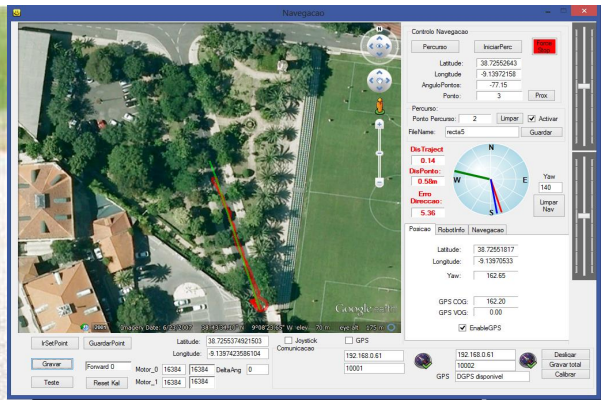
TÉCNICO LISBOA

ACADEMIA MILITAR
DULCE ET DECORUM EST PRO PATRIA MORI



Piloto Automático

Sistema de Navegação Autónoma

Filipe Alexandre Valdeira João

Dissertação para obtenção do Grau de Mestre em

Engenharia Electrotécnica e de Computadores

Júri

Presidente: Professor Doutor João Manuel Torres Caldinhas Simões Vaz

Orientador: Professor Doutor José António Beltran Gerald

Vogal: Professor Doutor Rodrigo Martins de Matos Ventura

Setembro 2013

Agradecimentos

Gostaria de começar por agradecer aos meus orientadores, Professor Doutor José António Beltran Gerald e ao Professor Doutor António Joaquim Serralheiro, o apoio dado durante o projecto, a sua participação activa e os esforços realizados para disponibilizar todos os recursos necessários.

Deixo também uma palavra especial ao Director de Curso de Transmissões da Academia Militar, Tenente-Coronel de Transmissões, Engenheiro João Pedro Pereira Bastos Rocha, pelo apoio e pela preocupação empregues na minha formação.

Agradeço a todos os meus amigos da Academia Militar, que me apoiaram constantemente durante o meu percurso militar e académico ao longo destes seis anos e ainda aos meus amigos de longa data pelo apoio e amizade de sempre.

Quero agradecer à pessoa que esteve sempre comigo, compreendendo e motivando mesmo nos momentos mais críticos, a minha namorada, Joana Vieira.

Agradeço aos meus irmãos, Susana João, Tiago João e Reinaldo César João por quem tenho grande estima e que sempre me acompanharam, aconselharam e ajudaram.

Finalmente, e como os últimos são os primeiros, agradeço à minha mãe, Maria Aurora João e ao meu pai, Reinaldo Dias João, o carinho e apoio incondicional prestados ao longo da minha vida, pelos ensinamentos e por me proporcionarem todas as condições para chegar até aqui.

A todos os mencionados e aos injustamente esquecidos, o meu muito obrigado!

Resumo

Desde o início do crescente interesse na área de robótica que a navegação autónoma se apresenta como um problema de complexa resolução que, por isso, desperta vasto interesse no meio científico. Além disso, as capacidades da navegação autónoma aliadas à robótica permitem o desenvolvimento de variadas aplicações.

O objectivo da navegação autónoma é conferir, a um dispositivo motor, capacidade de decisão relativa à locomoção. Para o efeito, utilizam-se sensores, como os sensores IMU, o receptor GPS e os *encoders*, para fornecer os dados essenciais à navegação. A dificuldade encontra-se no correcto processamento destes sinais uma vez que são susceptíveis a fontes de ruído.

Este trabalho apresenta um sistema de navegação autónomo aplicado ao controlo de um robot. Para tal, desenvolveu-se uma aplicação que alberga todo o sistema de localização, navegação e controlo, acrescido de uma interface gráfica, que permite a visualização em mapa da movimentação autónoma do robot. Recorre-se ao Filtro de *Kalman* como método probabilístico de estimação de posição, em que os sinais dos vários sensores são conjugados e filtrados.

Foram realizados vários testes de modo a avaliar a capacidade do robot atingir os pontos traçados e a sua autonomia no seguimento da trajectória pretendida.

Palavras-chave: navegação autónoma, localização, filtro de *Kalman*, filtragem de ruído, sensores de movimento

Abstract

Ever since the beginning of the growing interest shown in robotics, autonomous navigation presented itself as a complex problem. It has instigated a great deal of interest among the scientific community. Moreover, various applications arise from the combined capacities offered by the autonomous navigation and robotics.

The goal of autonomous navigation is to provide cognition in motor device locomotion. In order to provide the required navigation data sensors as the IMU sensors, GPS receptor and encoders, are used. These signals are prone to noise and, therefore, their correct processing is a main issue.

This work aims at studying in detail an autonomous navigation system and its implementation in order to empower a robot to control itself autonomously. For this propose, an application was developed which accommodates the entire system positioning, navigation and control systems. It also provides a graphical interface to allow the robot movement to be tracked in a map. This is achieved using a Kalman filter, where the different sensors signals are combined and filtered, in order to obtain a probabilistic evaluation of the position.

Tests were conducted to assess the ability of the vehicle to reach the designated points and to evaluate its autonomy on following the desired trajectory.

Keywords: autonomous navigation, autonomous robot, position estimation, Kalman filter, noise filtering, motion sensors.

Índice

Índice de figuras

Índice de Tabelas

Lista de Siglas e Abreviaturas

2D	Duas dimensões
3D	Três dimensões
AM	Academia Militar
AP	Access Point - Ponto de Acesso
API	Application Programming Interface
COG	Course Over Ground
DDP	Diferença de Potencial
DOF	Degrees of Freedom
DOP	Dilution of precision
ENU	East, North, Up coordinates, Cartesian coordinate system
EOD	Explosive Ordnance Disposal
EP	Exército Português
GPRMC	Recommended minimum specific GPS/Transit data
GPS	Global Position System - Sistema de posicionamento global
IA	Inteligência Artificial
IMU	Inertial measurement unit
IST	Instituto Superior Técnico
MRUV	Movimento Rectilíneo Uniformemente Variado
NASA	National Aeronautics and Space Administration – Administração Nacional da Aeronáutica e do Espaço
NAVSTAR	Navigation Satellite with Time and Ranging
PROMETHEUS	PROgraMme for a European Traffic of Highest Efficiency and Unprecedented Safety
SI	Sistema Internacional de Unidades
SRI	Standford Research Institute

UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
VOG	Velocity Over Ground
vSLAM	Visual Simultaneous Localization and Mapping

1 Introdução

Este trabalho descreve a realização de um sistema de navegação autónoma de um robot móvel terrestre. Este robot comporta vários sensores de navegação, nomeadamente, giroscópio, bússola magnética, acelerómetro, encoders e receptor de GPS. De forma a possibilitar que o Robot siga uma trajectória pré-definida, desenvolveram-se procedimentos e algoritmos de leitura dos sensores, filtragem dos dados provenientes destes e um modo de controlo dos motores.

A robótica autónoma móvel assenta na necessidade de navegar de forma não comandada através do espaço disponível, pelo que necessita do conhecimento da localização do robot em cada instante de tempo. Pressupõe-se, antes da navegação, a existência de um percurso ou trajectória que pode ser constituído por uma sucessão de pontos. No fundo, trata-se de saber para onde se vai (navegação), a partir de onde se está (localização) e de que forma (controlo).

Sendo a estimativa da localização e a navegação a partir dos dados de sensores um dos principais problemas da robótica móvel e somando à crescente importância destes robots autónomos em todo o mundo, esta dissertação adquire elevada relevância.

Neste primeiro capítulo apresentam-se as motivações, os objectivos, o trabalho realizado, o estado da arte e uma descrição sumária do robot utilizado na implementação dos algoritmos de navegação realizados.

1.1 Motivação

A robótica autónoma é um campo em grande desenvolvimento nos dias de hoje. O seu funcionamento depende de várias áreas da engenharia, nomeadamente a mecânica, a electrónica e a informática.

No meio militar, o interesse na capacidade autónoma de robots/veículos é de assinalado valor, pois, no futuro, pode significar a salvaguarda de vidas humanas ou a possibilidade de servir melhor os interesses da nação. Hoje em dia assiste-se ao desenvolvimento dos denominados UAV (Unmanned Aerial Vehicle) e UGV (Unmanned Ground Vehicle), (Figura 1.1 Veículos militares não tripulados, UGV (esquerda) e UAV (direita)), sendo estes de elevada importância, pois, como foi referido por Devdas Shetty[1], numa conferência nos Estados Unidos da América, *"The military role of UAV is growing at*

[1] Devdas Shetty Ph.D, PE, School of Engineering and Applied Sciences, Professor of Mechanical Engineering, University of Hartford, EUA

unprecedented rates…" [1], cada vez se recorre mais a estes meios independentes e não tripulados para as diversas actividades militares e civis.

Figura 1.1 Veículos militares não tripulados, UGV (esquerda) e UAV (direita)

Uma possível aplicação de robots com sistema autónomo no Exército Português (EP) é a videovigilância. No EP, as unidades militares possuem o seu sistema de segurança para o qual são destacados diariamente efectivos cuja finalidade é a vigilância das mesmas. Para reduzir o número de efectivos empenhados nestas funções, seria útil a existência de uma plataforma móvel autónoma para a vigilância de instalações militares.

Existem ainda outras aplicações possíveis onde empenhar estes robots autónomos móveis, como, por exemplo, em busca e desactivação de engenhos explosivos (EOD) ou em situações de combate.

Também em Abril de 2013 surgiu um projecto no âmbito da NASA, *"NASA Challenges Student and Citizen Inventor Teams to Build Autonomous Robots (...) Fourteen teams from across the country and around the globe are perfecting their hardware and software to compete for $1.5 million in prize money at NASA's 2013 Sample Return Robot Challenge"* [2], provando a importância conferida a este tema na actualidade.

Neste âmbito e sabendo das potenciais aplicações, obteve-se grande motivação na pesquisa e aprendizagem deste tema, bem como na elaboração desta dissertação.

1.2 Objectivos

Como anteriormente se referiu, a navegação autónoma de veículos, tanto terrestre como marítima ou aérea [3] tem obtido grande relevância na comunidade científica.

Por autonomia entende-se sistemas capazes de, por longos períodos de tempo, operarem em tempo real sem nenhuma forma de controlo externo [4].

O robot utilizado neste estudo, pertencente à Academia Militar, fornece à sua saída os sinais dos sensores de movimento instalados (giroscópios, acelerómetros, bússola magnética, encoders e receptor GPS), permitindo obter informações do seu movimento em tempo real. No entanto, esses sinais apresentam erros, pois são corrompidos por ruído, pelo que se torna difícil obter a sua localização real.

Esta dissertação visou estudar os métodos necessários (algoritmos de filtragem) e aplicá-los aos sinais recebidos, a fim de obter um conjunto de dados mais precisos, que permitiram obter uma melhor localização do robot. Neste trabalho intenta-se ainda estudar os procedimentos necessários para sincronizar os dados dos vários sensores de modo a obter um controlo adequado dos motores do robot, obtendo-se no final uma melhor navegação autónoma.

O algoritmo desenvolvido usa como base o modelo geral de Siegwart [5], Figura 1.2, onde se apresentam os vários blocos constituintes do sistema de navegação autónoma. Em suma, teremos dois blocos distintos, a saber: o bloco de controlo de movimento, constituído pela execução do trajecto e pela actuação dos motores e o bloco de percepção onde os dados dos sensores são extraídos, analisados e filtrados. Os capítulos 2, 3 e 4 descrevem estes blocos detalhadamente.

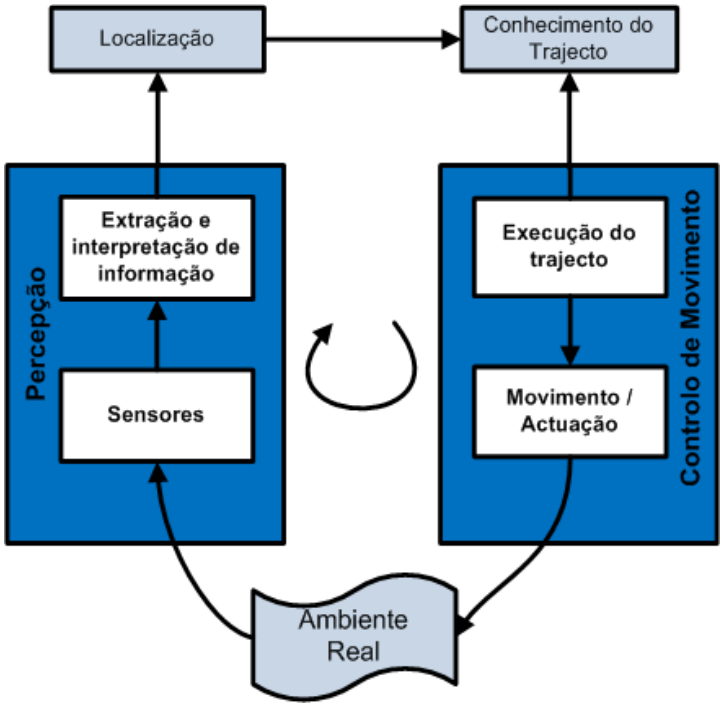


Figura 1.2 Esquema genérico de controlo de um robot autónomo

1.3 Estado da Arte

O aparecimento de robots autónomos iniciou-se há largos anos com o desenvolvimento da robótica e da inteligência artificial (IA). Com o decorrer dos anos, estudos e trabalhos proporcionaram evolução suficiente para o aparecimento de autonomia em robots. Actualmente, esta área continua a ser alvo de estudo e inovação, sendo aplicável a inúmeros casos da robótica, facilitando variadas aplicações em diferentes contextos.

Um dos primeiros, senão o primeiro grande estudo de desenvolvimento de robots móveis foi de Nilsson e a sua equipa em 1969, com o do robot *Shakey*, em Standford Research Institute (SRI) [6]. A (Figura 1.3)[2] retrata o robot Shakey, ao centro, rodeado pela respectiva equipa de investigação.

Figura 1.3 Equipa de desenvolvimento e estudo do primeiro robot autónomo.

Este robot consistia numa plataforma com rodas tendo sensores de distância e de contacto, sendo idealizado para tarefas de navegação e exploração. Ainda que este robot (Shakey) tenha sido considerado um falhanço na sua época, por nunca ter atingido autonomia de operação, o projecto estabeleceu patamares funcionais importantes, identificando as deficiências tecnológicas e ajudando a inserir novas áreas de planeamento e processamento na pesquisa da IA.

Também nesta época surgiu o projecto *Standford Cart* que tinha como objectivo ser utilizado na Lua, no entanto, com o envio de uma missão tripulada, esta tecnologia foi abandonada. Contudo alguns anos mais tarde, em 1975, o robot foi utilizado para estudos de navegação e desvio de obstáculos, com uso de sistemas de visão que proporcionavam uma reconstrução do ambiente em 3D. [7]

[2] Fonte:*http://approach.rpi.edu/2011/01/18/the-shakey-connection/, consultado em 22/07/2013*

Mais tarde entre 1987 e 1995, a Comissão Europeia criou um projecto designado *EUREKA Prometheus Project (PROgraMme for a European Traffic of Highest Efficiency and Unprecedented Safety),* na área de veículos terrestres não tripulados, tendo sido o maior projecto Europeu nesta área até à actualidade, com um orçamento na ordem dos 1000 milhões de euros. Dois projectos deste programa atingiram bons resultados, sendo eles os veículos gémeos VaMP e Vita e o veículo ARGO. Na demostração final do Prometheus Project, os veículos VaMP e Vita, liderados por Ernst Dickmanns, conseguiram acompanhar uma faixa a 130km/h e ainda mudar de faixa com capacidade de decisão sobre quando poderia ser realizada em segurança. O veículo ARGO foi desenvolvido em Itália na Universidade de Parma. O projecto deste veículo continuou em desenvolvimento mesmo após o fim do Prometheus Project, e em 1998, numa demostração publica realizou uma viagem autónoma de 2 mil km pela Itália. [8]

Com a evolução tecnológica dos últimos 10 anos, a robótica foi alvo de grandes progressos, como exemplo, nos Estados Unidos da América, o projecto MDARS definiu e construiu um robot de segurança (Figura 1.4)[3] capaz de atingir 40km/h, patrulhar áreas definidas, de detectar intrusos até 200 metros, entre outras capacidades. Este robot é actualmente utilizado por algumas unidades do exército americano [9]

Figura 1.4 Robot MDARS

Hoje em dia existem várias empresas privadas que estudam e trabalham nesta área de robots autónomos. A título de exemplo um dos estudos mais sofisticados e conhecidos é o sistema vSLAM (Visual Simultaneous Localization and Mapping) [10] pela ActivMedia Robotics e da Evolution Robotics.

Com o crescimento e difusão do conceito de robots inteligentes e com o sucesso evidenciado pelos robots de exploração planetária, cada vez são mais as empresas que mostram interesse e se focam nesta área. Chegou-se assim a um ponto que a tecnologia actual possibilita a comercialização de alguns robots autónomos.

[3] Fonte:http://www.globalsecurity.org/military/systems/ground/mdars.html, consultado em 22/07/2013

A evolução assistida tornou-se possível graças às enormes capacidades de cálculo dos computadores actuais que, para além de permitirem a interligação de sensores muito diversos, possibilitam a execução em tempo real de algoritmos muito complexos. [11]

Também a evolução dos sensores de movimento e do sistema GPS permitiu uma melhoria nesta área. Por sensores de movimento refere-se os acelerómetros e giroscópios, bússola magnética.

Por exemplo, o acelerómetro que é um dispositivo capaz de medir a aceleração instantânea em relação a gravidade, era inicialmente composto por um sistema mecânico construído por molas. Presentemente são implementados com sistemas electrónicos que permitem atingir precisões muito mais elevadas e com melhor resolução. O mesmo acontece com os giroscópios actuais.

Mais importante e mais complexo é o sistema GPS, sendo este o principal meio de localização terrestre dos dias de hoje. A sua origem provém de 1957, ano em que a União Soviética lançou o primeiro satélite artificial da história, acontecimento que deu inicio aos primeiros estudos sobre o uso de satélites na localização de pontos na superfície terrestre. No entanto, seriam os EUA que anos mais tarde criariam o sistema com o desenvolvimento do projecto NAVSTAR em 1960. Este projecto estava inicialmente disponível apenas para o exército Americano. Contudo com o passar dos anos e depois dos vários ajustes e correcções tornou-se disponível para uso civil. Presentemente a constelação completa tem 24 satélites, o que permite que o sistema de navegação tenha uma cobertura de GPS global.

O estado da arte de sistemas de navegação autónoma encontra-se num nível de complexidade elevado impulsionado por sistemas implementados por empresas como a *iRobot* [12] ou a *NASA* [2]. Deste modo, esta dissertação que aborda uma pequena parte da robótica autónoma, insere-se num ambiente sujeito a constante desenvolvimento e inovação, sendo por isso alvo de interesse.

Para o Exército Português revelar-se-á uma mais-valia a aquisição de conhecimentos nesta área de modo a que se mantenha actualizado e na vanguarda das novas tecnologias em compasso com as outras forças armadas internacionais.

1.4 Descrição do Robot

O robot móvel utilizado nesta dissertação para estudo e implementação da navegação autónoma, foi o Jaguar 4x4 Wheel, apresentado na Figura 1.5. Sendo este robot o meio disponível para a implementação do algoritmo desenvolvido e para a realização de testes, é

importante ter conhecimento das suas características básicas (velocidade máxima, comunicação) e dos seus principais constituintes (sensores, motores).

Figura 1.5 Robot Jaguar 4x4 Wheel

Este robot é constituído por quatro motores DC de 80 W, um por cada roda. Possui uma plataforma exterior rígida e resistente, com certa de 20 kg e pode atingir uma velocidade máxima de 15 km/h. Tem uma distância ao solo de 88 mm, é compacto e impermeável. O jaguar 4x4 wheel tem ainda a capacidade de subir elevações que não ultrapassem os 155mm e consegue subir degraus com elevação inferior a 110 mm. Tem integrado um sistema de GPS exterior e sensores de 9 DOF IMU (Giroscópio/ Acelerómetro /Bússola magnética). Tem ainda incorporado um sistema de vídeo e áudio de alta resolução. [13] A comunicação com o robot é feita por tecnologia wireless (802.11G).

O ritmo de transmissão dos dados provenientes do receptor de GPS é de 5 leituras por segundo, ou seja, uma leitura em cada 200ms. Estes dados têm uma exactidão de posição menor ou igual a 15 m em 95% dos casos. De notar que 15m representa uma precisão muito pobre da verdadeira localização.

Relativamente ao sistema de sensores de 9 DOF IMU (Degrees of Freedom, Inertial measurement unit), o mesmo é constituído por um giroscópio com saída digital de três eixos (ITG3200) [14], por um acelerómetro ADXL345 [15] de 3 eixos com 13bit de resolução com capacidade máxima suportada de +/-16G e ainda uma bússola magnética de 3 eixos, HMC5843 [16]. O tempo de actualização no giroscópio e no acelerómetro é de 20ms, enquanto o tempo de actualização da bússola magnética é de 220ms. Acresce ainda referir que Informações detalhadas relativamente a estes sensores estão disponíveis no manual de utilizador do robot Jaguar 4x4 Wheel [13] .

1.5 Trabalho Realizado/Descrição Geral de Conteúdo

Nesta dissertação foi realizada uma aplicação que permite a comunicação, o tratamento e filtragem de sinais dos sensores, a estimação de posição e o controlo do robot.

Foram estudados o sistema GPS, os sensores IMU e os *encoders,* sendo analisadas quais as suas benesses e quais os seus defeitos para se efectuar a estimativa da posição.

Foi ainda estudado o método de localização de Markov, que, apesar de não ser aplicado na prática, abordou o problema estatístico de estimar a posição. Analogamente foi estudado e explicado o método matemático criado por Rudolf Kalman, este foi aplicado em duas circunstâncias nesta tese: na filtragem de sinal ruidoso e como meio de integração dos valores de vários sensores. Este método denominado Filtro de Kalman permitiu-nos obter uma estimativa da posição real do robot através da associação dos valores do sistema GPS, dos sensores IMU e dos *encoders*. No final, foi comparado o método de localização de Markov com o de Kalman e explicada a razão da utilização deste último.

Para navegar autonomamente foi necessário estimar a posição actual e a orientação do robot em cada instante e saber controlar os seus motores de modo a seguir o percurso desejado.

A estimativa da posição do robot em cada instante foi calculada no Filtro de Kalman de duas dimensões. No entanto, antes da implementação deste foram filtrados e calibrados os dados dos sensores IMU. A calibração permitiu que, logo à partida, não se tivessem erros de leitura a incrementar ao ruído existente. O ruído por sua vez foi retido com um Filtro de Kalman de uma dimensão. Os passos para adaptação do filtro foram também expostos.

Para controlo do deslocamento do robot foi necessário trabalhar os ângulos de direcção actual e de direcção desejada, sendo obtidos com o apoio do giroscópio e da bússola magnética. Consoante o erro existente entre estes foram estimados os valores adequados a enviar aos motores do robot.

No final realizaram-se os testes de desempenho, de modo a verificar a capacidade de autonomia do robot em várias condições, analisando o erro existente e verificando as capacidades e problemas existentes.

1.6 Estrutura da Tese

A dissertação está estruturada da seguinte forma:

Inicialmente, no primeiro capítulo, foi introduzido e contextualizado o tema, referindo-se a motivação, os objectivos e o estado da arte deste. Foram ainda descritas as características gerais do robot utilizado nos estudos e nos testes desta dissertação.

No segundo capítulo, serão descritos os sistemas de localização, mencionando-se os sensores e sistemas disponibilizados pelo robot. Segue-se uma breve explicação do funcionamento do método de localização de Markov como modo de introdução à abordagem do problema da localização. É então descrito detalhadamente o Filtro de Kalman que foi implementado no sistema de localização desenvolvido. No final deste capítulo, segue-se uma comparação entre os dois métodos referidos de modo a justificar a escolha efectuada.

No terceiro capítulo, expõe-se o sistema de navegação implementado, enfatizando o modo de comunicação com o robot e o funcionamento dos seus motores. Descreve-se também o modo de funcionamento de toda a aplicação desenvolvida e apresenta-se a interface desenvolvida.

No quarto capítulo explicam-se os métodos utilizados para implementar a navegação autónoma, desde a calibração e a filtragem dos valores dos sensores, ao cálculo dos erros de direcção e ao controlo dos motores do robot.

No quinto capítulo demostram-se os testes finais efectuados ao sistema de navegação do robot, bem como os resultados obtidos.

Por último apresenta-se, no capítulo seis, as conclusões desta dissertação evidenciando o conhecimento obtido e o trabalho desenvolvido. Importa ainda referir que neste capítulo se aborda também o trabalho que poderá ser desenvolvido posteriormente no seguimento deste tema.

2 Sistemas de Localização

O problema da localização em sistemas autónomos é complexo. O funcionamento conjunto dos vários sensores, a imprecisão e a ocasional ineficácia destes colocam desafios difíceis à obtenção precisa da localização.

Neste capítulo abordam-se os mecanismos, os dispositivos e o sistema de GPS disponibilizados pelo robot, de modo a obter a posição real deste em cada instante. De seguida, tratam-se duas técnicas probabilísticas existentes para melhoramento da posição estimada e portanto para uma melhor navegação, o método de Markov e o Filtro de Kalman. Descreve-se o método da localização de *Markov,* que, não sendo utilizado para implementação prática nesta dissertação é um sistema simples e acessível e que permite uma primeira abordagem aos problemas de localização.

No seguimento, é exposto o segundo método probabilístico, mais propriamente a localização por Filtro de Kalman. Este é um método de localização mais complexo, com características que lhe conferem capacidade para ser utilizado em diversas aplicações. Trata-se de um filtro que permite a integração de valores de vários sensores, sendo possível obter bons resultados para a estimação da posição e assim auxiliar a navegação autónoma.

No final deste capítulo, estabelece-se uma comparação entre os métodos de localização referidos, concluindo-se com a explanação dos pontos fortes e fracos de cada um deles.

2.1 Sensores de Movimento e Sistema GPS

Numa primeira avaliação, o sistema de localização de GPS parece ser uma boa solução, no entanto, este sistema por si só não é prático. A existente rede GPS disponibiliza uma precisão dentro de alguns metros, o que não é aceitável para a localização de robots móveis em escala humana. Além disso, as tecnologias GPS podem não funcionar em ambientes fechados ou em áreas obstruídas, devido a perdas de sinal ou sinais muito ruidosos.

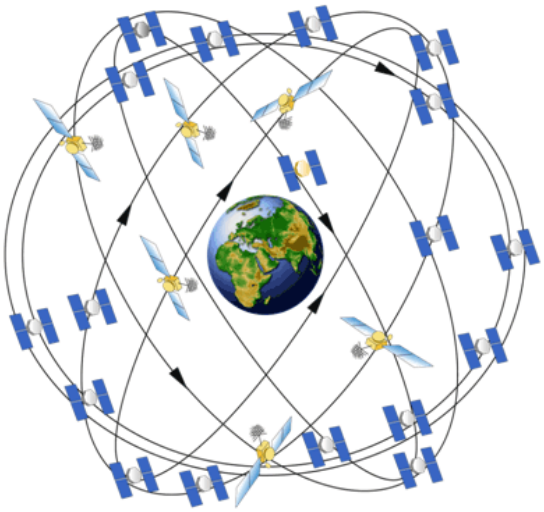


Figura 2.1 Sistema de Localização Global (GPS) [33]

Contudo, olhando além das limitações do GPS, a localização implica mais do que saber a posição absoluta no sistema de referência terrestre. Considerando, por exemplo, um robot que interaja com seres humanos, pode ser relevante conhecer a sua posição relativa a uma referência específica. Esta posição relativa, bem como as correcções na posição absoluta podem ser obtidas pela análise dos sensores de movimento (IMU).

2.1.1 Sistema GPS

Um sistema importante que apoia e fornece a localização em tempo real é o sistema GPS (*Global Position System*), sistema de posicionamento global. É composto por uma rede de estações terrestres e satélites colocados na órbita do planeta Terra, estando actualmente disponível para uso civil, de forma generalizada. [17] Os satélites GPS orbitam em torno da terra duas vezes por dia com órbita definida, transmitindo informação para o Planeta. Para cálculo da posição os receptores necessitam de ter acesso a pelo menos quatro satélites GPS.

O receptor GPS utilizado neste robot é o dispositivo da Garmin *GPS 18x (*Figura 2.2*).* Este receptor disponibiliza 5 leituras por segundo, ou seja, uma leitura em cada 200ms. Estes dados têm uma exactidão de posição menor ou igual a 15m, em 95% dos casos. Outras especificações técnicas encontram-se descritas em "*GPS 18x technical specifications"*. [18]

Figura 2.2 Receptor GPS 18x

O receptor mencionado consiste na principal fonte de informação relativa à localização, no entanto existem limitações e fontes de ruído interferente que condicionam o funcionamento deste sistema. [19].

Alguns dos factores comuns que afectam a precisão do sistema GPS:

- Técnica GPS utilizada;
- Condições envolventes (visibilidade do satélite e *multipath*);
- Número de satélites em linha de vista;
- Geometria do satélite
- Condições Ionosféricas (ocorrentes na Ionosfera e troposfera que podem introduzir atrasos significativos);
- Qualidade do receptor de GPS;
- Erros de órbita (fraca precisão da posição do satélite GPS).

Também as obstruções (edifícios, árvores, cercas, cabos, etc.) existentes na zona de recepção dos sinais do GPS podem causar uma redução muito significativa na precisão. Estas obstruções podem induzir os seguintes efeitos:

- Número reduzido de satélites alcançados pelo receptor;
- Redução da potência do sinal dos satélites (*Dilution of precision* (DOP));
- Sinal de satélite *multipath*;
- Corrupção de medições de GPS;

2.1.2 Sensores IMU

O sistema GPS fornece informação em tempo real da posição estimada, contudo essa informação tem ruído associado às suas limitações e ainda não nos possibilita saber a direcção actual para onde o robot se encontra orientado. Assim justifica-se recorrer a sensores complementares.

No caso do robot em questão, empregam-se três sensores de ajuda ao movimento. A este conjunto de sensores é dado o nome: Sensores de 9 DOF IMU. Isto significa ter 9 graus de liberdade em unidades de medição de inércia. Este conjunto de sensores permite melhorar significativamente a precisão da localização, sendo o modo utilizado para a correcção de falhas de sinal GPS e para dar continuidade de localização no caso de perda de sinal do mesmo.

Este conjunto de sensores 9 DOF IMU é constituído por um giroscópio com saída digital de três eixos (ITG3200) [14] (Figura 2.3).

Figura 2.3 Giroscópio ITG-3200

Também possui um acelerómetro ADXL345 [15] de 3 eixos, com 13bit de resolução e com capacidade máxima suportada de +/-16g (Figura 2.4).

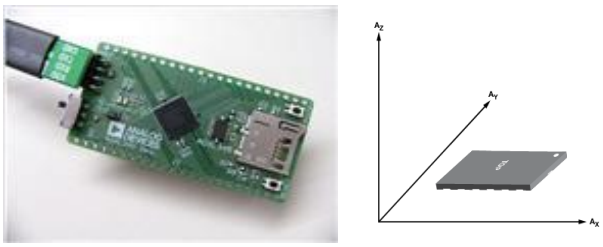


Figura 2.4 Acelerómetro ADXL345

Contém ainda uma bússola magnética de 3 eixos, HMC5843 [16](Figura 2.5).

Figura 2.5 Bússola magnética IC HMC5843

2.2 Localização de Markov

2.2.1 Noção básica do funcionamento da localização de Markov

De um modo geral, a localização de Markov é aplicável ao problema de estimação de estado a partir de dados dos sensores, como é referido por Maria Ribeiro e Pedro Lima, em "*Markov Localization"*: "*Represent the robot's belief by a probability distribution over possible positions and uses Bayes rule and convolution to update the belief whenever the robot senses or moves*" [20].

Ainda a partir de outra fonte podemos verificar: "Markov localization is a special case of probabilistic state estimation, applied to mobile robot localization" [21], que o algoritmo utilizado na localização de Markov é probabilístico, em que, em vez de manter uma única hipótese sobre a real localização do robot, utiliza uma distribuição de probabilidade sobre o espaço que engloba todas as hipóteses.

A representação probabilística permite atribuir medidas a estas diferentes hipóteses de modo matemático. Para dar a entender de modo simples o funcionamento matemático do algoritmo, serão explicados inicialmente os conceitos gerais a partir de um exemplo.

Considerando o ambiente desenhado na Figura 2.6, assume-se que o robot se movimenta a uma dimensão, ou seja, o robot movimenta-se apenas horizontalmente. É ainda suposto que o robot é colocado aleatoriamente neste ambiente, sendo a sua posição desconhecida. A localização de Markov representa este estado de incerteza com uma distribuição uniforme sobre todas as posições, verificando-se, a verde, na recta da Figura 2.6.

Figura 2.6 Localização equiprovável do robot

Supondo agora que o robot recebe a informação de que se encontra perto de uma porta (assinalada a azul nas figuras seguintes) isto irá fazer com que a localização de Markov modifique a sua convicção (belief), aumentando a probabilidade junto aos locais onde existem portas e baixando-a nos restantes lugares. Esta situação corresponde ao segundo diagrama

da Figura 2.7. Pode-se observar que o resultante gráfico apresenta várias elevações reflectindo o facto de ainda não existir informação suficiente para uma localização global exacta. Observa-se também que nos restantes locais que a probabilidade não é zero. Isto acontece devido ao ruído das leituras dos sensores e porque uma medição apenas sobre a proximidade de uma porta é normalmente insuficiente para excluir a possibilidade de não se estar próximo de uma porta.

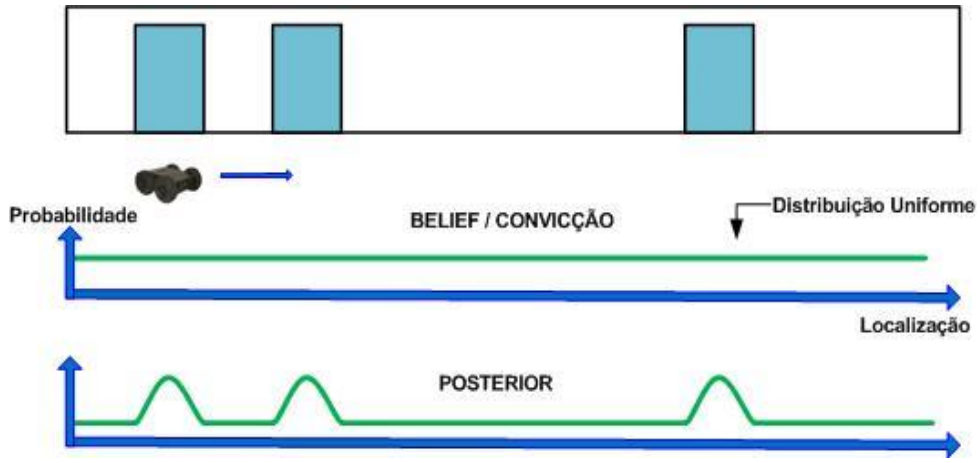


Figura 2.7 Posterior reconhecimento do robot

Agora assume-se que o robot se continua a deslocar em frente. A localização de Markov irá incorporar esta informação movendo a distribuição anterior em concordância, como podemos observar na Figura 2.8 no terceiro diagrama. Pode-se notar que, de acordo com o ruído inerente ao movimento do robot, existe um decaimento de informação, obtendo-se assim uma representação mais dispersa que a anterior.

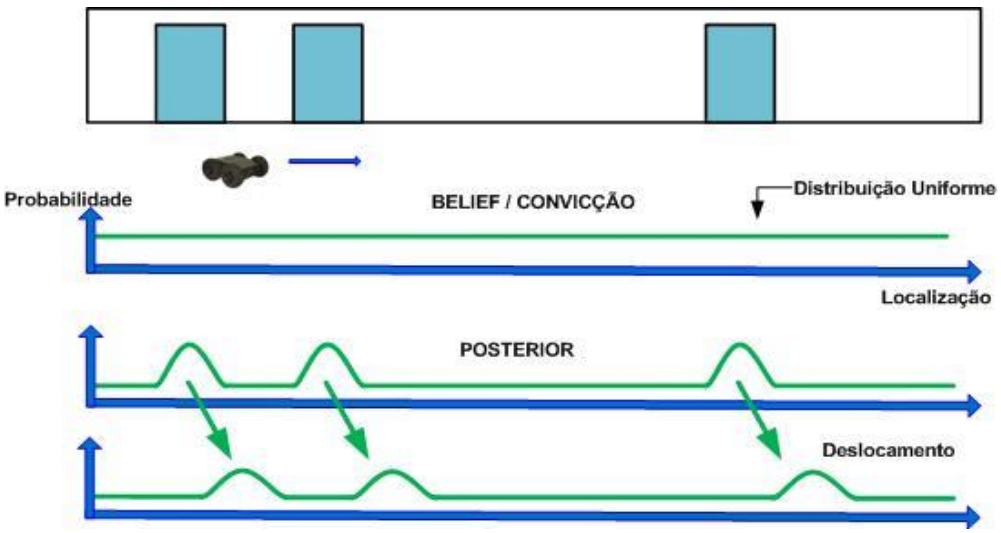


Figura 2.8 Deslocamento do Robot

Finalmente, assumindo que os sensores do robot detectam a porta seguinte, a localização de Markov multiplica esta observação com a amostra anterior, o que nos leva ao último diagrama da Figura 2.9. Nesta altura, a maior parte da probabilidade está centrada numa única localização e, neste momento o robot tem quase a certeza da sua posição.

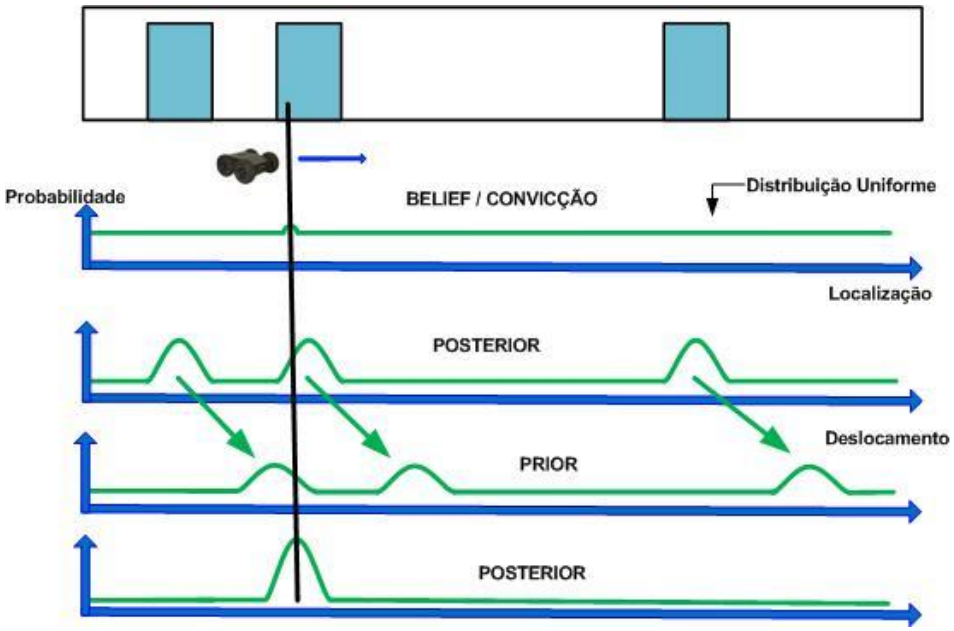


Figura 2.9 Conceito da localização de Markov.

2.2.2 Aplicação teórica para localização

De acordo com Ribeiro [20], Dieter [21] e Scheneider [22] irá ser explicado, de modo resumido, o funcionamento da localização de Markov.

De modo a expressar o estado de convicção do robot *(belief*), é necessário fazer corresponder para cada posição possível uma determinada probabilidade de como o robot está realmente nessa posição. Assim sendo, utilizar-se-á o termo *p(A)* para definir a probabilidade de *A* ser verdadeiro, facto este designado de probabilidade *a priori* de A, pois mede a probabilidade de *A* ser totalmente independente de qualquer conhecimento anterior que possa existir. Portanto, utiliza-se $p(r_t = l)$ para indicar a probabilidade do robot *r* estar na posição *l* no instante *t*.

Na prática, pretende-se calcular a probabilidade de cada posição individualmente a partir dos dados recolhidos pelos sensores do robot. Seja *p(A|B)* a probabilidade do acontecimento *A* sabendo o condicionamento da ocorrência de *B*, então, $p(r_t = l \mid i_t)$ denota a probabilidade do robot estar na posição *l,* sabendo os dados *i* dos sensores do robot no instante *t*.

Como se sabe:

$$p(A \wedge B) = p(A|B)p(B) \quad (2.1)$$

A equação (2.1), relaciona a probabilidade de ambos, A e B serem simultaneamente verdadeiras com a probabilidade de A sabendo B ser verdadeira e a probabilidade de B ser verdadeiro; igualmente se tem:

$$p(A \wedge B) = p(B|A)p(A) \quad (2.2)$$

Juntando as duas equações anteriores, (2.1) e (2.2) obtém-se a conhecida fórmula de Bayes (2.3)

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \quad (2.3)$$

Utiliza-se então a regra de Bayes para calcular os novos estados de convicção (*belief*) do robot em função das saídas dos sensores e o seu antigo estado de convicção. Para uma correcta demostração, considera-se que o conjunto de possíveis posições do robot é representado por *L* e o estado de convicção do robot deve atribuir uma probabilidade $p(r_t = l)$ para cada localização. É preciso então actualizar as probabilidades associadas a cada posição *l* em *L,* aplicando a fórmula de *Bayes* a cada uma dessas posições *l.*

$$p(l\,|i) = \frac{p(i|l)p(l)}{p(i)} \quad (2.4)$$

O valor de $p(i|l)$ é a chave da equação (2.4). Esta probabilidade dos valores dos sensores para cada posição deve ser calculada segundo um modelo. Uma estratégia óbvia seria consular o mapa do robot, identificando a probabilidade de particularidades identificadas nas leituras do sensor em cada posição do mapa. O valor de p(*l*) é fácil de obter neste caso, sendo simplesmente a probabilidade de $p(r_t = l)$ associada ao estado de convicção. O denominador p(*i*) não depende de *l* e, assim sendo, cada vez que aplicarmos a equação (2.4) para cada posição *l* em L, não iremos variar o denominador.

Considere-se agora que a partir do estado de convicção anterior e dos dados obtidos dos sensores obtém-se o novo estado de convicção. Para calcular a probabilidade da posição *l* no novo estado de convicção, deve-se integrar todos os caminhos possíveis que o robot poderá ter atingido, de acordo com as potenciais posições expressas no estado de convicção anterior. Este facto é importante, pois o mesmo local *l* pode ser acedido a partir de várias localizações diferentes com a mesma medida *o* dos sensores.

$$p(l_t|o_t) = \int_{ti-1}^{ti} p(l_t|l'_{t-1}, o_t)p(l'_{t-1})\, dl'_{t-1} \quad (2.5)$$

Assim, a probabilidade para uma específica posição *l* é construída a partir de contribuições individuais de todas as localizações *l'* no estado de convicção anterior e pela medida *o* dos sensores.

As equações (2.4) e (2.5) formam a base da localização de Markov. Formalmente significa isto que a saída é gerada a partir do estado anterior do robot e das mais recentes acções e percepções. Em geral, numa situação Markoviana o estado do sistema não depende de toda a sua história. Os valores dos sensores do robot no instante de tempo t não têm de depender apenas da sua posição no momento t. Estes valores dependem de alguma maneira de toda a trajectória do robot ao longo do tempo. Pelo mesmo raciocínio, a posição do robot no momento t não depende apenas da posição no instante t-1 e das suas medidas odométricas. De acordo com a história do movimento, um motor pode ter andado mais que outro, causando descalibração ao longo do tempo, o que afectará a estimativa da posição actual.

Conforme Siegwart: "So the Markov assumption is, of course, not a valid assumption" [5], contudo, embora incorrecta, a mesma simplifica extremamente o raciocínio, o planeamento e a monitorização e por isso é ainda uma aproximação que continua muito popular na robótica móvel.

2.3 Filtro de Kalman

Outra forma de aumentar a precisão da localização passa pela utilização do Filtro de Kalman, o qual teve origem nos trabalhos de Rudolf Kalman. Será utilizado no sistema de localização desenvolvido e na consequente navegação:

"In 1960 and 1961 Rudolf Emil Kalman published his papers on a recursive predictive filter that is based on the use of state space techniques and recursive algorithms and therewith he revolutionized the field of estimation." [23]

Desde a sua publicação, há mais de 50 anos, o Filtro de Kalman continua a ser objecto de investigação, desenvolvimento e aplicação para diversas áreas. Neste capítulo e com base em alguns artigos já publicados por Walter [24], Bilgin [25], Bishop [26] e Makhijani [27] serão apresentadas as componentes básicas do Filtro de Kalman, explicando-se o seu modo de funcionamento, as suas aplicações e os resultados decorrentes da sua utilização.

2.3.1 Informação básica do Filtro de Kalman

O Filtro de Kalman é um filtro de predição recursiva que se baseia no uso de técnicas de estados e em algoritmos recursivos, estimando o estado de sistemas dinâmicos. Estes sistemas dinâmicos são por exemplo as equações de movimento, que, juntamente com as entradas de controlo conhecidas e com as medições provenientes de vários sensores, são utilizadas pelo Filtro de Kalman para gerar uma estimativa das variáveis do sistema, ou seja, os seus estados.

A estimativa das variáveis do sistema, obtida deste modo, é melhor que a estimativa obtida utilizando-se cada uma das medições unicamente. É portanto um algoritmo ideal para combinar dados provenientes de vários sensores.

Citando Schneider: *"The Kalman filter provides an estimate for past, present and future system-states even if the precise internal structure of the system in question is unknown."* [22] Representa-se na Figura 2.10 uma aplicação típica deste filtro.

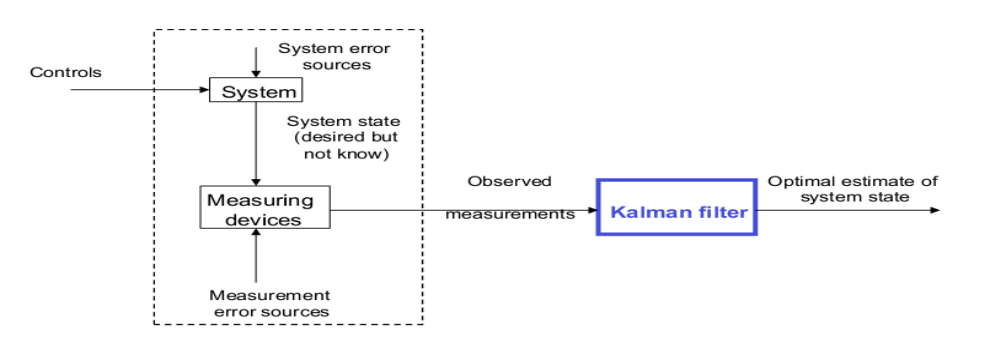


Figura 2.10 Aplicação típica do Filtro de Kalman [32]

Antes da explicação do processo e do algoritmo é importante adquirir alguns conhecimentos gerais sobre o funcionamento geral das partes integrantes deste filtro. É conhecido que todas as medições e cálculos baseados em modelos são, de certo modo, estimativas. Os sinais ruidosos de sensores, as aproximações nas equações que descrevem o comportamento do sistema e os factores externos não considerados introduzem incerteza sobre os valores obtidos para o estado de um sistema.

O Filtro de Kalman combina uma predição do estado de um sistema com a nova medida, utilizando uma média ponderada e forma a obter o estado estimado. São ainda utilizados "pesos" para as medições e para os estados, em que o propósito dos pesos é atribuir maior importância aos valores em que existe maior confiança. Os pesos são

calculados através da covariância, uma medida da incerteza estimada a partir da predição do estado do sistema.

O resultado da média ponderada é uma nova estimativa de estado, que se localiza entre o estado predito e o estado medido, apresentando uma melhor certeza estimada que qualquer um dos dois individualmente. Este processo é repetido em cada nova iteração, com a nova estimativa e a nova covariância[4], originando assim a predição a ser utilizada na próxima iteração. Pode-se então concluir que o Filtro de Kalman funciona recursivamente e baseia-se apenas na última estimativa do estado de um sistema, para calcular o estado seguinte, e não no histórico completo.

O estado posterior obtido pelo Filtro de Kalman é representado por uma distribuição Gaussiana e dependendo da dimensão, a distribuição toma as seguintes formas:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

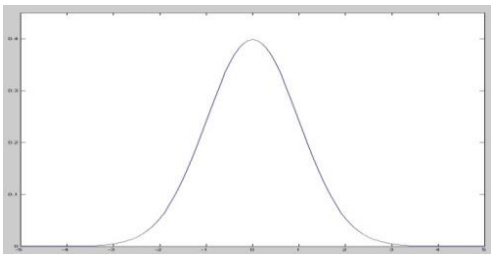


Figura 2.12 Distribuição Gaussiana a 1-D

$$P(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

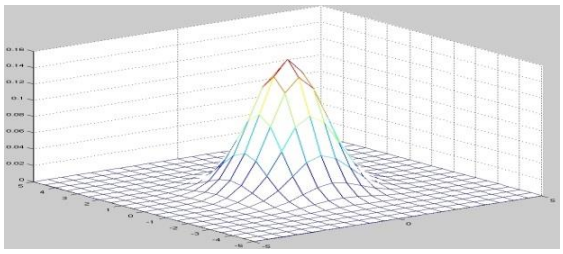


Figura 2.12 Distribuição Gaussiana a 2-D

Procede-se com a explicação detalhada do algoritmo, das funções e dos cálculos utilizados para a determinação dos estados.

O Filtro de Kalman funciona em duas fases distintas:

1. Actualização temporal / A predição (*prediction/ time update*)
2. Actualização de medidas / A correcção (*correction/ measurement update*)

Na primeira fase o estado é predito com o modelo dinâmico. Na segunda fase é corrigido com o modelo de observação, onde o erro de covariância do estimador é minimizado [25].

Este processo é repetido para cada intervalo de tempo com o estado anterior como valor inicial (Figura 2.13). Entende-se assim que o Filtro de Kalman é também um filtro recursivo.

[4] A covariância é uma medida de associação (relação) linear entre duas variáveis aleatórias

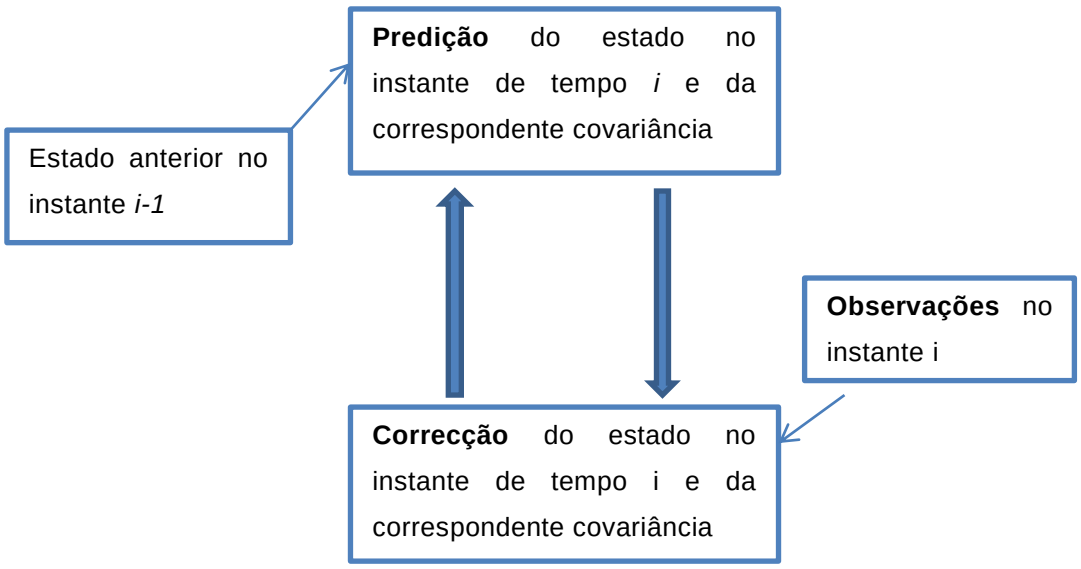


Figura 2.13 Filtro de Kalman, recursividade

Os componentes básicos do Filtro de Kalman consistem no vector de estados, no modelo dinâmico e no modelo de observação, que serão abordados de seguida.

O vector de estados, é onde são mantidas as variáveis de estado. Este vector descreve o estado dinâmico do sistema e representa os seus graus de liberdade. As variáveis do vector de estados não podem ser medidas directamente dos sensores, mas podem ser calculadas a partir de valores que são mensuráveis.

Os elementos do vector de estados podem ser, por exemplo, a posição, a velocidade, o ângulo de orientação, por entre outros. A título de exemplo, o caso de um veículo que siga a velocidade constante e sempre na mesma direcção origina um vector de estados que terá dois graus de liberdade, a saber: a distância s e a velocidade v. Sabe-se que a velocidade corresponde à derivada da posição, $\dot{s} = v$, seja para o vector de estado:

$$\underline{x} = \begin{bmatrix} s \\ v \end{bmatrix} \tag{2.6}$$

O vector de estados tem dois valores ao mesmo tempo, o valor *a priori*, que será o valor predito antes da actualização, e o valor *a posteriori*, ou seja, o valor correcto depois da actualização. Para se diferenciar estes dois valores, utilizar-se-á para o valor *a priori* $\underline{x}^-$ e para o valor a *posteriori* $\underline{x}^+$.

Relativamente ao modelo dinâmico, este descreve as transformações do vector de estados ao longo do tempo e pode ser representado por um sistema de equações diferenciais (2.7).

$$\dot{\underline{x}}(t) = \frac{d}{dt}\,\underline{x}(t) = f(\underline{x}(t), \underline{m}(t)) \tag{2.7}$$

Do modo linear, o modelo dinâmico pode ser rescrito da forma em (2.8).

$$\dot{\underline{x}}(t) = F \cdot \underline{x}(t) + \underline{n}(t) \tag{2.8}$$

Onde *F* é uma matriz dinâmica e constante, $\underline{x}(t)$ é o vector de estados e $\underline{n}(t)$ é o ruído dinâmico, que normalmente é assumido como ruído branco, com matriz de covariância correspondente *Q(t)*.

Aborda-se agora o modelo de observação, este representa a relação entre o estado e as medidas. No caso linear, as medições são descritas por um sistema linear de equações, que apenas dependem do estado das variáveis. As observações $\underline{l}(t_i)$ são feitas em instantes de tempo $t_{i,}$ como se mostra em (2.9)

$$\underline{l}(t_i) = h(\underline{x}(t_i), \underline{v}(t_i)) \tag{2.9}$$

O vector deste sistema é representado por (2.10).

$$\underline{l}(t_i) = H \cdot \underline{x}(t_i) + \underline{w}(t_i) \tag{2.10}$$

Onde $\underline{l}(t_i)$ é o vector de observações no instante t_i, *H* é a matriz de observação e $\underline{w}(t_i)$ é o ruido do processo de medição com a matriz de covariância $\mathrm{R}(t_i)$. Tal como na matriz dinâmica, no sistema linear a matriz de observação *H* é igualmente constante.

2.3.2 Actualização temporal / Predição

Como referido anteriormente, a primeira fase do Filtro de Kalman é a predição. O estado predito, ou o estado *a priori,* é calculado desprezando o ruído e resolvendo a equação diferencial que descreve o modelo dinâmico:

$$\dot{\underline{x}}^{-}(t) = F \cdot \underline{x}^{-}(t) \tag{2.11}$$

O vector de estados no instante *t* pode ser expresso pela série de Taylor para uma aproximação ao estado $\underline{x}^{-}(t_0)$.

$$\underline{x}^{-}(t) = \underline{x}^{-}(t_0) + \dot{\underline{x}}^{-}(t_0)(t - t_0) + \frac{1}{2}\ddot{\underline{x}}^{-}(t_0)(t - t_0)^2 + \cdots \tag{2.12}$$

Esta pode ser reescrita com recurso à equação (2.11):

$$\underline{x}^{-}(t) = \underline{x}^{-}(t_0) + F \cdot \underline{x}^{-}(t_0)(t - t_0) + \frac{1}{2}F^2 \cdot \underline{x}^{-}(t_0)(t - t_0)^2 + \cdots \tag{2.13}$$

Como se mostra, a solução de $\underline{x}^{-}(t)$ da equação diferencial, ou o actual estado predito, é uma combinação linear do estado inicial $\underline{x}^{-}(t_0)$.

$$\underline{x}^{-}(t) = \Phi_0^t \cdot \underline{x}^{-}(t_0) \tag{2.14}$$

Onde Φ_0^t representa a matriz de estado de transição que transforma o estado inicial $x(t_0)$ no correspondente estado $x(t)$ no instante *t*.

A partir das equações anteriores, (2.11) e (2.14) obtém-se:

$$\dot{\underline{x}}^{-}(t) = F \cdot \underline{x}^{-}(t) = F \cdot \Phi_0^t \cdot \underline{x}^{-}(t_0) \tag{2.15}$$

Partindo de novo da equação (2.14) ,

$$\frac{d}{dt}\underline{x}^{-}(t) = \frac{d}{dt}\left[\Phi_0^t \cdot \underline{x}^{-}(t_0)\right] = \left[\frac{d}{dt}\ \Phi_0^t\right] \cdot \underline{x}^{-}(t_0) \tag{2.16}$$

Comparando as duas últimas equações (2.15) e (2.16) conclui-se,

$$\frac{d}{dt}\ \Phi_0^t = F \cdot \Phi_0^t \tag{2.17}$$

Com matriz inicial $\Phi_0^0 = I$, pois $\underline{x}(t_0) = I \cdot x(t_0)$, em que I representa a matriz identidade.

Relativamente à matriz de covariância $P^{-}(t_i)$ do vector de estado predito, esta é obtida a partir da lei da propagação de erro. A matriz de covariância com ruído $Q(t)$, em função do tempo, é expressa em:

$$P^{-}(t_i) = \Phi_{ti-1}^{ti} \cdot P(t_{i-1}) \cdot (\Phi_{ti-1}^{ti})^T + \int_{ti-1}^{ti} Q(t)dt \tag{2.18}$$

2.3.3 Actualização de medidas /Correcção

Na fase de correcção, o vector de estados predito $\underline{x}^{-}(t_i)$ é melhorado com observações feitas no instante t_i , fazendo deste modo com que o estado *a posteriori* tenha a seguinte forma:

$$\dot{\underline{x}}^{+}(t_i) = \underline{x}^{-}(t_i) + \Delta\underline{x}(t_i) \tag{2.19}$$

com matriz de covariância:

$$P^{+}(t_i) = P^{-}(t_i) + \Delta P(t_i) \tag{2.20}$$

O Filtro de Kalman é um filtro óptimo, o que significa que as variâncias[5] de estado na matriz de covariância P^{+} são minimizadas. Como P^{-} já é conhecido pela fase de predição, implica que o ΔP seja minimizado,(2.21).

$$\Delta P(t_i) = E\left[\Delta \underline{x}(t_i)\underline{x}(t_i)^T\right] \tag{2.21}$$

Esta condição é obtida por:

$$\Delta \underline{x}(t_i) = P^{-}H^T(HP^{-}H^T + R(t_i))^{-1} \cdot (\underline{l}(t_i) - H\underline{x}^{-}(t_i)) \tag{2.22}$$

Obtendo,

$$\Delta \underline{x}(t_i) = K(t_i) \cdot (\underline{l}(t_i) - \underline{l}^{-}(t_i)) \tag{2.23}$$

Onde K é denominada por *matriz de ganho* de Kalman e é por definição:

$$K(t_i) = P^{-}H^T(HP^{-}H^T + R(t_i))^{-1} \tag{2.24}$$

A diferença entre $(\underline{l}(t_i) - \underline{l}^{-}(t_i))$ é designada de resíduo. Este reflecte a discrepância entre a medida predita $\underline{l}^{-}(t_i) = H\,\underline{x}^{-}(t_i)$ e a medida actual $\underline{l}(t_i)$.

Por fim, o estado corrigido é obtido por:

$$\underline{x}^{+}(t_i) = \underline{x}^{-}(t_i) + K(t_i) \cdot (\underline{l}(t_i) - \underline{l}^{-}(t_i)) \tag{2.25}$$

Nesta equação, o estado estimado e as medidas são avaliados e combinados para cálculo do estado correcto. Por exemplo, se a covariância medida for muito inferior à considerada no estado predito, será atribuído um peso mais alto às medidas e um peso muito inferior ao estado predito, reduzindo-se deste modo o grau de incerteza.

A matriz de covariância do estado *a posteriori* é dada segundo a lei da propagação do erro por:

$$P^{+}(t_i) = P^{-}(t_i) + K(t_i)HP^{-}(t_i) = (I - K(t_i)H)\,P^{-}(t_i) \tag{2.26}$$

Apresenta-se agora em tabela as expressões obtidas e as suas relações:

[5] A variância é uma medida de dispersão das variáveis. Mede a distância entre as observações e a sua média. Por definição mede o quadrado da distância entre uma observação e sua média. Dito de outra forma, a variância de um conjunto de dados é o somatório das distâncias entre cada observação e a média desse conjunto elevado ao quadrado.

Tabela 1 - Ciclo do Filtro de Kalman [25]

Predição *(Time update)*	**Correcção / Actualização** *(Measurement Update)*
Estado *a priori*: $\dot{\underline{x}}^{-}(t) = F \cdot \underline{x}^{-}(t) + \underline{n}(t)$ Matriz de covariância *a priori:* $P^{-}(t_i) = \Phi_{ti-1}^{ti} \cdot P(t_{i-1}) \cdot (\Phi_{ti-1}^{ti})^{T} + \int_{ti-1}^{ti} Q(t)dt$	Matriz de ganho: $K(t_i) = P^{-}H^{T}(HP^{-}H^{T} + R(t_i))^{-1}$ Estado *a posteriori*: $\underline{x}^{+}(t_i) = \underline{x}^{-}(t_i) + K(t_i) \cdot (\underline{l}(t_i) - \underline{l}^{-}(t_i))$ Matriz de covariancia *a posteriori*: $P^{+}(t_i) = (I - K(t_i)H)\, P^{-}(t_i)$

Com base em vários artigos, foi apresentada uma descrição matemática do Filtro de Kalman, com o intuito de tornar perceptível os princípios de funcionamento deste. Uma explicação detalhada desta matéria encontra-se em [28] e em [29].

2.4 Comparação de Algoritmos de Localização Probabilística

"The Markov localization model can represent any probability density function over robot position. This approach is very general but, due to its generality, inefficient". [5]

Como citado anteriormente, no caso da localização de Markov, o estado de convicção do robot é representado por diferentes atribuições de probabilidade para cada posição do robot possível. Durante os processos de actualização e percepção, a probabilidade deve ser actualizada para cada posição.

Na localização por Filtro de Kalman, o estado de convicção do robot é representado utilizando apenas a função de densidade de probabilidade Gaussiana mantendo, assim, apenas os parâmetros da média μ e variância σ do estado de convicção do robot, em relação à respectiva posição. Assim é apenas necessário modificar os parâmetros da distribuição Gaussiana em cada actualização.

Como é referido por Siegward, "Kalman Filter is in fact more efficient than Markov localization because of key simplifications when representing the probability density function of the robot's belief state and even its individual sensor readings" [5]

Enumeram-se os principais aspectos da localização de Markov:

- Permite inicializar a localização a partir de uma posição inicial desconhecida;
- Permite recuperar de situações ambíguas, pois o robot consegue seguir múltiplas e completamente distintas posições possíveis;
- Requer representação discreta do espaço, de modo a ser possível actualizar todas as posições a qualquer instante;
- Requer elevados requisitos de memória e grande capacidade computacional.

Referem-se as principais características do Filtro de Kalman:

- Requer posição inicial conhecida;
- É preciso e eficiente;
- Pode ser usado para representações contínuas;
- Se o estado de incerteza for demasiado elevado (por exemplo devido a uma colisão) pode levar o filtro a falhar na captura de múltiplas posições possíveis e ficar irreversivelmente perdido.

A maioria dos robots, tal como o robot utilizado no desenvolvimento desta tese, possui um elevado número de sensores, cada um fornecendo dados úteis para aperfeiçoar a estimativa da sua posição. Uma localização óptima deve ter em conta a informação proveniente de todos estes sensores e, desta forma, o Filtro de Kalman é um mecanismo eficiente para relacionar a informação destes sensores.

Tabela 2 - Comparação de Sistemas de Localização

Sistema de Localização	Markov	Kalman
Permite posição inicial desconhecida?	Sim	Não
Baixos requisitos de memória?	Não	Sim
Requer representação do espaço/ mapa?	Sim	Não
Recupera de situações ambíguas?	Sim	Não
Integração de vários sensores?	Sim	Sim
Elevada precisão?	Sim	Sim

3 Sistema de navegação aplicado

*"Navigation is one of the most challenging competences required of a mobile robot."(*Siegwart, 2004*)* [5]

Os sistemas de navegação tem um impacto cada vez mais significativo na vida quotidiana, são empregues em automóveis, barcos, mísseis, submarinos, etc. A maioria destes dispositivos utiliza não só o Sistema de Localização Global (GPS), como também um Sistema de Navegação Interno (INS) para ajuda e complemento da navegação.

A existência do sistema de navegação interno possibilita uma melhor obtenção de estimativas de posição, como por exemplo, em locais de fraco sinal GPS ou com sinal muito ruidoso. Estes sistemas de navegação interna são constituídos por diversos sensores, tais como acelerómetros, giroscópios, sensores laser, câmaras, etc.

O sucesso na navegação requer o funcionamento conjunto de quatro blocos distintos: percepção, localização, capacidade de decisão e controlo de movimento. No bloco *percepção* é onde o robot interpreta o sinal proveniente dos sensores e retira deles informação útil, de modo a determinar a sua posição no terreno (*localização)* e conseguir decidir (*capacidade de decisão*) como actuar para atingir o seu destino. Por último, o *controlo de movimento* é onde o robot modula as saídas do motor de acordo com o conhecimento e a trajectória desejada. [5]

Na Figura 3.1 representam-se as interligações dos referidos blocos.

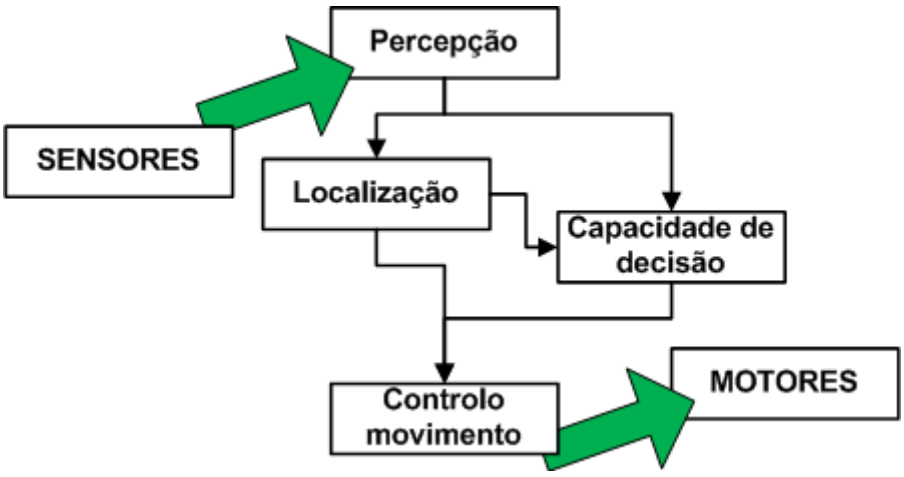


Figura 3.1 Esquema geral do sistema de navegação para robots móveis [5]

Neste capítulo, serão descritas as características do robot utilizado com enfâse nas funcionalidades disponibilizadas. No seguimento, é exposto o sistema de navegação autónomo proposto, detalhando a sua estrutura.

3.1 Funcionamento do robot

Para proporcionar uma percepção integral das capacidades do robot *Jaguar*, explicita-se como se processa a comunicação com os módulos deste e o seu processo de controlo.

3.1.1 Comunicação com o robot Jaguar

Para comunicação com o robot é normalmente utilizada uma rede *wireless*, dada a inclusão de um *Access Point/Router* com wireless 802.11 no Jaguar. No entanto, o computador, como dispositivo controlador, consegue conectar-se ao robot, dependendo das necessidades, via:

- cabo de rede: ligando directamente a placa AP/router;
- wireless: para conexão ao AP/router é necessário configurar a rede do computador do controlador, fixando o endereço de IP como 192.168.0.XXX e máscara 255.255.255.0.

Depois de estabelecida a ligação ao computador "controlador", o utilizador pode, a partir de um comando, controlar os movimentos do robot em tempo real, com um alcance tão elevado quanto a rede wireless o permitir.

De referir ainda que os módulos internos do robot estão disponíveis para configuração individual a partir de endereços IP fixos, enumerados na Tabela 3.

Tabela 3 - Endereços IP dos vários módulos do robot

Controlador de movimento	IP: 192.168.0.60 Porto: 10001
GPS	IP: 192.168.0.61 Porto: 10002
Sensores IMU	IP: 192.168.0.61 Porto: 10001
Câmara	IP: 192.168.0.65 Porto: 8081

De modo a obter uma noção básica do hardware que constitui o robot e entender o modo de comunicação anteriormente referido, observe-se o seguinte esquema da arquitectura e comunicação dos vários módulos do robot, (Figura 3.2):

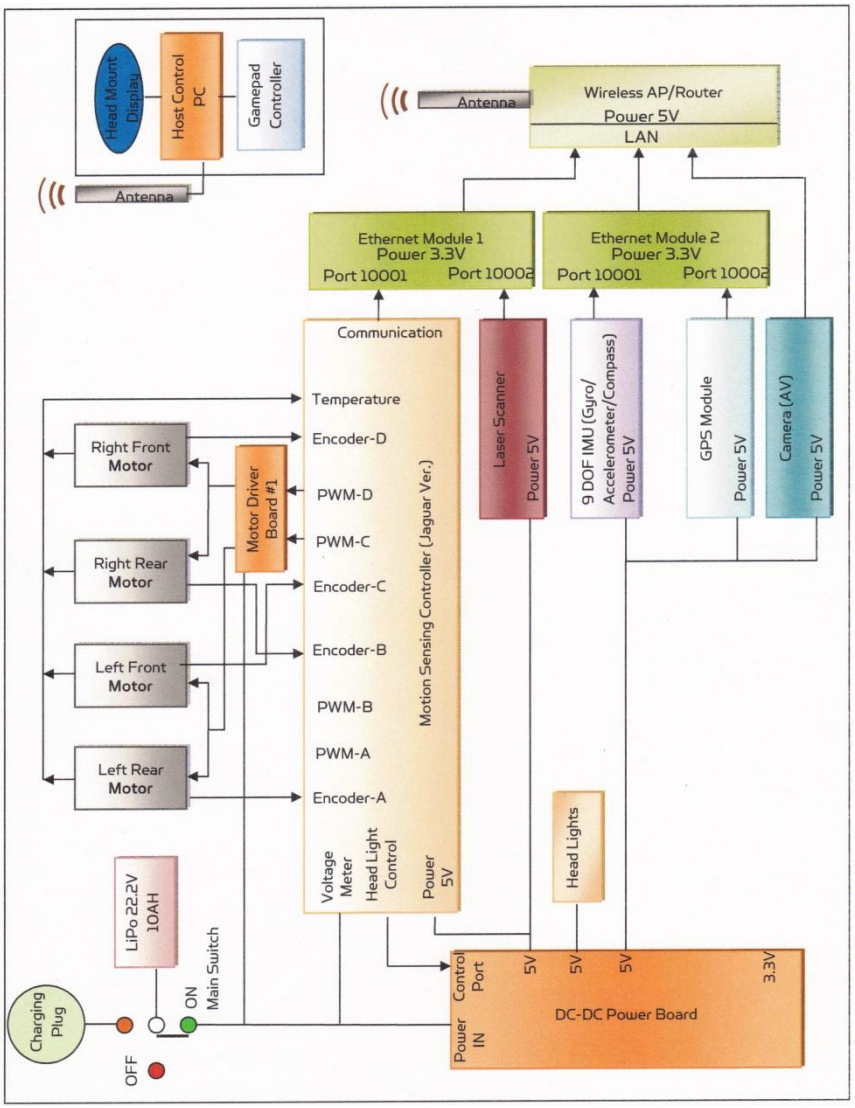


Figura 3.2 Esquema da arquitectura hardware do robot

O robot é constituído por quatro módulos físicos, que podem ser acedidos individualmente, depois de estabelecida a ligação entre o utilizador e o *AP*, sendo eles o módulo de controlo de movimento, o de GPS, o de Sensores IMU e a câmara.

Para comunicação com estes módulos foi disponibilizado pelo fabricante uma API (*Application Programming Interface)* em C# que permite a comunicação com os vários módulos do robot. Para uma observação detalhada desta verifique-se o *WiRobot SDK API Reference Manual* [30].

A partir do esquema da Figura 3.2 podemos observar que este robot também permite integração de um sensor laser de medição de distância (laser range finder). No entanto neste robot não esta integrado.

3.1.2 Motores

A ligação aos motores é efectuada a partir do módulo de controlo de movimento (Motion Sensing Controller), representado no esquema da Figura 3.2. Para a comunicação com o robot são utilizadas funções disponibilizadas pelo fornecedor, onde estão disponíveis dois canais para enviar os dados referentes à potência.

Os dois canais de comunicação, os canais 3 e 4, correspondem à *potência0* e *potência1* explicados mais à frente.

O controlo dos motores do robot funciona de modo diferencial. São controlados pela informação transmitida pelos canais anteriormente referidos, estes aceitam valores que variam entre 0 e 32768 com valor intermédio mútuo de 16384 para paragem. A variável *potência0 (PW0)* é utilizada para controlar a velocidade e o andamento do robot, para a frente ou para trás, enquanto que a variável *potência1 (PW1)* é responsável pelo ângulo de rotação, de modo a virar mais ou menos à esquerda ou à direita.

A compreensão deste funcionamento foi alvo de estudo, tornando-se essencial para se conseguir escolher, em cada instante, o melhor conjunto de valores, intentando-se assim obter um bom controlo para o deslocamento autónomo.

Na Tabela 4, é demonstrado o esquema de valores enviados e a resposta dos motores aos mesmos.

Tabela 4 - Resposta dos motores

		PW 1								
		0	**4**	**8**	**12**	**16**	**20**	**24**	**28**	**32**
PW 0	**0**	0 PF 100	25 FF 100	50 FF 100	75 FF 100	100 FF 3.5m/s	100 FF 75	100 FF 50	100 FF 25	100 FP 0
	4	25 TF 100	0 PF 100	25 FF 100	50 FF 80	70 FF 2.5m/s	80 FF 50	100 FF 25	100 FP 0	100 FT 25
	8	50 TF 100	20 TF 100	0 PF 80	25 FF 60	50 FF 1.2m/s	60 FF 25	80 FP 0	100 FT 20	100 FT 50
	12	75 TF 100	40 TF 80	20 TF 60	0 PF 40	25 FF 0.75m/s	40 FP 0	60 FT 20	80 FT 40	100 FT 75
	16	100 TF 100	60 TF 60	40 TF 40	20 TF 20	**0 PP 0**	20 FT 20	40 FT 40	60 FT 60	100 FT 100
	20	100 TF 75	80 TF 40	60 TF 20	40 TP 0	25 TT - 0.75m/s	0 PT 40	20 FT 60	40 FT 80	75 FT 100
	24	100 TF 50	100 TF 20	80 TP 0	60 TT 40	50 TT - 1.2m/s	40 TT 60	0 PT 80	20 FT 100	50 FT 100
	28	100 TF 25	100 TP 0	100 TT 25	80 TT 60	75 TT - 2.5m/s	20 TT 80	60 TT 100	0 PT 100	25 FT 100
	32	100 TP 0	100 TT 25	100 TT 50	100 TT 75	100 TT - 3.5m/s	75 TT 100	50 TT 100	25 TT 100	0 PT 100
		0	**4**	**8**	**12**	**16**	**20**	**24**	**28**	**32**

FF	avança frente direita
FF	avança frente esquerda

FT	roda direita e recua
TF	roda esquerda e recua

FF	avança frente a direito
TT	recua a direito

TT	recua trás esquerda
TT	recua trás direita

3.2 Algoritmo de navegação

Conjuntamente com o robot, é disponibilizada uma *Application Programming Interface* (*API*) que proporciona uma interface entre o *software* ao nível da aplicação e o *hardware* do robot [30]. Facilita portanto o desenvolvimento de aplicações, como é o caso da que aqui se discute. Para o desenvolvimento deste algoritmo beneficiou-se inicialmente, ainda, de várias funções e rotinas implementadas em código numa aplicação de navegação, também disponibilizada com o robot, de modo a servir de exemplo das capacidades de desenvolvimento que o robot proporciona.

Algumas das funções mais relevantes são: as funções de comunicação com os motores, de sincronização entre a bússola magnética e o acelerómetro, de conversão instantânea entre coordenadas geográficas e posição, tendo como referencia as coordenadas iniciais dadas pelo utilizador, do cálculo do ângulo desejado de direcção, do cálculo da distância entre coordenadas, entre outras. Estas funções em particular são disponibilizadas através de uma *Dynamic-Link Library*, a sua descrição detalhada encontra-se no Anexo A.

Além destas funções, foi disponibilizada uma interface gráfica com sincronização com o mapa da aplicação *Google Earth,* de modo a permitir uma representação gráfica dos percursos pretendidos. Estão ainda incluídas na aplicação referida, e das quais se fez uso, as rotinas de leitura de dados dos sensores e do sistema GPS, as rotinas de reconhecimento e de leitura do *joystick*, bem como a conversão dos seus dados para controlo do robot.

Dado que o funcionamento do programa é extenso, a sua exposição será decomposta em três partes. Na primeira, (Figura 3.4) explicam-se o controlo manual e o procedimento de ligação ao robot. Abordam-se, na segunda parte (Figura 3.5), a ligação aos sensores e a adaptação dos mesmos. Na mais extensa (Figura 3.6) e última parte, descreve-se o funcionamento e controlo da navegação autónoma, desde a definição do percurso a percorrer até ao método de controlo dos motores, a partir do erro de direcção.

Na Figura 3.3, representa-se o fluxograma que descreve o funcionamento da aplicação. Cada bloco representado corresponde respectivamente, por ordem numérica, às partes acima referidas.

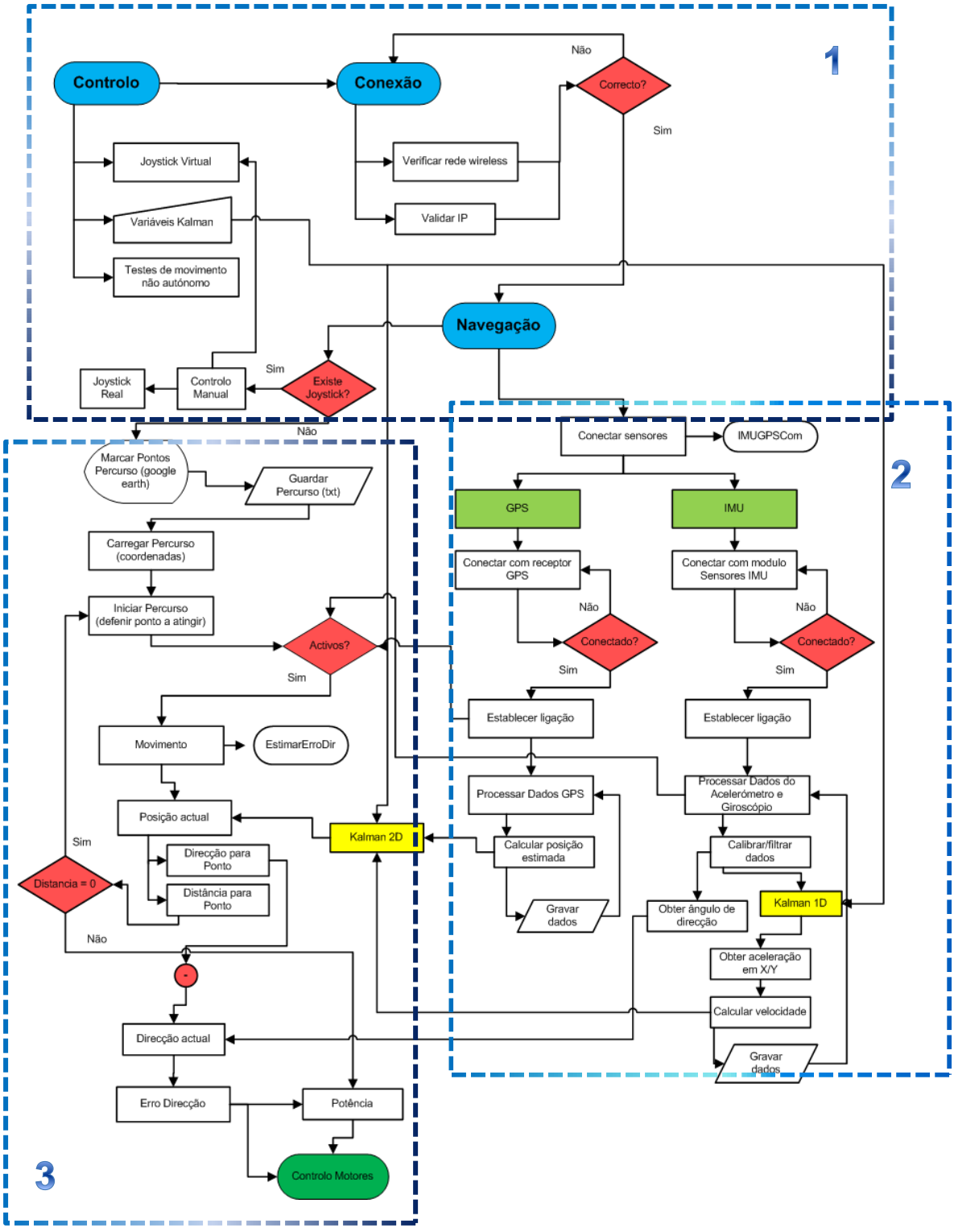


Figura 3.3 Fluxograma da aplicação desenvolvida para controlo autónomo

3.2.1 Controlo e conexão

O módulo do controlo é o primeiro a ser executado sendo que a partir deste é possível aceder ao segundo módulo de conexão com o robot. Supondo que a conexão é estabelecida com sucesso, o módulo de controlo é onde reside a possibilidade do próprio utilizador controlar o robot, não autonomamente, mas a partir da aplicação ou de um comando (*joystick*) virtual. Por comando virtual entende-se uma aplicação externa capaz de simular e enviar dados como um comando real.

Pode-se, ainda no módulo de controlo e em tempo real, alterar os valores de variância (Q) e de ruído mínimo (R) para utilização no Filtro de Kalman. Por fim, este módulo serviu de plataforma para efectuar os testes ao controlo dos motores, ou seja, verificar as variações de velocidade dos motores para cada valor de potência enviado. A conexão permite verificar e validar os endereços de IP e estabelecer a ligação com o AP do robot.

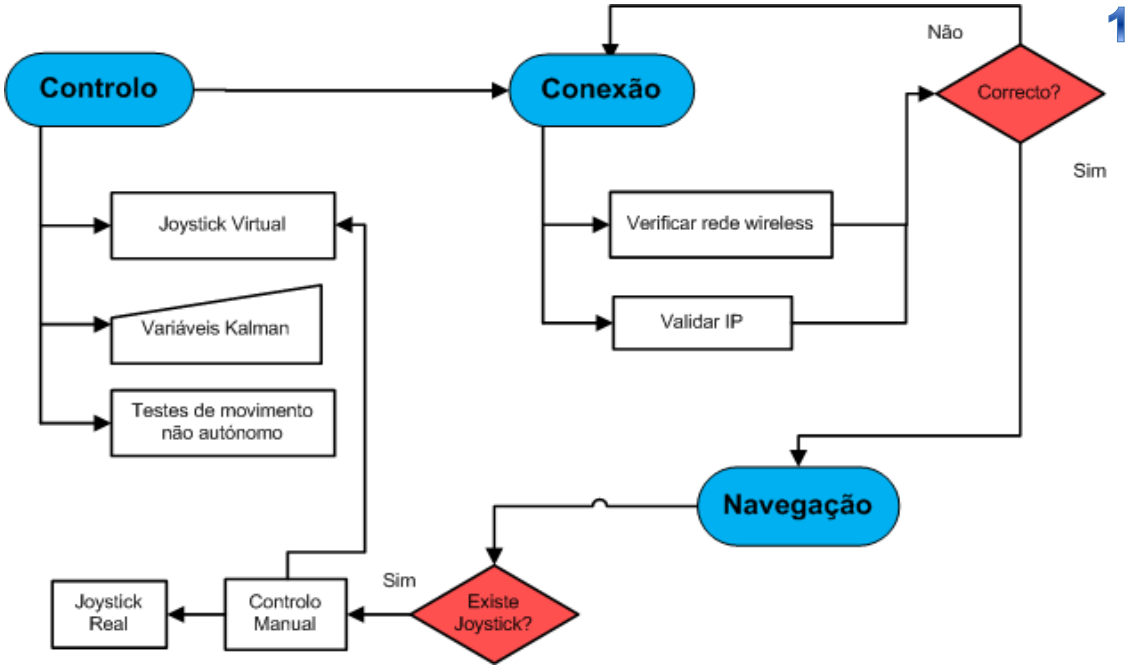


Figura 3.4 Controlo e conexão com o Robot

3.2.2 Ligação aos sensores e tratamento de valores

Na segunda parte da aplicação e já dentro do módulo de navegação, encontra-se o procedimento de ligação aos sensores IMU e ao sistema GPS.

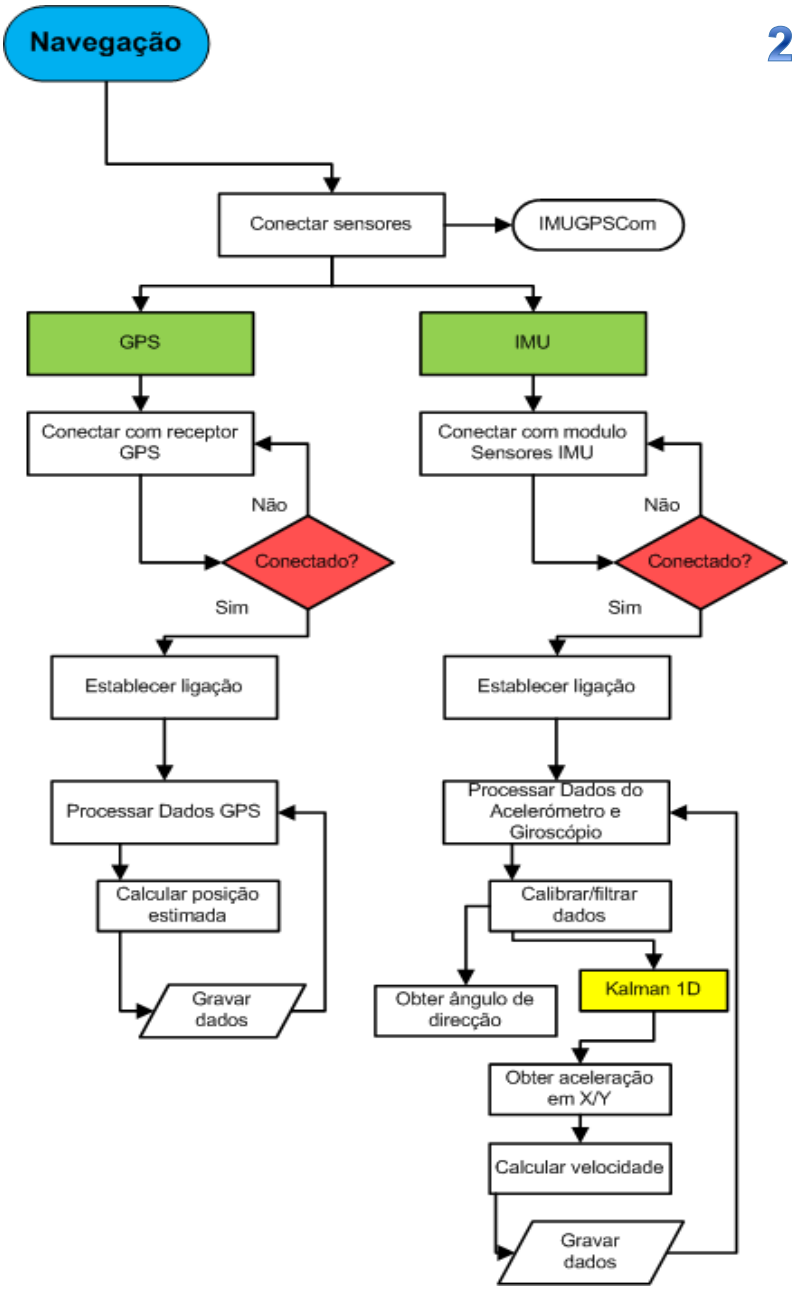


Figura 3.5 Conexão aos sensores e tratamento de dados

A rotina para ligação aos sensores foi facultada pela API e consiste na criação de duas *threads*[6] que se executam concorrentemente, uma para o sistema GPS e outra para os sensores IMU. Estas *threads* são colocadas em modo "adormecido" caso não existam dados a receber.

Em relação ao processamento dos dados dos sensores IMU, são recebidos valores de três sensores diferentes, do acelerómetro, do giroscópio e da bússola. Os dados do acelerómetro e giroscópio são calibrados e, no caso do acelerómetro, são posteriormente

[6] Threads: é uma forma de um processo se poder dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente.

filtrados com o Filtro de Kalman a uma dimensão. A partir dos valores do giroscópio e da bússola magnética é obtido o ângulo de direcção. Este é calculado com base no norte magnético e na informação do giroscópio. Do acelerómetro é calculada a aceleração instantânea do robot.

É então aplicado um sistema baseado em *"dead reckoning"* (método de calcular a posição actual do robot, com base na posição anterior, na velocidade e na sua orientação) obtendo-se uma posição estimada. Finalmente resta a possibilidade de gravar os dados e retorna-se ao ponto inicial, aguardando por novos valores destes sensores.

Em relação ao processamento de sinal GPS, a situação é mais simples. Os valores de vários tipos de informação que o receptor GPS disponibiliza são separados à entrada, aproveitando-se os valores de latitude e longitude do sinal GPRMC. De seguida, e caso o modo de navegação autónoma esteja activo, estas coordenadas são convertidas para um referencial XY. Por fim, existe a possibilidade de gravar estes dados em formato .txt e retorna-se ao ponto inicial, aguardando novo sinal GPS.

3.2.3 Controlo autónomo

Esta última parte da aplicação é exclusivamente dedicada ao controlo da navegação autónoma. Inicialmente define-se o percurso desejado, marcando os pontos no mapa e guardando as suas coordenadas. Estas são obtidas a partir do Google Earth, não sendo necessário neste caso interacção com o sistema GPS do robot. A sincronização com o Google Earth foi disponibilizada, não tendo sido efectuado código nesta parte. Depois de carregado o percurso e de os sensores estarem activos, é necessário conhecer a posição actual, motivo pelo qual é introduzido o Filtro de Kalman.

O Filtro de Kalman recebe como entradas a aceleração, a velocidade e a posição retornando à saída o valor da posição estimada. Obtida a estimativa da posição é calculada a distância para o ponto desejado e a distância ao percurso desejado. É ainda calculado o erro de direcção, ou seja, a diferença entre a nossa orientação e a direcção para onde desejamos ir. A partir deste erro de direcção, consegue-se controlar a potência dos motores conseguindo assim movimentar o robot para a rota pretendida.

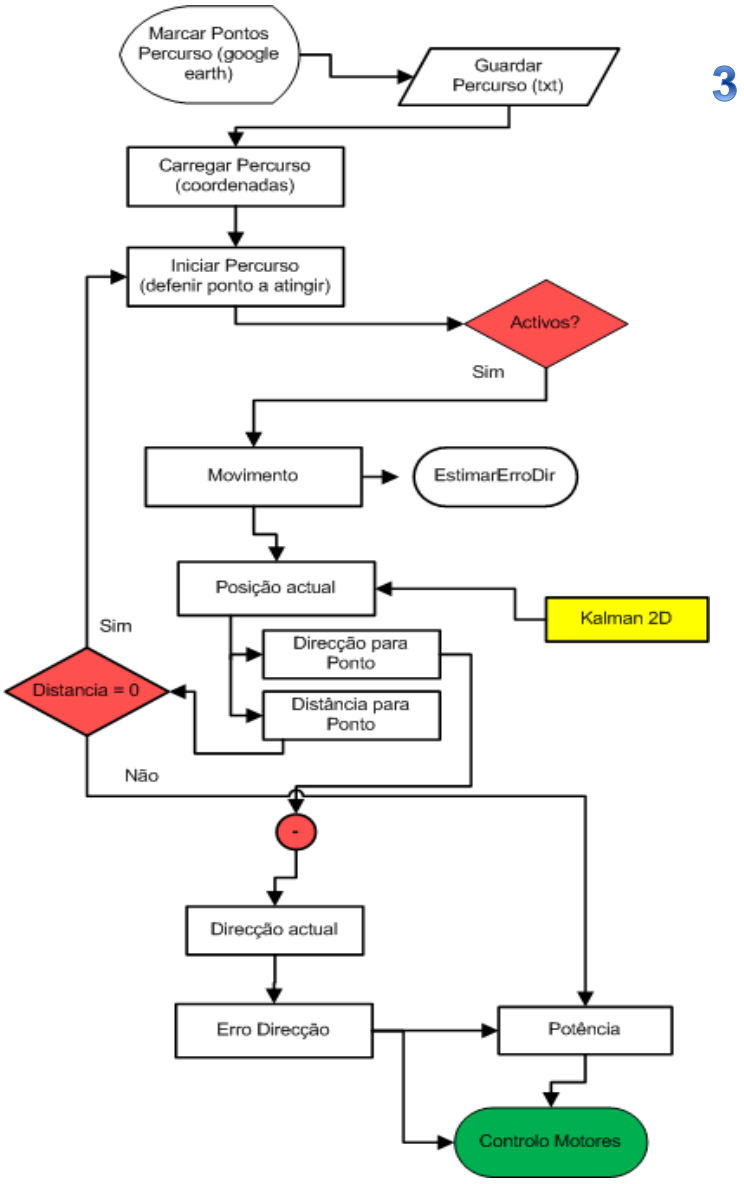


Figura 3.6 Controlo da autonomia do robot

3.2.4 Procedimento da aplicação para a navegação autónoma

Descrevendo em tópicos o procedimento para navegação autónoma, temos:

1) Marcação dos pontos a seguir no mapa (GOOGLE MAPS – coordenadas geográficas);
2) Guardar trajecto (guardar coordenadas geográficas), e carregá-lo a partir do ficheiro.
3) Activar GPS e sensores IMU.
4) Activar robot:
    i. Leitura da posição GPS –> conversão para posição XY

ii. Leitura dos sensores acelerómetro e giroscópio -> obter aceleração em X e Y e ângulo yaw (direcção do robot)

5) Calibrar o Giroscópio, filtrar dados do Acelerómetro e calcular velocidade instantânea;
6) Aplicação do Filtro de Kalman para correcção da posição.
    a. Entradas:
        i. Posição lida pelo GPS.
        ii. Velocidade calculada.
        iii. Aceleração filtrada e calibrada.
    b. Saída:
        i. Posição estimada
7) Calcular o ângulo entre a posição do robot e a posição desejada;
8) A partir do ângulo de direcção (yaw) obtido a partir do giroscópio e da bússola magnética e do ângulo calculado anteriormente, encontrar o erro de direcção;
9) Iniciar deslocamento do Robot;
10) Com base no erro de direcção calculado, guiar o robot para esquerda ou direita;
11) Quando a distância entre a posição actual e a posição desejada for mínima, actualizar ponto desejado para próximo ponto.

3.3 Interface gráfica desenvolvida

A aplicação de controlo e navegação foi programada em Microsoft Visual Studio 2010 sobre C# com .Net framework 3.5. Foi utilizado o Matlab R2012b para construção gráfica, investigação e análise dos dados obtidos.

O programa desenvolvido está dividido graficamente em dois módulos, o primeiro de conexão e controlo e o segundo de navegação. O primeiro subcapítulo relaciona as secções da aplicação de maior relevância, explicadas no capítulo anterior, com o ambiente gráfico.

O primeiro módulo do programa serviu de base para o controlo manual do robot. Esta interface (Figura 3.7) permitiu criar um comando virtual, estabelecer a conexão com o robot e possibilitar a alteração em tempo real das variáveis de calibração do Filtro de Kalman. Foi também o modo desenvolvido para efectuar os testes ao controlo de movimento como por exemplo criar e adaptar a relação entre os ângulos e potência dos motores. A forma gráfica desta interface é a seguinte:

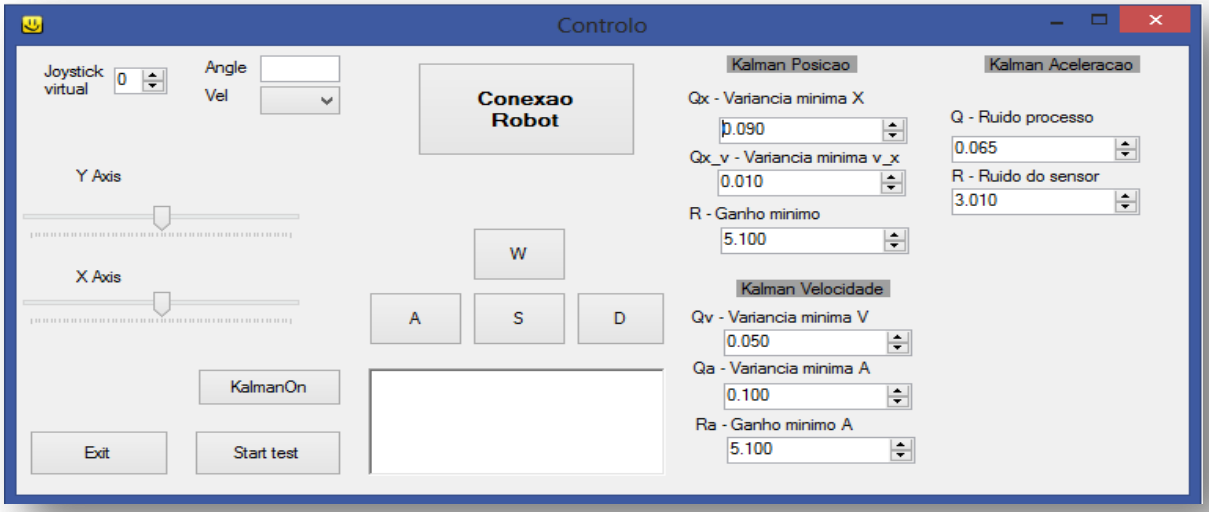


Figura 3.7 Interface gráfica desenvolvida do controlo e conexão do robot

O segundo módulo do programa é o de navegação. Esta interface (Figura 3.8) permite, de um modo abrangente, visualizar graficamente informação relevante, alterar variáveis de navegação, estabelecer ligações a módulos independentes no robot e suportar todos os algoritmos internos inerentes à navegação autónoma.

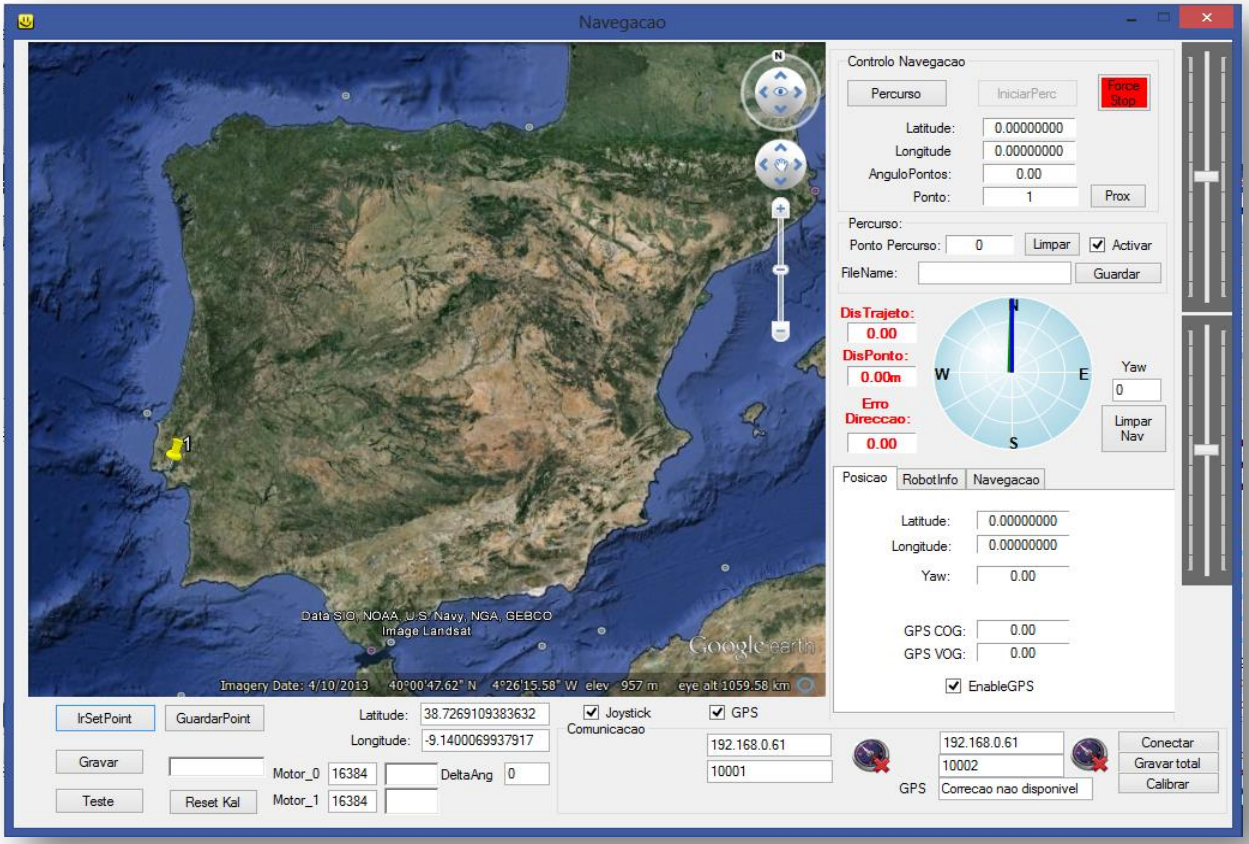


Figura 3.8 Interface Gráfica desenvolvida do bloco de Navegação do robot

Especificando, a visualização gráfica desta interface permite ao utilizador adquirir informação sobre os vários componentes do robot, pelo que se expõe:

- Os valores de potência enviados aos motores em cada correcção;
- Os valores de diferença de potencial da bateria;
- A temperatura dos motores;
- Outros dados;

É disponibilizada ainda informação relativa à navegação, a saber:

- O ângulo de direcção actual;
- O erro de direcção calculado entre a direcção actual e a direcção desejada;
- A distância entre a posição estimada e o trajecto ideal;
- A distância entre a posição estimada e o ponto final a atingir;
- A latitude e a longitude obtidas pelo sinal GPS.

Além de disponibilizar esta informação relativa à navegação autónoma, a interface permite definir os pontos do percurso desejado, facilitar a interacção com o mapa e apresentar graficamente os locais do itinerário, tanto o desejado como o realmente percorrido.

As definições de navegação são neste módulo definidas por (Figura 3.9):

- *DisPonto*: Quando a distância entre a posição estimada e o ponto desejado for menor que o valor definido nesta variável, o algoritmo assume que o ponto foi atingido e é definido o ponto desejado;
- *DistPercLimite*: Quando o erro de posição for maior que o valor definido nesta variável, o algoritmo efectuará correcções de posição;
- *AnguloAjuste*: Quando o erro de direcção for superior ao ângulo definido nesta variável, o algoritmo calcula novos valores de potência a enviar aos motores para o manter na direcção desejada;
- *MinDif*: O valor utilizado para definir o mínimo de diferença entre a potência dos motores do lado esquerdo e direito, ou seja, controla a intensidade de rotação. Em terrenos mais irregulares é necessária aumentar este valor;
- *Nível de atuação*: Valor para alterar o nível de atuação a enviar para o robot, o que implicitamente altera a velocidade do robot. Com esta variável a 100% o robot funciona à potência máxima, atingindo aproximadamente 3.5 (m/s).

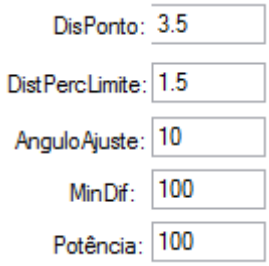


Figura 3.9 Definições de navegação

Neste módulo, é ainda efectuada a ligação e a leitura dos sensores IMU e do sistema GPS. São calibrados e filtrados os seus valores, obtendo-se a partir destes uma estimativa da localização do robot em tempo real. São ainda realizados todos os cálculos para apresentação da informação anteriormente referida e calculados os ângulos, as distâncias e as potências para o controlo dos motores do robot.

Por fim, é possível guardar em ficheiros todos os dados referentes aos sensores, aos filtros e aos cálculos efectuados, facilitando o posterior estudo de todo o processo.

4 Implementação do sistema de Navegação

Nas secções seguintes, é explicada a construção do algoritmo de navegação. Aborda-se a calibração e filtragem dos dados dos sensores, a utilização do Filtro de Kalman, o cálculo dos ângulos de direcção, o controlo dos motores do robot e a escala temporal de acontecimentos.

4.1 Leitura dos sensores e tratamento do sinal

Nesta primeira secção, será explicado o tratamento, a filtragem e a calibração dos sinais provenientes do acelerómetro e do giroscópio. Explica-se ainda a implementação do Filtro de Kalman para filtragem do ruído do acelerómetro. Por fim, estuda-se um sistema de estimativa de posição do robot a partir dos seus valores de aceleração/velocidade, da sua orientação e do seu ponto de partida.

4.1.1 Calibrar e filtrar dados do acelerómetro e do giroscópio

Para determinar o mais próximo da realidade a posição instantânea do robot e para efectuar um controlo sobre este que lhe permita seguir o percurso desejado, é fundamental integrar os dados de todos os seus sensores. Utilizaram-se então os dados do acelerómetro, do giroscópio e da bússola magnética. No entanto, estes dados estão corrompidos por ruído e necessitam de calibração e de filtragem.

Para calibrar os valores obtidos do acelerómetro, fez-se a recolha dos dados com o robot parado, obtendo-se o gráfico da Figura 4.1 para o eixo X:

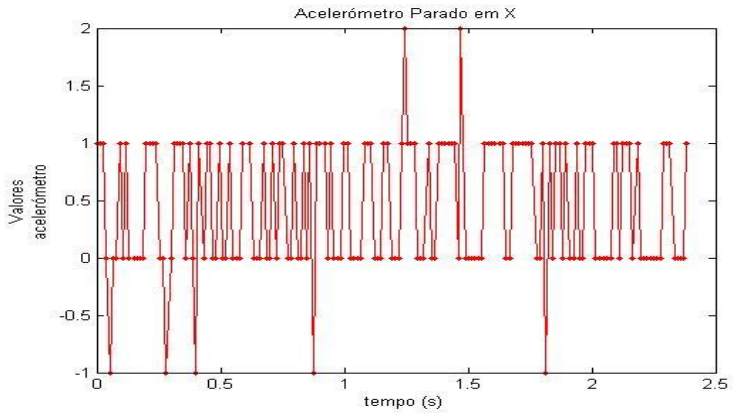


Figura 4.1 Valores do acelerómetro no eixo X, com robot parado.

O gráfico da Figura 4.2 para o eixo Y:

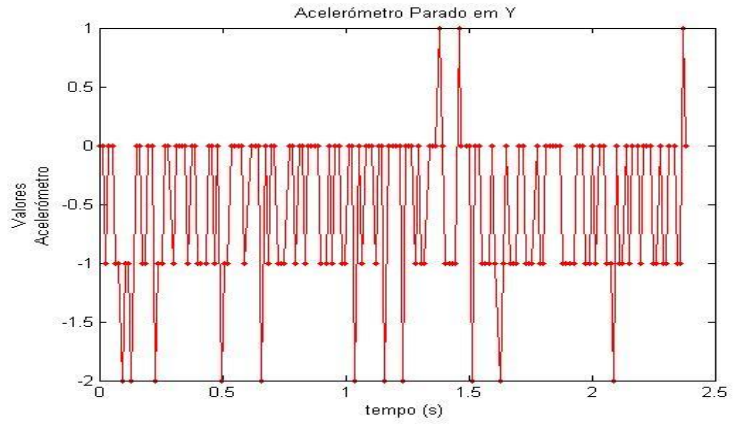


Figura 4.2 Valores do acelerómetro no eixo Y, com robot parado

E o gráfico da Figura 4.3 no eixo Z

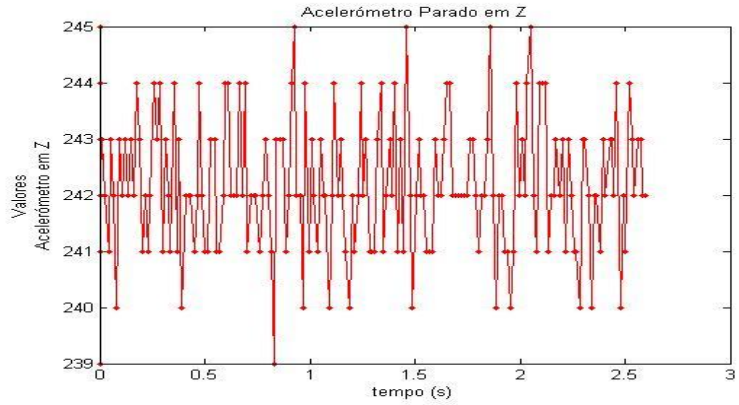


Figura 4.3 Valores do acelerómetro no eixo Z, com robot parado

Conclui-se, pela observação das figuras anteriores, que os dados tendem para um valor diferente de 0, mesmo com robot parado. Pode-se pensar que os valores são de uma ordem tão baixa que pouco afectariam, no entanto, estes valores de aceleração não têm

média nula. Deste modo com o passar do tempo provocam uma variação na velocidade, levando a crer que o robot está em movimento, quando ainda se encontra parado. O giroscópio apresenta os mesmos problemas: valores ruidosos.

Na tentativa de resolver este problema, entendeu-se que seria necessária uma calibração, pelo que se efectuou a média de valores em cada eixo com o robot parado e posteriormente subtraíram-se aos valores obtidos. O resultado final obtido é mostrado na Figura 4.4. Os valores obtidos para aceleração em x e y, após os 3 segundos, têm média nula. O método utilizado para o giroscópio foi idêntico, pelo que se apresentam apenas os valores do acelerómetro.

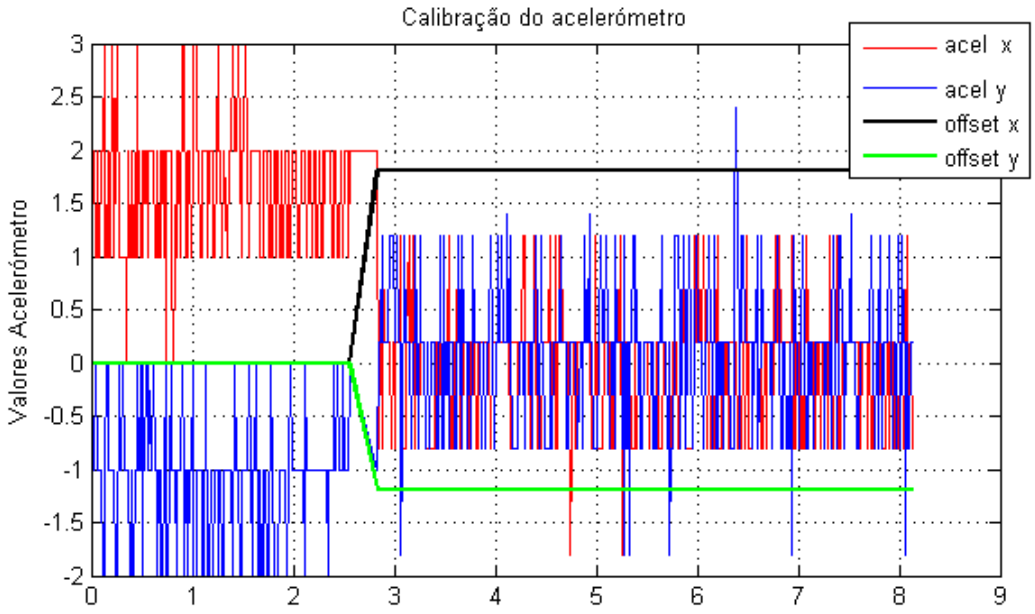


Figura 4.4 Calibração do acelerómetro

O acelerómetro foi ainda testado em linha recta durante, aproximadamente, 4 segundos, forçando uma aceleração seguida de travagem, pelo que se conseguiu adquirir os seguintes resultados do gráfico da Figura 4.5.

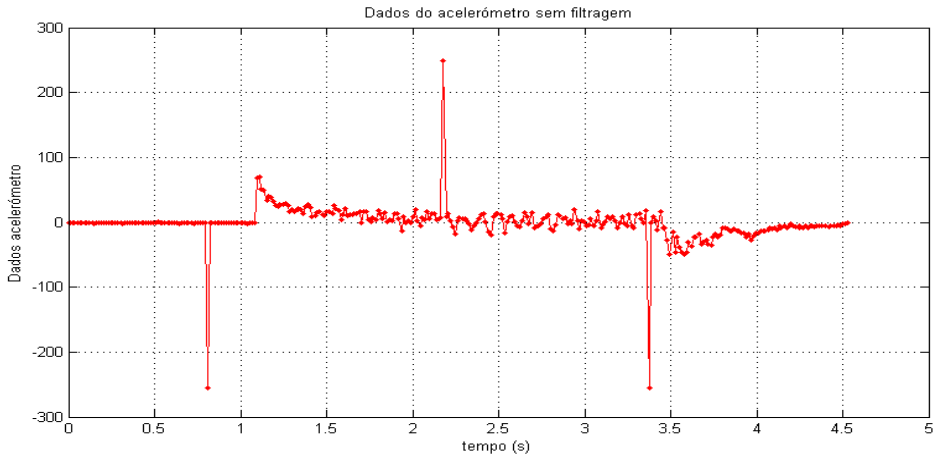


Figura 4.5 Valores do acelerómetro sem filtragem em andamento

Podemos reparar que existem vários picos incompreensíveis e ainda algum ruído associado ao movimento do robot. Reconhece-se no entanto que existiu aceleração aproximadamente aos 1,2 segundos e travagem aos 3,5 segundos.

Pode-se observar ainda o gráfico de velocidade, na Figura 4.6, obtido através destes dados.

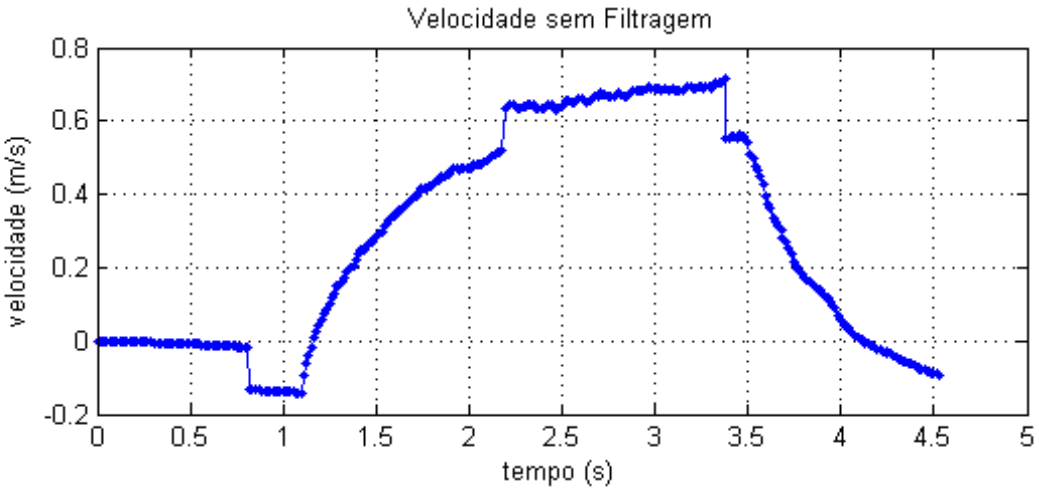


Figura 4.6 Valores de velocidade antes de filtragem e calibração do acelerómetro

Verifica-se a existência de erros desde os 0,8 segundos. No final o robot apresenta velocidade negativa, quando na realidade se encontrava parado. Todavia, com este gráfico, entende-se que existe correlação entre os dados do acelerómetro e a velocidade calculada.

4.1.2 Incorporação do Filtro de Kalman 1D para acelerómetro

A partir dos dados obtidos anteriormente percebeu-se que seria necessário proceder a uma filtragem e calibração. Os valores extremamente elevados e fora de contexto da aceleração foram desconsiderados. Aos restantes dados, optou-se por implementar um Filtro de Kalman. Sabemos que o Filtro de Kalman apresenta elevada complexidade matemática, no entanto, como felizmente apenas se pretende implementar a uma dimensão, a complexidade é muito mais reduzida.

Como foi estudado no subcapítulo 2.3, o Filtro de Kalman envolve cálculos complexos com matrizes e vectores. Todavia, neste caso, torna-se mais simples dado que pretendemos apenas filtrar os dados provenientes do acelerómetro. O acelerómetro tem 3 eixos com total independência entre si, o que permite realizar a filtragem de 3 sinais diferentes, com três filtros de Kalman de uma dimensão. As equações determinadas anteriormente, confira Tabela 1, são agora simplificadas para uma dimensão ficando reduzidas às equações da Tabela 5.

Tabela 5 - Formulas de Kalman simplificadas para 1D

Predição *(Time update)*	**Correcção / Actualização** *(Measurement Update)*
Estado *a priori*: $\underline{\dot{x}}^{-}(t) = \underline{x}^{-}(t)$ Matriz de covariância *a priori:* $P^{-}(t_i) = P(t_{i-1}) + Q(t)$	Matriz de ganho: $K(t_i) = P^{-}(P^{-} + R(t_i))^{-1}$ Estado *a posteriori*: $\underline{x}^{+}(t_i) = \underline{x}^{-}(t_i) + K(t_i) \cdot (\underline{l}(t_i) - \underline{l}^{-}(t_i))$ Matriz de covariância *a posteriori*: $P^{+}(t_i) = (1 - K(t_i))\, P^{-}(t_i)$

Recapitulando, as duas fórmulas da esquerda representam a predição do Filtro de Kalman e as três fórmulas da direita calculam as actualizações de medição. As variáveis representam: *X* para os valores filtrados, *Q* para o ruído do processo, *R* para o ruído do sensor, *P* para a estimativa de erro e *K* para o ganho de Kalman.

O filtro é aplicado a cada novo valor do acelerómetro e inicializado com os valores de ruído do processo, ruído do sensor, o valor inicial de erro estimado e o valor inicial de X. O valor inicial de P (erro estimado) não é relevante, pois é ajustado ao longo do processo bem como o valor de X, que também será actualizado ao longo do processo.

Restam apenas os valores do ruído do processo e do sensor. O ajuste destes valores é então essencial para se obter resultados aceitáveis. Para testar e ajustar o Filtro de Kalman, colocou-se o robot a movimentar-se ao longo de uma linha recta, onde sofre uma aceleração

brusca seguida de travagem, passados poucos segundos. Nos gráficos das figuras seguintes estão representados, a vermelho, os valores de aceleração originais e, a azul, os valores desta filtrados.

Primeiro observou-se a importância do ajuste da variável do ruído do processo (Q), começando com um valor bastante elevado, Q= 100, pelo que se obteve o gráfico da Figura 4.7.

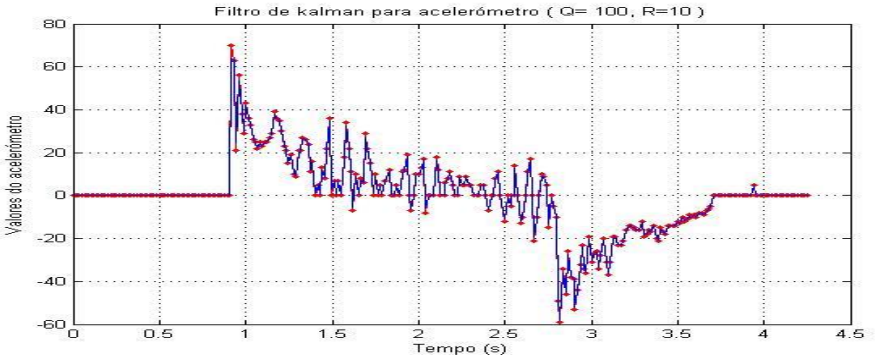


Figura 4.7 Filtro de Kalman para acelerómetro (Q=100, R=10)

Pode-se observar que não existe quase diferença entre os dados filtrados (azul) e os dados originais (vermelho). Se baixarmos o ajuste do ruído de processo, Q=3, obtém-se o gráfico da Figura 4.8.

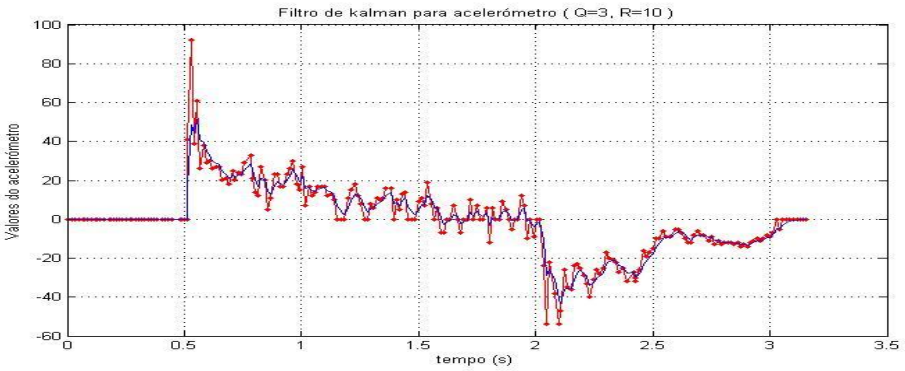


Figura 4.8 Filtro de Kalman para acelerómetro (Q=3,R=10)

Consegue-se com estes valores obter menor variação e remover algum ruído dos dados originais. Contudo como se pretende uma saída mais estática, alisando a maioria do ruído e obtendo um sinal mais constante, baixou-se mais um pouco o valor de Q para 0,065, obtendo-se o gráfico da Figura 4.9.

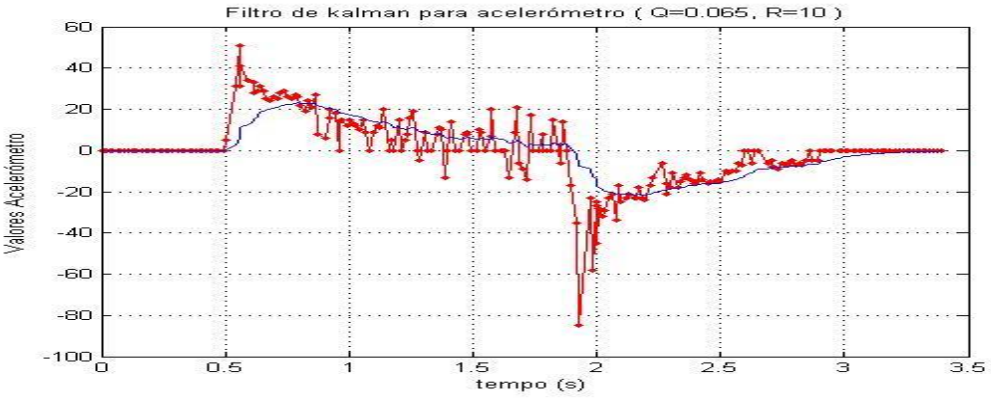


Figura 4.9 Filtro de Kalman para acelerómetro (Q=0,065, R=10)

Desta forma conseguiu-se um sinal bastante limpo, no entanto existem algumas incorrecções nos valores obtidos em relação aos dados reais, principalmente nos locais onde existem grandes variações. Contudo em comparação ao ruído inicial é uma boa aproximação. Alterando agora o valor de ajuste do ruído do sensor (R) e começando com R=1, obteve-se o gráfico da Figura 4.10:

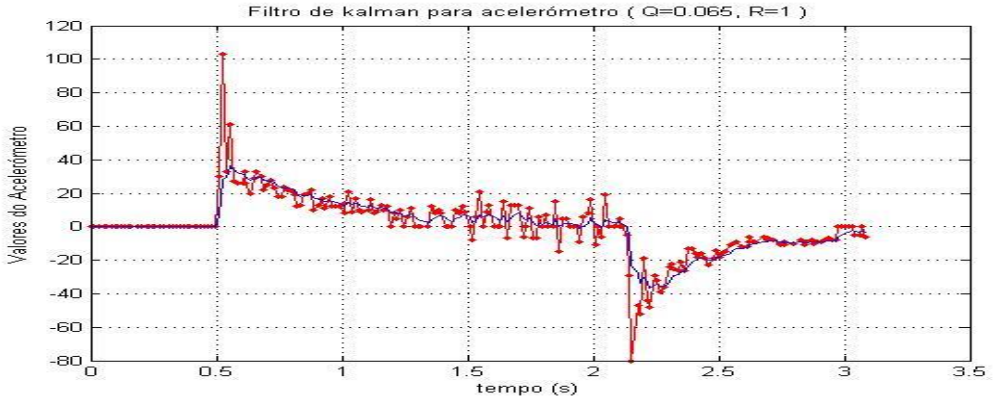


Figura 4.10 Filtro de Kalman para acelerómetro (Q=0,065, R=1)

Como era esperado, se for assumido que o ruído do sensor é bastante baixo, o Filtro de Kalman “confia” mais nos dados do sensor, fornecendo resultados mais ruidosos. Se aumentarmos um pouco o factor de ruído do sensor, obter-se-á um resultado mais estável. Com R=5, obteve-se o gráfico da Figura 4.11.

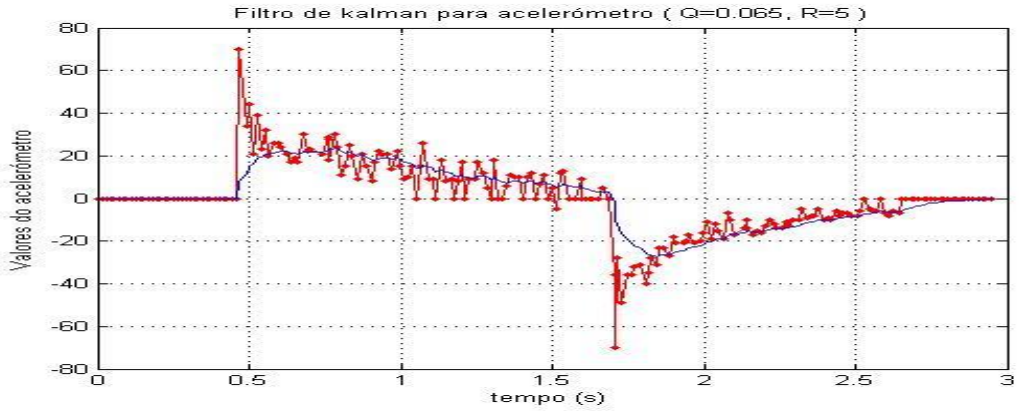


Figura 4.11 Filtro de Kalman para acelerómetro (Q=0,065, R=5)

Tal como se previa, os valores do filtro são mais estáveis. Todavia esta melhoria acarreta a desvantagem de se ter um atraso em relação ao valor real. Obteve-se assim uma filtragem simples e robusta, que permitiu obter resultados menos extremos do que os que poderíamos encontrar com um filtro passa-baixo. Deste modo, existe ainda a possibilidade de poder adaptar os resultados, conforme a situação actual do robot, confiando de forma relativa nos dados do acelerómetro.

Para a mesma situação estudada anteriormente com ruído (aceleração seguida de travagem), testaram-se agora novamente os valores do acelerómetro com os seus valores filtrados, a partir do Filtro de Kalman adaptado (Q=0,065 e R= 4), obtendo-se o gráfico da Figura 4.12.

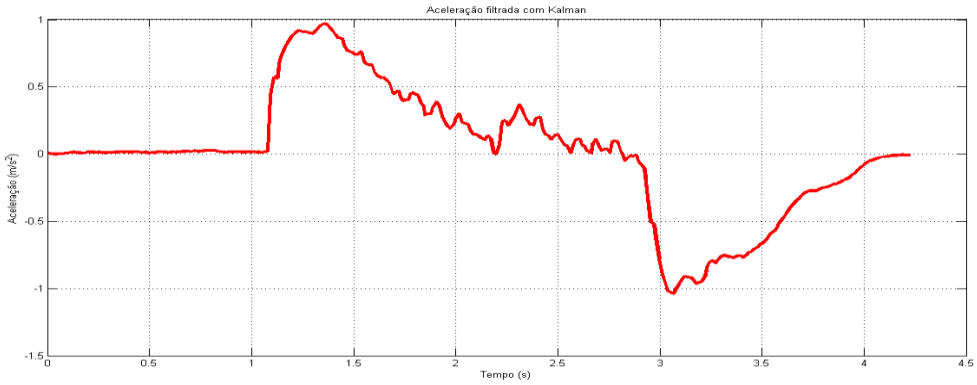


Figura 4.12 Aceleração filtrada com Kalman

Como podemos verificar, o ruído que anteriormente existia na aceleração foi praticamente limpo, conseguindo-se valores mais suaves e menos irregulares. A partir destes dados, calculou-se a velocidade, obtendo-se o gráfico da Figura 4.13.

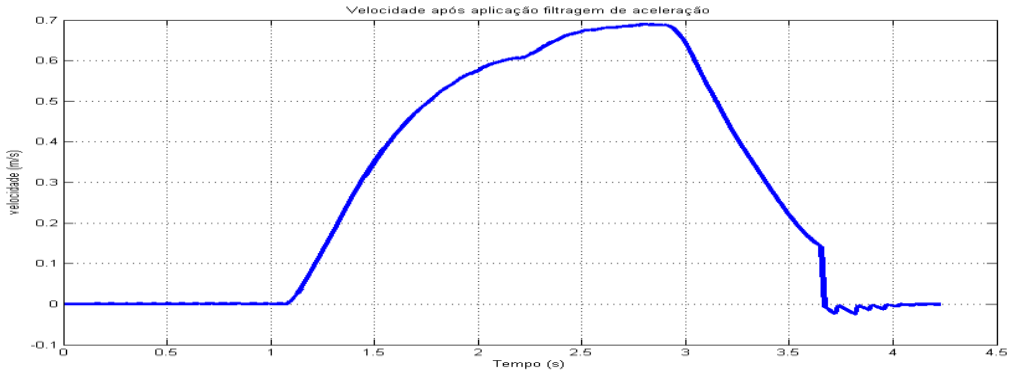


Figura 4.13 Velocidade após filtragem da aceleração com Filtro de Kalman

Em relação à velocidade, pode-se constatar que desta vez os valores são plausíveis, podendo corresponder deste modo à realidade.

4.1.3 Valores de aceleração globais

Uma vez que o robot se desloca apenas no plano horizontal, pode-se constatar que são apenas necessários os dados da aceleração, no eixo X e Y. Para além disso, sabendo ainda que o robot se movimenta conforme a aceleração a que está sujeito, pode-se utilizar esta aceleração para ter uma noção da velocidade e posição deste ao longo do tempo. No entanto, esta aceleração é calculada pelo sensor, em relação ao referencial relativo do robot, mesmo que este, vire e altere a sua posição no referencial global de posição. Por referencial global de posição, entende-se o sistema de coordenadas geográficas terrestre. [31]

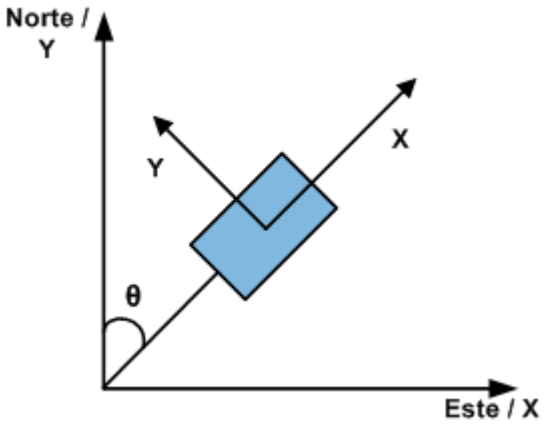


Figura 4.14 Referenciais de movimento

Pode-se, a partir da Figura 4.14, entender dois referenciais distintos. Utilizar estes dois referenciais é complicado quando se quer combinar medições de vários sensores. Deste

modo, aplicando as leis da física, é convertida a aceleração obtida no referencial do robot, para referencial global, em X na equação (4.1) e em Y na equação (4.2):

$$Ax\_ref\_global = Ax * sin(\theta) - Ay * \cos(\theta) \quad (4.1)$$

$$Ay\_ref\_global = Ax * \cos(\theta) + Ay * sin(\theta) \quad (4.2)$$

Sendo o objectivo primordial obter a posição estimada do robot em cada instante, utiliza-se o referencial global, de maneira a poder sincronizar estes dados com as coordenadas obtidas pelo receptor GPS, com os valores do giroscópio e com os dados da bússola magnética, definindo o eixo Y como Norte e o eixo X como Este. O ângulo formado entre o Norte e a direcção actual do robot é o ângulo de direcção (θ). Este ângulo de direcção, é conhecido como *yaw (terminologia em inglês que exprime o ângulo de rotação),* é obtido a partir dos dados do giroscópio e da bússola magnética. Este ângulo tem valor 0 quando coincidente com o eixo Y ou o eixo Norte, podendo obter valores de –π a π (rad), para a esquerda e direita do eixo Y, respectivamente.

É ainda importante referir que esta estimativa do ângulo de direcção (*yaw*) é calculada a partir da informação obtida da bússola magnética e da sua referência do Norte, pelo que este ângulo está também sujeito a erros, devido à influência magnética de objectos próximos do robot.

Intentando-se então comprovar a veracidade destas equações, colocou-se o robot a percorrer uma trajectória circular durante 12 segundos, realizando duas circunferências, testando assim a aceleração em X e Y. Os gráficos seguintes (Figura 4.15 e Figura 4.16) correspondem aos valores dos acelerómetros em X e Y, no referencial do robot.

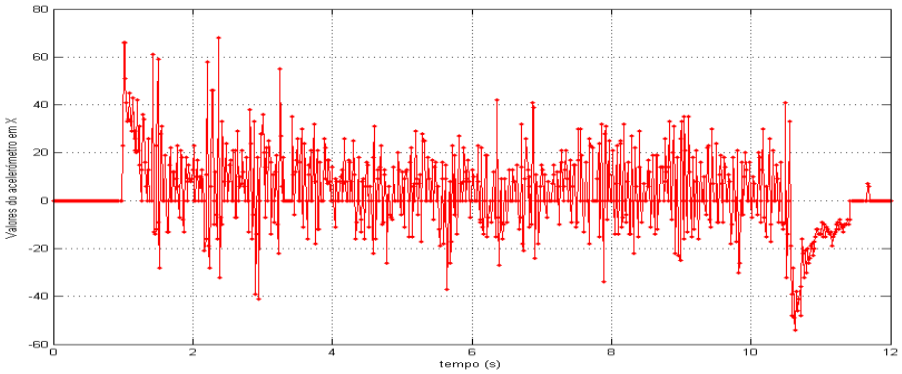


Figura 4.15 Gráfico de valores do acelerómetro em X

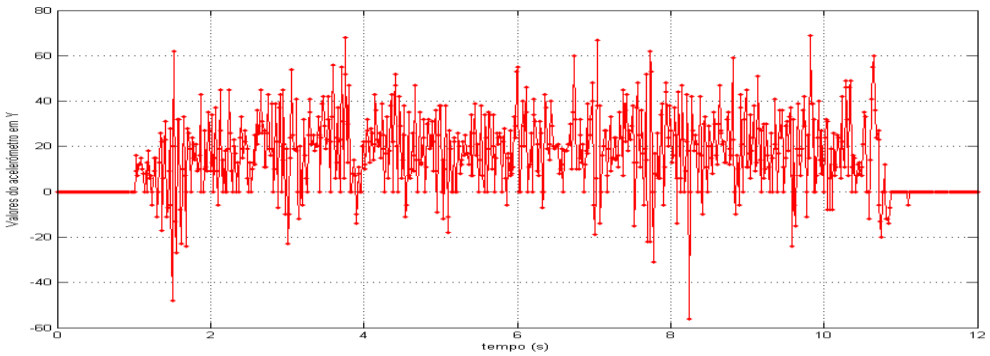


Figura 4.16 Gráfico de valores do acelerómetro em Y

No gráfico da Figura 4.17 e da Figura 4.18, apresenta-se, a vermelho, os valores de aceleração global em X e os valores da aceleração global em Y, respectivamente, e a azul os valores filtrados a partir do Filtro de Kalman, com baixo factor de coeficiente R, covariância mínima, ou seja, valores filtrados bastante ligados aos valores do sensor.

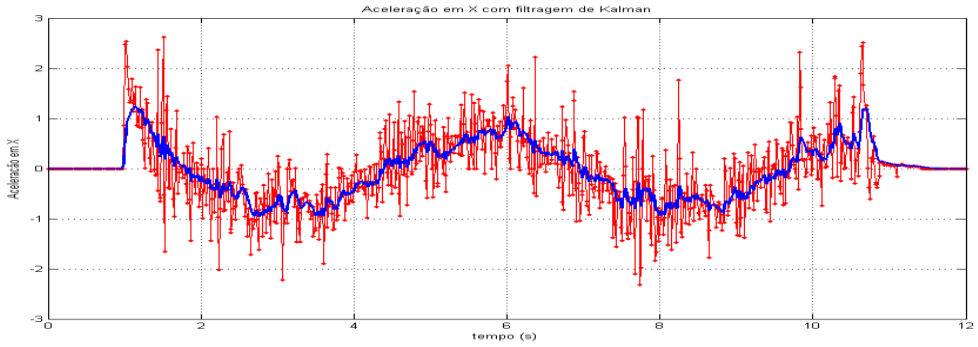


Figura 4.17 Aceleração com referencial global em X com Filtro de Kalman

Estes dados confirmam as expectativas iniciais, já que na realidade foram efectuadas duas circunferências com o robot (posição), prevendo-se que a aceleração em X e Y correspondesse a funções sinusoidais.

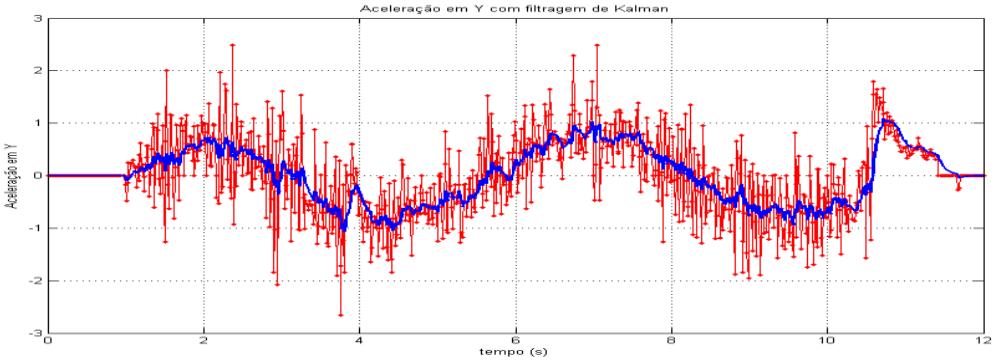


Figura 4.18 Aceleração com referencial global em Y com Filtro de Kalman

Neste ponto, findaram-se os cálculos com a aceleração, tendo-se obtido os dados pretendidos, podendo-se por isso, neste momento, determinar a velocidade e ter uma estimativa da posição instantânea.

4.1.4 Calcular velocidade e estimar posição a partir dos sensores IMU

Tendo anteriormente calibrado e filtrado os dados do acelerómetro, pretende-se agora determinar a velocidade instantânea do robot.

Para cálculo da velocidade, optou-se inicialmente por utilizar duas posições GPS consecutivas, no entanto, este método provocava um atraso de 0,4 seg, sendo que o tempo de actualização do GPS é 0,2 seg. Para além disso, os erros relacionados com o ruído do sinal GPS ir-se-iam propagar. Por assim ser, optou-se pelo cálculo da velocidade instantânea a partir dos dados do acelerómetro, este com um tempo de actualização de 0,02 seg. Sendo a velocidade calculada desta forma, é totalmente livre dos erros do sinal de GPS, mas ainda assim tem associado o indesejado ruído proveniente do acelerómetro e ainda o erro relativo à recursividade dos cálculos efectuados.

Calculando especificamente a velocidade X e Y, foram utilizadas as equações do movimento rectilíneo uniformemente variado (MRUV)[7], apresentadas em (4.3) e (4.4), respectivamente. Estas têm como entrada a aceleração global calculada anteriormente.

$$vel\_final\_x = vel\_anterior\_x + acel\_x\_global * deltaT \quad (4.3)$$

$$vel\_final\_y = vel\_anterior\_y + acel\_y\_global * deltaT \quad (4.4)$$

Importa referir que o teste realizado anteriormente para cálculo da aceleração no referencial global em X e Y, foi agora aplicado para cálculo da velocidade. Obteve-se o gráfico da Figura 4.19, onde temos a vermelho a velocidade em X e a azul a velocidade em Y. A preto está representada a direcção do robot.

[7] MRUV, Equação de posição: $V = V_o + at$

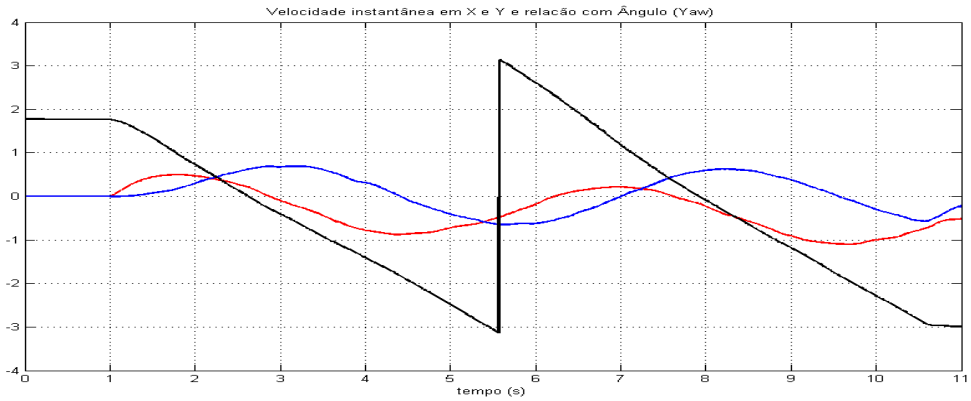


Figura 4.19 Velocidade em X e Y obtida a partir do acelerómetro e factorizada em X e Y a partir do ângulo de direcção YAW

Com a velocidade calculada a partir da aceleração, é possível obtermos uma posição estimada da localização do robot, no entanto, sendo um método recursivo, com o acumular de erros, associa um ruído considerável que não se pode ignorar.

A partir das seguintes equações de posição do MRUV (movimento rectilíneo uniformemente variado) calculou-se a posição estimada do robot (4.5) para a posição no eixo X e (4.6) para posição no eixo Y:

$$Pos\_X = Pos\_ant\_X + vel\_final\_x * deltaT + 0.5 * acel\_x\_global * deltaT\text{^}2; \quad (4.5)$$

$$Pos\_Y = Pos\_ant\_Y + vel\_final\_y * deltaT + 0.5 * acel\_y\_global * deltaT\text{^}2 \quad (4.6)$$

Realizando-se o mesmo teste, que anteriormente foi utilizado para cálculo da velocidade e aceleração, obteve-se o seguinte gráfico de posição:

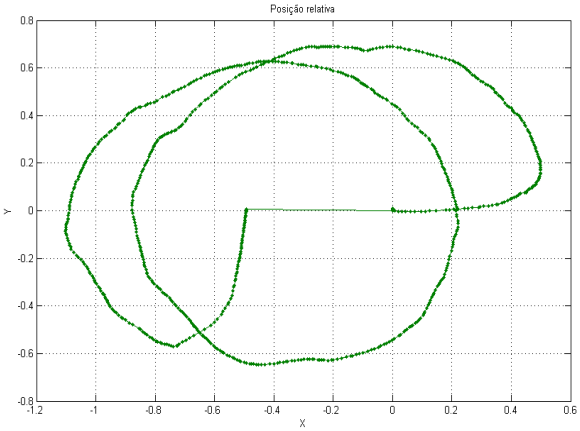


Figura 4.20 Posição obtida a partir do acelerómetro e giroscópio

Na realidade, o robot realizou um percurso circular, correspondente a duas circunferências concêntricas, sendo os seus pontos de partida e chegada coincidentes. Pela análise do gráfico verifica-se que as circunferências correspondentes ao percurso circular estão bastante deformadas, uma vez que não são totalmente circulares e não têm o ponto final coincidente com o inicial. Por esta razão, a posição estimada por este método não é uma boa aproximação.

4.1.5 Velocidade acelerómetro/encoder/gps

A velocidade calculada a partir dos dados do *encoder* não é óptima, podendo ser alvo de estudo mais aprofundado. Para cálculo desta velocidade, é realizada uma média entre o número de voltas obtidas dos *encoders* das rodas da esquerda e da direita e multiplicado pelo perímetro da roda em cada segundo.

É calculada:

$$velocidadeEncoders = leftMotorS + rightMotorS \ / 2.0 * RODA\_P; \quad (4.7)$$

```
RODA_R = 0.27/2 // raio da roda
RODA_P = PI * RODA_R  * 2; //perímetro da roda
```

Após o cálculo da velocidade média obtida pelos *encoders,* calcula-se a velocidade em referencial global, de forma análoga ao realizado anteriormente, com os valores de aceleração.

Neste caso:

$$velocidadey\_enc = velocidadeEncoders * Cos(Yaw) \quad (4.8)$$

$$velocidadex\_enc = velocidadeEncoders * Sin(Yaw) \quad (4.9)$$

Para efeitos de estudo, foi ainda obtida a velocidade, *Speed over ground* em *Knots,* $(1\ nó = 0.5144444\ m/s)$ pelos dados *GPRMC* do receptor GPS. Esta velocidade foi convertida em metros por segundo e referenciada globalmente.

$$velocidadex\_gps = (vog * 0.514444444) * Sin(Yaw) \quad (4.10)$$

$$velocidadey\_gps = (vog * 0.514444444) * Cos(Yaw) \quad (4.11)$$

Para o mesmo percurso, obteve-se o seguinte gráfico de velocidades, Figura 4.21:

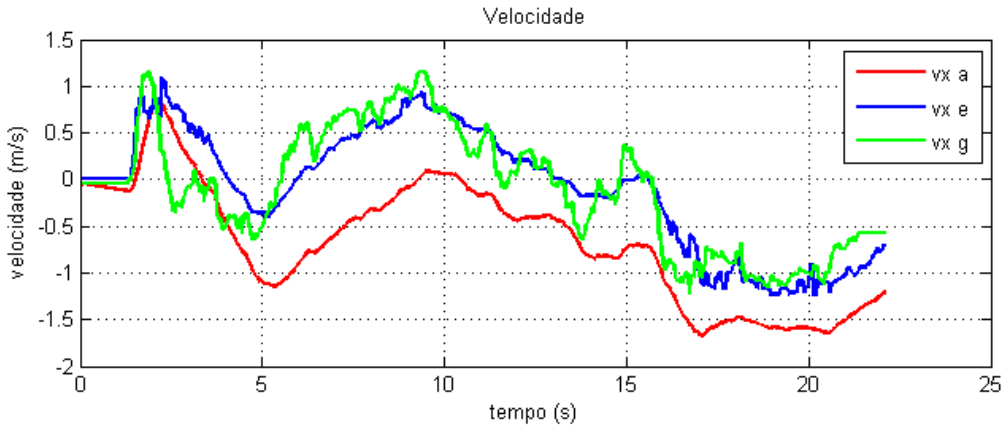


Figura 4.21 Velocidade obtida pelo acelerómetro, *encoders* e GPS

A vermelho está representada a velocidade calculada a partir dos dados do acelerómetro, a azul pelos *encoders* e a verde pelo receptor GPS. Dado que os valores de velocidade calculados pelo acelerómetro tem elevado erro associado e a como os valores de velocidade obtidos pelo sistema GPS tem muito ruído, optou-se por utilizar a velocidade dos *encoders* para implementação no Filtro de Kalman.

4.2 Determinar Posição Estimada Com Filtro de Kalman 2D

Tal como já foi referido, o Filtro de Kalman foi alvo de uma abordagem teórica apresentada no Capítulo 2.3, no entanto, da teoria até à implementação prática deste, foi percorrido um longo trajecto.

Este filtro permite filtrar o ruído e combinar diferentes medidas, de modo a calcular uma resposta. O Filtro de Kalman, anteriormente aplicado a uma dimensão, é relativamente simples, todavia, quando se pretende incorporar e conjugar valores de vários sensores, torna-se muito mais complexo.

O robot está equipado com um receptor GPS que prevê uma estimativa de posição, com precisão de alguns metros. A posição também pode ser estimada, como já foi demostrado, integrando-se a velocidade e a direcção ao longo do tempo, técnica conhecida como dead reckoning. Esta técnica gera uma estimativa da posição, mas afasta-se do valor real com o tempo, devido à acumulação erros.

O Filtro de Kalman opera em duas fases distintas: predição e actualização. Na fase de predição, a posição anterior é modificada, através das equações do movimento. As predições para a posição e a covariância dos erros são obtidas nesta fase. A covariância deve ser

proporcional à velocidade do robot, pois haverá maior incerteza em maiores velocidades. Depois, na fase de actualização, a medição da posição do robot é obtida pelo GPS. Esta medida também contém incertezas e a sua covariância em relação à covariância predita, na fase anterior, determinará o peso com que essa medição irá afectar a estimativa actualizada de posição. A estimativa, a partir da velocidade e da direcção, afastar-se-á da posição real ao longo do tempo, mas a sua combinação com a medição do GPS irá trazer a estimativa mais perto do valor real, sem se tornar ruidosa e com saltos bruscos em cada medição.

A chave para uma boa projecção do Filtro de Kalman baseia-se na capacidade de transformar as medições em estados e na forma como os estados e as medições se correlacionam ou não.

Anteriormente estudou-se o Filtro de Kalman, calculando as suas equações. Nesta fase, serão abordadas estas mesmas equações, ainda que de forma simplificada, com vista a ser mais explícita a sua aplicação ao código produzido. De referir também que foram utilizadas rotinas já desenvolvidas de cálculo matemático de matrizes. Segue-se:

Predição:

X = F*X + H*U Estado *a priori* (2.11)

P = F*P*F^T + Q Matriz de covariância *a priori* (2.18)

Actualização/correcção

Y = M – H*X	Inovação = Medida – estado transformado por H.
S = H*P*H^T + R	Covariância residual = covariância transformada por H + R
K = P * H^T *S^-1	Ganho de Kalman = variância / covariância residual (2.24)
X = X + K*Y	Actualização do estado com o ganho (2.25)
P = (I – K * H) * P	Actualização da covariância (2.26)

Onde :

X = Representa o Estado, é um vector onde X[0] = posição e X[1]= velocidade

F = Prediz estado futuro – matriz 2x2. = [1, dt], [0, 1]

P = Covariância – matriz 2x2 (alterada e calculada em cada iteração)

Y = Resíduo/Inovação - Variação entre a medição e o último estado. – Matriz 1x2

M = Medição

S = Covariância Residual.

R = Actualização mínima de covariância – Representa ruído dos sensores

K = Ganho de Kalman

Q = Covariância mínima de actualização de P – matriz 2x2 – Representa o ruído do processo.

I = Matriz identidade 2x2.

Os valores de posição obtidos pelo sinal GPS são ruidosos, mas os valores de velocidade sofrem alterações mais lentas e suaves. O tempo entre medições varia, pois, cada vez que um sensor disponibiliza um determinado valor, é aplicado o filtro para correcção da posição.

No código desenvolvido, poder-se-ia ter utilizado um Filtro de Kalman em 3D, (posição, velocidade, aceleração) no entanto, o nível de complexidade e a exigência de tempo de computação seria muito penalizador, pelo que se evidenciou ser mais simples utilizar dois filtros de duas dimensões e optimizar cada um individualmente. Assim sendo, um filtro calcula a velocidade com um Filtro de Kalman 2D (velocidade e aceleração), depreendendo-se então que, utilizando a velocidade obtida pelos dados do *encoder* (a azul) e a aceleração pelos valores do acelerómetro (a vermelho) obtém-se a velocidade estimada pelo filtro (a preto), representados na Figura 4.22.

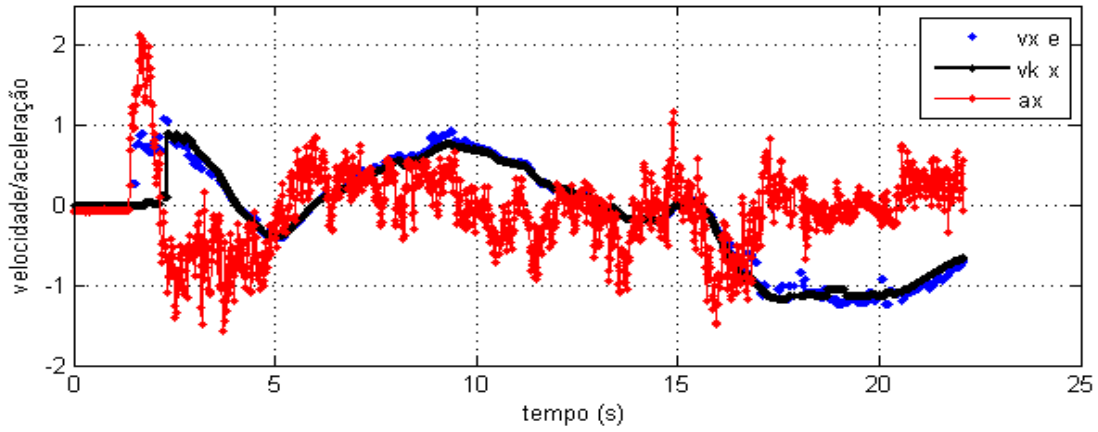


Figura 4.22 Velocidade estimada pelo Filtro de Kalman

O segundo filtro utiliza esta informação mais precisa da velocidade, para filtrar os dados na posição. Finalmente, para podermos trabalhar no plano horizontal, utilizaram-se dois filtros para o eixo X (velocidade e posição) e dois filtros para o eixo Y. Deste modo, utilizando a velocidade calculada anteriormente e o sinal GPS com elevado ruído induzido (a vermelho), obteve-se uma estimativa da posição instantânea do robot (a preto). Na Figura 4.23, pode-se comparar a posição estimada com a posição calculada, a partir das equações MRUV[8] (a azul), e a posição desejada (verde).

[8] MRUV, Equação de posição: $S = S_o + V_o t + at^2/2$

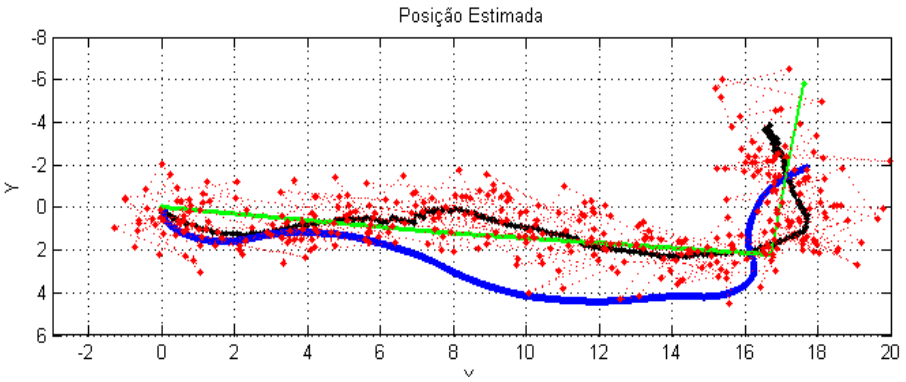


Figura 4.23 Posição Estimada com Filtro de Kalman

O Filtro de Kalman capta a velocidade ao longo do tempo e confia essencialmente na informação da velocidade, para actualizar a posição estimada, em detrimento das medidas de posição do GPS. Tais factos sucedem pois, quando a aceleração e velocidade são detectadas existe imediatamente correcção na posição, enquanto o GPS tem um atraso muito maior nesta correcção, além disso o ruído inerente ao sistema GPS também afecta estimativa da posição.

Por fim é de referir a extrema importância dos coeficientes de configuração do filtro (Figura 4.23) sendo necessário um bom equilíbrio entre estes. Para se atingirem bons valores de configuração foram realizados vários testes, analogamente aos realizados no subcapítulo 4.1.2.

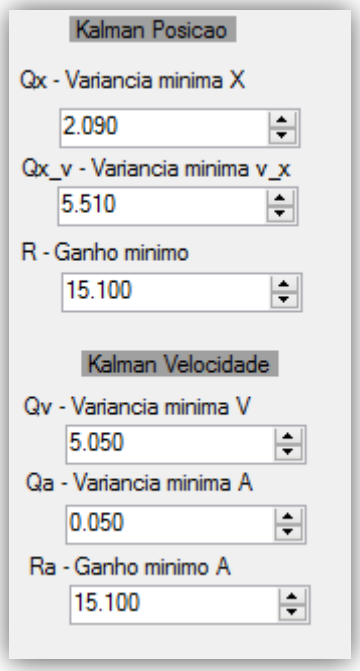


Figura 4.24 Coeficientes de configuração do Filtro de Kalman

Podemos entender a correspondência entre estes coeficientes de adaptação do filtro e as funções utilizadas no código:

Qx ou Qv_x- Valor mínimo de variância de estado de posição/velocidade

Qv ou Qa - Valor mínimo de variância de estado de velocidade/aceleração.

R ou Ra - Valor covariância (Ganho mínimo – (Erro receptor GPS / Erro acelerómetro/ Erro *encoders*)

pd - Variância inicial (Alterada em cada iteração)

IX - Posição inicial

Relembrando, sendo o filtro de duas dimensões, calcula-se primeiro a velocidade estimada e, a partir desta, a posição estimada. Por esta razão é que se pode utilizar as mesmas variáveis de adaptação do filtro para valores diferentes.

4.3 Controlo De Movimento

Pretendendo-se que o robot siga a trajectória desejada, após se ter calculado a posição deste, é necessário conseguir dar ordens precisas para o mover correctamente. O controlo do movimento é um dos blocos principais da navegação autónoma e consiste em obter uma correcta adaptação dos motores, relativamente aos dados obtidos de posição.

4.3.1 Cálculo da direcção de condução

Para controlo do movimento do robot, importa efectuar a leitura dos valores do giroscópio e da bússola magnética, para a obtenção do ângulo entre o Norte e a direcção em que o robot está orientado (*yaw*). Urge ainda calcular o ângulo de condução (*dirCond*), ângulo esse que representa a direcção de para onde desejamos ir. A partir destes dois ângulos, determina-se o erro de direcção (*errodir*), o qual é o ponto de partida para o cálculo dos valores de potência a enviar a cada motor.

Os valores obtidos a partir da bússola magnética e do giroscópio (Figura 4.25) estão compreendidos entre –π e π, sendo que 0 corresponde ao Norte e π/2 a Este.

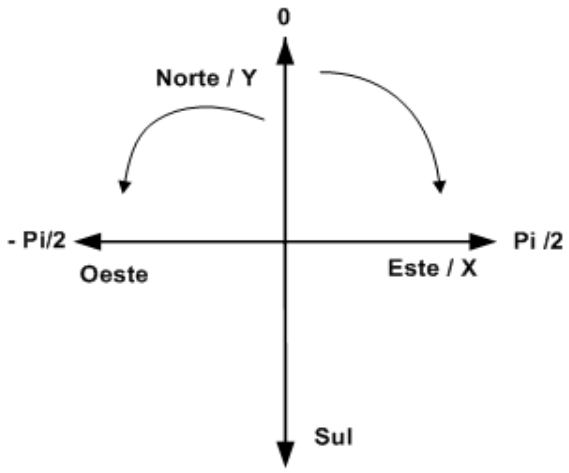


Figura 4.25 Ângulo de direcção (YAW) obtido pelo giroscópio

Para cálculo da direcção desejada, é necessário definir e calcular alguns parâmetros: a recta correspondente ao percurso entre o ponto inicial e o final, a distância entre a posição actual e o ponto final e a distância entre a posição actual e o ponto mais próximo da recta definida.

Com os parâmetros anteriores conhecidos, verifica-se se a distância entre a posição actual e o ponto final é menor que a distância entre a posição actual e o ponto mais próximo do trajecto desejado. Analisa-se ainda se a distância ao ponto final é menor que uma distância previamente determinada. Nestes dois casos, o ângulo de direcção de condução é calculado entre o Norte e o ponto em questão. Caso não se verifique nenhuma destas situações, o ângulo de direcção é calculado entre o Norte e o ponto da recta desejada mais próximo da posição actual.

Esta situação, que já foi referida, é colocada em prática caso a distância entre a posição actual e a recta desejada seja superior a uma distância previamente definida. Na Figura 4.26, encontra-se um exemplo dos ângulos e percursos que se explicaram anteriormente.

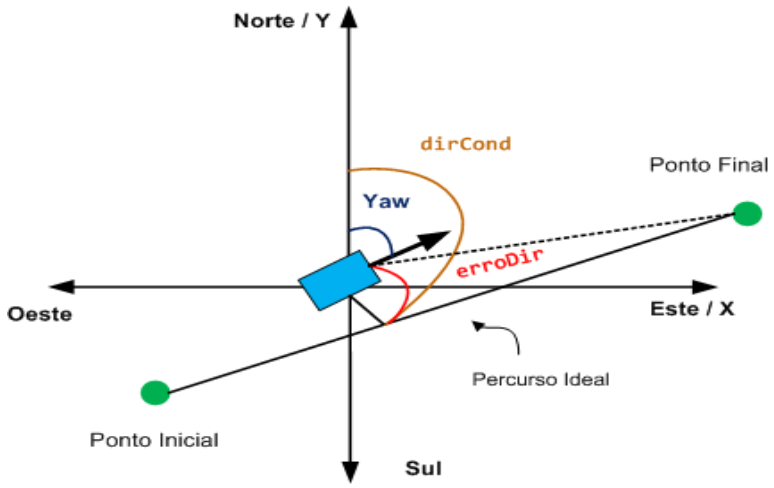


Figura 4.26 Calculo do erro de direcção

Com o ângulo de direcção actual (yaw) e o ângulo de direcção de condução desejada (dirCond) é calculado o erro de direcção (errodir). Com o intuito de obter um erro de direcção que seja sempre o mínimo possível, ou seja, que corresponda sempre ao lado para onde o erro seja menor, utilizou-se o seguinte algoritmo:

```
erroDir = dirCond - yaw // se o erro for maior ou menor que pi rad

Se valor absoluto (erroDir) > π
E se yaw > 0
        erroDir = 2π + dirCond - yaw; //> 0
Se não
        erroDir = - 2π + dirCond - yaw;//< 0
```

A comparação do erro com π serve para confirmar que o ângulo entre a direcção actual e a pretendida é sempre o menor possível, ou seja, serve para garantir que o robot nunca precisa de virar mais do que π radianos (180 graus).

Na interface gráfica desenvolvida está disponível, em tempo real, a informação da estimativa de direcção actual do robot, a vermelho, e a direcção para o ponto desejado, a azul, conforme mostrado na Figura 4.27.

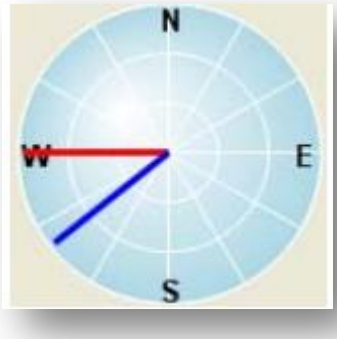


Figura 4.27 Demostração gráfica dos ângulos de controlo do robot

4.3.2 Movimentação do robot: Relação ângulo – motores

No capítulo 3.1.2, estudou-se o funcionamento e a comunicação com os motores. Estes motores funcionam a partir de dois canais de comunicação, denominados aqui *potência0* e *potência1*, podendo variar com valores entre 0 e 32768, sendo o valor mútuo de 16384 para paragem. Para relembrar, observe-se a Tabela 4 do subcapítulo 3.1.2.

A variável *potência0* influencia a velocidade e o andamento do robot, em frente (>16384) ou para trás (<16384) do robot, aumentando de intensidade à medida que se afasta do valor intermédio (16384). A variável *potência1* é responsável por virar com maior ou menor intensidade, para a esquerda (<16384) ou para a direita (>16384).

Para calcular o valor da variável *potência0* a enviar no canal de comunicação, utilizou-se a seguinte expressão:

*Potencia[0] = PARADO - MOTDIR * POTMAX * limPotencia;*

Onde:

PARADO = 16384
MOTDIR = 1 para a frente, -1 para trás.
POTMAX = 16384
limPotencia = definido pelo utilizador

Variando o limitador de potência e mantendo a direcção do motor para a frente obteve-se o seguinte gráfico Figura 4.28. De notar, que com a *potência0* igual a 0, o robot atinge a velocidade máxima.

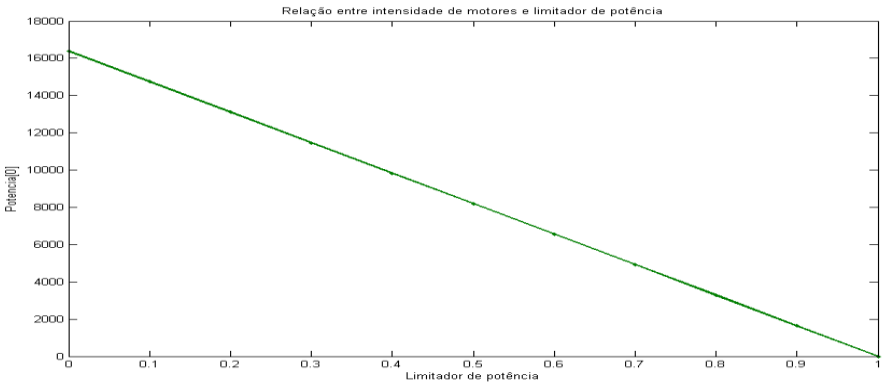


Figura 4.28 Relação entre a potência e limitador

Tendo o modelo de controlo da intensidade dos motores para a frente e para trás, interessa saber controlá-los para a esquerda e para a direita. Pretende-se então poder variar a intensidade de rotação, dependendo do erro do ângulo, entre a orientação actual e a direcção para a posição desejada.

Calculou-se a variável *difPotencia* que permite controlar os motores do robot, de modo que este vire para a direcção pretendida mais ou menos acentuadamente, de acordo com o erro de direcção:

$$difPotencia = Valor\ absoluto(erroDir / \pi * (maxDifPot - minDifPot)) + minDifPot;$$

O valor *minDifPot* pode ser definido pelo utilizador e corresponde ao valor mínimo de diferença entre os valores de potência dos motores da esquerda e da direita. Resumindo, o *minDifPot* limita a intensidade mínima com que o robot roda, ou seja, fá-lo rodar mais bruscamente à medida que aumenta este valor.

O erro é calculado em radianos, variando entre –π e π, aqui divide-se por π de modo a variar este erro entre valores de -1 a 1. O valor da *difPotencia* é sempre positivo. Por exemplo para *minDifPot=10000* e *maxDifPot=16384* obtêm-se os valores da Figura 4.29.

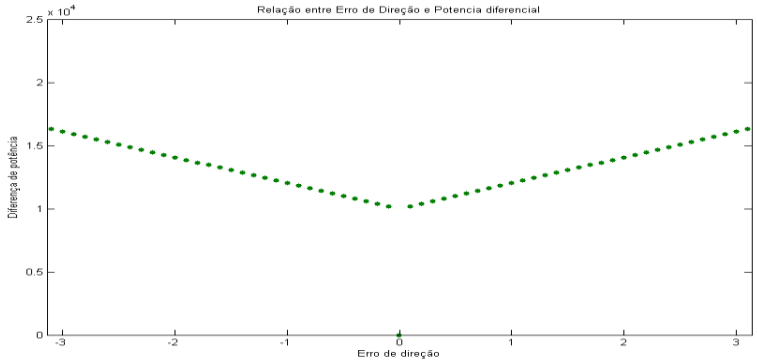


Figura 4.29 Relação entre erro de direcção e diferença de potência de rotação

Por fim, para diferenciar a esquerda da direita, bastou verificar o sinal do erro de direcção (*erroDir)* e somar ou subtrair o valor da diferença de potência (*difPotencia)* calculado.

```
SE (erroDir >= 0)
{
     //Virar para a direita
     Potencia[1] = PARADO +  MOTDIR * difPotencia);
}
senão
{
     //Virar para a esquerda
     Potencia[1] = PARADO - MOTDIR * difPotencia);
}
```

Observando o gráfico da Figura 4.30, onde se varia o erro de direcção entre -4 e 4, obtém-se os valores da *potencia1*.

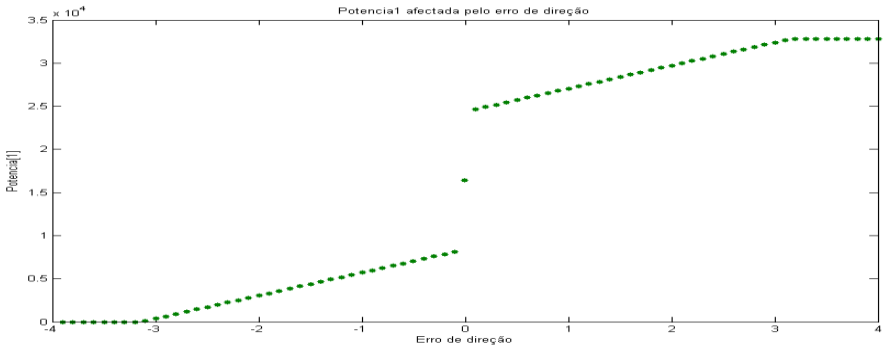


Figura 4.30 Valores de *Potencia1* de acordo com o erro de direcção

Se o erro de direcção for 0, a potência mantém o valor de 16384 como previsto. Com erro negativo, a *potencia1* toma valores menores que 10000, rodando para a esquerda. Com erro positivo, a *potencia1* toma valores maiores que 25000, rodando para a direita. O declive da recta é afectado pela variável `minDifPot`.

4.3.3 Escala temporal

O algoritmo desenvolvido comunica e interage com o robot em tempo real, recebe os dados dos sensores, transforma-os e transmite informações para o controlo dos motores.

Os tempos de execução são:

1. Tempo de actualização dos motores do robot = 100 ms;
2. Tempo de espera pelos sensores:
    a. GPS = 200 ms;
    b. Sensores (acelerómetro e giroscópio) = 20 ms;
    c. Bússola magnética = 220 ms;
3. Posição estimada obtida em média a cada 56.0 ms.
4. Tempo de comunicação com o robot (via wireless, 802.11) praticamente nulo;

A partir do esquema da Figura 4.31 torna-se compreensível a ordem de acontecimentos.

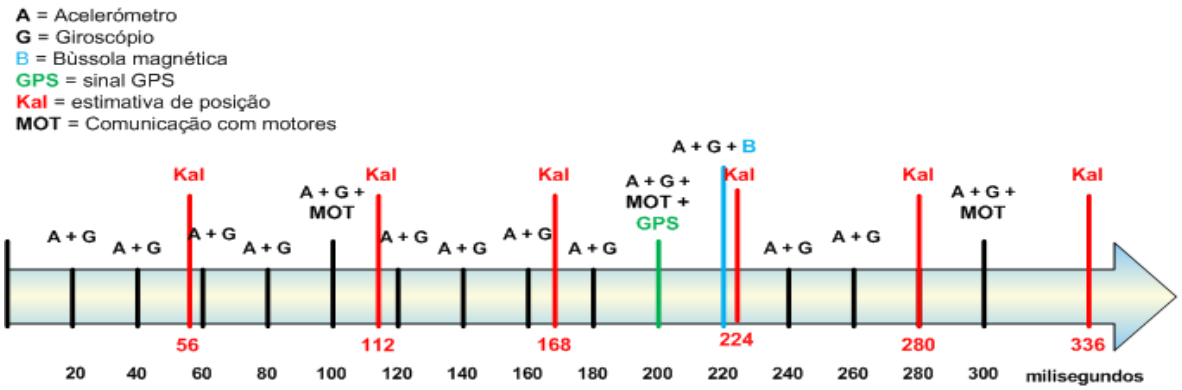


Figura 4.31 Escala temporal

Dado que em cada 20 ms são disponibilizados novos valores do acelerómetro e do giroscópio, estes são na sua maioria desprezados, devido à falta de tempo para os tratar. Durante o processamento da informação dos sensores no algoritmo são corridos 56ms, adquirindo-se no final destes a posição estimada. A comunicação com o robot ocorre a cada 100 ms, pois é o tempo mínimo necessário para o robot processar a informação enviada, o que se afigura portanto como uma restrição técnica deste robot.

O sistema GPS fornece valores a cada 200 ms, sendo muito mais lento que os outros dois sensores referidos anteriormente, no entanto, o sensor com maior tempo de actualização é a bússola magnética, o que influencia o giroscópio no cálculo da orientação do robot.

Atende-se a situação do robot seguir a 1,5 m/s, velocidade esta correspondente à metade da potência dos motores. Sendo a posição estimada obtida a cada 60ms e os motores actualizados a cada 100 ms, pode-se chegar à situação de ter uma posição estimada com erro de 0,10 metros. O valor não é significativo, mas, para futuros testes a velocidades superiores, este erro poderá ser relevante.

5 Testes finais e Resultados

Neste capítulo serão apresentados os testes efectuados ao funcionamento da aplicação, sendo que estes testes foram divididos em dois grupos, por forma a serem avaliadas metas diferentes.

No primeiro grupo deste capítulo, serão realizados os testes necessários para verificar a autonomia do robot, em vários ambientes e em diferentes circunstâncias.

Quando se atingir autonomia de deslocamento com êxito, iniciar-se-á então o seu aperfeiçoamento, principalmente melhorando a posição real estimada e afinando o controlo dos motores.

Na segunda parte deste capítulo, apresentar-se-ão os testes finais, testando a autonomia do robot em situações não óptimas, ou seja, facultando condições que dificultam uma leitura sem erros por parte dos sensores. Finalmente verificar-se-á o erro de localização em relação ao percurso desejado e comentar-se-ão os principais factores que conduziram a estes erros.

De salientar que, para cada teste aqui apresentado, muitos outros nas mesmas condições foram realizados, no entanto, apenas os testes, que melhor informação disponibilizam, serão apresentados.

5.1 Testes à navegação autónoma do robot

No primeiro grupo de testes realizados, utilizaram-se condições de navegação óptimas, isto é, realizam-se num local amplo, onde se podia obter um bom sinal GPS, num terreno plano e a uma velocidade moderada (sensivelmente a 50% da potência máxima do robot, +/- 1.5 m/s). Foram escolhidas estas condições, pois neste primeiro caso o objectivo é demostrar que o robot consegue realmente ter autonomia de navegação.

Foram testados nestas condições três tipos de percurso diferentes recorrendo-se a uma recta, a um rectângulo e a uma circunferência de forma a analisar o comportamento do robot nas diferentes situações. No primeiro caso, o ângulo de erro de direcção não sofre grandes alterações, no segundo caso, tem algumas alterações mais bruscas, no caso da circunferência é alterado um maior número de vezes, mas essas alterações sucedem-se de um modo mais suave.

No segundo grupo de testes, alteraram-se as condições de navegação, introduzindo-se fontes de erro, ou seja, recorrendo a um sinal GPS mais ruidoso e navegando a maior

velocidade (potência máxima correspondente a uma velocidade de 3.5 m/s). Deste modo podemos entender como é afectada a trajectória consoante as fontes de erro.

Finalmente falta referir que os parâmetros analisados baseiam-se na posição estimada pelo Filtro de Kalman, no erro de direcção e na velocidade/aceleração do robot em cada instante.

Nas figuras retiradas da nossa aplicação e apresentadas seguidamente, encontrar-se-á a verde o percurso desejado e a vermelho o percurso realmente percorrido, marcado com as posições estimadas do Filtro de Kalman. Os primeiros testes realizaram-se na Parada da Academia Militar, em Lisboa, enquanto os segundos testes são realizados na estrada de calçada, rodeada por vegetação, também na Academia Militar.

5.1.1 Teste à capacidade de movimentação autónoma

Parâmetros de teste:

- Percurso, Erro de Direcção, Direcção de Condução, Posições estimadas

Condições ideais:

- Bom sinal GPS, terreno plano, velocidade de 1.5 m/s (50%)

Percursos:

- Recta, rectângulo, circunferência

No primeiro teste efectuado, escolheram-se dois pontos para definir, como percurso desejado, uma recta com aproximadamente 35m. Esta recta foi marcada num local plano, afastado de edifícios para evitar falhas de sinal GPS. Obteve-se o percurso definido na Figura 5.1.

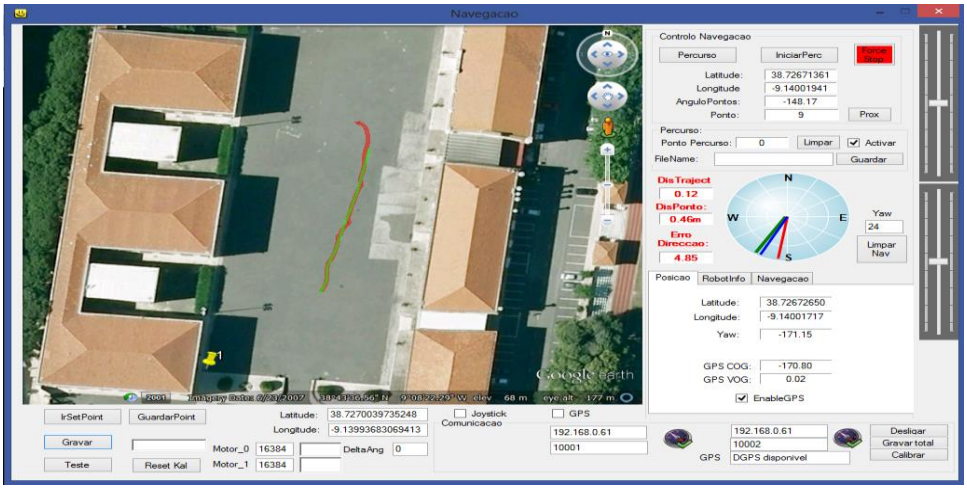


Figura 5.1 Percurso rectilíneo com condições ideais

Observa-se que o robot reconhece o primeiro ponto e efectua uma trajectória suave para o atingir. Durante o seguimento do percurso desejado existe algum erro, representado como oscilação em torno do percurso desejado. Este erro deve-se principalmente ao controlo dos motores do robot, porque não existe grande variação do erro de direcção, Figura 5.2, nem da direcção de condução, Figura 5.3, dado que o robot segue quase sempre em frente.

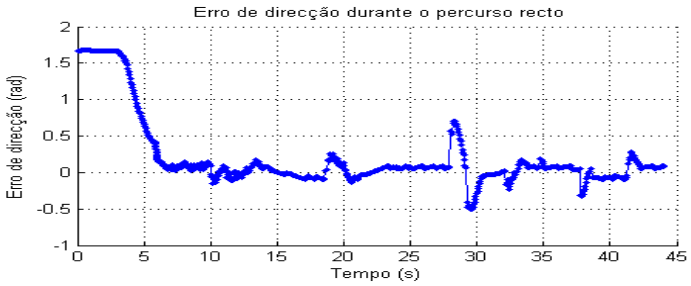


Figura 5.2 Erro de direcção durante o percurso rectilíneo

Pode-se ainda verificar que o robot seguia em direcção a sudoeste, sendo que ultrapassa a barreira dos π rad (180º) pelos 14 segundos. Esta passagem de ângulo, de π para –π também é fonte de alguns erros, dado que afecta os cálculos da velocidade e posição estimada, nomeadamente na passagem de referencial. Outras falhas ocorrem por erros no sistema de localização, no entanto a realização da recta pelo robot de forma autónoma foi um sucesso, uma vez que cumpriu o pretendido com um erro mínimo.

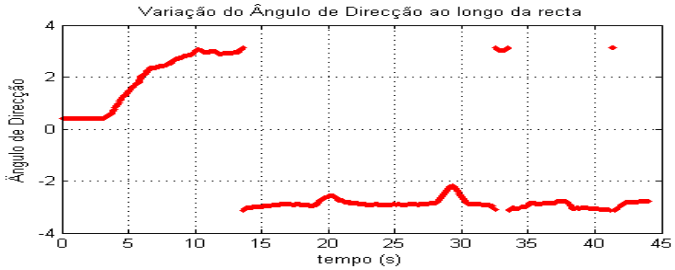


Figura 5.3 Variação do ângulo de direcção ao longo do percurso

No segundo teste, definiu-se um rectângulo com 15m de largura e 25 de comprimento (Figura 5.4).

Figura 5.4 Percurso rectangular com condições ideais

Este teste teve como interesse o estudo da autonomia para longos percursos, o estudo das variações bruscas de direcção e a capacidade do algoritmo aplicado estimar uma boa localização e controlar os motores, para assim se obter a menor distância possível entre o percurso desejado e o realizado.

Após a realização do teste, obteve-se o seguinte gráfico de posição da Figura 5.5.

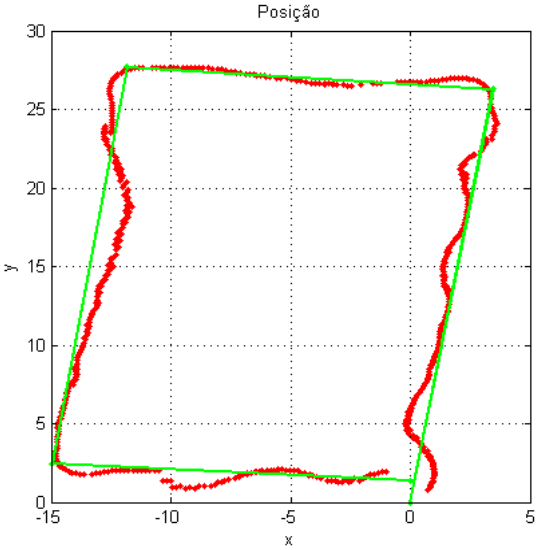


Figura 5.5 Posições estimadas (metros)

Com este gráfico pode-se verificar que a posição estimada (a vermelho), a partir do Filtro de Kalman, está bastante próxima do percurso desejado (a verde), em todo o percurso. Existem algumas oscilações, principalmente devido ao controlo dos motores do robot e à estimação do erro de direcção no momento de mudança de ângulo, aquando da passagem pelo ponto cardeal Sul, aproximadamente no ponto (-14,24).

Interessa por isso verificar os erros de direcção relacionados com a direcção do robot em cada instante do percurso. Observa-se pelo gráfico da Figura 5.6 que as curvas do percurso ocorrem nos 17, 25 e 40 segundos, instantes em que o erro de direcção sofre grande alteração. Pode-se verificar ainda que, entre os instantes 22 a 24, existem oscilações de ângulo de direcção entre $-\pi$ e π, o que incita a erros no cálculo da posição estimada e como tal no controlo do robot.

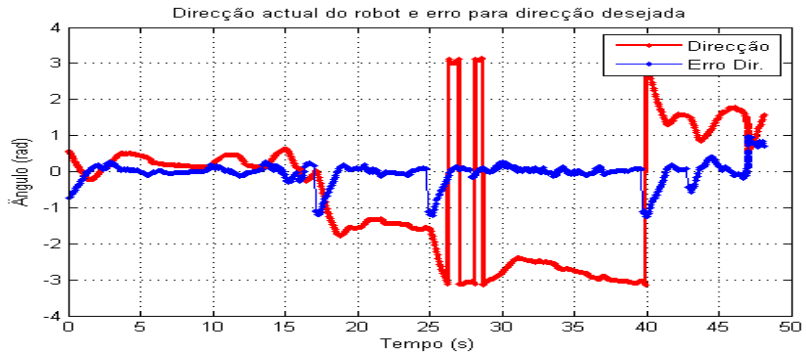


Figura 5.6 Erro de direcção e Direcção de condução do robot

Em relação à distância máxima entre o percurso real e o desejado, verificou-se que, em módulo, o trajecto percorrido nunca se afasta mais de 1,3 metros do percurso desejado (Figura 5.7) e que o maior erro ocorre após as oscilações de fase referidas anteriormente. Existe ainda algum erro causado pelas mudanças de direcção referentes às curvas do trajecto desejado.

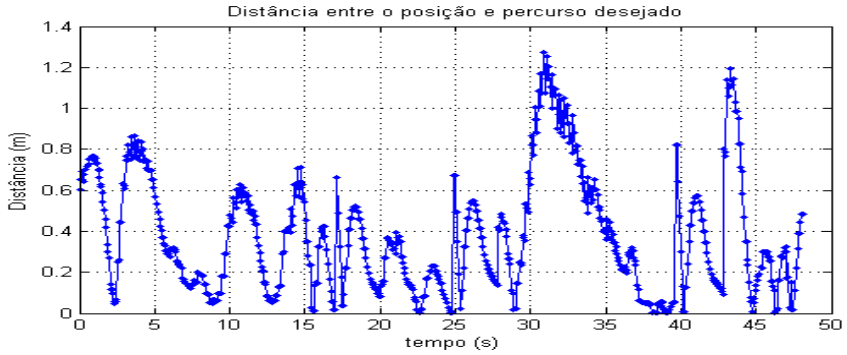


Figura 5.7 Distância em módulo entre posições actuais e percurso desejado

No terceiro teste, marcaram-se 9 pontos próximos, de modo a construir uma circunferência de aproximadamente 5 metros de raio, como se pode verificar na Figura 5.8. Consegue-se verificar que o trajecto executado pelo robot foi próximo do percurso desejado.

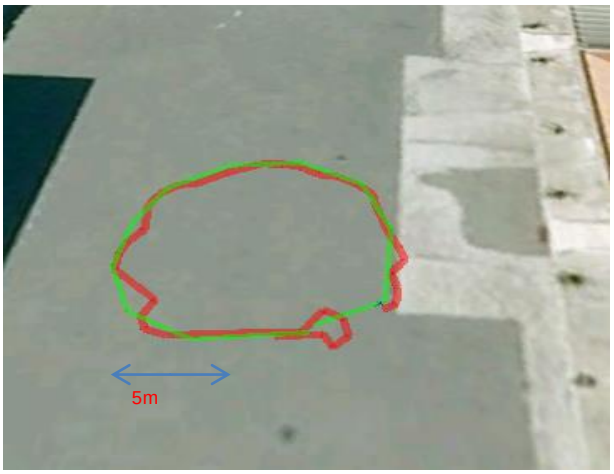


Figura 5.8 Percurso circular em condições ideais

Analisando o sinal GPS e a posição estimada no final (Figura 5.9), verifica-se que a posição fornecida pelo filtro é muito mais suave e contínua, devido ao facto de o filtro integrar a informação do acelerómetro e encoder, (aceleração e velocidade) e impedir que posições sucessivas tenham demasiado desfasamento.

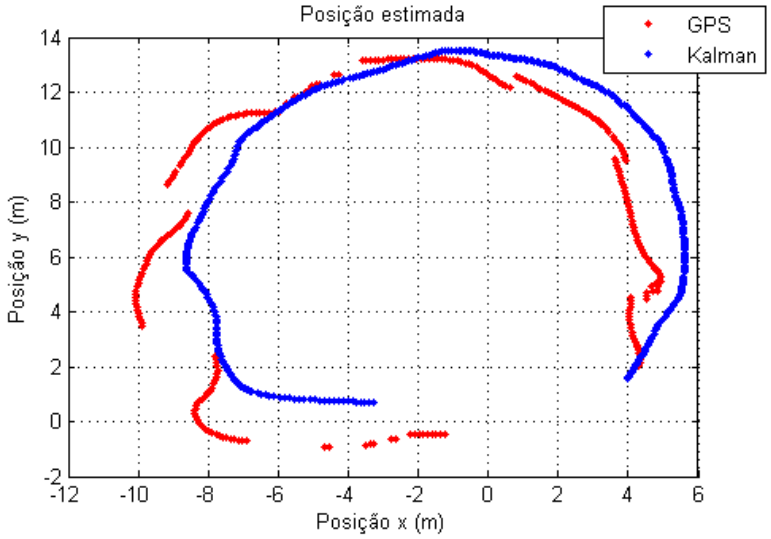


Figura 5.9 Posições GPS e Kalman

Neste caso, verifica-se que o erro de direcção (Figura 5.10) tem alguma oscilação, o que é normal já que o percurso requer constantes actualizações da direcção de condução (Figura 5.11). Também neste caso confirma-se um maior erro de direcção na transição de –π para π.

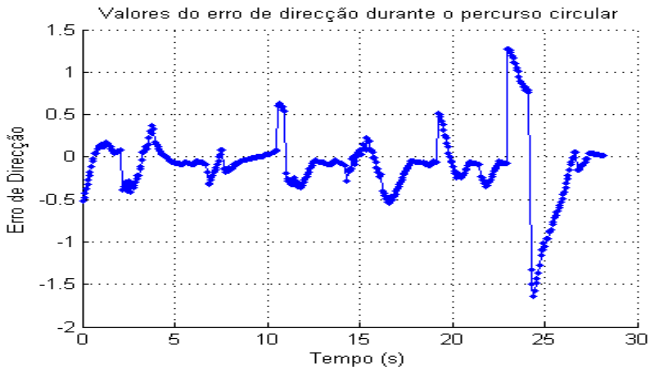


Figura 5.10 Erro de direcção do Robot em percurso circular

Como era suposto, a direcção do robot (Figura 5.11) é quase linearmente decrementada (uma vez que o sentido do deslocamento é realizado para o lado esquerdo) dando uma volta de 2π rad (360º).

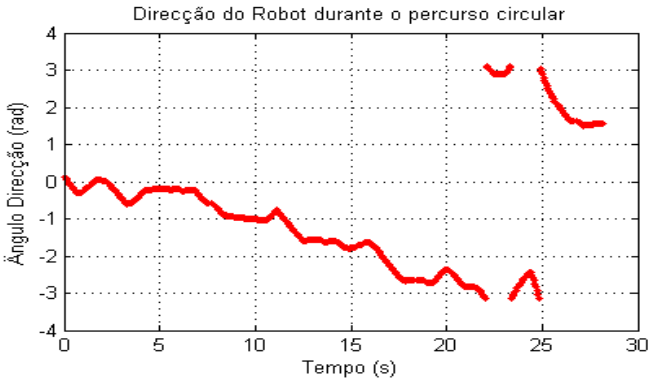


Figura 5.11 Direcção instantânea do robot em percurso circular

5.1.2 Teste à capacidade de resposta do robot em situações não ideais

Parâmetros de teste:

- Percurso, Erro de Direcção, Direcção de Condução, Posições estimadas, erro de posição

Condições não ideais:

- Fraco sinal GPS, terreno irregular, velocidade de 1.5 m/s (50%) a 3.2m/s (90%)

Percursos:

- Recta com fraco sinal GPS, rectângulo a elevada velocidade

Nos últimos dois testes que serão aqui apresentados, introduziu-se propositadamente algum erro no sistema, sem ser no entanto em demasia para não se perder por completo o controlo da navegação autónoma do robot. Como se pode entender, foram realizados outros testes com resultados menos interessantes, contudo não são aqui alvo de estudo.

Este caso de estudo foi efectuado num percurso rectilíneo, sobre uma estrada com algumas irregularidades, coberta de árvores, onde o sinal GPS obtido era mais fraco. Pode-se observar o local na Figura 5.12. Note-se que o percurso está representado a verde exemplificando o percurso desejado e a vermelho o percurso percorrido. O percurso foi executado duas vezes, uma em cada sentido da recta.

Para uma melhor percepção do trajecto estudado, verifique-se o gráfico da Figura 5.12. O percurso desejado era rectilíneo, com aproximadamente 60 metros. A vermelho pode observar-se a posição estimada (percurso percorrido), nas condições não ideais. Os dois traços indicam que se testou o robot nas duas direcções. De um modo geral, os dados obtidos para a posição estimada do robot não fogem muito do percurso desejado.

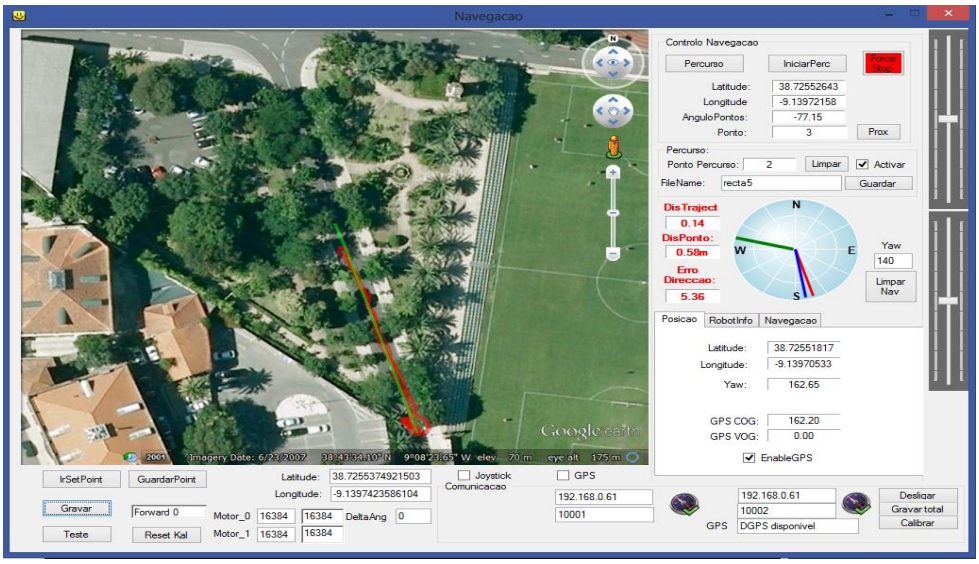


Figura 5.12 Percurso rectilíneo em situação não ideal

Observando o gráfico de posição da Figura 5.13, consegue-se verificar que o trajecto percorrido se encontra com pouco erro, ainda que tenha mais oscilações que, a recta efectuada anteriormente nas condições ideais. Pode-se reparar ainda que as posições estimadas não sofrem oscilações bruscas, mesmo com o sinal de GPS ruidoso.

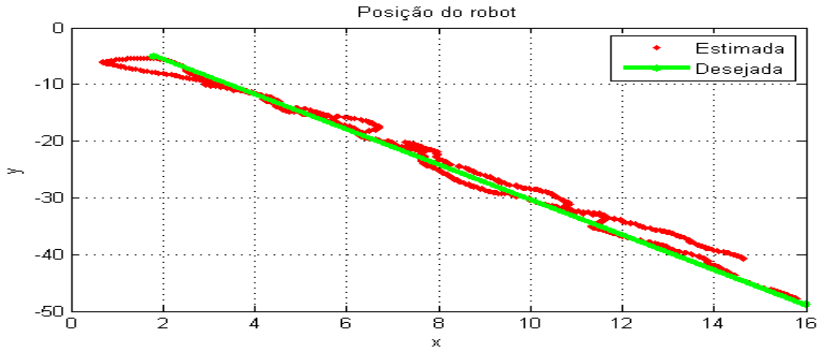


Figura 5.13 Posição do robot para percurso rectilíneo não ideal

Relativamente ao erro de direcção (Figura 5.14), tal como era previsto, é apresenta um erro ligeiramente superior ao erro obtido na recta efectuada, nas condições ideais, sendo ainda notório um maior pico após a mudança de ângulo aos π rad (180º).

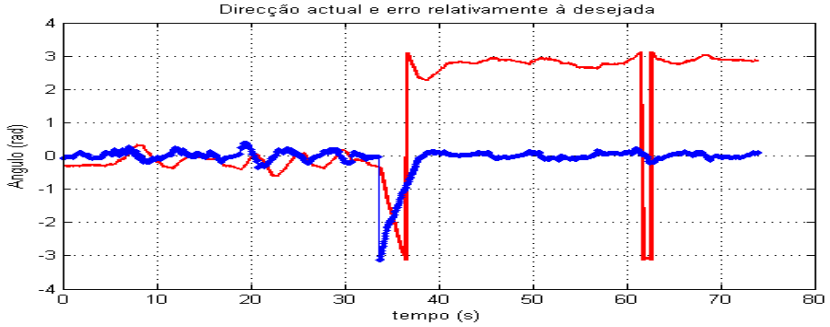


Figura 5.14 Erro de direcção e direcção de condução em percurso não ideal

No que concerne à distância entre a posição estimada do robot e o trajecto desejado, esta não ultrapassa 1,5metros e atinge apenas aproximadamente esse valor, quando o robot altera o sentido do seu percurso (Figura 5.15).

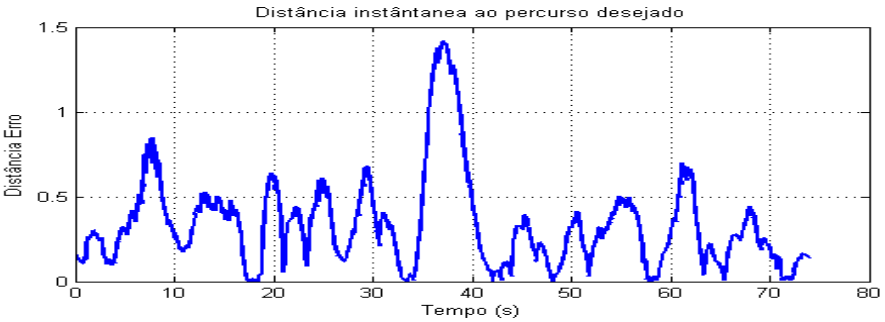


Figura 5.15 Distância em módulo entre o percurso desejado e as posições instantâneas

O último teste tem como particularidade o facto de ser efectuado com a potência/velocidade máxima do robot (aprox. 3.5m/s). Na Figura 5.16 está apresentado o trajecto percorrido.

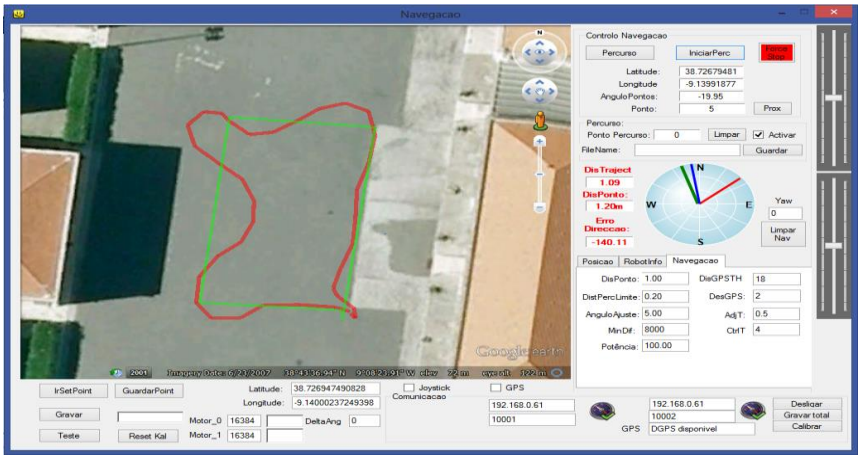


Figura 5.16 Percurso rectangular com velocidade máxima

O gráfico de posição da Figura 5.17 permite demonstrar que o robot não consegue actualizar atempadamente os seus motores para poder actualizar a direcção de deslocamento.

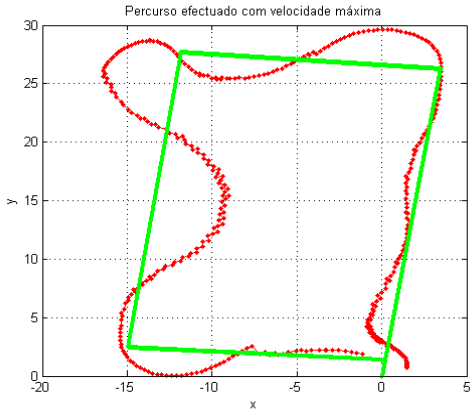


Figura 5.17 Percurso efectuado pelo robot com velocidade máxima

Do mesmo modo, entende-se que o gráfico relativo ao erro de direcção (Figura 5.18) também sofre um grande aumento, juntamente com as grandes variações de direcção de condução do robot.

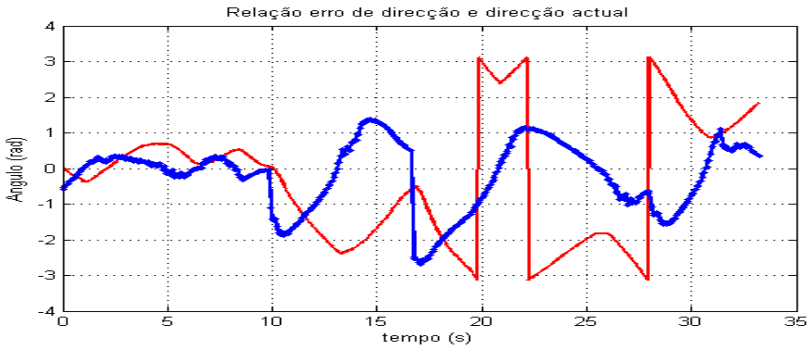


Figura 5.18 Erro de direcção e direcção actual em teste com velocidade máxima

A distância em módulo, do percurso realizado ao percurso desejado (Figura 5.19) apresenta neste caso valores superiores àqueles que foram obtidos nos testes efectuados anteriormente. Atingiram-se os maiores picos de erro junto aos locais onde existiam curvas no percurso desejado, chegando a atingir uma distancia máxima em relação ao trajecto desejado de 4 metros.

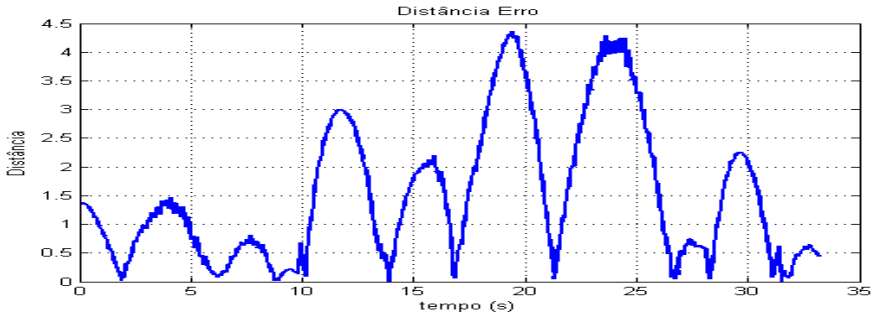


Figura 5.19 Módulo da distância entre trajecto efectuado e percurso desejado

5.1 Análise do desempenho

Analisar-se-á agora o desempenho do robot, ou mais concretamente, o algoritmo desenvolvido para a navegação autónoma, focando os objectivos cumpridos, as falhas detectadas e os melhoramentos possíveis.

O primeiro grupo de testes permitiu observar que o robot consegue em terreno regular, com velocidade moderada e com bom sinal GPS, seguir autonomamente um percurso pré-definido, sem grandes erros de percurso.

No primeiro teste (recta), concluiu-se que, na ausência de grandes alterações de direcção, o robot consegue dirigir-se ao percurso desejado e segui-lo sem dificuldades. No

segundo teste (rectângulo), o robot, na maioria dos casos, executa a sua missão de forma extremamente satisfatória, tendo no entanto, alguns problemas na alteração brusca de direcção. Tais factos significam que, ao nível de estimativa de posição, o Filtro de Kalman pode estar a limitar a variação brusca de valores de posição. Em relação ao controlo dos motores, existe um atraso entre os cálculos da posição estimada, dos erros dos ângulos e dos valores a enviar aos motores. No momento em que se pretende de novo atingir a rota certa, torna-se difícil calibrar motores de acordo com a direcção pretendida.

No último teste do primeiro grupo, pode-se observar pelo teste do círculo que a adaptação do Filtro de Kalman na estimativa da posição foi uma boa aposta, pois suaviza os valores de posição obtidos do sinal GPS, permitindo uma melhor aproximação da realidade.

Em suma, neste grupo de testes, os resultados obtidos foram bastante satisfatórios, visto comprovar que a aplicação desenvolvida consegue navegar autonomamente.

No segundo grupo de testes, o algoritmo teve um comportamento melhor que o esperado, uma vez que se conseguiu mesmo em condições desfavoráveis, seguir o trajecto desejado.

Com o primeiro teste do segundo grupo, pode-se concluir que a conjugação dos sensores no Filtro de Kalman teve os seus lucros permitindo que mesmo com fraco sinal GPS, o robot conseguisse seguir a trajectória desejada.

No segundo teste deste grupo, é possível concluir-se que o atraso do sistema autónomo tem influência na navegação. As actualizações ao percurso do robot ocorrem mais lentamente do que seria de esperar, ou seja, a posição utilizada para calcular os valores de controlo dos motores será diferente da posição do robot, quando este realmente sofrer a actualização.

6 Conclusões

Para alcançar os objectivos desta tese de mestrado, descritos no subcapítulo 1.2, estudaram-se os métodos de estimação de posição e de filtragem de ruído. Estes aplicaram-se aos sinais recebidos, a fim de obter um conjunto de dados mais precisos, relativos à localização do robot. Neste trabalho, foram ainda objecto de estudo os procedimentos necessários para sincronizar os dados dos vários sensores, de modo a conseguir um controlo adequado dos motores do robot, permitindo assim uma melhor navegação autónoma. Para a implementação do sistema de navegação autónomo pretendido, foi concretizada uma aplicação que, para além de incluir uma interface gráfica, permitiu englobar todo o sistema de comunicação, tratamento e filtragem de sinais, estimação de posição e controlo dos motores.

No capítulo 2, foram investigados o sistema GPS e os sensores IMU, com o intuito de apreender quais as suas benesses e os seus defeitos no contexto da estimativa da posição em tempo real. Entendeu-se que o sistema GPS é o principal recurso para a obtenção de dados, relativos à localização, e que, se utilizado em conjunto com os restantes sensores, permite obter uma melhoria na estimativa da posição.

Foi estudado o método de localização de *Markov*, reconhecendo-se que é um método simples e que permite obter uma boa estimativa estatística da posição. No entanto, concluiu-se que o método matemático criado por *Rudolf Kalman*, aplicado como filtragem de sinais ruidosos e como meio de integração dos valores de vários sensores, é uma melhor solução. Foi portanto escolhido para a implementação na aplicação desenvolvida, de maneira a obter uma estimativa da posição real do robot, através da filtragem e associação dos sinais provenientes do sistema GPS, dos Sensores IMU e dos *encoders*.

Com o estudo do robot, principalmente dos seus motores, concluiu-se que estes funcionam de modo diferencial e assimilou-se quais eram as suas respostas aos diversos impulsos enviados. Foi também analisado o modo de comunicação com o robot. Para navegar autonomamente, foi necessário estimar a posição actual e a orientação do robot em cada instante e saber controlar os seus motores para assim seguir o percurso desejado.

Com o desenvolvimento da aplicação de navegação, foi possível construir uma interface de comunicação com o robot, que também serviu de base de implementação dos algoritmos desenvolvidos. Esta aplicação é deste modo o resultado prático deste trabalho, uma vez que é o meio que permite implementar a navegação autónoma no robot.

A primeira abordagem, para estimar a localização actual do robot, não utilizou o sistema GPS, baseando-se apenas na utilização da aceleração/velocidade estimada pelo acelerómetro e da orientação do robot em cada instante de tempo. Esta abordagem foi estudada mas não desenvolvida, pois concluiu-se que, além de requerer o conhecimento da

posição inicial, acumula demasiados erros nas constantes actualizações de velocidade e posição.

Após se verificar que a velocidade calculada pelo acelerómetro tendia, com o passar do tempo, a divergir da velocidade real optou-se por utilizar os valores dos *encoders* para cálculo desta. Em suma o Filtro de Kalman teve como entradas a aceleração obtida pelos valores do acelerómetro, a velocidade obtida pelos dados dos *encoders* e a posição obtida pelo sistema GPS.

Uma melhor estimativa da posição do robot em cada instante foi então calculada no Filtro de Kalman de duas dimensões, sendo que, antes da implementação deste, foram filtrados e calibrados os dados dos sensores IMU. A calibração permitiu que logo à partida não existissem erros de leitura a incrementar ao ruído existente. O ruído por sua vez foi retido com um Filtro de Kalman adaptado de uma dimensão. Nesta fase do projecto, apurou-se as capacidades do filtro na prática, verificando-se que, a partir da adaptação deste, podemos obter resultados muito distintos.

Concluiu-se também que, para controlar o deslocamento do robot, seria necessário trabalhar os ângulos de direcção actual e de direcção desejada. Estes ângulos foram obtidos com apoio do giroscópio e da bússola magnética. Consoante o erro existente entre o ângulo de direcção actual e o ângulo desejado, foram calculados os valores adequados a enviar aos motores do robot.

Com o sistema de navegação integrado no nosso programa, foram efectuados os testes e afinados os vários blocos da aplicação. Relativamente aos testes, concluiu-se que o robot tem autonomia, sendo no entanto possível o seu melhoramento.

Focando de novo nos objectivos atingidos a partir dos testes efectuados, estes corresponderam às espectativas. Foi provado no Capitulo 5 Testes e Resultados que, em várias situações diferentes, o robot consegue manter a rota pretendida sem que, relativamente ao percurso desejado, exista um erro de distância elevado. Dos testes realizados, pode-se concluir que o atraso entre a obtenção dos dados dos sensores até ao controlo dos motores afecta o percurso efectuado, isto porque, como foi estudado no subcapítulo 4.3.3, a actualização dos motores não ocorre simultaneamente com a estimativa da posição. Este tempo de atraso fará com que a posição utilizada para calcular os valores de controlo dos motores seja diferente da posição do robot, quando este realmente sofrer a actualização.

Também os erros associados ao cálculo da velocidade pelo acelerómetro são significativos. Deste modo, para longos percursos, é benéfico parar o robot por breves momentos e reiniciar os valores de velocidade, permitindo assim eliminar os valores calculados erradamente.

Outro factor que pode provocar erros no deslocamento autónomo provém dos ângulos de orientação do robot. A informação do norte disponibilizada pela bússola magnética é afectada pela proximidade de objectos metálicos do robot. Os ângulos de direcção do robot que são calculados pelo giroscópio, baseando-se no norte magnético, estão assim sujeitos a valores errados.

É de interesse referir que este projecto, desde o seu início até ao momento actual, sofreu muitas alterações e passou por fases distintas. De início, o objectivo seria apenas filtrar dados provenientes dos sensores do robot e obter uma melhor localização deste. Com este intuito, foi inicialmente estudado o Filtro de Kalman e a localização de Markov. No entanto, ficaria a faltar um modo de aplicar na prática o estudo desenvolvido. A primeira abordagem passou pela criação de um programa de simulação de um joystick. Com este seria possível posteriormente, após saber a posição actual, controlar o robot. Contudo, tornou-se demasiado complexa esta abordagem, dado que seria necessário que o programa de origem do robot reconhecesse este programa, tal como reconhece um joystick real.

Após se ter estudado a API disponibilizada pelo fabricante e se ter verificado que eram disponibilizadas várias funções de comunicação com o robot, decidiu-se criar este programa de navegação autónoma. A partir deste momento, os objectivos finais foram alargados e muito mais matéria se adicionou ao estudo.

Por assim ser, devido à grande abrangência de matéria desta tese, foi difícil focar e estudar ao pormenor todos os temas e assuntos necessários para se obter uma melhor navegação autónoma. Como referido, mesmo o objectivo inicial desta dissertação, a filtragem de sinais dos vários sensores para uma melhor localização, poderia ser alvo de um estudo mais aprofundado, contudo para se obter resultados práticos foi necessário o desenvolvimento de todo o sistema de navegação e controlo, motivo pelo qual escasseou o tempo, tornando-se assim difícil focar de forma mais intrínseca o assunto principal.

Resumindo, o objectivo de criar uma aplicação capaz de controlar autonomamente o robot Jaguar 4x4 wheel foi cumprido com êxito, na medida em que, após os vários testes realizados, se observou que o robot seguiu todos os percursos previamente definidos. Espera-se ainda que o trabalho desenvolvido seja uma mais-valia para o Exército Português e para a área da navegação autónoma.

6.1 Perspectivas de trabalho futuro

A área da robótica móvel autónoma encontra-se em constante evolução, sendo actualmente alvo de inúmeras investigações por toda a comunidade científica. No caso específico do robot utilizado nesta dissertação, Jaguar 4x4 wheel, novas e melhores capacidades podem ser incluídas.

Caso se pretenda introduzir um sensor de distância/sensor laser, é possível desenvolver um sistema autónomo com capacidade de evitar a colisão com vários objectos, durante a navegação do robot.

No caso de se pensar em criar um sistema de videovigilância a ser implementado numa unidade militar, é importante o desenvolvimento de um posto de comando, que, podendo ser ou não móvel, teria de ter alcance para comunicação com o robot sobre toda a área de navegação.

Enumeram-se de seguida alguns dos projectos que poderão melhorar a aplicação desenvolvida nesta dissertação:

- Efectuar uma calibração precisa do mapa com as coordenadas geográficas, evitando observar o robot num local do mapa que não corresponde ao local correcto na realidade.
- Criar um modo de segurança automático para que o robot pare caso ocorram problemas durante o seu percurso.
- Possibilitar o controlo manual do robot, interrompendo a navegação autónoma e permitindo, após findado o controlo manual, a prossecução do percurso automaticamente.
- Facilitar o mecanismo de criação de percursos, possibilitando por exemplo mover trajectos ou incrementá-los alguns metros.
- Criar um sistema de alarme, que informe o utilizador sobre o estado de baterias ou sobre falhas na comunicação.

Em relação às melhorias no algoritmo, é necessário desenvolver um maior estudo específico nos vários módulos desenvolvidos, podendo-se obter resultados mais precisos, mesmo a maior velocidade.

Bibliografia

1] D. Shetty e L. Manzione, Unmanned Aerial Vehicles (UAV): Design Trends, Denver, Colorado, USA: ASME, 2011.

2] B. D. -. NASA, “NASA,” 28 Julho 2013. [Online]. Available: http://spaceref.com/nasa-hack-space/2013-sample-return-robot-challenge.html. [Acedido em 2013].

3] Academia da Força Aérea, “Pitvant,” 2011/2013. [Online]. Available: http://www.academiafa.edu.pt/index.php?bd0b6f49=011.005.004&lang=PT. [Acedido em Maio 2013].

4] G. Bekey, Autonomous robots : from biological inspiration to implementation and control, Massachusetts: The Mit Press, 2005.

5] R. Siegwart e I. R. Nourbakhsh, Introduction to Autonomous Mobile Robots, London: Massachusetts Institute of Technology, 2004.

6] J. N. Nilsson, A Mobile Automation: An application of artificial intelligence techniques, Menlo Park, California: Standford Research institute, 1969.

7] L. Earnest, “Standford Cart,” Dezembro 2012. [Online]. Available: http://www.stanford.edu/~learnest/cart.htm. [Acedido em Abril 2013].

8] “EUREKA Prometheus Project:all facts at a glance,” Maio 2010. [Online]. Available: http://vionto.com/show/info/en/EUREKA%20Prometheus%20Project/EUREKA. [Acedido em Abril 2013].

9] “Autonomous Robotics Programs,” General Dynamics - Robotic Systems, [Online]. Available: http://www.gdrs.com/robotics/programs/program.asp?UniqueID=27. [Acedido em Abril 2013].

10] N. Karlsson, L. Gonçalves e M. Munich, The vSLAM Algorithm for Navigationin Natural Environments, Evolution Robotics, Inc., 2012.

11] L. F. Gonçalves, Desenvolvimento de sistema de navegação autônoma por GNSS, São Paulo, Brasil, 2011.

iRobot, “IRobot,” Irobot Corporation, 2013. [Online]. Available:

12] http://www.irobot.com/en/us/learn/defense.aspx. [Acedido em Maio 2013].

13] DrRobot, Jaguar 4x4 Whell - User Guide, Ontario, Canadá: Dr Robot Inc, 2011.

14] InvenSense Inc., ITG-3200 Product Specification Revision 1.4, U.S.A.: InvenSense Inc., 2010.

15] Analog Devices Inc., Digital Accelerometer - Data Sheet (ADXL345), Analog Devices, 2009-2013.

16] Honeywell International Inc., 3-Axis Digital Compass IC HMC5843, Plymouth, Reino Unido: Honeywell, Fevereiro 2009.

17] "Trimble," Trimble Navigation Limited, 2013. [Online]. Available: http://www.trimble.com/gps_tutorial/whatgps.aspx. [Acedido em Maio 2013].

18] Garmin International, Inc., GPS 18x technical specifications, Olathe, USA: Garmin, 2008.

19] "Earth Measurement Consulting," 2013. [Online]. Available: http://earthmeasurement.com/GPS_accuracy.html. [Acedido em Julho 2013].

20] M. Ribeiro e P. Lima, "Markov Localization," Instituto superior técnico, Instituto de sistemas e Robotica, Lisboa, Maio 2002.

21] D. Fox, W. Burgard e S. Thrun, Markov Localization for Mobile Robots in Dynamic Environments, Journal of Artificial Intelligence Research 11, 1999, pp. 391-427.

22] A. Schneider e D. Westhoff, "Autonomous Navigation and Control of a Mobile Robot in a Cell Culture Laboratory," University of Bielefeld, Germany, 2002.

23] R. Kleinbauer, "Kalman Filter Implementation with Matlab," Universität Stuttgart, Helsinki, 2004.

24] W. Lages, "Filtro de kalman," Universidade Federal do Rio Grande do Sul, Escola de Engenharia, Rio Grande Do Sul, 2005.

25] B. Esme, "Bilgin´s Blog," Março 2009. [Online]. Available: http://bilgin.esme.org/BitsBytes/KalmanFilterforDummies.aspx. [Acedido em Fevereiro 2013].

G. Bishop e G. Welch, An Introduction to the Kalman Filter, North Carolina:

26] Department od Computing Science, University of North Carolina at Chapel Hill, 2006.

V. Makhijani, “Artificial Intelligence for Robotics,” Udacity, 25 Março 2013. [Online].
27] Available: https://www.udacity.com/wiki/cs373/unit_2#unit-2-kalman-filters. [Acedido em Abril 2013].

P. S. Maybeck, The Kalman Filter: An Introduction to Concepts, New York:
28] Springer-Verlag, 1990.

A. Andrews e M. Grewal, Kalman Filter: Theory an Pratice Using MATLAB, John
29] Wiley & Sons, 2001.

DrRobot - WiRobot (the heart of robot), WiRobot SDK Application Programming
30] Interface, DrRobot, Janeiro 2012.

C. Suliman, C. Cruceru e F. Moldoveanu, Mobile Robot Position Estimation Using
31] the Kalman Filter, Brasov, Romania: inter-eng Transilvania University of Brasov, 2009.

I. Ribeiro, P. Lima e R. Ventura, “Introduction to kalman filtering,” Instituto Superior
32] Técnico/ Institudo de sistemas e Robótica, Lisboa, October 2008.

M. Johnson, 4 Janeiro 2010. [Online]. Available:
33] http://www.azavea.com/blogs/atlas/2010/01/2010-a-pivotal-year-for-gps/. [Acedido em 10 Jullho 2013].

“Udacity,” Udacity, 15 Junho 2013. [Online]. Available:
34] https://www.udacity.com/wiki/cs373/unit_1. [Acedido em 10 Julho 2013].

V. d. C. Santos e R. Romero, “Filtro de Kalman Extendido,” USP, São Paulo.
35]

Y. Liu, Navigation and control of mobile robot using sensor fusion, United States of
36] America: CareFusion, INTECH, 2010.

ANEXOS

A) Funções disponibilizadas pela API

```
using System.Windows.Forms;
using System.Runtime.InteropServices;
namespace DrRobot.IMUGPSNavigation
{
    public partial class navegacao : Form
    {

        //for DCM algorithm
        /// <summary>
        ///this function is for DCM reset, yaw is reset angle(in radian)
        ///before use this DLL, this function must be called first
        ///this will reset all the variables in DCM algorithm
        ///yaw -- initial yaw value, pitch and roll always set as "0"
        /// </summary>
        /// <param name="yaw"></param>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
        static extern void ResetDCM(double yaw);


        /// <summary>
        ///this function is for Yaw PID correction
        /// </summary>
        /// <param name="kp">now we set as 0.25</param>
        /// <param name="ki"> now 0.001</param>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
        static extern void SetPIDYaw(double kp, double ki);


        /// <summary>
        ///  this function is for Pitch&Roll PID correction
        /// </summary>
        /// <param name="kp">0.15</param>
        /// <param name="ki">0.001</param>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
        static extern void SetPIDPitchRoll(double kp, double ki);


        /// <summary>
        /// this function is for IMU sensor setting, now flag fixed as 0
        /// gravity_scale,gyro_scale, sampleTime are from IMU setting
        /// </summary>
        /// You could set these values by your test
        /// <param name="gravity_scale">0.0043478  AD tick 16g sensitivity</param>
        /// <param name="gyro_scale">1.2*GrausRad gyro AD tick</param>
        /// <param name="sampleTime">0.020 50Hz</param>
        /// <param name="flag">1</param>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
        static extern void SetDCMParameter(double gravity_scale, double gyro_scale,
double sampleTime, int flag);
```

```
        /// <summary>
        ///this function is for GPS sensor data update
        /// </summary>
        /// <param name="cog">cog -- course over ground, radian, -pI ~ PI</param>
        /// <param name="vog">vog -- velocity over ground, unit: m/s</param>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
        static extern void UpdateGPS(double cog, double vog);


        /// <summary>
        ///this function is for IMU sensor data update and output estimate Yaw angle
(in radian)
        ///accel[0] --- X    accel[1]--Y    accel[2]-- Z
        ///gyro[0] ---- X    gyro[1] --- Y   gyro[2] -- Z
        ///pry[0]--- pitch   pry[1]---- roll pry[2] --- yaw, unit: radian
        ///please notice the IMU mounting position, X-- points to forward, Y points
to right side, Z points down
        /// </summary>
        /// <param name="accel"></param>
        /// <param name="gyro"></param>
        /// <param name="pry"></param>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
        static unsafe extern void EstimatePRY(double* accel, double* gyro, double*
pry);


        /// <summary>
        ///this function will use GPS latitude and longitude reading to update
estimated position
        /// </summary>
        /// <param name="latitude">unit: degree</param>
        /// <param name="longitude">unit : degree</param>
        /// <param name="estPos">pointer to current position estimation, estPos[0] -
- E, estPos[1] -- N, estPos[2] -- Orientation(Yaw)</param>
        /// <returns></returns>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
          static unsafe extern int GPSUpdatePosition(double latitude, double
longitude,double* estPos);


        /// <summary>
        /// these 5 functions are used for coordinate transform, this function must
be called first
        /// GPS (latitude, longitude, altitude) <---> ECEF (XYZ) <----> Navigation
ENU (E,N,U)
        /// we need first set reference latitude, longitude (ENU (0,0,0) point
        ///
        /// </summary>
        /// <param name="latitude"> unit: degree</param>
        /// <param name="longitude"> unit: degree</param>
        /// <param name="altitude">0</param>
        /// <param name="xyzr">reference point in ECEF(XYZ)</param>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
          static unsafe extern void SetRefXYZ(double latitude, double
longitude,double altitude,double* xyzr);


        /// <summary>
        ///this function is for LLH to XYZ
        /// </summary>
        /// <param name="lat">unit: degree</param>
```

```
        /// <param name="lng">unit: degree</param>
        /// <param name="h">0</param>
        /// <param name="XYZ">position in ECEF</param>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
          static unsafe extern void LLH2XYZ(double lat,double lng,double h, double*
XYZ);


        /// <summary>
        /// this function is for XYZ to ENU
        /// </summary>
        /// <param name="XYZr">reference point XYZ</param>
        /// <param name="XYZ">point XYZ(ECEF)</param>
        /// <param name="ENU">point in ENU</param>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
        static unsafe extern void XYZ2ENU(double* XYZr, double* XYZ, double* ENU);

        //
        //
        //
        // setENU[2] --U 0 forced as "0" in function
        /// <summary>
        /// this function is ENU 2 XYZ
        /// setENU[0] - E
        ///  setENU[1] -- N
        ///  setENU[2] --U 0 forced as "0"
        /// </summary>
        /// <param name="setENU"></param>
        /// <param name="calXYZ"></param>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
          static unsafe extern void ENU2XYZ(double* setENU, double* calXYZ);


        /// <summary>
        /// XYZ to LLH
        /// latitude, longitude unit: degree, altitude = 0
        /// </summary>
        /// <param name="XYZ"></param>
        /// <param name="calLLH"></param>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
          static unsafe extern void XYZ2LLH(double* XYZ, double* calLLH);


        /// <summary>
        /// this function will calculate the distance between two points
        /// (latitude1, longitude1) to (latitude0, longitude0)
        /// </summary>
        /// <param name="latitude1"></param>
        /// <param name="longitude1"></param>
        /// <param name="latitude0"></param>
        /// <param name="longitude0"></param>
        /// <returns></returns>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
          static extern double DistanceLL(double latitude1, double longitude1,
double latitude0, double longitude0);


        /// <summary>
        /// this function will calculate the heading direction from (latitude1,
longitude1) to (latitude2, longitude2)
```

```
        /// the return value is in radian, -PI ~ PI, 0 is point to north, PI/2 is
point to East
        /// </summary>
        /// <param name="latitude1"></param>
        /// <param name="longitude1"></param>
        /// <param name="latitude2"></param>
        /// <param name="longitude2"></param>
        /// <returns></returns>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
          static extern double BearingToLL(double latitude1, double longitude1,
double latitude2, double longitude2);

        /// <summary>
        /// this function will estimate driving direction, based on current
estimated position and preset path
        /// the preset path is defined by (startENU) ---> endENU
        /// the return value is robot driving direction
        /// </summary>
        /// <param name="estPos"></param>
        /// <param name="distanciaErro"></param>
        /// <param name="startDir"></param>
        /// <param name="heading"></param>
        /// <param name="startENU"></param>
        /// <param name="endENU"></param>
        /// <returns></returns>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
          static unsafe extern double EstimateDrvDir(double* estPos, double
distanciaErro,double startDir, double heading, double* startENU, double* endENU);



        /// <summary>
        /// this function will set minimum differential PWM power,
        /// this value will make sure robot could turn in setting minimum driving
angle
        /// usually for tank, need bigger value, set as 3500, for wheel truck, set
as 2000,
        /// if the ground surface need more power, could set bigger value
        /// </summary>
        /// <param name="setValue"></param>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
        static extern void SetMinDiffPWM(int setValue);


        //
        /// <summary>
        /// this function will return Distance between current XYZ and present
path(startXYZ--endXYZ)
        /// </summary>
        /// <param name="curXYZ"></param>
        /// <param name="startXYZ"></param>
        /// <param name="endXYZ"></param>
        /// <returns></returns>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
        static unsafe extern double Distance2Path(double* curXYZ, double* startXYZ,
double* endXYZ);


        /// <summary>
        /// this function is setting Encoder orientation estimate parameter
        /// orientation = encoder_ori + kp * ( IMU_Yaw - encoder_ori)
```

```
        /// </summary>
        /// <param name="kp"> now we set kp = 1 or 0.8, we trust IMU estimation more
than Encoder estimation because wheel slip</param>
        [DllImport("DrRobotIMUGPSDCM.dll", SetLastError = true)]
           static extern void SetEncoderEstimateYawParameter(double kp);


        public static double latitude { get; set; }
    }
}
```