k+1} = x_k + V\Delta T cos(\phi_k + \theta) \tag{2.11}$$

$$\phi_{k+1} = \phi_k - \frac{V\Delta T}{b} tan(\theta) \tag{2.12}$$

onde (x_{k+1}, y_{k+1}) e (x_k, y_k) são o par de coordenadas das posições atual e anterior, respetivamente, ϕ_{k+1} e ϕ_k são as orientações atual e anterior, respetivamente, V é a velocidade, ΔT o incremento de tempo, θ o ângulo da direção e b a distância entre eixos das rodas.

A técnica de localização sem pontos de referência sem filtragem usa o modelo cinemático do sistema, que recebe as medidas obtidas, e calcula as mudanças de posição e operação do veículo. Este processamento é computacionalmente mais simples que a aplicação do filtro de Kalman estendido, mas, como seria de esperar, a incerteza associada aumenta mais rapidamente que no método com filtragem, isto porque o filtro suaviza os efeitos dos erros e desvios. O filtro de Kalman trata o modelo cinemático como o estado do sistema e desenvolve o processamento de previsão-correção associado, assim, as variáveis consideradas inicialmente no modelo cinemático (x, y e ϕ) em conjunto com a velocidade (V) e o ângulo de direção (θ) são as que compõem o estado $\mathbf{x}$ da equação 2.8 na secção 3.3.6.

Verificou-se, no entanto, que este conjunto de variáveis não era suficientemente sensível às variações de orientação do veículo, o que afeta consideravelmente a navegação porque o erro de orientação provoca um crescimento constante do erro de posição após a sua ocorrência. Para corrigir este problema acrescentaram-se variáveis ao vetor de estado $\mathbf{x}$. Adicionaram-se então a velocidade angular ($\dot{\phi}$), a aceleração angular ($\ddot{\phi}$) e a aceleração do veículo ($\dot{V}$). O vetor de estado final é, então, $\mathbf{x} = [x, y, \phi, \dot{\phi}, \ddot{\phi}, V, \dot{V}, \theta]^T$

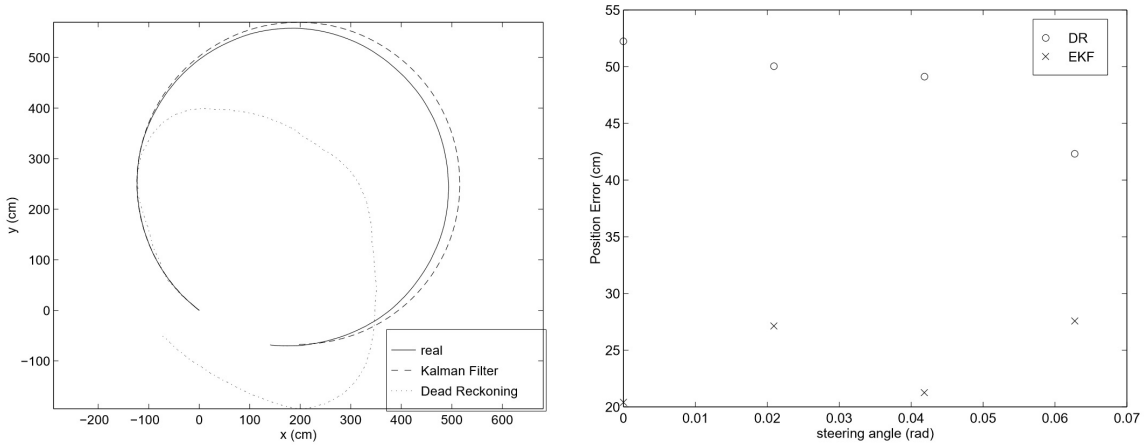
Relativamente ao vetor de medidas $\mathbf{y}$, é composto pelas seguintes quantidades medidas: V_1, V_2 (velocidades das rodas determinada pelos sensores de movimento), $\dot{\phi}$ (velocidade angular dada pelo giroscópio), e θ (ângulo da roda direciona dado pelo potenciómetro). Assim, o vetor de medidas é $\mathbf{y} = [V_1, V_2, \dot{\phi}, \theta]^T$

Para avaliar as diferenças de desempenho entre os métodos de localização com e sem filtragem, foram realizados testes cujos resultados se apresentam na Figura 2.12.

Na Figura 2.12a ilustra-se um exemplo de uma trajetória percorrida (com distância total de 18 m) recorrendo aos dois métodos referidos e na Figura 2.12b demonstra-se o erro médio absoluto na posição após 10 tentativas para diferentes ângulos de direção. Verificou-se que a localização determinada com recurso ao filtro de Kalman era muito mais precisa e o erro de orientação muito menor.

Após as análises sem recurso a pontos de referência testou-se a influência do sensor solar no sistema de localização, desta vez recorrendo sempre ao filtro de Kalman estendido. O resultado pode verificar-se na Figura 2.13, onde se observa a trajetória real para um determinado percurso em comparação com a estimativa que utiliza o sensor solar e a que não utiliza. Conclui-se, com o gráfico, que a utilização do sensor solar melhora consideravelmente a navegação, sobretudo com o aumento da distância, diminuindo o erro de posição acumulado.

Outro trabalho interessante no campo da localização e navegação autónoma é o de Motilal Agrawal, Kurt Konolige e Robert C. Bolles (Agrawal *et al.* , 2007) que utiliza sensores de visão para navegação autónoma em terrenos exteriores. O objetivo é que um robot consiga percorrer um espaço até alcançar determinado ponto construindo simultaneamente um mapa desse espaço. Desta forma, o mais desafiante será a construção correta de mapas enquanto o robot se mantém bem localizado no percurso até ao objetivo. Para isto, são implementados algoritmos de visão estéreo para construção dos mapas e obtenção de odometria visual para localização.



(a) Exemplo de trajetória (18 m de distância) *(b) erro de posição para diferentes ângulos de direção*

Figura 2.12: Comparação da navegação sem e com filtro de Kalman estendido (Roumeliotis & Bekey, 1997)

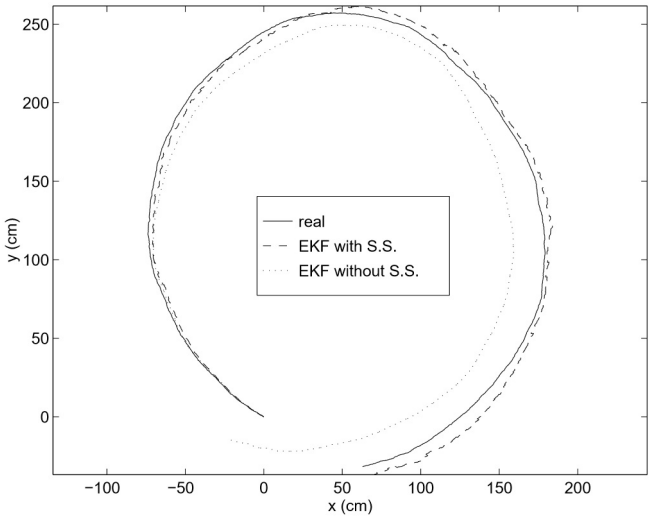


Figura 2.13: Comparação da trajetória entre o método de localização com e sem o sensor solar (Roumeliotis & Bekey, 1997)

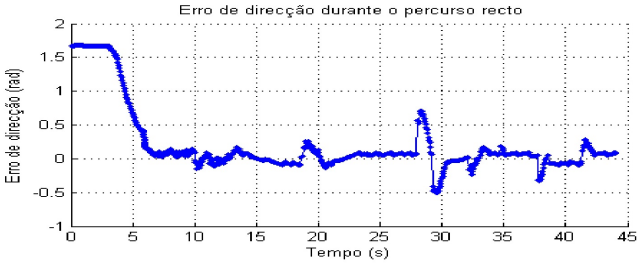
Apesar de, neste trabalho não se utilizarem filtros de Kalman, é interessante perceber que a aquisição de dados visuais permite também localizar objetos no espaço. No entanto, como referido no artigo, é necessário o recurso a algoritmos eficientes e precisos, cujo desenvolvimento é complexo. Isto deve-se ao facto de o registo do movimento do robot e os mapas de obstáculos são criados com base nas imagens recebidas. Estes dados podem, no entanto, ser integrados com outros de fontes variadas e fundidos com recurso a filtros de Kalman.

O trabalho desenvolvido com o robot *Pygmalion* difere deste relativamente ao tipo de imagens captadas e à forma de obtenção dos dados de odometria. No primeiro as imagens captadas são dados telemétricos do ambiente e a odometria é obtida através dos comandos dados aos motores. Neste trabalho as imagens captadas são imagens estéreo que permitem a definição do relevo do ambiente e a odometria é calculada com base nestas imagens.

Por fim refere-se também a dissertação realizada por Filipe João (João, 2013), cujo objetivo foi o desenvolvimento de um sistema de navegação autónoma também para o robot *Jaguar*, recorrendo ao filtro de Kalman linear para filtragem direta dos dados recebidos do GPS e acelerómetro. Nas Figuras 2.14, 2.15 e 2.16 podem observar-se o resultado de alguns testes de navegação e respetivo desvio em relação à

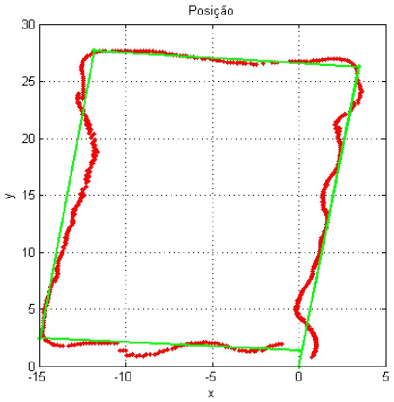
trajetória pretendida.

(a) Trajetória desejada (verde) e trajetória seguida (vermelho)

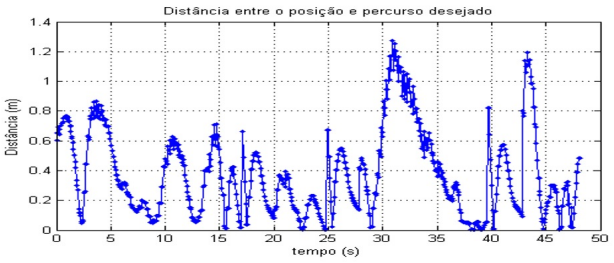


(b) Erro de direção durante o percurso

Figura 2.14: Testes realizados à navegação do robot Jaguar com filtro de Kalman linear num percurso retilíneo de 35m aprox. (João, 2013)

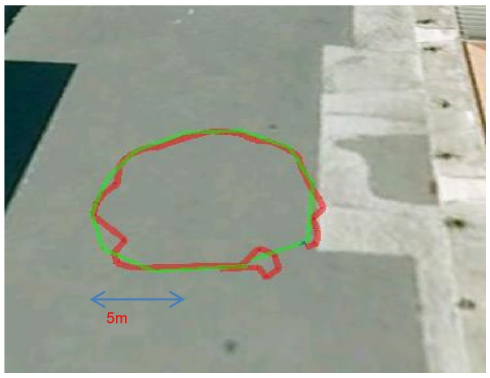


(a) Trajetória desejada (verde) e trajetória seguida (vermelho)

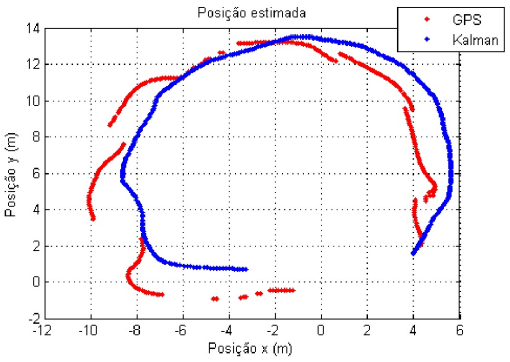


(b) Erro de posição durante o percurso

Figura 2.15: Testes realizados à navegação do robot Jaguar com filtro de Kalman linear num percurso retangular (15x25m)(João, 2013)



(a) Trajetória desejada (verde) e trajetória seguida (vermelho)



(b) Comparação da posição determinada pelo GPS (vermelho) e algoritmo (azul)

Figura 2.16: Testes realizados à navegação do robot Jaguar com filtro de Kalman linear num percurso circular (aprox. 5m de raio)(João, 2013)

2.4 Robot *Jaguar 4×4 Wheel*

O robot a utilizar é o robot *Jaguar 4x4 Wheel* fabricado pela empresa canadiana *Dr. Robot*, ilustrado na Figura 2.17. Este robot é destinado a operar dentro ou fora de edifícios, em terreno acidentado, em atividades que requeiram uma rápida operação e controlo.

Figura 2.17: Robot Jaguar 4x4 Wheel. (DrRobotInc, 2016)

Está equipado com um motor de 80W para cada uma das quatro rodas, atingindo uma velocidade máxima de 11km/h. Apresenta uma construção robusta, pesa cerca de 20kg e é resistente à água. A comunicação com o robot é efetuada via *wireless* (norma IEEE802.11G). O aparelho é originalmente alimentado por uma bateria de Polímero de Lítio (LiPo) de 22,2V com uma capacidade de 10Ah, que permite a sua operação durante cerca de 2 horas.

Integra um módulo GPS e um conjunto de sensores (giroscópio, acelerómetros e bússola) que permitem efetuar navegação autónoma. Realiza também a captura de vídeo e som e, apesar de existir um programa para controlo, é possível o desenvolvimento de software para essa finalidade.

Para obtenção da completa informação acerca do robot sugere-se a consulta do respetivo manual (DrRobotInc, 2014).

A interação do robot com o computador é necessária porque é no computador que se realiza todo o processamento de informação relativa à trajetória do robot. Desta forma, o robot deve, a cada instante de tempo, enviar a informação obtida nos seus sensores para o computador e este realizará o seu processamento. Depois são enviadas ao robot novas ordens para que este siga a trajetória pretendida. No computador existe uma plataforma de comunicação com o robot que promove a troca de informação entre ambos, e é necessário recorrer a um programa de cálculo que faça o tratamento dos dados recebidos e determine os estados do sistema. Utiliza-se o programa *Matlab* para a realização deste processamento.

A plataforma de comunicação do robot é desenvolvida com base num programa de navegação com ele fornecido. Este programa realiza originalmente um controlo muito rudimentar do robot baseando-se em dados de GPS e do programa *Google Earth*. Permite também o controlo manual do robot utilizando um comando de videojogos para computador, a visualização de informações relativas à energia da bateria e permite observar as imagens captadas pela câmara frontal do robot e a sua posição numa imagem do *Google Earth*. Originalmente, a aplicação possibilita também a gravação dos dados recebidos pelo sensor GPS, gravando num ficheiro de texto a informação recebida.

No manual (DrRobotInc, 2014), refere-se inclusivamente, que os utilizadores podem desenvolver o seu programa de controlo recorrendo à interface de programação de aplicações (API) e às ferramentas fornecidas. A alteração da aplicação é uma tarefa necessária para o desenvolvimento desta dissertação, uma vez que é necessária a recolha de informação não só relativa ao GPS mas também aos restantes sensores do robot. A aplicação a desenvolver deve também aceitar os dados determinados pelo *Matlab* e

enviá-los ao robot. Pode dizer-se que esta aplicação deve assumir um papel de intermediário entre robot e *Matlab*, sendo também responsável por mostrar dados ao utilizador acerca do estado do robot, não só em termos de navegação mas também em termos de energia disponível na bateria, temperatura dos motores, etc.

Como descrito no manual do robot (DrRobotInc, 2014), a comunicação no seu interior entre os componentes (controladores de motores, sensores, *motherboards*, etc) é realizada via *Ethernet* formando uma rede entre os componentes. Esta rede é uma *Local Area Network (LAN)* que integra também um *router wireless* que estabelece comunicação sem fios com o computador possibilitando o controlo à distância independente de ligações por fios. Se for utilizado o comando de videojogos para controlo do robot, este comando deve comunicar com o computador. No Apêndice A pode consultar-se um diagrama com as interconexões entre os vários circuitos e módulos do robot retirado do seu manual.

A comunicação com o computador implica a configuração de uma rede *wireless*, sendo que para isto se deve definir manualmente um endereço *IP* para o computador. Deve dar-se atenção ao facto de o *router* ter três saídas para a rede *Ethernet* no interior do robot que já têm, por definição, três *IPs* atribuídos. Desta forma, o endereço *IP* atribuído manualmente ao computador não pode ser igual a nenhum dos já atribuídos. Estes *IPs* predefinidos encontram-se indicados no manual do robot (DrRobotInc, 2014).

A aplicação original é, de acordo com o manual, desenvolvida com *Visual Studio 2008* em C#, sendo possível também a adaptação da programação da placa de controlo dos sensores e movimento do robot. Esta programação deve ser C++ ou java. De seguida descrevem-se com mais detalhe os sensores do robot que têm especial importância e cuja informação está diretamente relacionada com a sua navegação.

2.4.1 Sensor GPS

O sistema GPS é a principal referência do robot para a determinação da sua localização e, por conseguinte, para a navegação. De uma forma geral, este sensor utiliza os dados dos satélites GPS em órbita da terra para determinar a posição do robot.

O sensor GPS utilizado no robot é o Garmin GPS 18x 5Hz, ilustrado na Figura 2.18. Como descrito pelo fabricante (Garmin, 2016), este é um sensor de alta sensibilidade para uso em aplicações de controlo e orientação de máquinas que exigem relatórios de posição e velocidade a 5 Hz.

Figura 2.18: Garmin GPS 18x 5Hz. (Garmin, 2016)

Este GPS armazena informações de configuração em memória não volátil, pelo que arranca rapidamente quando é ligado. Possui também um relógio de tempo real e dados de saída em bruto para aplicações variadas. Faz medições com a frequência de 5 Hz, o que permite ter uma boa precisão na determinação da localização.

No entanto, os dados obtidos por este sensor são também afetados por ruído e erros de medida, sobretudo se as medições forem obtidas em locais com obstruções no espaço entre o sensor e os satélites. Estas obstruções podem ser, por exemplo, construções ou árvores, e os seus efeitos são, principalmente, a

redução da potência do sinal de satélite ou a receção de sinais refletidos levando à corrupção da informação.

2.4.2 Sensores *IMU* (*Inertial measurement unit*)

Outros sensores incluídos no robot que podem fornecer informação útil acerca dos seus movimentos e ajudar na navegação são os sensores inerciais. Estes sensores são um acelerómetro (ADXL345), um giroscópio (ITG-3200) e uma bússola magnética (HMC5843) e estão incluídos numa única plataforma (SEN-10125). Os três sensores têm a capacidade de recolher medidas nos eixos X, Y e Z, que são reunidas pelo circuito e enviado ao sistema de comunicação do robot. Assim, o acelerómetro mede as acelerações ocorridas em cada um dos três eixos, o giroscópio mede as velocidades angulares verificadas em torno de cada eixo e a bússola converte campos magnéticos em tensões relativas a cada um dos eixos.

Importa referir que as medições do acelerómetro e giroscópio são obtidas com uma frequência de 50Hz (20ms) enquanto que a bússola envia os seus dados a cada 220ms.

Na Figura 2.19 encontra-se ilustrada a placa que integra os referidos sensores. Para a obtenção de informação mais detalhada acerca de cada um deles sugere-se a consulta dos respetivos manuais.

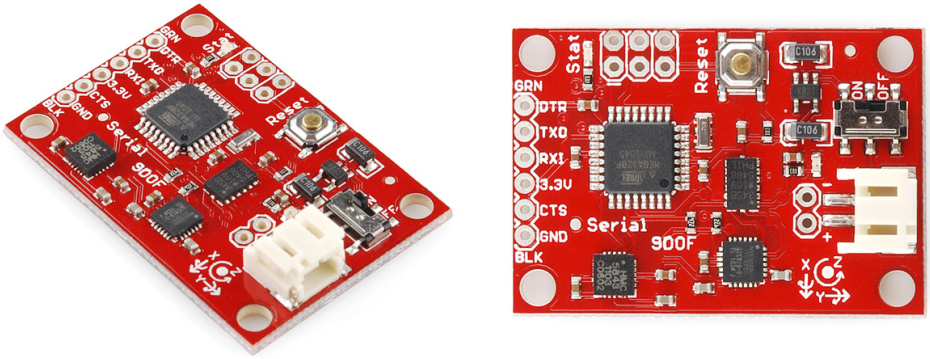


Figura 2.19: Sparkfun 9DOF-Razor SEN-10125 (Sparkfun, 2016)

2.4.3 Encoders das rodas

O terceiro tipo de sensores que pode ser utilizado na navegação do robot são os *encoders* dos motores das rodas. Estes sensores verificam a posição das rodas através da soma do número de passos que cada roda avança durante o movimento do robot, isto é, o ângulo total das rodas (360°) é dividido em pequenos espaços que são contabilizados durante o movimento circular de cada roda. Este sensor apresenta uma resolução de 150 contagens por revolução da roda o que corresponde a uma distância percorrida pelo robot de aproximadamente 0.57 cm entre cada contagem (tendo em conta que o diâmetro das rodas é de 27 cm). A sua leitura é realizada com eventos criados a partir do sistema de controlo do robot à frequência de 20 Hz.

O sistema de controlo do robot referido consiste numa versão especial da placa PMS5005 da *Dr. Robot*, na Figura 2.20, e é numa plataforma que faz a leitura de sensores de movimento e realiza o controlo dos motores. Esta placa recebe então os dados dos *encoders* das rodas e a temperatura dos motores e controla o seu movimento. Também a tensão da bateria é lida por esta placa.

Figura 2.20: Placa PMS5005 (DrRobotInc, 2016)

Capítulo 3

Implementação e trabalho realizado

3.1 Metodologia utilizada

O trabalho desenvolvido para esta tese de mestrado consiste na criação de um sistema de navegação autónoma para um robot móvel baseado na filtragem de Kalman da informação disponibilizada pelos sensores desse robot (*IMU*, GPS e *encoders* das rodas). Desta forma, o desenvolvimento do trabalho segue uma metodologia que permite o desenvolvimento gradual do sistema implementado.

Na Figura 3.1 pode observar-se, de forma geral, o algoritmo de desenvolvimento da tese de mestrado. As tarefas realizadas em cada passo do desenvolvimento do trabalho, bem como os detalhes da implementação das soluções são descritas no decorrer deste capítulo.

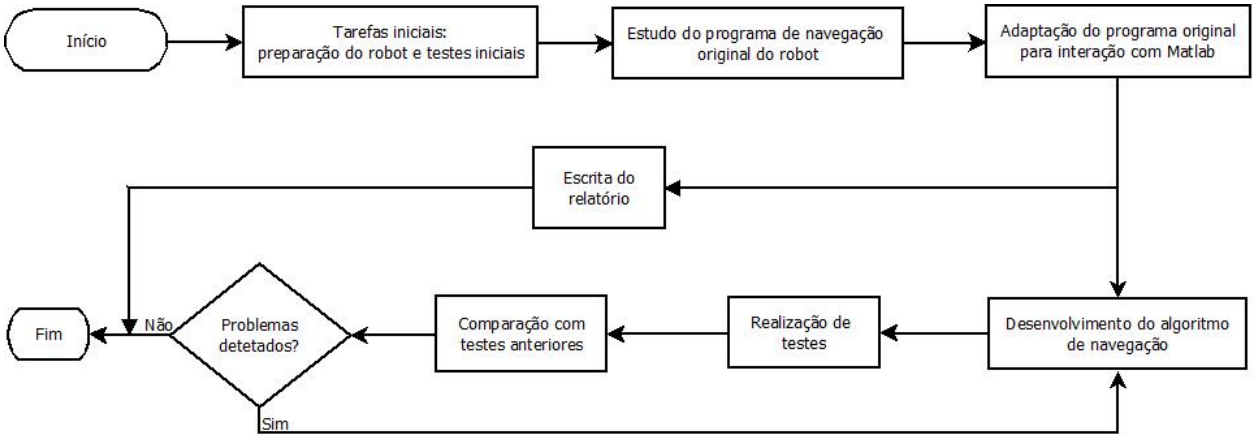


Figura 3.1: Metodologia para o desenvolvimento do trabalho realizado.

3.2 Tarefas iniciais

3.2.1 Bateria

Importa referir que o robot a utilizar para desenvolvimento da dissertação de mestrado já tinha sido previamente usado para outros trabalhos de investigação, mas, no entanto não se encontrava a funcionar. Desta forma, foi necessário realizar alguns procedimentos que o tornassem operacional para poder ser utilizado. Mais especificamente, foram inicialmente trocadas as baterias para alimentação do aparelho, uma vez que as que se encontravam instaladas (originais) apresentavam sinais de desgaste e má conservação. Adaptaram-se então as conexões para que fosse possível a colocação da nova bateria, tanto no próprio robot como no seu carregador.

Esta adaptação consistiu na conversão da conexão do tipo Bala (*Bullet connector*) da bateria para o tipo *Tamiya*, presente no robot. Para isso instalou-se uma extensão de cabo elétrico com esta conversão para evitar a alteração das extremidades originais do cabo do robot e da bateria. Utiliza-se também um fusível nesta ligação para evitar que uma eventual descarga possa danificar os circuitos do robot. Aplicou-se manga termoretrátil no isolamento das diversas ligações. Na Figura 3.2 pode observar-se a nova bateria já adaptada.

Esta nova bateria tem tensão nominal igual à original (22,2V), mas apresenta uma capacidade de 8 Ah - ligeiramente inferior à original (10 Ah) - o que exige uma maior atenção durante a realização de testes para evitar a descarga excessiva que acontece agora mais cedo que com a original. Esta descarga excessiva poderá ser prejudicial à bateria, podendo danificá-la permanentemente ou provocar derrames, aquecimento exagerado ou explosões. Assim, cada célula da bateria não deverá apresentar uma tensão inferior a 3V, o que se traduz numa tensão total da bateria de 18v (6 células).

Para o seu carregamento consideraram-se sempre diversos cuidados como a utilização do tipo de carregamento balanceado (em que o carregador monitoriza a tensão das células e tenta que todas apresentem o mesmo valor), não carregar acima dos 4,2V por célula (25,2v no total), corrente de carregamento máxima não superior a 1,5A, supervisão e boa ventilação do local de carregamento.

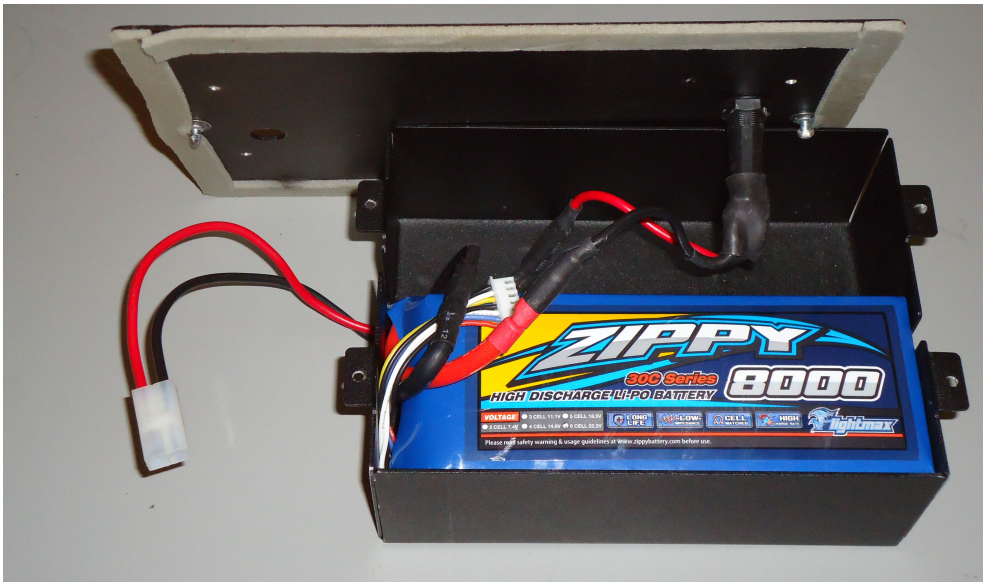


Figura 3.2: Adaptação da nova bateria no respetivo recipiente com conexões já convertidas.

3.2.2 Conexão com computador

Após a correta conexão e instalação da bateria carregada no chassis do robot iniciam-se os procedimentos para comunicação do robot com o computador. Para isto é necessário ligar o robot no comutador que se encontra na sua parte traseira. Após esta ação, o robot realiza automaticamente os procedimentos de inicialização de todos os seus componentes internos e a criação de uma rede *wireless* através do *router* existente no seu interior. Esta inicialização está programada no sistema de controlo interno do robot, que não foi alterado no decorrer do trabalho, por não ser realmente necessária essa alteração e para manter a consistência do funcionamento do robot e do seu equipamento.

Após alguns minutos é possível identificar a rede *wireless* num computador que se encontre no seu alcance. Para aceder a esta rede devem ser desativadas as configurações de servidores *proxy* existentes no computador e, uma vez que o *router* do robot não define automaticamente um endereço IP para novos elementos da rede, este deve ser manualmente inserido nas definições IPv4 do computador. Deve ser tido em conta que o *router* inicializa automaticamente diversos endereços IP para os componentes internos do

robot, identificados no seu manual (DrRobotInc, 2014), consequentemente, o endereço introduzido deverá ser diferente dos já definidos. A título de exemplo, na utilização do robot foi regularmente utilizado o endereço IP 192.168.0.200.

Após definidas as configurações referidas deve estabelecer-se ligação com a rede *wireless* do robot selecionando a rede com o nome *DriJaguar* com a palavra chave "*drrobotdrrobot*". A partir deste momento o computador e o robot encontram-se conectados na mesma rede sendo possível a troca de dados entre as duas entidades.

A comunicação utiliza o protocolo *WEP - Wired Equivalent Privacy* de 128 *bits*, pelo que se pode considerar que a ligação é realizada com certo grau de segurança.

Na Figura 3.3 ilustra-se, de forma esquemática, a sequência de operações a realizar para estabelecer a ligação via *wireless* entre o robot e o computador.

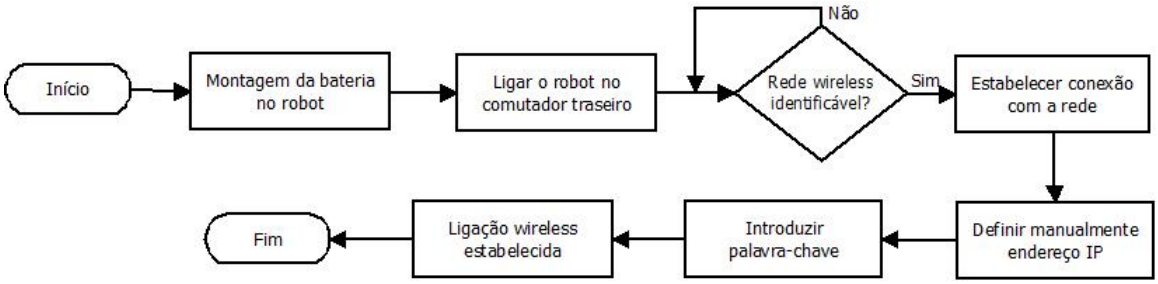


Figura 3.3: Sequência de operações a realizar para estabelecer ligação via wireless entre o robot e o computador.

3.2.3 Utilização dos programas de controlo/navegação

O *software* original fornecido com o robot inclui dois programas que permitem a sua utilização. Ambos os programas são escritos na linguagem C# e são editáveis, uma vez que os seus códigos fonte são também disponibilizados.

A aplicação *Jaguar Control* é um dos programas referidos e permite uma interação completa com o robot e a informação por ele disponibilizada. Assim, com a sua utilização, é possível observar imagens da câmara, receber e enviar sinais áudio de/para o robot, observar a variação da aceleração e da velocidade angular nos eixos X, Y e Z, verificar os dados GPS captados, estado da bateria, velocidade e temperatura de cada motor e valores dos encoders das rodas. Observa-se também uma imagem de *Google Earth* com o trajeto percorrido pelo robot e é possível controlar o seu movimento através de um comando de videojogos para computador ou através dos controlos virtuais na interface do programa. Também é possível guardar os dados em bruto recebidos do sensor GPS e do sistema *IMU* em forma de *string* num ficheiro de texto.

Na Figura 3.4 ilustra-se uma imagem da interface gráfica desta aplicação.

A aplicação *Jaguar Navigation Demo* é o segundo programa disponibilizado e permite a navegação do robot em trajetos definidos pelo utilizador. Esta aplicação, para além de realizar a maioria das tarefas da anterior, inclui também um algoritmo de navegação e permite que o robot se desloque num percurso definido.

Uma vez que este programa tem toda a estrutura para comunicação com o robot e implementa já um algoritmo para navegação, é utilizado como base para o desenvolvimento do trabalho desta tese de mestrado. Por este motivo descrevem-se no decurso do relatório os detalhes do seu funcionamento original e as alterações efetuadas para implementação do algoritmo de navegação com filtros de Kalman, tema do trabalho.

Na Figura 3.5 ilustra-se uma imagem da interface gráfica desta aplicação.

Com estas aplicações e após conexão com um computador testou-se o funcionamento geral do robot e foi possível verificar que tudo se encontrava funcional e que o aparelho obedecia corretamente aos controlos enviados. Desta forma, por intermédio de testes realizados, confirma-se a inexistência de

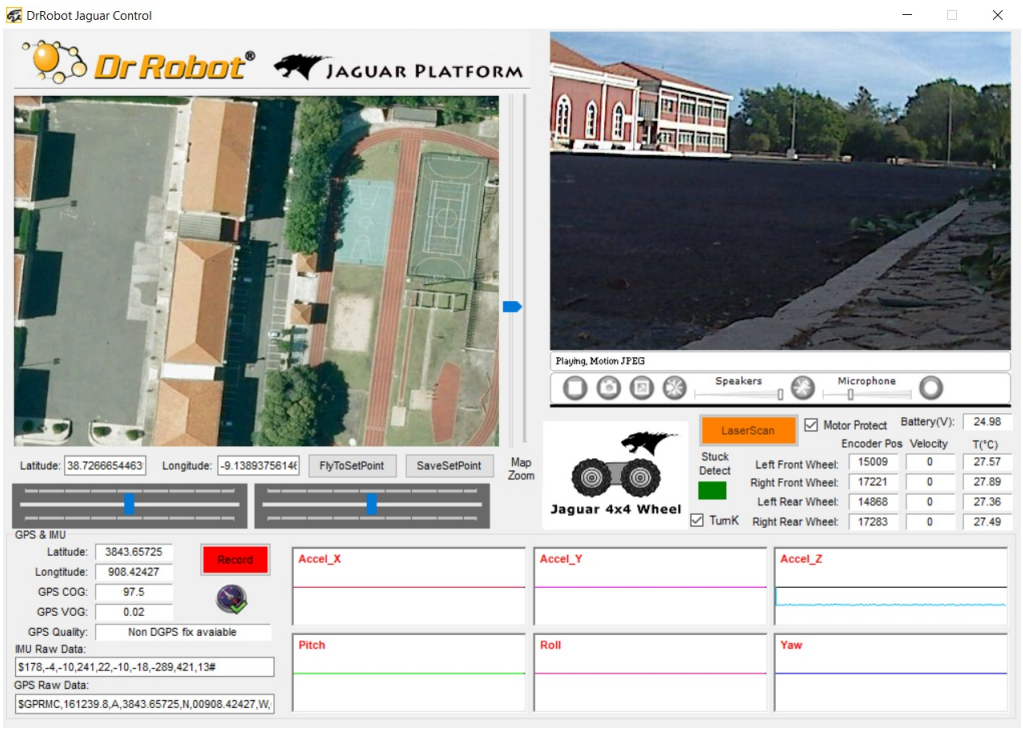


Figura 3.4: Aplicação Jaguar Control.

problemas de *hardware* que possam comprometer o correto desenvolvimento da dissertação. Com isto, é possível o aumento do conhecimento das capacidades do robot e dos programas disponibilizados. Na Figura 3.6 ilustram-se resultados da navegação realizada nestes testes iniciais sendo a linha a verde a trajetória que se pretende que o robot percorra, a vermelho a trajetória calculada pelo seu algoritmo de navegação e a azul o percurso definido pelos dados GPS recebidos. Apesar de serem testes meramente didáticos com o único objetivo de utilizar o programa original e verificar o seu funcionamento, podem observar-se já alguns problemas que podem existir com o sistema de navegação original, uma vez que há desvios consideráveis das rotas pretendidas.

Após a exploração destes programas na ótica de utilizador, foi realizado um estudo detalhado do código fonte das aplicações, sobretudo do programa a utilizar como base para a navegação (*Jaguar Navigation Demo*). Assim, reconheceram-se as especificidades das funções disponibilizadas pelo fabricante na linguagem de programação utilizada (C#) e compreendeu-se o seu funcionamento e estrutura. Este estudo possibilitou a adaptação do programa às diferentes necessidades que foram surgindo, bem como a introdução ou modificação de capacidades que se consideraram importantes no decorrer do desenvolvimento.

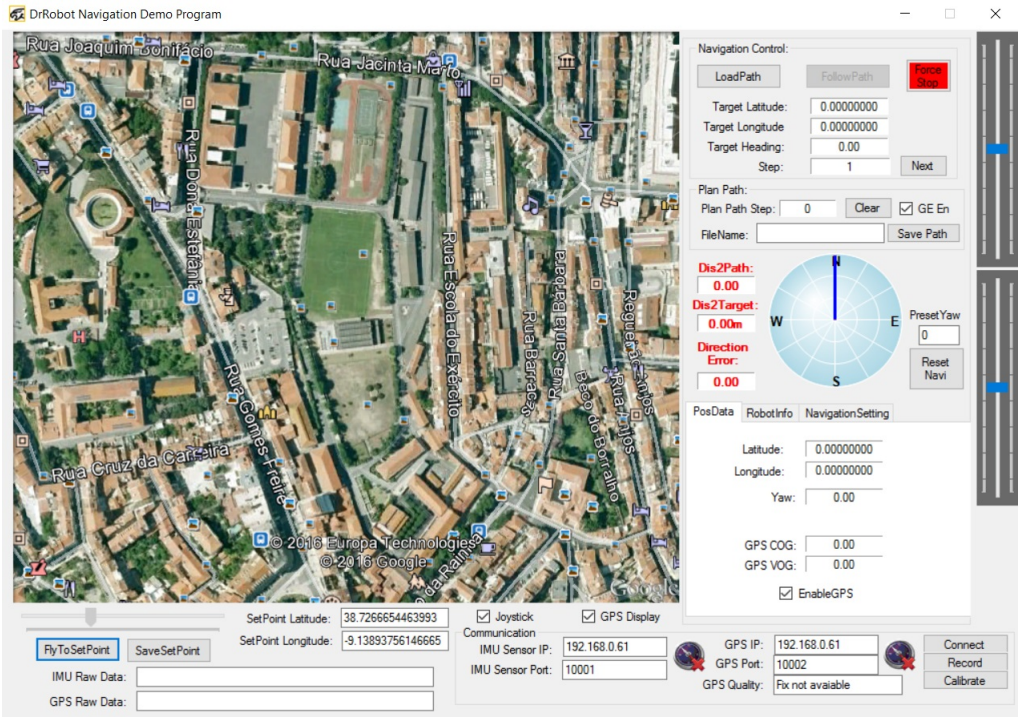


Figura 3.5: Aplicação Jaguar Navigation Demo.

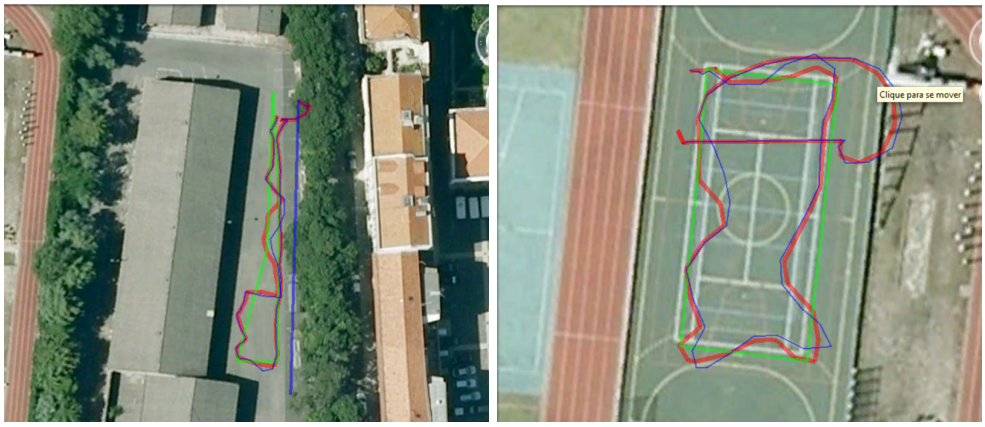


Figura 3.6: Testes realizados inicialmente para conhecimento do programa de navegação.

3.3 Programa de navegação original - *Jaguar Navigation Demo*

Como referido, o programa *Jaguar Navigation Demo* é uma aplicação fornecida pelo fabricante do robot *Jaguar* que permite navegar com o robot num percurso definido pelo utilizador. Esta aplicação é utilizada como base para o desenvolvimento do algoritmo de navegação com filtros de Kalman (tema deste trabalho) uma vez que já inclui toda a plataforma de comunicação com o robot e permite que o foco do esforço seja no desenvolvimento do algoritmo referido.

Assim, e uma vez que o código fonte está disponível, o programa é adaptado para que seja possível a troca de dados com o programa *Matlab* que possui grande capacidade de cálculo e ferramentas importantes na manipulação de dados matemáticos. De uma maneira geral, o algoritmo implementado em *Matlab* tem a função de receber os dados em bruto provenientes dos sensores (*IMU* e GPS) e devolver uma estimativa filtrada da posição instantânea do robot. Após a receção da posição filtrada, fica a cargo da aplicação a realização dos procedimentos de navegação.

Dada a importância deste programa para o desenvolvimento do trabalho e uma vez que o estudo do seu funcionamento é uma parte importante desta tese, serão descritos os pormenores de funcionamento do *Jaguar Navigation Demo*. Posteriormente descrevem-se as alterações e adaptações efetuadas e o funcionamento do algoritmo implementado em *Matlab*.

Uma vez que este programa de navegação se trata de uma aplicação desenvolvida na linguagem C# orientada a objetos, a adaptação e desenvolvimento do código fonte é realizado em ambiente *Microsoft Visual Studio 2015*.

3.3.1 Preparação da ligação ao robot

As primeiras tarefas realizadas pelo *Jaguar Navigation Demo* são definição de todas as variáveis para o programa funcionar, a criação da interface gráfica (Figura 3.5) e a preparação para ligação ao *software* que corre no robot, nomeadamente na placa de controlo PMS5005 (referida na Secção 2.4.3). Para isso serve-se da rede *wireless* criada (Secção 3.2.2) para definir um *gateway* que possibilita a troca de dados com a placa. Esta definição de variáveis é necessária para colocar em memória todos os valores especificamente relacionados com este tipo de robot. Estes valores são utilizados em cálculos e nos algoritmos de controlo e comunicação tendo em conta as características do robot *Jaguar*. Isto justifica-se com a possibilidade de utilizar este mesmo programa com outros tipos de robot da *Dr. Robot*, mas nesse caso, as variáveis a definir teriam valores diferentes.

Alguns exemplos dessas variáveis são os diversos endereços IP existentes na rede interna do robot, o diâmetro das rodas, o ângulo de viragem do robot e a potência inicial aplicada nos motores.

Na Figura 3.7 ilustra-se a janela de iniciação de ligação ao robot.

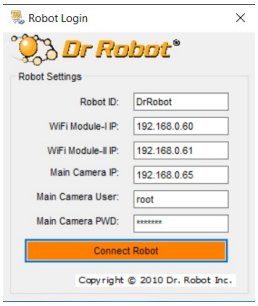


Figura 3.7: Janela de iniciação de ligação ao robot.

3.3.2 Estabelecimento da ligação ao robot

Depois da preparação inicial de ligação ao robot é possível, através da interface gráfica criada (Figura 3.5), iniciar a troca de dados com o robot. Esta troca de dados acontece com o premir do botão *Connect* no seu canto inferior direito. Após este momento o programa recebe e exibe os dados enviados pelo robot, que são os dados do sensor GPS, dados do sensor *IMU* e dados da placa PMS5005. Importa referir que a informação dos dois primeiros sensores é transmitida via *sockets* unidirecionais (do robot para o computador) criados no momento em que se pressiona o botão *Connect*. A comunicação com a placa PMS5005 é bidirecional e mais complexa (inclui instruções específicas para os seus vários componentes) sendo portanto realizada via *gateway* referido na secção anterior. Na Figura 3.8 ilustra-se, de forma esquemática, a sequência de operações realizadas na iniciação do programa *Jaguar Navigation Demo*.

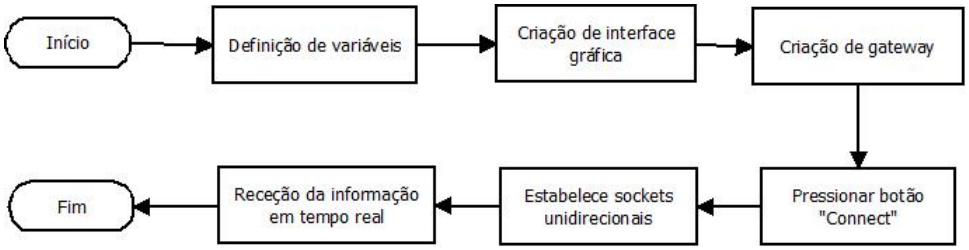


Figura 3.8: Sequência de operações realizadas na iniciação do programa Jaguar Navigation Demo.

A partir deste momento o programa mostra a seguinte informação em tempo real (algarismos e letras da Figura 3.9): os dados em bruto recebidos pelos sensores GPS e *IMU* (1); as coordenadas da posição atual do robot e a orientação do seu movimento (2); os endereços IP e portas que está a utilizar para ler os dados desses sensores (3); a qualidade do sinal GPS (4); dados relativos ao estado do robot, por exemplo temperatura dos motores, a tensão da bateria, a posição dos *encoders* das rodas, etc (A); opções configuráveis no método de navegação, por exemplo potência dos motores, diferencial de potência para realizar curvas, etc (B).

É também possível fazer o robot deslocar-se com o comando de videojogos para computador (5) e manipular a imagem visível do programa *Google Earth* (6). No canto inferior direito da interface (7), é possível desconectar do robot, gravar os dados que estão a ser recebidos ou proceder à calibração do sensor *IMU* (sumariamente, esta calibração é importante para definir o comportamento do giroscópio).

3.3.3 Configuração do percurso a realizar

Para definir um percurso para o robot percorrer deve criar-se um ficheiro de texto com as coordenadas em graus decimais dos locais por onde se pretende que o robot passe. Este ficheiro deve conter, em cada linha, um par de coordenadas (latitude, longitude) relativos aos pontos pretendidos. O programa de navegação forma um percurso definido pela sequência de pontos indicados da primeira linha para a última. Para criar este ficheiro pode ser utilizada uma ferramenta já incluída no programa (campo `Plan Path` da Figura 3.10).

O ficheiro de texto agora definido deve ser carregado no programa de navegação através do botão *LoadPath* na interface gráfica (Figura 3.10). O percurso então definido surge na imagem de *Google Earth* na forma de uma linha a verde e é indicado na interface gráfica o ponto para o qual o robot se dirige em cada instante (`Target`). O robot inicia o percurso ao pressionar o botão `FollowPath` (note-se que a função de navegação com *Joystick* deverá estar dasativada). Numa situação de emergência pode-se parar imediatamente o robot pressionando o botão `Force Stop`.

Nota: Na Figura 3.10, o caminho definido (linha verde) está deslocado para a direita em relação ao centro das pistas (local onde se pretende que o robot circule). Isto justifica-se por existir um desfasamento

Figura 3.9: Aparência da aplicação quando conectada ao robot.

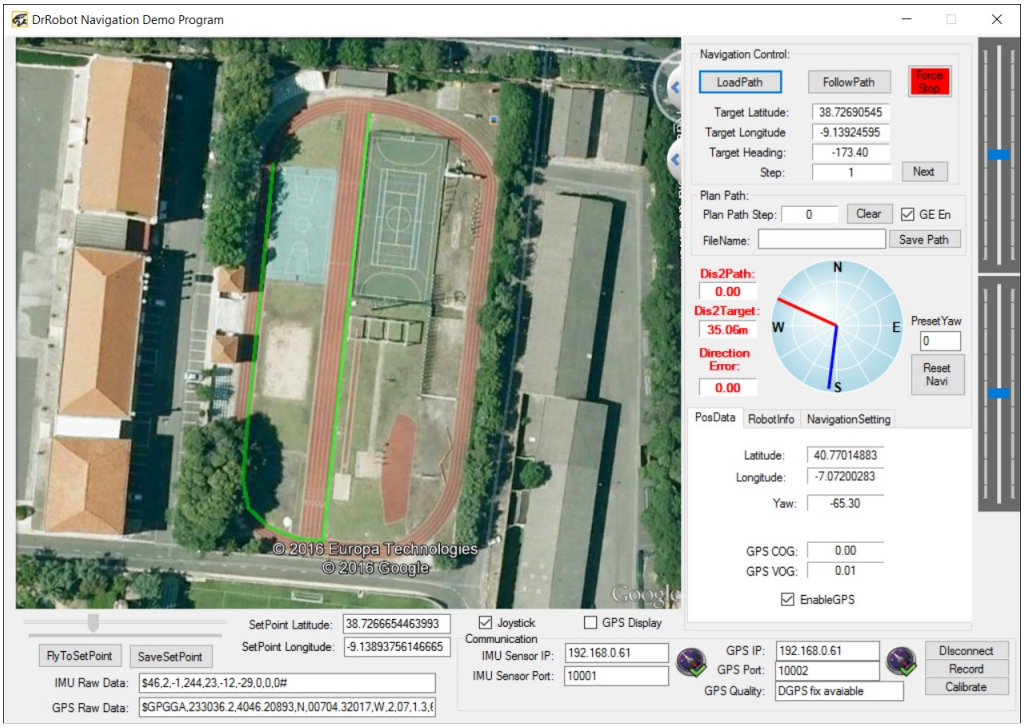


Figura 3.10: Aparência da aplicação na definição de um percurso.

entre as coordenadas GPS de um local e a sua localização na imagem *Google Earth*. No terreno, o percurso está efetivamente definido na zona central das pistas.

3.3.4 Seguir o trajeto

O processo de seguir o trajeto definido é onde se evidencia a necessidade da utilização de algoritmos e métodos para calcular a posição instantânea do robot e para determinar a evolução do seu movimento. Desta forma, o programa de navegação deve estimar com precisão a posição em que o robot se encontra em cada instante a partir dos dados disponíveis, deve ter o conhecimento da posição a alcançar e deve então determinar as ações a tomar para chegar à posição desejada.

Este conjunto de procedimentos é conseguido pelo *Jaguar Navigation Demo* da seguinte forma:

- Determinação da posição instantânea do robot: Para determinar a posição instantânea do robot são utilizados os dados dos *encoders* das rodas, que, ao serem recebidos do robot, são processados para atualizar a posição conforme as alterações nos valores recebidos. Quando são recebidos dados do GPS a posição até então estimada pelos *encoders* é atualizada com a informação recebida. Este processo é realizado com recurso a chamadas de funções definidas num ficheiro do tipo *Dynamic-link library* (*DLL*), cujo código não é disponibilizado (segundo a *Dr. Robot*, devido a direitos de autor de terceiros). Assim, não é possível compreender em detalhe os métodos matemáticos utilizados neste processamento.

- Determinação do trajeto a seguir: Para atingir o ponto de destino a partir da posição em que o robot se encontra, o programa baseia-se, sobretudo, no cálculo de duas orientações. Uma dessas orientações é a definida do ponto onde se estima que o robot se encontra para o local de destino, a segunda é a definida pelo seu movimento instantâneo.

Quando as duas direções diferem mais de 5º (definido por defeito mas este valor pode ser alterado) - Figura 3.11a - o programa inicia medidas corretivas para virar o robot de forma a que a diferença seja inferior a esse valor - Figura 3.11b. Este procedimento é repetido até que a posição estimada do robot se encontre a menos de uma determinada distância do ponto de destino (1 metro, por defeito). Nesta situação, o programa passa a considerar o ponto seguinte como ponto de destino, ou seja, considera agora que o destino é definido pelas coordenadas indicadas na linha seguinte do ficheiro de texto que contém os pontos do percurso.

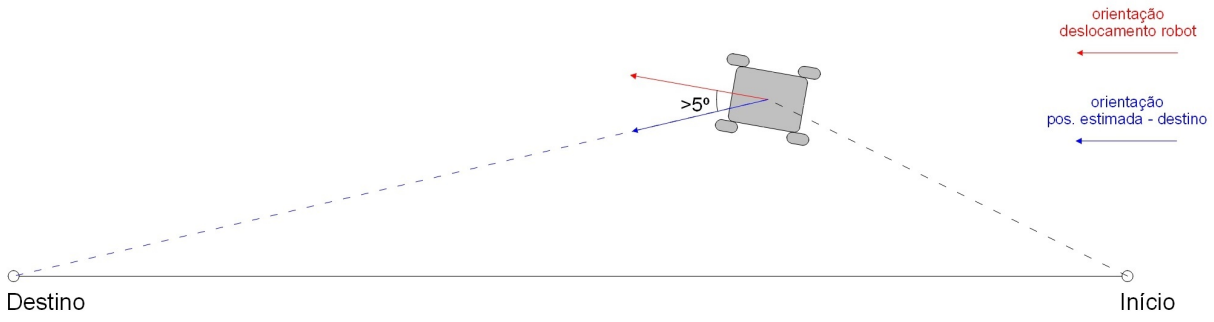
Neste processo, o programa vai enviando para o robot as instruções que permitem a alteração da sua rota, através da diferenciação da potência enviada às rodas de cada lado, o que permite ao robot realizar curvas.

A necessidade da determinação correta das duas orientações referidas torna-se evidente com a compreensão deste método de navegação:

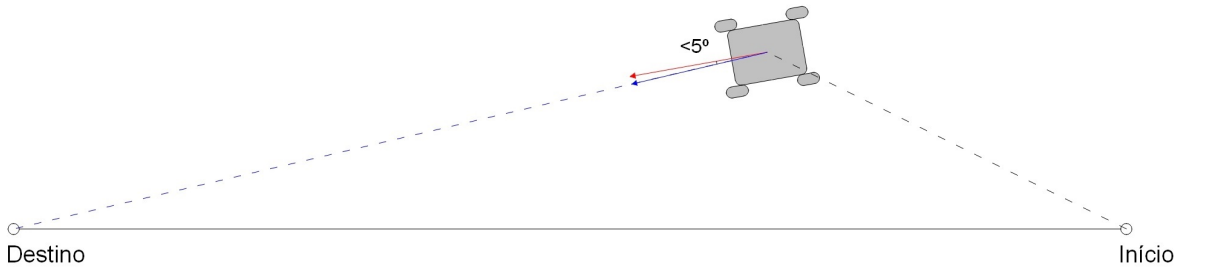
- É importante que a estimativa da posição instantânea do robot seja obtida com o mínimo de erro, livre de ruídos e com o menor atraso possível, para então se obter uma correta orientação da posição instantânea para o ponto de destino.

- A segunda orientação, aquela em que o robot se desloca a cada instante, é também muito importante. Ela é o que define se o seu movimento está a evoluir no sentido que se pretende. No programa original é calculada a partir dos dados do GPS e do giroscópio do robot. Assim, com a evolução das coordenadas GPS o programa determina qual a direção do deslocamento. Os dados do giroscópio ajudam em situações de curvas repentinas que o GPS não tem resolução para determinar. Devido a isto, o programa desativa os dados GPS momentaneamente na proximidade de pontos do percurso e passa a navegar só com base nas informações dos *encoders* e giroscópio que são mais detalhadas em situações onde é necessária mais resolução.

Esta combinação de dados pode trazer alguns problemas, nomeadamente ruído que pode ser captado pelo giroscópio durante o deslocamento do robot em terreno rugoso. Também em situações em que o sinal de GPS possa estar afetado (reflexões, obstáculos) e existam variações nas posições recebidas, a orientação instantânea do robot irá ter valores errados e serão ruidosas.



(a) Situação não corrigida.



(b) Situação corrigida.

Figura 3.11: Demonstração da correção efetuada pelo programa de navegação no alcance do ponto de destino.

Esta fusão de dados do GPS e giroscópio é também realizada, tal como a estimativa de posição, com recurso às funções do ficheiro *DLL*, não sendo então possível determinar o processo matemático utilizado.

Durante a navegação do robot, é possível acompanhar, na interface gráfica do *Jaguar Navigation Demo* e em tempo real, a variação de ambas as direções. Na Figura 3.12 ilustra-se um exemplo desta indicação, onde a linha vermelha representa a direção do movimento instantâneo do robot e a linha azul a direção da sua posição estimada para a posição de destino.

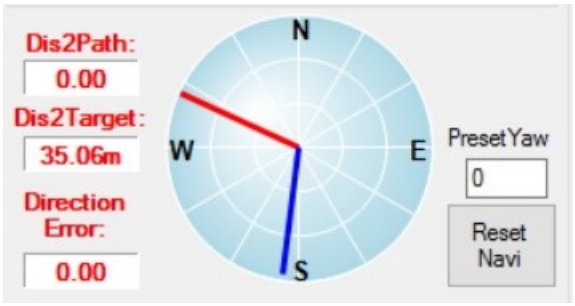


Figura 3.12: Indicação das direções utilizadas para navegação na interface gráfica do Jaguar Navigation Demo.

3.3.5 Chamada do algoritmo *Matlab* a partir do programa de navegação e obtenção de resultados

Para ser possível a chamada de ficheiros *Matlab* a partir de programas em C# é necessária a inclusão da referência à biblioteca de funções *MLApp* (*Matlab Application Type Library*). Esta biblioteca encontra-se nos ficheiros de instalação do *Matlab* e permite a troca de dados e a manipulação de *Scripts Matlab* a partir do código em C#. Com a definição de uma instância do tipo *MLApp* (passo inicial

necessário para chamar funções) é automaticamente criado um *workspace* de *Matlab* e um terminal que pode ser usado como consola de comandos ou para visualizar informação. Este *workspace* é o ambiente onde são definidas as variáveis das funções *Matlab* chamadas.

A definição da instância referida requer a adição das seguintes instruções ao código fonte do *Jaguar Navigation Demo*:

- `using MLApp` - utilizado inicialmente para definir a biblioteca *MATLAB Type Library*;

- `MLApp.MLApp matlab = new MLApp.MLApp()` - Cria instância do tipo *MATLAB Type Library*.

As instruções da biblioteca *MLApp* utilizadas foram as seguintes:

- `Execute(String x)` - executa comando definido pela *string* x no *workspace Matlab*;

- `PutWorkspaceData(String varname, String workspace, Object data)` - coloca o objeto `data` no espaço definido por `workspace` (*Base* ou *Global*) numa variável com o nome `varname`;

- `GetVariable(String varname, String workspace)` - devolve o valor da variável `varname` definida em `workspace` (*Base* ou *Global*) do ambiente *Matlab*;

Com o objetivo de se utilizar a maior quantidade possível da informação que é adquirida dos sensores, este algoritmo é chamado quando se recebem dados de qualquer um deles. Para isso é necessário compreender a forma como o programa de navegação atua para receber a informação disponível dos sensores, para então ser possível enviá-la ao *Matlab* e fazer correr o algoritmo.

Relembra-se que os dados dos sensores *IMU* e GPS são gerados com frequências definidas e são recebidos em *stream* via *socket TCP*. Já a informação dos *encoders* é recebida através do *gateway* criado na iniciação do programa e é lida com base em eventos gerados pela placa de controlo no robot. Assim, o programa de navegação define duas *threads* para receção periódica dos dados dos sensores *IMU* e GPS e um *event handler* para os dados do *encoder*. Uma ilustração grosseira deste tipo de processamento pode ser observado na Figura 3.13.

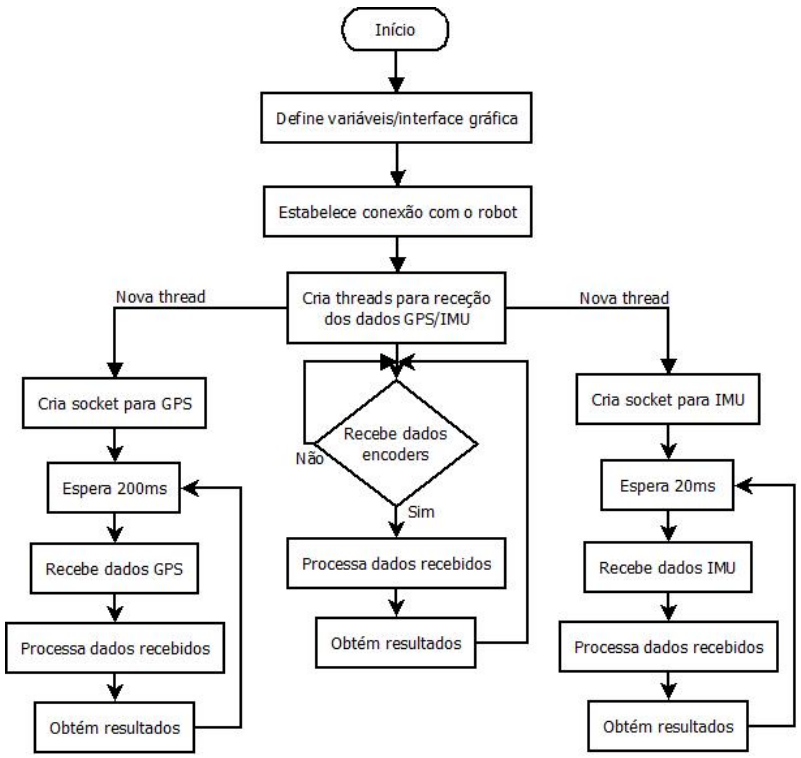


Figura 3.13: Exemplificação geral do método de receção dos dados dos vários sensores utilizando threads.

A forma de o programa tratar os dados recebidos consiste em colocar numa variável do ambiente *Matlab* a *string* de informação recebida (dos *sockets* ou *Gateway*) - utilizando o comando `PutWorkspaceData`

- e de seguida enviar um comando com a designação da função que deve correr, que neste caso é o referido algoritmo. Para isso utiliza-se agora o comando `Execute`. Esta função tem um argumento de entrada que é a variável com a *string* definida no *workspace Matlab*. Após a conclusão do processamento do algoritmo, este devolve os dados determinados através de argumentos de saída que ficam em variáveis também nesse *workspace*. Agora é possível obter estes dados com a função `GetVariable` e utilizá-los em ambiente C#.

Esta forma de processamento de informação em paralelo gera, como esperado, problemas de concorrência de recursos, uma vez que a colocação de *strings* no ambiente *Matlab* e a chamada do algoritmo ocorre em duas *threads* e num evento iniciado exteriormente. Para evitar estes problemas implementam-se *mutexes* que isolam todo o processo de definição das variáveis *Matlab*, cálculo do algoritmo e recolha dos resultados.

Outro problema relacionado com a chamada de instâncias *Matlab* a partir de outro ambiente de programação é a manutenção das variáveis internas das funções *Matlab*, ou seja, não é possível manter o estado dessas funções se as variáveis declaradas não forem todas globais. Esta dificuldade foi ultrapassada gravando as variáveis utilizadas em disco sempre que a função termina e assim que é chamada novamente essas variáveis são lidas do ficheiro em disco. No entanto, verificou-se que este procedimento atrasa significativamente o processamento do algoritmo (passa da ordem dos 0.01ms para cerca de 10ms). O programa de navegação não sai afetado uma vez que a frequência máxima de receção de dados é de 50Hz (20ms - sensor *IMU*).

3.3.6 Considerações sobre o filtro de Kalman estendido

Com o desenvolvimento das tarefas iniciais e a busca por uma solução concreta para a questão da filtragem da informação, chegou-se à conclusão que apesar de se reconhecer que a melhor maneira de lidar com todas as variáveis e não-linearidades do robot *Jaguar* passaria pela utilização de filtros de Kalman estendidos, seria preferível a sua não implementação. Esta escolha deveu-se às seguintes questões relacionadas com estes filtros, bem conhecidas na literatura desta área do conhecimento (Simon, 2006): filtragem não linear pode ser complexa e é mais difícil de implementar que a filtragem linear; a utilização do filtro de Kalman estendido implica a definição de um modelo matemático de representação do sistema; são necessários modelos de ruído de sistema precisos e uma maior quantidade de recursos computacionais. Contudo, apesar de o robot se tratar de um sistema não linear, pode ser razoavelmente aproximado a um sistema linear e, portanto, espera-se que os resultados de uma estimativa linear sejam bastante aproximados aos que se obteriam com estimativas não lineares.

3.4 Análise dos dados gerados

3.4.1 Sensor *IMU*

Os dados gerados pelo sensor *IMU* são recebidos na forma de *String* em formato ASCII através de um *Socket* definido no programa de navegação para endereço onde este sensor se encontra ligado na rede interna do robot (porto 10001, IP 192.168.0.61, ver Apêndice A). São gerados com uma frequência de 50Hz, no entanto, os dados da bússola magnética são gerados a cada 220ms.

O formato da *string* gerada é o seguinte:

"$seq,acelX,acelY,acelZ,giroX,giroY,giroZ,bussolaX,bussolaY,bussolaZ#"

- "seq"é o número de sequência do conjunto de dados gerado e varia de 0 a 255.

- Os valores "acelX,acelY,acelZ"são gerados pelo acelerómetro para os eixos X, Y e Z, respetivamente.

- Os valores "giroX,giroY,giroZ"são gerados pelo giroscópio para os eixos X, Y e Z, respetivamente.

- Os valores "bussolaX,bussolaY,bussolaZ"são gerados pela bússola magnética para os eixos X, Y

e Z, respetivamente.

Exemplo: `"107,-1,-4,243,20,-15,-12,-1,96,55"`

A orientação do sistema de eixos é a seguinte: o eixo X é paralelo ao chão e é longitudinal ao robot, ou seja, é orientado para a frente do robot; o eixo Y é paralelo ao chão e transversal ao robot; o eixo Z é perpendicular ao solo (ver Figura 3.14).

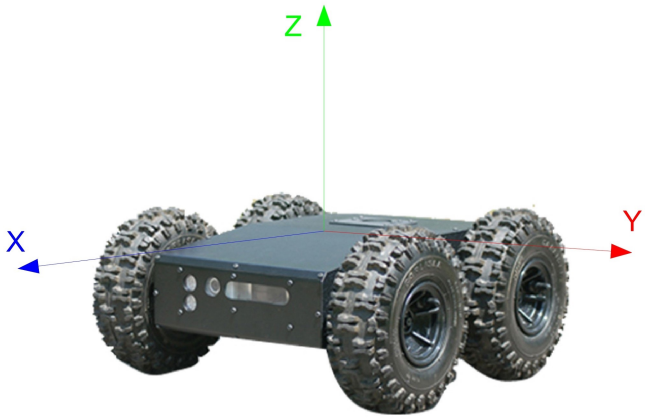


Figura 3.14: Esquema da orientação dos eixos utilizados nos dados odométricos.

Os vários componentes da informação *IMU* (aceleração, giroscópio e bússola) serão agora analisados, com o objetivo de referir as suas características e justificar a sua importância para o sistema de navegação. Para isso, efetuou-se, manualmente, um percurso padrão que permite comparar os dados em bruto enviados com o robot com o movimento efetivamente realizado. Este percurso, aproximadamente retangular, foi realizado em terreno plano alcatroado (estrada) e é ilustrado na Figura 3.15. Tem inicio no ponto A (0,0) e terminou em B. Este esquema foi obtido com base na conversão das coordenadas GPS obtidas durante o movimento para distância em metros entre os pontos definidos por essas coordenadas. Realça-se a necessidade de considerar a latitude do local para determinação da distância em longitude, por exemplo, um grau de longitude num local de latitude 0 (equador) representa maior distância que um grau de longitude num local mais a Norte ou a Sul.

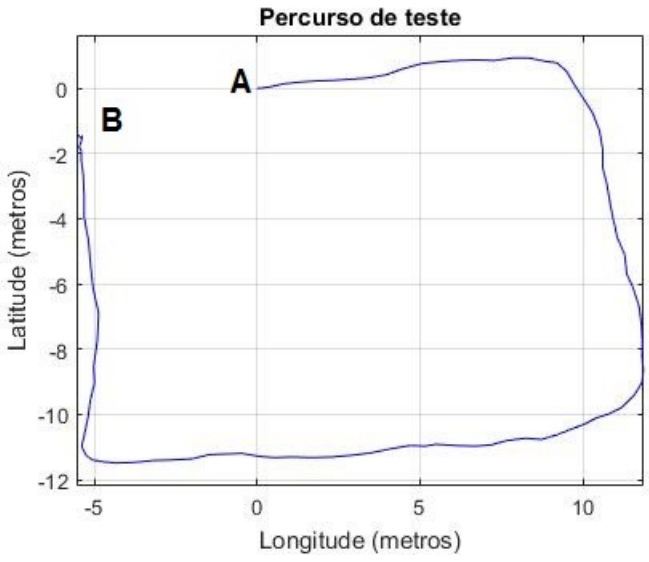


Figura 3.15: Percurso de teste realizado manualmente para observação dos dados enviados pelos sensores.

Aceleração

Os dados relativos à aceleração dão informação acerca da variação da velocidade do robot nos eixos X, Y e Z. O ótimo aproveitamento dessa informação requer a utilização dos dados dos três eixos, no entanto, apenas se considera o eixo X que é aquele que coincide com o movimento do robot e que mais diretamente reflete as variações de velocidade em condições normais, ou seja, considerando os casos mais prováveis em que o robot se possa deslocar (superfícies planas e niveladas sem deslizamentos). Os dados relativos ao eixo Y dão informação acerca da aceleração lateral e o eixo Z dá a aceleração na vertical.

O sensor produz valores com 10 *bit* de resolução e é programável para uma sensibilidade de ±2 g, ±4 g, ±8 g ou ±16 g. Uma vez que este valor não se encontra especificado na documentação, realizaram-se diversas experiências e utilizou-se o valor que se verificava constantemente no eixo Z, que refletia a aceleração da gravidade, para a determinação do correto valor em m/s^2 a partir dos dados numéricos gerados. Chegou-se assim à conclusão de que o sensor se encontra configurado para uma sensibilidade de ±2 g.

Nos gráficos da Figura 3.16 podem observar-se os dados em bruto (já em m/s^2) gerados pelo acelerómetro para o percurso da Figura 3.15.

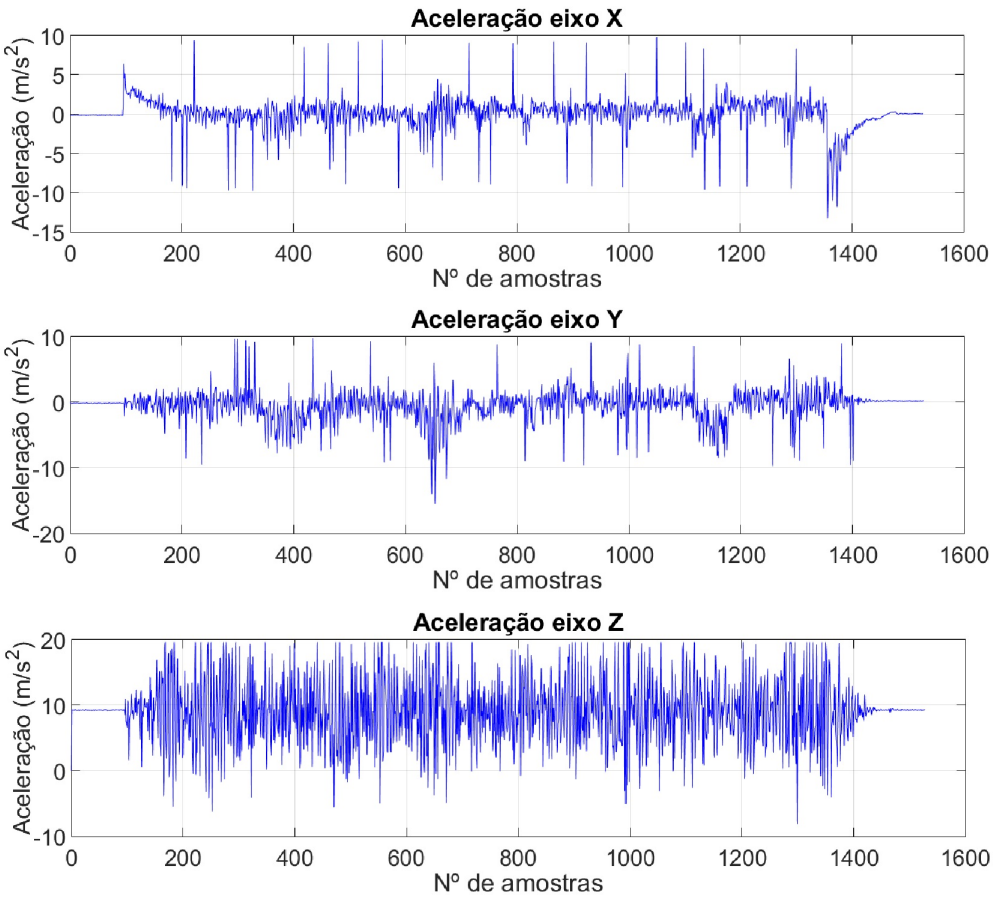


Figura 3.16: Dados em bruto para aceleração nos três eixos relativos ao percurso da Figura 3.15.

Com a integração dos valores da aceleração do eixo X é possível obter a variação da velocidade e com a integração desta determina-se a variação da posição do robot. Como pode ser observado, os dados do acelerómetro podem ser significativamente ruidosos, uma vez que a deslocação do robot provoca vibrações que afetam muito o sensor. Para além disso, o próprio sensor introduz regularmente valores pico que não têm qualquer significado. Também se deve considerar que o facto de se obterem valores de aceleração de forma discreta, implica que seja necessário ter em conta a duração de cada intervalo

entre amostras. Sabendo que esses intervalos são de duração constante (20ms) é possível obter resultados relativamente corretos, no entanto, se ocorrerem atrasos na comunicação ou no processamento ou se existir perda de pacotes de dados, a determinação correta da aceleração será muito afetada originando resultados incorretos. Uma vez que se estão a utilizar *mutexes* no programa de navegação, a variação da frequência na receção de pacotes é uma ocorrência quase certa.

Refere-se também que os sensores não apresentam um valor absolutamente nulo quando o robot não está em movimento, pelo que se deverá efetuar uma calibração aquando da utilização dos dados. Destaca-se novamente o facto de a aceleração em Z apresentar um desvio constante de cerca de 10 m/s, que se deve, naturalmente, à presença da aceleração da gravidade. A variação da aceleração no eixo Z que se verifica, apesar de ter variações consideráveis tendo em conta a escala, é apenas ruído, uma vez que não houve qualquer aceleração vertical no percurso efetuado para além da introduzida pela vibração das rodas.

Neste tipo de dados é francamente compreensível a utilização de filtros com o objetivo de limpar o sinal para obter dados muito mais corretos. Na Figura 3.17 pode observar-se, a azul, a aceleração no eixo X do percurso de teste (Figura 3.15) e a vermelho o resultado da filtragem de Kalman dessa informação. Também se representa o cálculo da velocidade com base na integração temporal dos dados das acelerações. Neste resultado podem observar-se alguns problemas derivados dessa integração, por exemplo, o facto de quando o robot termina o movimento a velocidade não ser de 0 m/s e a velocidade máxima obtida neste gráfico ser superior àquela que o robot pode efetivamente alcançar. Tudo isto se deve ao acumular de erros ao longo do tempo que não são corrigidos, sendo assim necessária a fusão de dados de outros sensores para ir amenizando a propagação de erros.

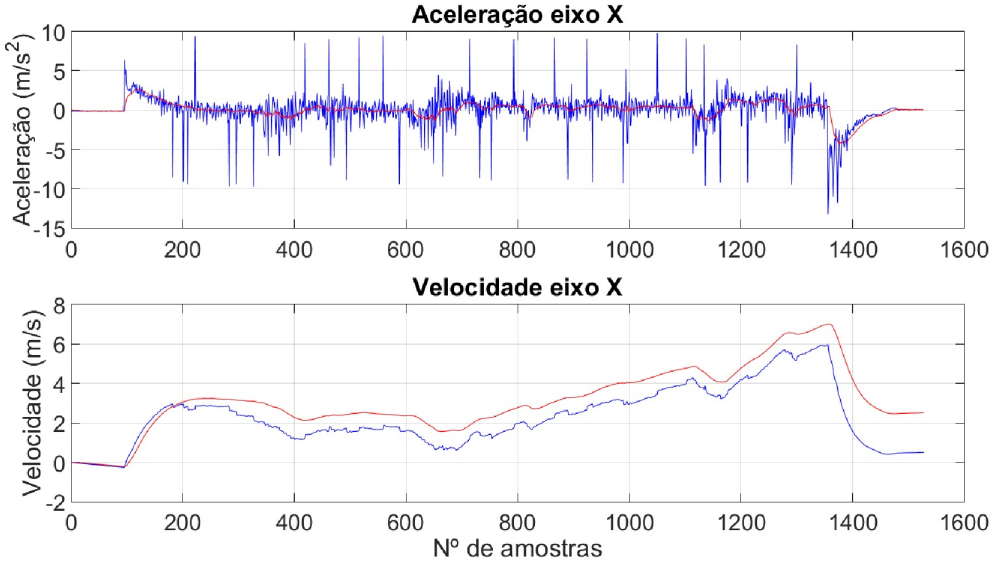


Figura 3.17: Gráficos com aceleração filtrada (vermelho) e não filtrada (azul) e determinação da velocidade por integração dos valores de aceleração obtidos.

Giroscópio

Tal como o sensor de aceleração, também o giroscópio devolve valores relativos ao sistema de eixos da Figura 3.14. Estes valores são a medida da velocidade angular em torno de cada um desses eixos. Neste caso, a medida mais importante é a que reflete a velocidade angular em torno do eixo Z, uma vez uma mudança de orientação na trajetória do robot se reflete neste eixo. Mais uma vez, é necessária a integração dos valores obtidos (velocidade angular) para ser possível determinar a variação da orientação (ângulo).

O giroscópio gera valores com 16 bit de resolução e tem uma sensibilidade de $\pm 2000^{o}/s$. Na Figura 3.18 podem observar-se os dados obtidos para o percurso de teste da Figura 3.15.

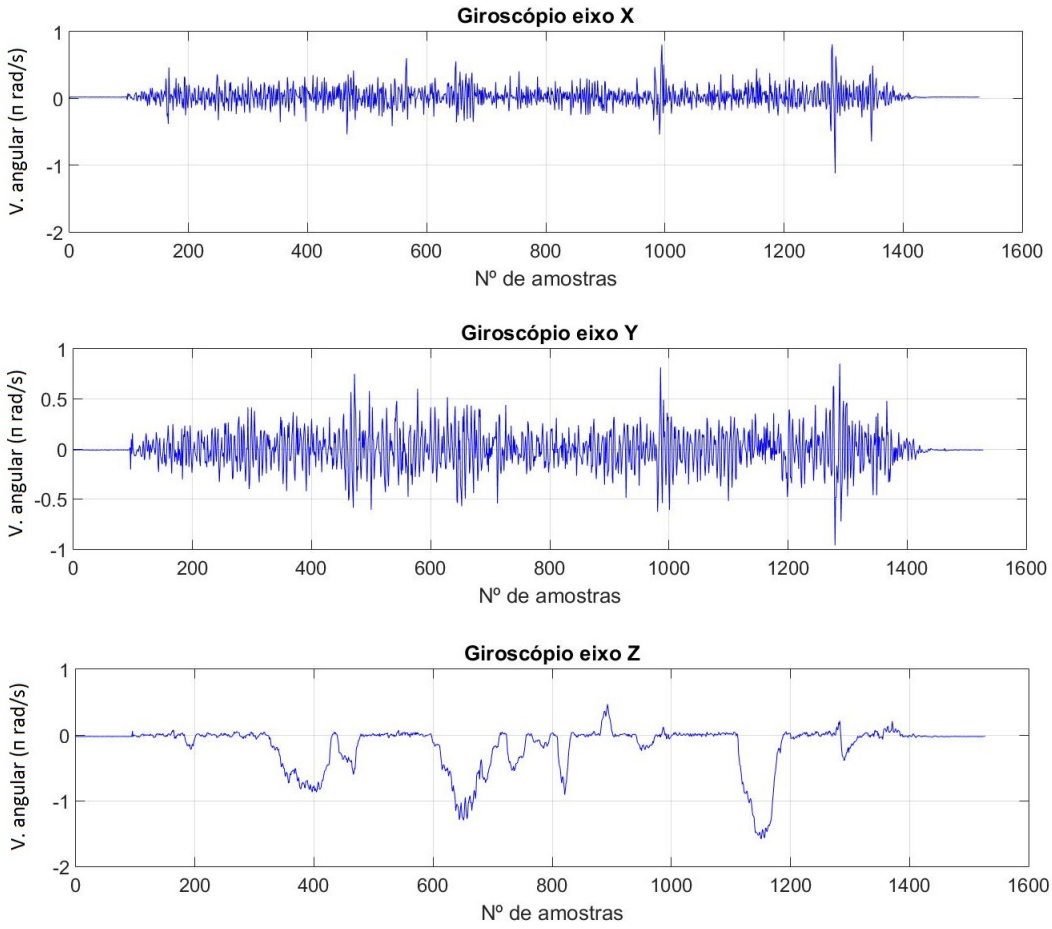


Figura 3.18: Dados em bruto para a velocidade angular nos três eixos relativos ao percurso da Figura 3.15.

Os dados relativos aos eixos X e Y dão informação relativa à rotação (inclinação) longitudinal e transversal do robot respetivamente. Os do eixo Z relacionam-se, como referido, com a variação de orientação do robot.

Naturalmente pode observar-se ruído nas medidas, sobretudo nos eixos X e Y. O eixo Z não é tão afetado, sobretudo em comparação com os dados do acelerómetro, nem apresenta valores pico aleatórios elevados. Algumas variações bruscas da velocidade angular que possam ser observadas devem-se, principalmente, a pequenas correções que foram sendo efetuadas durante o percurso de teste, controlado manualmente com o comando de videojogos.

Para complementar esta informação, ilustra-se agora, na Figura 3.19, a velocidade angular do eixo Z não filtrada em comparação com esses dados filtrados e a posterior integração temporal para determinação da orientação do robot.

Pode observar-se que, apesar da utilização do filtro nesta situação, a melhoria verificada não é substancial, uma vez que o cálculo da orientação do robot não é muito afetada pelo ruído nas medidas.

Por outro lado pode verificar-se que as medidas obtidas por este sensor são bastante fiáveis, pois, é possível identificar facilmente as mudanças de direção realizadas pelo robot no percurso realizado, que correspondem aos vértices do retângulo. Para além disso, os valores de orientação em radianos estão bastante próximos do real, pois o primeiro patamar tem um valor de aproximadamente 0 radianos, o segundo aproximadamente $\pi/2$ radianos, o terceiro π radianos e o quarto cerca de $3\pi/2$ radianos. Isto corresponde à mudança de direção de aproximadamente $\pi/2$ radianos realizada nos cantos do percurso.

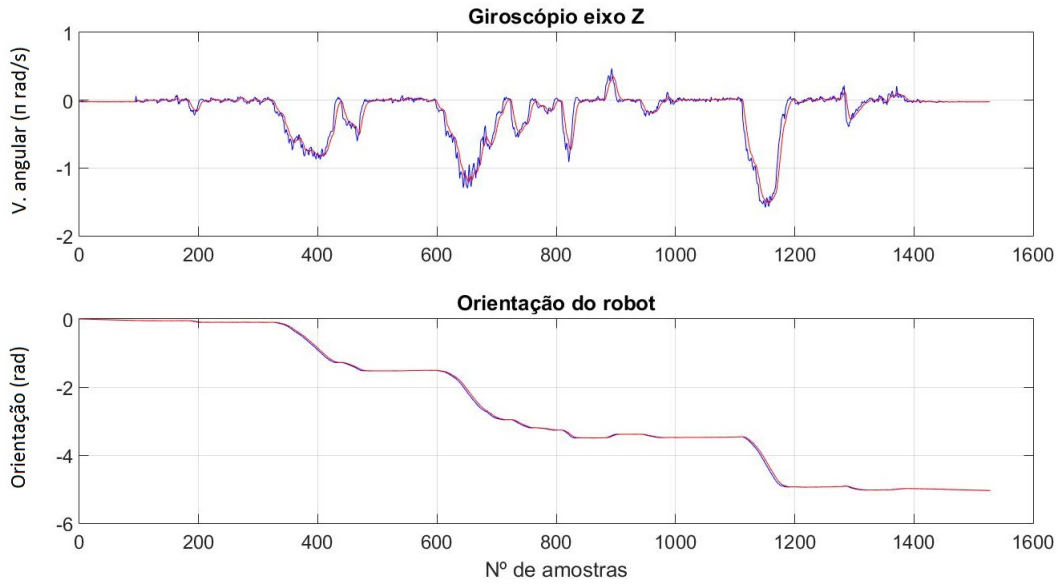


Figura 3.19: Gráficos com velocidade angular filtrada (vermelho) e não filtrada (azul) e determinação da orientação do robot por integração dos valores velocidade angular obtidos.

Neste sensor, existe também a tendência para gerar valores não nulos quando o robot está parado, ou seja, requer (à semelhança do acelerómetro) uma calibração inicial.

Um aspeto importante que deve ser levado em consideração é que a informação produzida pelo giroscópio no eixo Z e pelo acelerómetro no eixo X por si só não apresenta importância para a navegação do robot, uma vez que podem conhecer-se as variações de orientação e reconstituir-se o percurso realizado, mas não é possível enquadrá-lo num referencial conhecido (por exemplo coordenadas GPS). Para isso seriam úteis os dados gerados pela bússola que permitiriam enquadrar o trajeto relativamente ao campo magnético terrestre.

Na Figura 3.20 ilustra-se a reconstituição do percurso tendo em conta a evolução da posição em X (obtida através do acelerómetro) e as mudanças de direção obtidas pelo giroscópio.

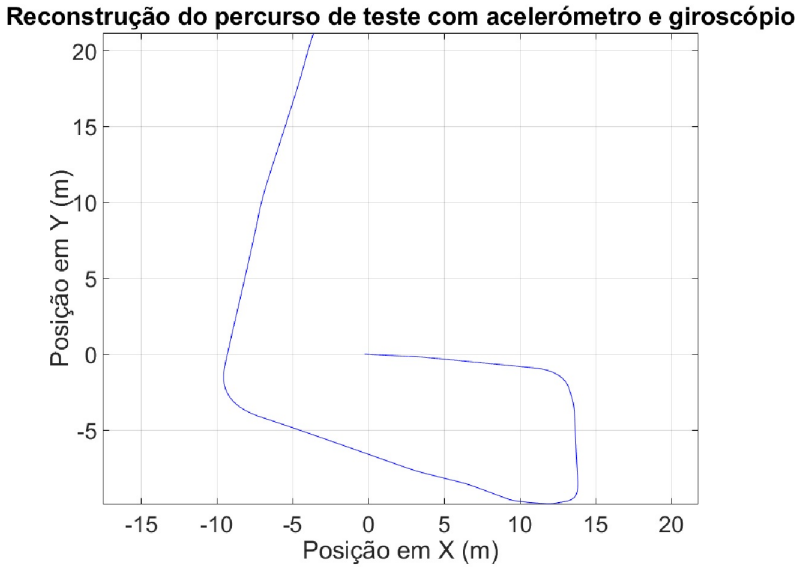


Figura 3.20: Reconstituição do percurso de teste com base nos dados do acelerómetro e giroscópio.

Nesta reconstituição pode facilmente reconhecer-se o trajeto efetivamente realizado, no entanto

podem destacar-se alguns pormenores interessantes associados ao processamento da informação. Pode observar-se que a acumulação de erros associados à aceleração provoca que a velocidade do robot calculada por integração temporal vá aumentando ao longo do tempo em relação à realidade e, por conseguinte, a posição determinada afasta-se da verdadeira. Isto justifica o comprimento exagerado do último trajeto retilíneo. Relativamente aos dados do giroscópio podem considerar-se bastante certos, como também já se tinha verificado, uma vez que as curvas estão representadas corretamente sem grandes desvios relativamente ao original. Sublinha-se, novamente, o facto de este percurso não estar enquadrado com um referencial conhecido. Por acaso, nesta situação, a direção com que o robot iniciou o movimento coincidiu com o resultado deste cálculo da sua posição, mas se tivesse realizado o mesmo trajeto num outro ângulo relativamente às coordenadas terrestres, já não se verificava esta situação.

Bússola

A bússola funciona através da conversão dos campos magnéticos em cada eixo em tensões, sendo possível conhecer a orientação de cada um relativamente a um ponto de referência - o Norte magnético da Terra. Esta medida é especialmente importante, pois permite orientar o movimento do robot num plano de referência bem definido, facilitando a posterior integração da informação dos restantes sensores sem referência (*encoders* e *IMU*) com o GPS. Uma vez que os dados da bússola não são gerados com a mesma frequência dos restantes dados *IMU*, são indicados com zero na *string IMU* recebida quando não existem novos valores.

No entanto, após a análise dos dados efetivamente gerados pelo sensor, chegou-se à conclusão de que não poderiam ser utilizados, uma vez que se apresentam muito ruidosos e sem qualquer sentido, como se pode perceber na Figura 3.21, relativamente ao percurso da Figura 3.15.

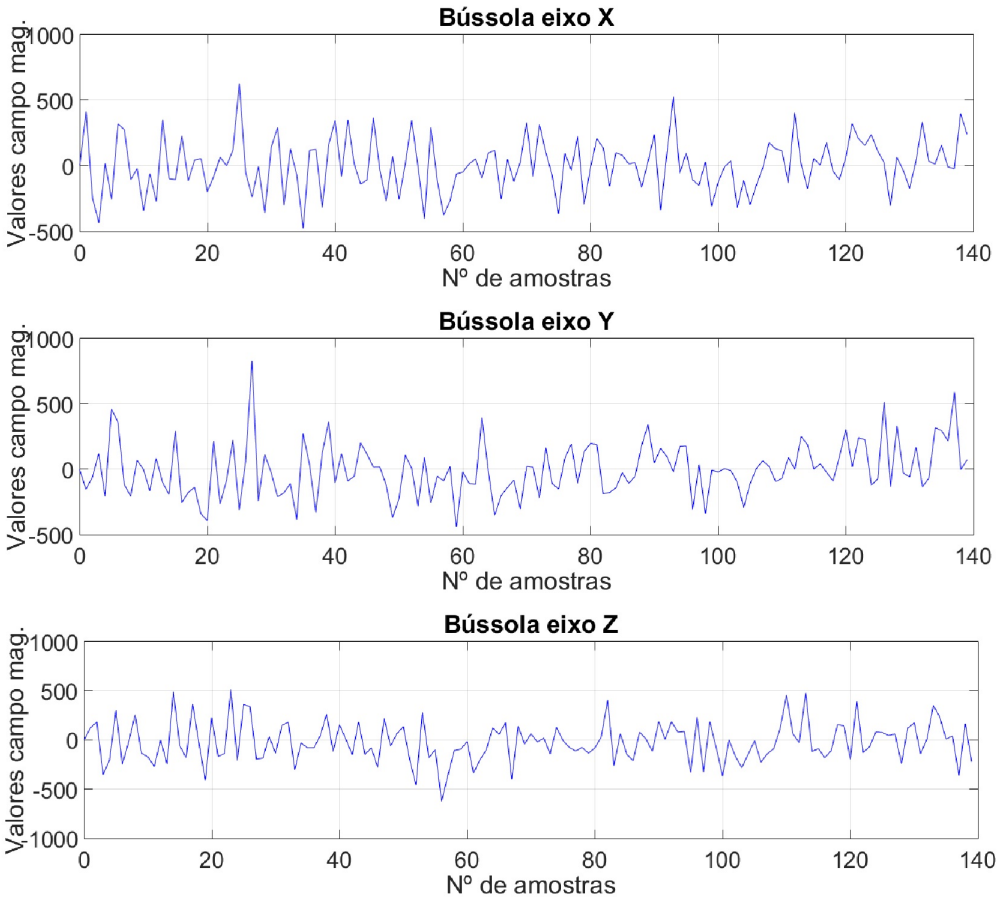


Figura 3.21: Dados em bruto gerados pela bússola magnética nos três eixos relativos ao percurso da Figura 3.15.

Neste contexto, uma vez que o robot circulou com, sensivelmente, 3 direções principais, esperava-se que os dados, pelo menos no eixo X e Y, apresentassem ao longo do gráfico, leituras aproximadas a 3 valores distintos, o que traduziria essas 3 mudanças de direção mais significativas, o que, efetivamente, não se verifica.

Excluindo a possibilidade de existir uma avaria no sensor, este conjunto de dados errados poderá dever-se ao facto de o interior do robot estar repleto de elementos que podem gerar campos magnéticos parasitas e corromper as leituras que deveriam ser relativas ao campo magnético da Terra. Alguns dos componentes que podem causar os erros de leitura são os motores elétricos das rodas, cabos elétricos, bateria ou modem *wireless*.

Como se decidiu descartar os dados da bússola, é necessário outro método para determinar a direção do robot face ao referencial Terra. Assim, são utilizados os dados GPS com esse objetivo.

3.4.2 *Encoders*

Os dados dos *encoders* são recebidos em forma de *string* com caracteres *ASCII* via *gateway* definido pelo programa de navegação com a placa de controlo do robot. Estes dados contêm os valores dos *encoders* das rodas da frente esquerda e direita. Com eles é possível saber o quanto cada roda girou entre cada verificação. Assim, é recebido um valor de 0 a 32767 ao qual vai sendo somado o número de passos que os *encoders* andaram entre verificações. Considerando que uma volta completa de uma roda corresponde a 150 passos (DrRobotInc, 2014) e que as rodas têm um diâmetro de 27cm, é possível calcular a distância em metros percorrida.

Um exemplo da *string* gerada:

`"#M,17530,11395"`

Uma vez que os eixos de rodas de lados diferentes giram em direções diferentes, o valor relativo a uma das rodas é crescente no intervalo referido (0 a 32767) enquanto que o outro valor é decrescente. Quando os limites do intervalo são atingidos, a contagem recomeça do limite oposto. Pode observar-se na Figura 3.22 esta evolução dos valores das contagens dos passos dos *encoders* para cada roda para o percurso da Figura 3.15.

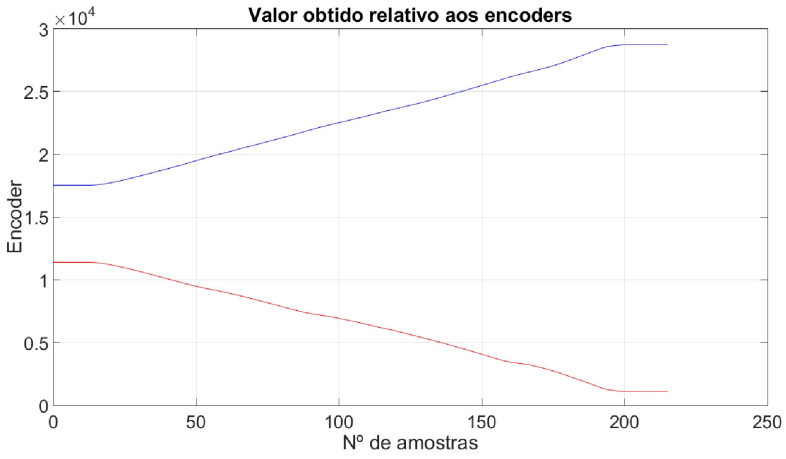


Figura 3.22: Evolução dos valores das contagens dos passos dos encoders o percurso da Figura 3.15

Reconstitui-se agora, na Figura 3.23 o percurso de teste realizado (Figura 3.15) com base nos dados dos encoders e giroscópio.

Nesta reconstituição (Figura 3.23), é possível visualizar uma reconstrução do percurso realizado bastante próxima da realidade, sobretudo se se comparar com a reconstituição obtida a partir dos dados do acelerómetro. Neste caso, a distância percorrida é determinada com a média da distância de cada uma das rodas, para se obter o percurso do centro do robot, mais certo do que se se considerar apenas uma

Figura 3.23: Reconstituição do percurso de teste com base nos dados dos encoders e giroscópio.

roda. Nesta situação evita-se a propagação de erros que acontecia com a utilização do acelerómetro, no entanto, este problema continua nos dados do giroscópio.

Outra abordagem que pode ser feita é a utilização da diferença de rotação entre as rodas para determinar as curvas realizadas pelo robot e, uma vez mais, obter uma reconstrução do percurso realizado. Na Figura 3.24 pode observar-se o resultado desta nova abordagem.

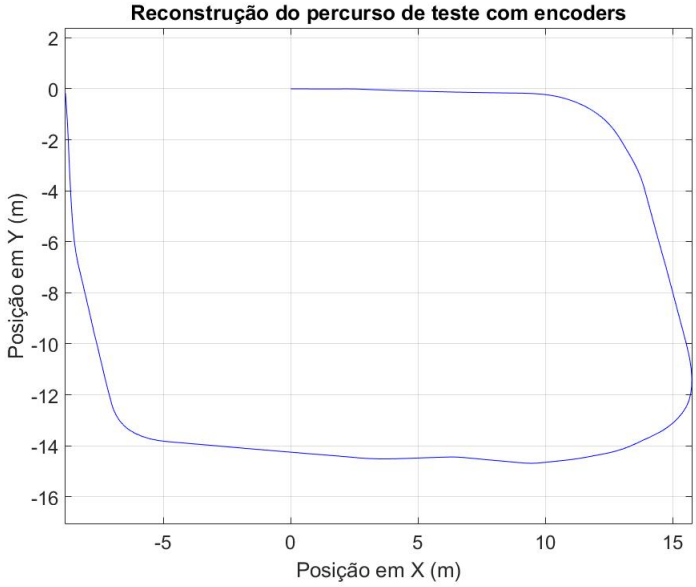


Figura 3.24: Reconstituição do percurso de teste com base nos dados dos encoders.

Esta forma de determinar o percurso exige que se obtenham valores para o ângulo de viragem do robot, que não é uma medida definida, uma vez que não tem rodas direcionais e está dependente de fatores como a velocidade os as características do terreno. Realizou-se então um conjunto de testes que permitiram determinar o valor que melhor se ajustava na maioria das situações. Considerou-se então que o raio de viragem do robot é de 1.1 metros.

O principal problema que se relaciona com a utilização dos *encoders* nos métodos das Figuras 3.23

e 3.24 é o escorregamento das rodas verificado durante o movimento do robot. No decorrer de diversos testes, verificou-se que este escorregamento pode ser significativo, sobretudo quando o robot troca a direção do deslocamento e realiza curvas. Isto deve-se ao elevado binário que é característica de motores elétricos como os do robot, as características do terreno e o peso do robot que não é suficiente para fazer os pneus aderirem convenientemente ao pavimento. Este escorregamento vai comprometer gravemente o cálculo da distância percorrida e a definição das curvas realizadas. Outro problema que persiste é a falta de referência para enquadrar o percurso reconstituído num referencial conhecido.

3.4.3 Sensor GPS

Os dados do sensor GPS são, então, muito importantes para a estimativa de posição realizada pelo algoritmo, pois são a referência que se dispõe para situar o movimento do robot na superfície terrestre. Estes dados são recebidos via *socket TCP* em forma de *String* com caracteres *ASCII* e baseiam-se no protocolo NMEA 0183, que é uma especificação de dados para comunicação entre dispositivos eletrónicos e define estruturas de informação que são enviadas pelos sistemas (para obtenção de informação detalhada sugere-se a consulta da sua documentação).

A *string* iniciada por "$GPRMC"é uma das estruturas definidas pelo protocolo e é aquela que tem relevância para o algoritmo, uma vez que contém os dados obtidos pelo GPS relativos à posição do robot. Entre outras informações, esta frase contém a hora e data da obtenção da informação e os valores atuais para a latitude e longitude. Um exemplo:

"`$GPRMC,232842.0,V,3843.62200,N,00908.32178,W,,150416,003.2,W*70`".

Com a obtenção de novas coordenadas GPS é também possível determinar a direção do movimento do robot, relativamente ao referencial Terra, ou seja, é possível saber se o robot se desloca para Norte ou Oeste, por exemplo.

Como referido na secção 2.4.1, os dados obtidos pelo sensor GPS são significativamente afetados por diversos fatores que podem vir a comprometer de forma importante a localização calculada com base nesta informação. A determinação da direção instantânea do robot - valor importante para o método de navegação implementado - depende diretamente dos dados recebidos do sensor GPS, e sofre com a eventual falta de precisão destes dados.

Se a precisão dos dados do sensor GPS estiver estiver afetada, a posição obtida pode distanciar-se alguns metros da verdadeira e até mesmo "saltar"para posições diversas em torno da posição verdadeira, o que provoca alterações muito acentuadas na orientação do veículo, mesmo se estiver parado.

Devido a estas condicionantes, é necessário encontrar um equilíbrio entre os dados disponibilizados pelos sensores *dead reckoning* e o GPS que é aquele que permite obter os pontos de referência necessários para enquadrar a restante informação. A utilização de filtros de Kalman é determinante na definição deste equilíbrio.

Assim, descreve-se de seguida, a metodologia desenvolvida no sentido de se equilibrar a informação recebida pelos vários sensores com o objetivo de utilizar filtros de Kalman para reduzir o erro associado á informação gerada com a fusão dos dados dos diversos sensores.

3.5 Algoritmo *Matlab* desenvolvido

Sublinha-se que o algoritmo desenvolvido na realização deste trabalho tem o principal objetivo de estimar a posição instantânea do robot. Esta estimativa substitui a que é realizada pelo programa original por intermédio das funções incluídas no ficheiro *DLL* referido. Com isto, não só se pretende a definição de um novo método de processamento de dados que seja compreendido e adaptado às diferentes necessidades, bem como a utilização de métodos de filtragem que envolvam filtros de Kalman para que seja possível tirar vantagens das suas características e mais-valias.

Assim, a partir da informação dos sensores disponível (*IMU*, GPS e *encoders*), foi desenvolvido um

algoritmo que procurou usar a maior quantidade de dados possível, com o objetivo de minimizar a perda de informação na determinação dos resultados. No entanto, não foi utilizada a totalidade da informação por motivos que serão explicados no decorrer desta secção.

O modelo base para o desenvolvimento deste algoritmo foi o referido no trabalho de (Cappelle *et al.* , 2006). Este trabalho propõe uma utilização do filtro de Kalman linear na filtragem do erro entre os dados odométricos (*INS - Inertial Navigation System*) e os recebidos do sensor GPS e a realimentação deste erro filtrado nos dados de localização odométrica. Na Figura 3.25 pode observar-se o diagrama esquemático desse algoritmo. Esta metodologia de *feedback* direto é então aplicada neste algoritmo para estimativa da posição instantânea do robot *Jaguar*.

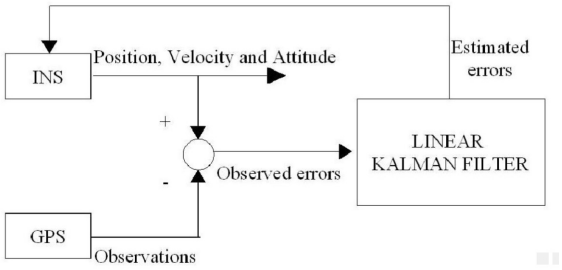
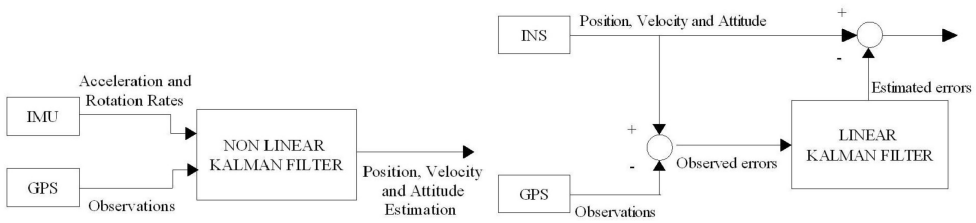


Figura 3.25: Modelo de feedback direto proposto por (Cappelle et al. *, 2006) para fusão de dados GPS/IMU.*

Outras opções, também enunciadas no artigo de (Cappelle *et al.* , 2006), são as ilustradas na Figura 3.26.



(a) Filtragem direta. *(b) Feedback indireto*

Figura 3.26: Alternativas ao modelo proposto para fusão de dados GPS/IMU. (Cappelle et al. *, 2006)*

No entanto, consideram-se menos aconselháveis relativamente ao *feedback* direto da Figura 3.25 pelo seguinte: a filtragem direta (Figura 3.26a) apresenta como desvantagem o facto de ser necessária a implementação de filtros de Kalman não lineares, opção que foi colocada de parte pelos motivos enunciados na Subsecção 3.3.6. O método de *feedback* indireto (Figura 3.26b) tem o problema de nunca corrigir o cálculo dos valores de *INS*, o que provoca um crescimento contínuo do erro entre estes e o GPS (Cappelle *et al.* , 2006).

Descrevem-se, agora, os detalhes da implementação e desenvolvimento deste algoritmo e quais os objetivos de cada pormenor dessa implementação.

3.5.1 Funcionamento geral do algoritmo implementado

Na Figura 3.27 observa-se um esquema do funcionamento geral do algoritmo implementado em *Matlab*. Este algoritmo substitui as chamadas realizadas pelo programa de navegação original (*Jaguar Navigation Demo*) às funções do ficheiro *DLL* que eram necessárias para a estimativa da posição instantânea do robot. Assim, este algoritmo passa a ser chamado na etapa "Processa dados recebidos"da Figura 3.13, substituindo então as funções originais.

Relembra-se que quando o programa de navegação *Jaguar navigation demo* recebe uma *string* do *socket* do GPS ou *encoders* coloca-a, sob a forma de variável, no *workspace Matlab*. Esta variável

tem o nome "entrada"e, após isso, o programa de navegação chama o algoritmo *Matlab* que tem como argumento esta variável "entrada". Neste momento dá-se o processamento em *Matlab* para a informação existente na variável. Entretanto, uma vez que as chamadas de funções *Matlab* são bloqueantes, o código do programa de navegação fica parado até receber resultados. Como se estão a utilizar *mutexes* (como referido na Subsecção 3.3.5), garante-se que o algoritmo *Matlab* não é chamado a partir de uma outra *thread* do programa de navegação.

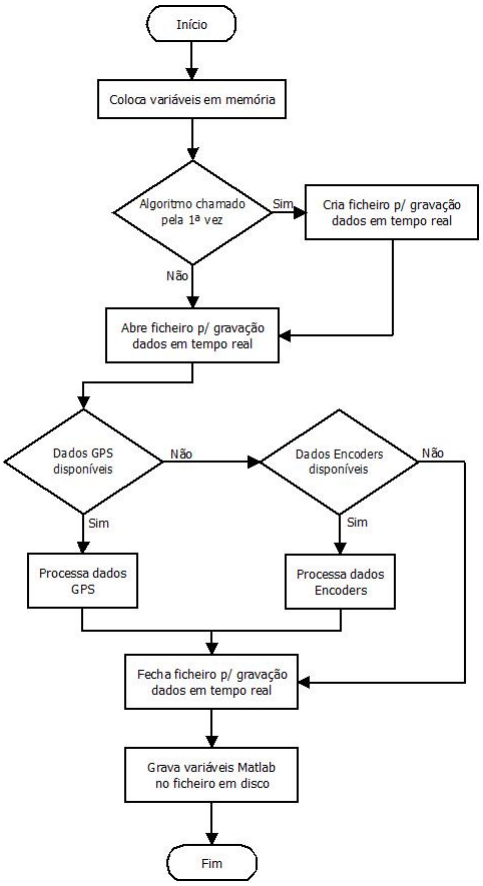


Figura 3.27: Descrição do funcionamento geral do algoritmo.

Como descrito na Figura 3.27, o algoritmo começa por colocar em memória as variáveis guardadas no final da chamada anterior num ficheiro em disco. Isto deve-se ao facto de não ser possível manter o estado do ambiente *Matlab* entre as chamadas das funções se as variáveis não forem todas globais. Na primeira vez que o algoritmo é chamado, o programa de navegação envia uma instrução ao *Matlab* criando um ficheiro com todas as variáveis necessárias definidas com o valor zero que pode então ser lido a partir deste momento.

De seguida é aberto ou criado um ficheiro de texto que é utilizado para gravar os valores recebidos e calculados no decorrer dessa sessão de navegação do robot. Isto permite guardar os dados em tempo real facilitando análises posteriores e a observação dos resultados.

Dependendo do evento que chama o algoritmo *Matlab* (*thread* do programa de navegação que trata os dados recebidos pelo GPS ou evento iniciado pela receção dos dados dos *encoders*), é realizado o processamento correspondente a cada tipo de dados (explicado no decorrer desta Secção). Se, por outro lado, o algoritmo for chamado por outro motivo (acontecimento que não deve ocorrer) não é realizado qualquer tipo de processamento.

Após as ações descritas, encerra-se o ficheiro de gravação de dados em tempo real, são gravadas as variáveis utilizadas nesta sessão e o algoritmo é terminado.

Por aqui pode observar-se que os dados gerados pelo *IMU* não são utilizados. Esta escolha deve-se

aos erros que advêm da necessidade de integração dos dados dos sensores que são muito significativos e prejudicam o cálculo da posição estimada. Na verdade, os dados obtidos dos *encoders*, apesar de não estarem livres de erros, fornecem informação mais correta que substitui aquela que poderia ser obtida com o acelerómetro e giroscópio.

3.5.2 Processamento dos dados GPS

Como se indica na Figura 3.28, assim que se inicia o processamento dos dados GPS, os valores relativos às coordenadas recebidas são convertidos para graus decimais. No caso de ser a primeira vez que o algoritmo está a receber dados do GPS, as coordenadas são guardadas como sendo o ponto inicial e o algoritmo é encerrado.

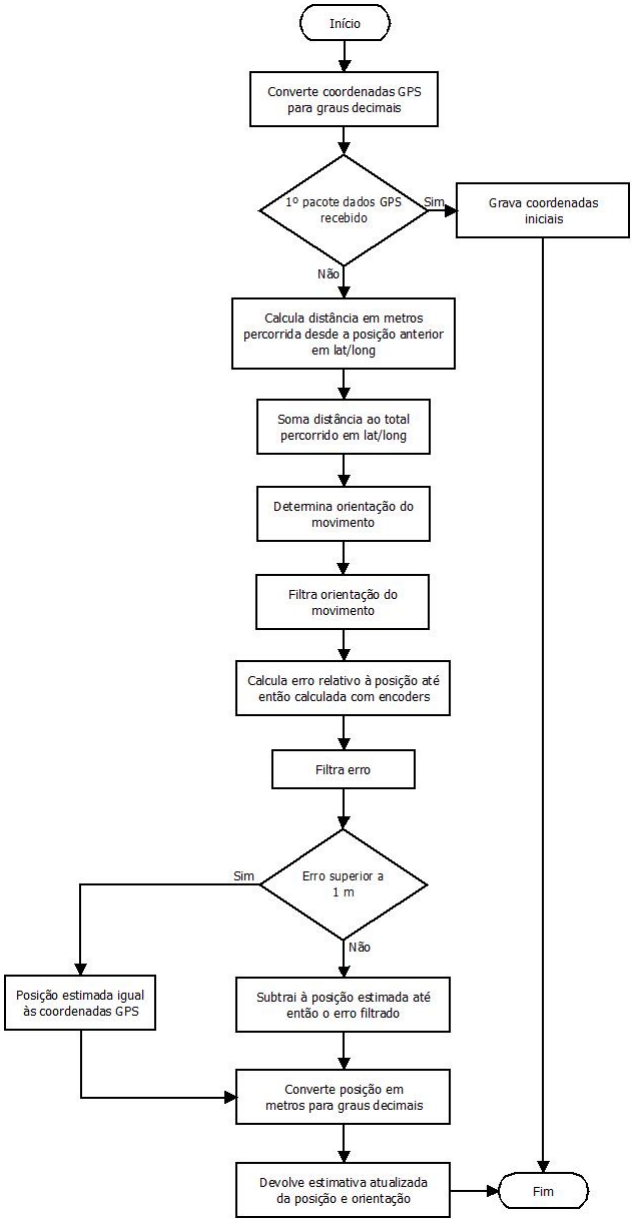


Figura 3.28: Descrição do processamento geral dos dados do GPS.

Se já não for a primeira vez que o algoritmo recebe informação deste tipo, é calculada a variação de distância em metros com base na diferença das coordenadas da posição anterior para a atual. Esta distância é calculada relativamente à longitude e latitude e é somada à distância até então percorrida.

Isto permite criar uma representação 2D em latitude/longitude do movimento do robot cujo início se dá na coordenada (0,0) e a sua evolução é representada pela variação da distância em metros percorrida em latitude e longitude, como representado na Figura 3.15. Para converter esta representação em coordenadas considera-se que a sua origem corresponde ao ponto inicialmente guardado e os pontos percorridos têm as coordenadas correspondentes à sua distância relativamente à origem. Desta forma é possível utilizar a informação dos *encoders* diretamente, uma vez que devolvem a distância percorrida pelas rodas. No entanto, esta representação 2D gerada pelo GPS não deve ser confundida com uma outra, do mesmo género, que é criada com o avanço determinado pelos dados das rodas.

Com base na inovação das coordenadas é também calculada a direção do movimento do robot em radianos face ao sistema de coordenadas, por exemplo, se um par de coordenadas recebido apresentar uma variação positiva da latitude face às coordenadas anteriores e uma variação nula na longitude sabe-se que o robot andou na direção do Norte geográfico e a sua orientação é de zero radianos. Se, noutro caso, só existir variação da longitude sabe-se que o robot se deslocou ou para Este ou Oeste e a sua orientação será de $\pi/2$ ou $-\pi/2$ radianos. Esta orientação é especialmente importante para o processamento dos dados dos *encoders*, pois, este valor é utilizado para orientar o avanço por eles determinado relativo ao movimento das rodas, substituindo a informação que poderia ser obtida pela bússola no enquadramento da deslocação no referencial de coordenadas.

Esta orientação pode ser especialmente ruidosa se o sinal GPS estiver afetado, uma vez que a localização determinada pode não coincidir exatamente com a real e pode alternar em posições em torno desta. Para evitar este problema, utiliza-se um filtro de Kalman 1D para retirar ruído a esta medida, sendo possível obter variações mais suaves desta orientação.

O próximo passo realizado pelo algoritmo consiste na comparação da última posição determinada (obtida com dados dos *encoders*) com a agora recebida do GPS. É calculada a diferença (erro) de latitude e longitude das duas posições e este erro relativo a cada uma é filtrado com um filtro de Kalman 1D. O erro filtrado de latitude e longitude é então subtraído aos valores de latitude e longitude da posição atual estimada e obtém-se assim um par de coordenadas que contém a informação filtrada e atualizada acerca da posição atual do robot. Este par de coordenadas é incluído na segunda representação 2D referida (relativa aos dados das rodas).

Recordando as equações (2.1) e (2.2), no método aqui utilizado considera-se que o sistema linear variável no tempo representa o erro entre a posição obtida pelo GPS e a determinada com os dados dos *encoders*. Assim, nas equações referidas, y_k é o erro efetivamente observado, x_k é o erro real no instante k e x_{k+1} é o erro no instante $k+1$. O filtro de Kalman estima o erro neste instante $k+1$, para ser possível atualizar a posição instantânea do robot.

No entanto, se o erro determinado for superior a um determinado valor, presume-se que pode existir um problema grave com a determinação da posição, pelo que se faz um recomeço da posição estimada igualando-a à posição GPS recebida. Isto evita que a posição calculada seja muito afetada no caso de, por exemplo, o robot se encontrar em terreno muito irregular e uma das rodas deixe de ter contacto com o solo e rode indefinidamente fazendo o algoritmo determinar um falso avanço de posição.

O algoritmo termina com a devolução do par de coordenadas da posição estimada atualizado para o programa de navegação e grava as variáveis de ambiente *Matlab* em disco.

3.5.3 Processamento dos dados dos *encoders*

Quando o algoritmo recebe informação do *encoder* calcula o quanto as rodas se deslocaram desde a última medida recebida e utiliza a orientação determinada pelo GPS para diferenciar o deslocamento em longitude e latitude, o que permite acrescentar informação à representação 2D do trajeto percorrido pelas rodas (referido na Subsecção 3.5.2). Esta orientação é utilizada se o algoritmo tiver obtido dados do GPS na iteração anterior, o que pode nem sempre acontecer porque os dados dos *encoders* são obtidos com maior frequência que os do GPS. Caso a informação obtida na iteração anterior não tenha sido do GPS, é provável que a orientação esteja desatualizada, então são utilizados os valores dos *encoders* de ambas as rodas para se determinar a diferença de distância percorrida por cada uma delas e então saber a variação da orientação do robot.

Após se determinar a variação da distância em metros percorrida pelo robot em latitude e longitude, as coordenadas são convertidas para graus decimais e devolvidas ao programa de navegação.

Pode observar-se na Figura 3.29 o esquema dos passos realizados para processamento dos dados dos *encoders*.

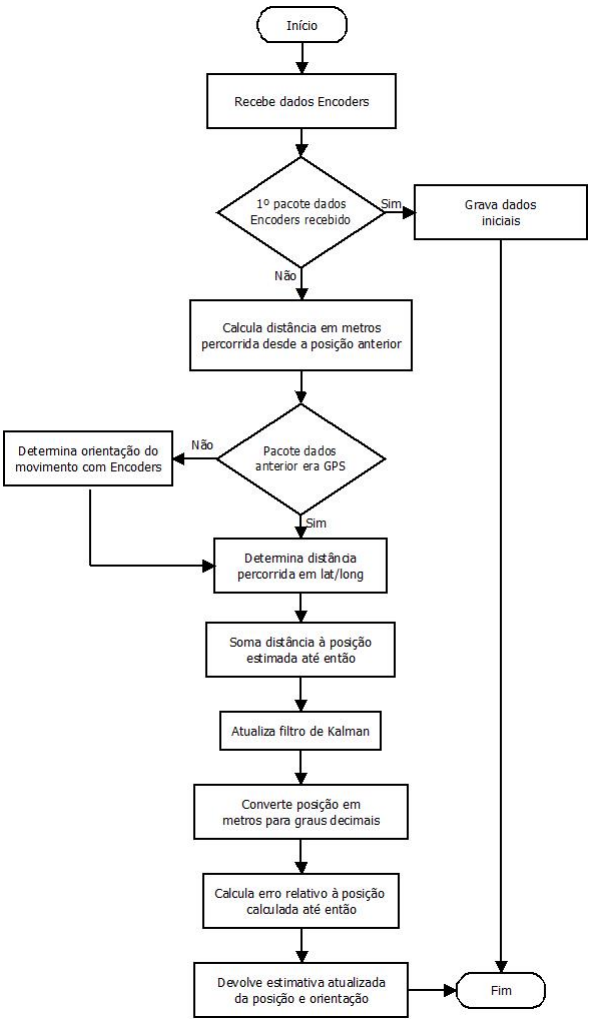


Figura 3.29: Descrição do processamento geral dos dados dos encoders.

Capítulo 4

Resultados experimentais

Após o desenvolvimento do trabalho descrito na Secção 3.5 obteve-se um algoritmo de navegação automática para o robot *Jaguar 4x4 Wheel* que funciona de forma bastante satisfatória. Isto significa que o robot é capaz de seguir uma trajetória previamente definida com erros reduzidos, sendo possível realizar tarefas com duração considerável mantendo a estabilidade do sistema.

No decorrer deste Capítulo serão descritos alguns resultados obtidos com o trabalho desenvolvido e serão referidos os principais fatores relevantes para cada situação. Será feita uma análise aos casos com o objetivo de se destacarem as qualidades e problemas que podem ser identificados.

Destaca-se que todos os testes foram realizados a 45% da potência máxima, que limita a velocidade máxima do robot a cerca de 6 km/h em terreno plano e rígido (estrada). Isto permite uma maior poupança da energia da bateria durante os testes e permite acompanhá-lo durante a realização dos percursos e pará-lo rapidamente no caso de se iniciar uma trajetória perigosa para o robot ou para o ambiente envolvente. Também se evita o aquecimento exagerado dos motores das rodas, e disponibiliza mais tempo para o processamento da informação.

Estes testes foram realizados no interior da Sede da Academia Militar em Lisboa em espaços abertos destinados à realização de desporto. Nestes locais foi possível operar o robot sem causar danos no equipamento nem em outras pessoas/objetos.

De seguida serão então descritos os testes realizados, sendo que a sua ordem/numeração pretende apenas a organização da informação.

Teste 1

Neste teste, ilustrado na Figura 4.1 realizou-se um percurso de cerca de 220 m. A linha verde indica o percurso previamente definido, que o robot deve seguir. A azul representa-se a posição determinada diretamente pelo GPS (com os erros associados) e a vermelho, a posição estimada pelo algoritmo. O percurso tem início na posição designada pela letra A e acaba na posição B.

Nas Figuras 4.2a e 4.2b podem observar-se partes desse percurso com maior detalhe.

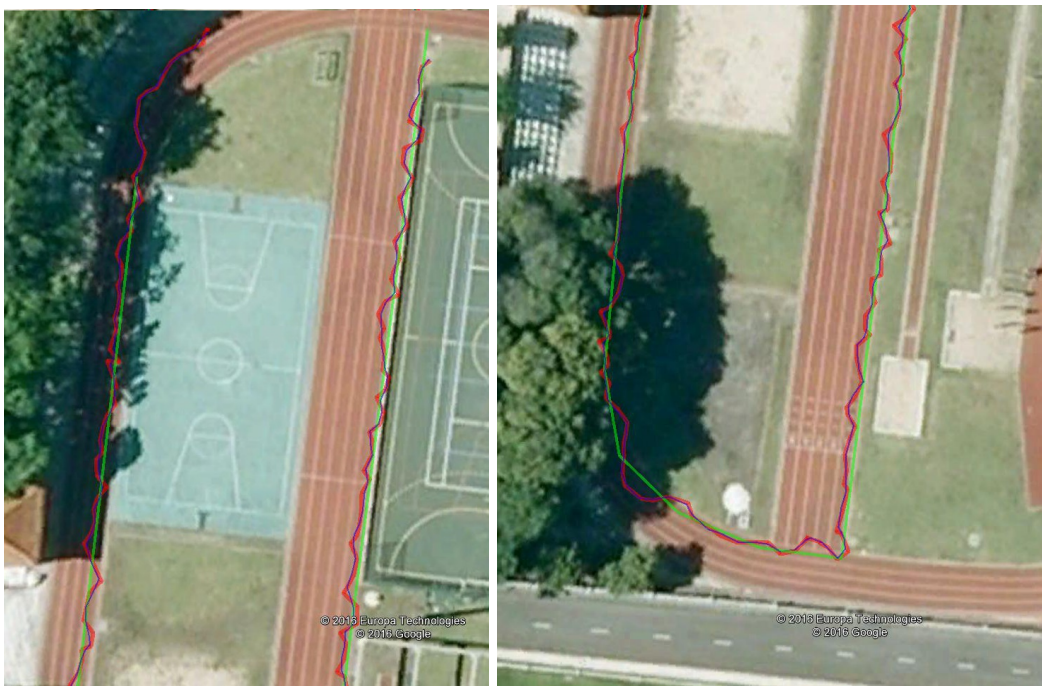
Os parâmetros mais relevantes deste teste são: - Utilização de uma margem de diferença de ângulos de direções (Figura 3.11) de 1º; - Filtragem da orientação obtida do GPS com filtro de Kalman; - Filtragem do erro entre a latitude e longitude estimadas e obtidas pelo GPS com Filtro de Kalman com um erro de medida de 5m e um erro de processo de 1 m.

Observa-se que o robot foi capaz de realizar este percurso sem se desviar mais de 1 m do percurso definido durante praticamente todo o trajeto. No entanto, existem localizações em que o erro é mais evidente, como por exemplo a curva na parte inferior da Figura 4.1, representada com maior detalhe na Figura 4.2b. Este erro está associado a perturbações no sinal GPS devido às árvores que existem nessa zona. O resto do percurso tem variações bruscas, que se devem às subtrações do erro filtrado entre as

Figura 4.1: Percurso realizado para o Teste 1.

posições GPS obtidas e as calculadas com os *encoders*. Relembra-se que a posição indicada nos mapas não corresponde exatamente a essa posição na realidade, o robot, no percurso percorrido, circulou sempre no centro das pistas e não junto aos extremos, como representado.

Na Figura 4.3 ilustra-se os valores para o erro de latitude e longitude gravados em tempo real no percurso realizado. A azul representa-se o erro entre a posição real e estimada não filtrado, a vermelho representa-se o mesmo erro após aplicação do filtro de Kalman. Este erro filtrado é aquele que é utilizado como *Feedback* no sistema (Figura 3.25) para ajuste da posição instantânea do robot determinada.



(a) *(b)*

Figura 4.2: Detalhes do percurso realizado para o Teste 1.

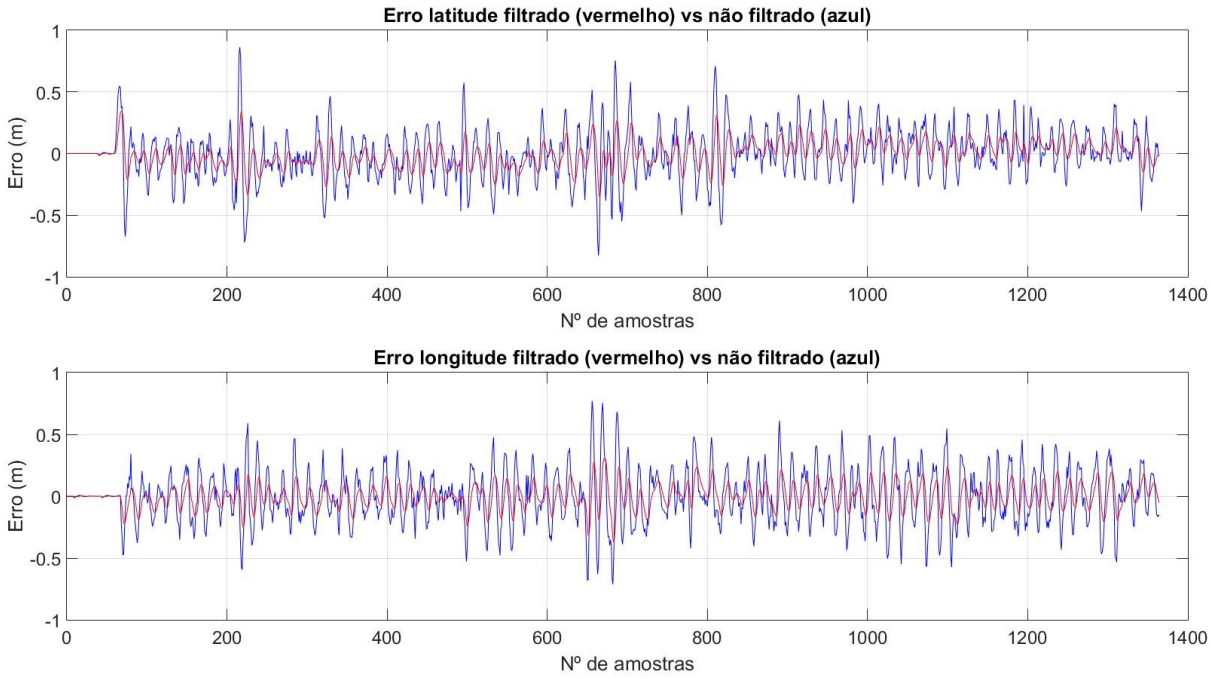


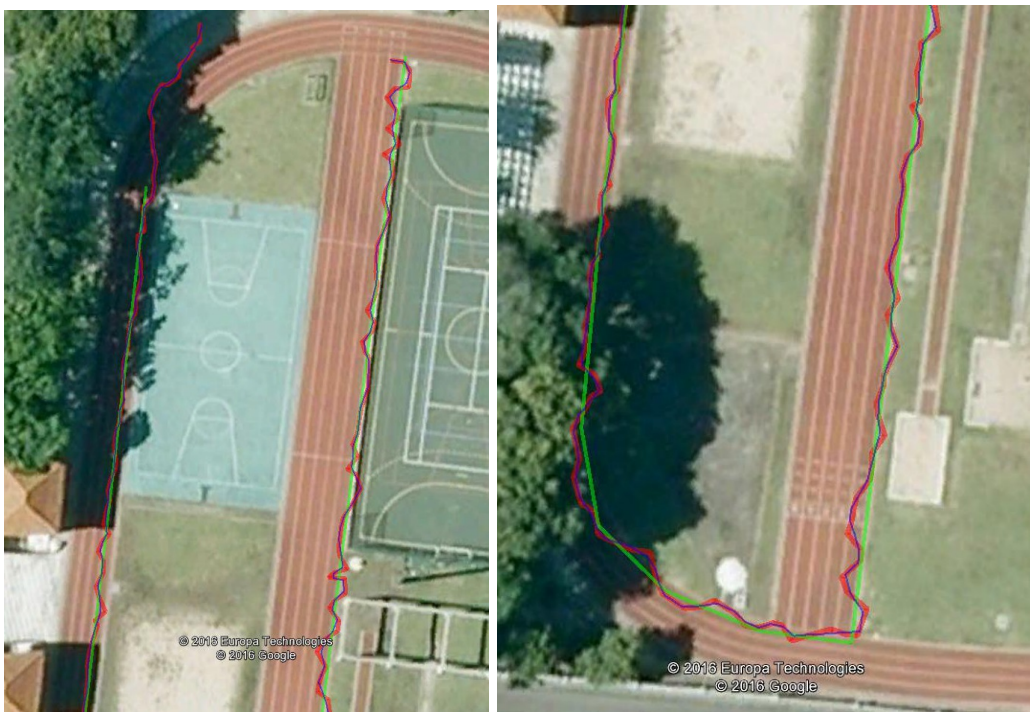
Figura 4.3: Erro de Latitude (em cima) e Longitude (em baixo) filtrado (vermelho) e não filtrado (azul) para o percurso da Figura 4.1.

Teste 2

Neste teste, repetiu-se o caso do Teste 1 com uma margem de diferença de ângulos de orientações de 3º (Figura 3.11), dando ao robot uma maior margem de erro na realização de curvas.

Na Figura 4.4 pode observar-se o percurso então realizado e na Figura 4.5 os detalhes do trajeto.

Figura 4.4: Percurso realizado para o Teste 2.



(a) *(b)*

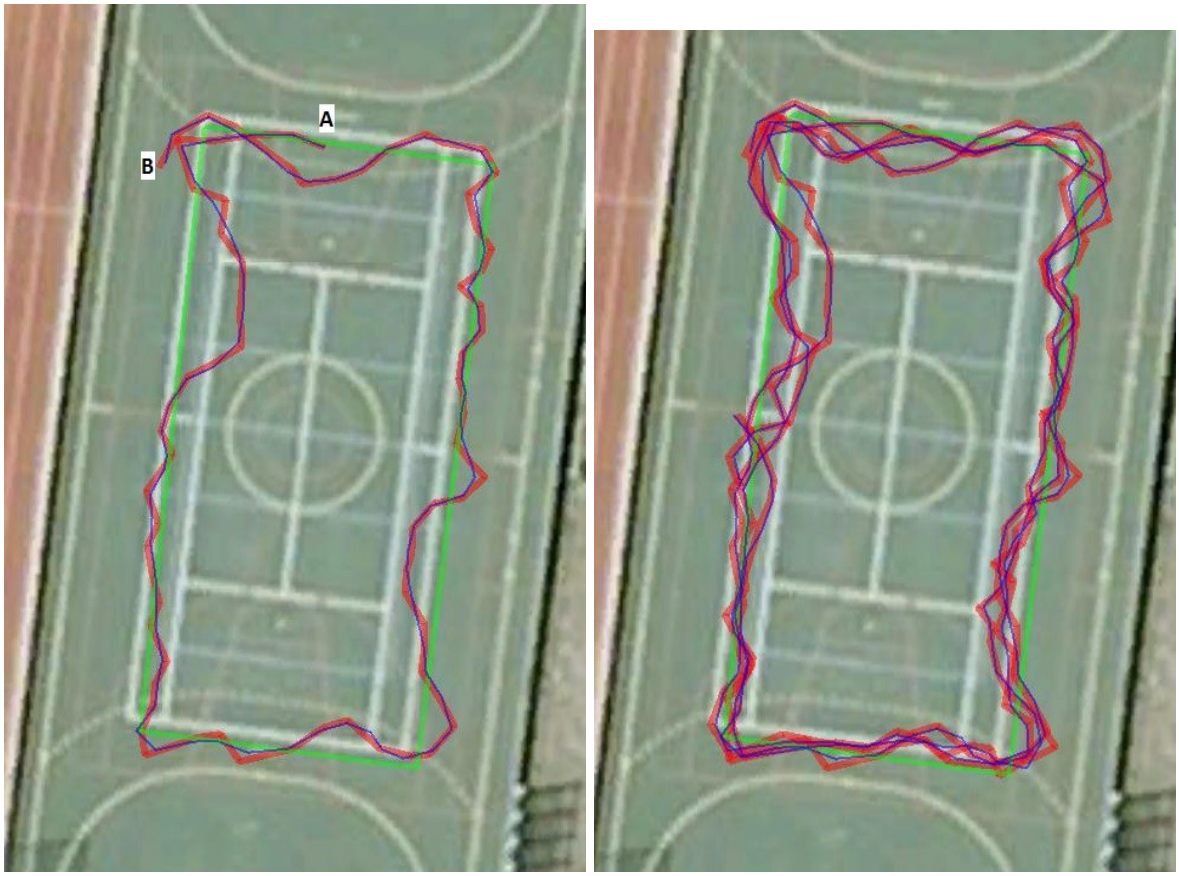
Figura 4.5: Detalhes do percurso realizado para o Teste 2.

Dada a alteração efetuada pode verificar-se que o efeito se reflete sobretudo na redução da quantidade de correções de direção que o robot realiza durante o percurso. No entanto, em determinadas situações, como o erro de direção pode ser superior ao do Teste 1, pode ocorrer um maior desvio no trajeto.

Teste 3

Neste caso, Figura 4.6, utilizaram-se as configurações do Teste 2 mas alterou-se o percurso a realizar. Este trajeto consiste num retângulo com dimensões aproximadas de 24m x 11m, com perímetro de cerca de 70m. Pretende-se, com este teste, observar o comportamento do robot num trajeto significativamente mais curto que os anteriores, onde a influência da incerteza associada aos erros do GPS aumenta devido à diminuição do percurso.

Na Figura 4.6a observa-se o resultado de uma única passagem com início em A em término em B. Na Figura 4.6b ilustra-se o resultado de três passagens consecutivas para verificação da constância da navegação, sendo possível observar que o trajeto efetivamente percorrido se mantém relativamente constante nas diversas passagens.



(a) *(b)*

Figura 4.6: Percurso realizado para o Teste 3.

Teste 4

Desta vez, como ilustrado na Figura 4.7, testou-se o robot num percurso bastante mais longo que nos testes anteriores. Este trajeto tem cerca de 620m e foi realizado em cerca de 10 minutos com início em A e fim em B. Pode observar-se que o robot conseguiu fazer o percurso na sua totalidade sem se desviar de forma excessiva do pretendido, como se pode ver nos detalhes nas Figuras 4.8 e 4.9.

Neste teste comprova-se a capacidade do algoritmo de navegação para ultrapassar zonas de sombra do sinal de GPS (provocada pelas zonas de árvores e edifícios) e para lidar com os erros e ruído da informação dos sensores, sendo possível manter a estabilidade e o funcionamento do sistema durante todo o percurso.

Figura 4.7: Percurso realizado para o Teste 4.

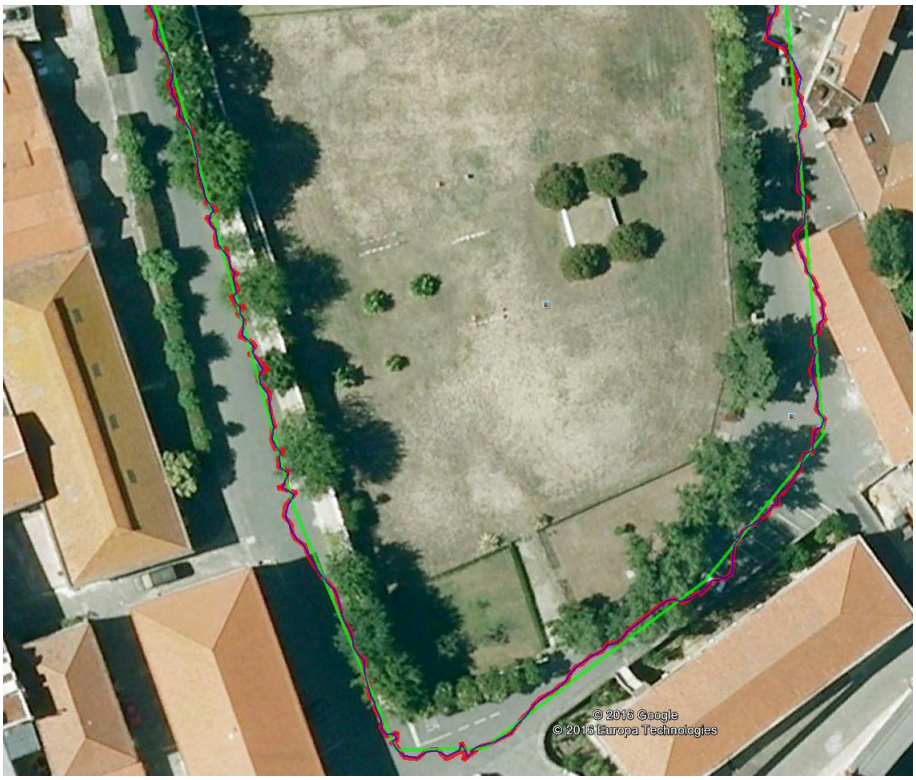


Figura 4.8: Percurso realizado para o Teste 4.

Figura 4.9: Percurso realizado para o Teste 4.

Capítulo 5

Conclusões

Este trabalho foi realizado no âmbito da Dissertação para obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores. Teve como objetivo o desenvolvimento de um sistema de navegação autónoma para o robot *Jaguar 4x4 wheel* recorrendo a filtros de Kalman.

Neste documento foi reunida alguma informação acerca do Estado da Arte dos métodos de navegação para sistemas móveis e utilização de filtros de Kalman, uma vez que este foi o conceito do trabalho realizado.

Esta reunião de informação permitiu um melhor conhecimento do tema e possibilitou o desenvolvimento mais cuidado de todo o trabalho em si, uma vez que permitiu a sua evolução com uma melhor noção das diversas soluções e estudos já realizados permitindo dar referências do que poderão ou não ser boas soluções e práticas.

A descrição do trabalho realizado procurou refletir a sequência de execução das tarefas, para que seja possível estabelecer uma ordem coerente e compreensível da evolução do trabalho. As dificuldades encontradas foram também referidas à medida que foram surgindo bem como as soluções encontradas.

Os testes realizados permitiram verificar o correto funcionamento do trabalho realizado mas, no entanto, naturalmente não estão livres de erros nem se poderão considerar ideais, pelo que existe sempre algo mais que poderá ser realizado ou melhorado. Todavia, o objetivo da dissertação foi plenamente alcançado, uma vez que os resultados são bastante satisfatórios.

Este relatório pretende não só a descrição do trabalho realizado para esta Dissertação, mas também o registo de aspetos importantes acerca do robot *Jaguar* e o seu funcionamento, podendo ser também utilizado como manual para apoio a outros trabalhos que podem vir a ser realizados no futuro que se relacionem com o assunto aqui tratado.

Destacam-se agora os principais aspetos mais trabalhosos que afetaram o desenvolvimento previsto da Dissertação:

- A inicial não disponibilidade funcional do robot levou a que fosse necessário resolver os problemas existentes. No entanto não afetou significativamente o trabalho porque foi possível resolver essas questões no semestre anterior ao da Dissertação.

- A não existência de documentação detalhada acerca dos programas originais levou a que fosse necessária a exploração exaustiva do seu código fonte, o que consumiu bastante tempo que não foi utilizado no desenvolvimento do algoritmo de navegação.

- A impossibilidade de utilização dos dados da bússola foram determinantes, uma vez que teriam sido muito úteis no processo de navegação. Como inicialmente se contava que poderiam ser utilizados, foi necessário empenhar algum tempo na tentativa de os determinar corretamente e, após isso, encontrar alternativas para os substituir.

- Como seria de esperar, a quantidade de dados envolvidos e o número elevado de parâmetros ajustáveis nos algoritmos tornou o processo de adaptação e ajuste do sistema de navegação mais lento,

exigindo uma grande quantidade de testes e revisões aos processos desenvolvidos. Também a existência de erros ou a falta de precisão em diversas medidas não considerados inicialmente levou a que fosse, muitas vezes necessário encontrar soluções para questões sobre as quais ainda não tinha existido reflexão.

Por outro lado, o apoio constante e motivação incansável dos professores orientadores permitiu o contínuo desenvolvimento do trabalho e a exploração de novas soluções. Também os seus conselhos e dicas foram determinantes para melhorar o trabalho, encontrando e resolvendo problemas que até então permaneciam desconhecidos.

Também a disponibilidade dos espaços da Academia Militar foi muito importante para a realização dos inúmeros testes necessários, que não poderiam ter sido realizados noutro espaço público pelo risco envolvido para o robot e demais utilizadores desse espaço. A robustez do robot e qualidade/estabilidade dos seus sensores foi também decisiva, uma vez que não houve necessidade de resolver problemas extra (eventualmente intermitentes) relacionados com *hardware* ou instabilidade nos sensores, permitindo observar melhor as alterações relativas a determinado parâmetro sem que fossem confundidas com outro pormenor qualquer. Destaca-se também a importância do programa *Google Earth* que permite o registo do movimento do robot e dos resultados da navegação e o programa *Matlab* que muito facilita a manipulação de dados numéricos e a gestão da imensa informação gerada.

Por fim, e apesar de, como referido, o trabalho ter alcançado os seus objetivos, torna-se necessário concluir que a conceção do Jaguar, baseada em 4 rodas motrizes fixas na carroçaria, apresenta sérias limitações à sua movimentação. Na verdade, as alterações a efetuar às trajetórias retilíneas resultam forçosamente numa derrapagem de, pelo menos, uma das rodas. Assim, os dados resultantes da odometria são sempre corrompidos o que muito dificulta estimativa da posição/orientação do robot. Resumindo, este robot não é adequado para navegação automática baseada numa estimativa da posição e orientação obtidas a partir de dados odométricos. Sabe-se agora que, para os efeitos e objetivos da tese, teria sido preferível a utilização de um robot móvel com sistema de direção em que, no mínimo, duas das rodas pudessem ser orientadas.

5.1 Principais contribuições do trabalho

Apesar de esta dissertação ter o objetivo principal de desenvolver um sistema de navegação para o robot *Jaguar*, focando-se sobretudo na filtragem dos dados obtidos pelos seus sensores, podem enumerar-se diversas outras contribuições e inovações que foram conseguidas no desenvolvimento do trabalho, sendo elas:

- Reunião de informação sobre sistemas de navegação desenvolvidos em trabalhos similares ao desta dissertação;

- A reparação do robot, que inicialmente não se encontrava funcional;

- Compreensão e documentação do comportamento do principal *hardware* do robot;

- Compreensão e documentação do funcionamento e da estrutura geral do programa de navegação originalmente fornecido com o robot;

- Descrição de um método de integração de rotinas em C# com programas desenvolvidos em *Matlab*;

- Descrição do funcionamento dos vários sensores e análise dos dados gerados.

5.2 Trabalho futuro

Como trabalho futuro sugere-se a adaptação de uma bússola externa ao sistema original do robot que esteja livre de interferências magnéticas e que possa comunicar com o computador por outra via que não *wireless* para não ser necessário alterar os componentes internos do robot. Esta comunicação poderia ser por exemplo *bluetooth* causando, no entanto uma significativa redução no raio de ação do

robot. A utilização da bússola que existe atualmente apenas será viável se se conseguirem isolar os campos magnéticos parasitas gerados pelos componentes internos do robot ou se se conseguir colocar o circuito noutro local do robot livre destas mesmas interferências. Se se detetar um funcionamento incorreto do dispositivo atualmente montado, poderá tentar-se uma troca direta do sensor mantendo toda a restante configuração do robot inalterada.

Se fosse possível enquadrar o movimento do robot com dados da bússola magnética, seria possível estimar a sua posição sem dependência do GPS, sendo apenas necessário para ir atualizando as coordenadas de posição. No caso deste trabalho, as orientações utilizadas carecem de precisão porque estão sempre dependentes do GPS que não tem resolução suficiente para interpretar corretamente os pequenos movimentos do robot.

Outra alteração interessante passa pelo desenvolvimento do algoritmo implementado em *Matlab* num formato executável (*DLL*, por exemplo), possibilitando a utilização do algoritmo fora deste ambiente *Matlab* e permitindo a execução mais rápida das instruções e a redução dos requisitos para o computador. Desta forma ficaria mais perto a possibilidade de implementar todo o sistema numa unidade de controlo reduzida (*System on Chip*) que pudesse ser embarcada no robot tornando-o completamente autónomo e independente de um computador permanentemente localizado no alcance da rede *wireless*.

Bibliografia

Agrawal, Motilal, Konolige, Kurt, & Bolles, Robert C. 2007. Localization and mapping for autonomous navigation in outdoor terrains: A stereo vision approach. *In: Applications of Computer Vision, 2007. WACV'07. IEEE Workshop on*. IEEE.

BostonDynamics. 2013. *BigDog - The Most Advanced Rough-Terrain Robot on Earth*.

Brega, R., Tomatis, N., & Arras, K.O. 2000. The Need for Autonomy and Real-Time in Mobile Robotics: A Case Study of XO/2 and Pygmalion.

Cappelle, C, Pomorski, D, & Yang, Y. 2006. GPS/INS data fusion for land vehicle localization. *Page 2127 of: Proceedings of the IMACS Multiconference on Computational Engineering in Systems Applications, Beijing, China*, vol. 46.

Cornell. 2008. *Kalman filter tank filling, subject MI63*.

DrRobotInc. 2014. *Jaguar 4x4 Wheel, All-Terrain Autonomous Navigation Robot with GPS-IMU*.

DrRobotInc. 2016. *Jaguar 4x4 Wheel*.

Garmin. 2016. *GPS 18x 5Hz*.

Hernández, S., Torres, J.M., Morales, C.A., & Acosta, L. 2003. A New Low Cost System for Autonomous Robot Heading and Position Localization in a Closed Area. *Autonomous Robots*, **15**(2), 99–110.

Honda. 2015. *The World's Most Advanced Humanoid Robot*.

João, Filipe. 2013 (Setembro). *Piloto Automático Sistema de Navegação Autónoma*. M.Phil. thesis, Instituto superior Técnico.

Ko, Joong Hyup, Kim, Wan Joo, & Chung, Myung Jin. 1996. A method of acoustic landmark extraction for mobile robot navigation. *Robotics and Automation, IEEE Transactions on*, **12**(3), 478–485.

Leonard, John J., & Durrant-Whyte, Hugh F. 1991. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, **7**(3), 376–382.

Leonard, John J., & Durrant-Whyte, Hugh F. 1992. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers.

Maybeck, Peter S. 1982. *Stochastic models, estimation, and control*. Vol. 1. Academic press.

Melo, Leonimer Flávio de, Rosário, João Mauricio, & Junior, Almiro Franco da Silveira. 2013. Mobile Robot Indoor Autonomous Navigation with Position Estimation Using RF Signal Triangulation. *Positioning*, 20–35.

Ning, Xiaolin, & Fang, Jiancheng. 2008. Spacecraft autonomous navigation using unscented particle filter-based celestial/Doppler information fusion. *Measurement Science and Technology*, **19**(9), 095203.

Ribeiro, Maria Isabel. 2004. Kalman and extended kalman filters: Concept, derivation and properties. *Institute for Systems and Robotics*, **43**.

Roumeliotis, Stergios I, & Bekey, George A. 1997. Extended Kalman filter for frequent local and infrequent global sensor data fusion. *Pages 11–22 of: Intelligent Systems & Advanced Manufacturing*. International Society for Optics and Photonics.

Siegwart, Roland, & Nourbakhsh, Illah R. 2004. *Introduction to Autonomous Mobile Robots*. The MIT Press.

Simon, Dan. 2001. Kalman filtering. *Embedded systems programming*, **14**(6), 72–79.

Simon, Dan. 2006. Using nonlinear Kalman filtering to estimate signals. *Embedded Systems Design*, **19**(7), 38.

Sparkfun. 2016. *9 Degrees of Freedom - Razor IMU*.

Suliman, Caius, Cruceru, Cristina, & Moldoveanu, Florin. 2009. Mobile robot position estimation using the Kalman filter. *Department of Automation, Transilvania University of Brasov, Brasov*, 1–3.

Welch, Greg, & Bishop, Gary. 2006. An introduction to the kalman filter. 2006. *University of North Carolina: Chapel Hill, North Carolina, US*.

Yamaha. 2015. *MOTOBOT ver.1*.

Yi, Soo-Yeong, & Choi, Byoung-Wook. 2007. Autonomous Navigation of Indoor Mobile Robot Using Global Ultrasonic System. *Mobile Robots: Perception and Navigation*.

Apêndice A

Módulos do robot Jaguar

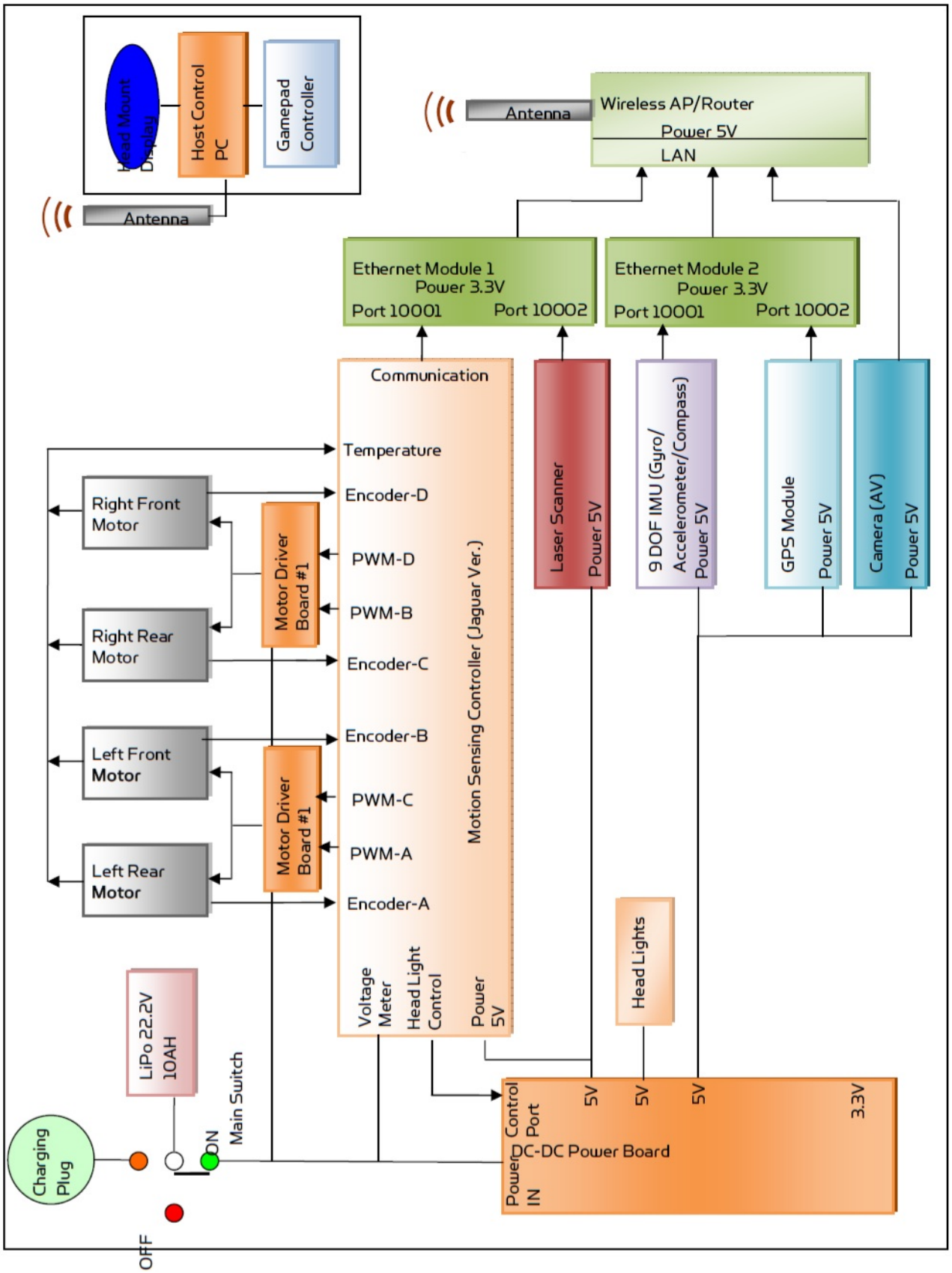


Figura A.1: Diagrama ilustrativo das interconexões entre os circuitos eletrónicos e módulos do robot (DrRobotInc, 2014)