FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Localização de Robôs Autónomos em Ambiente Industrial

Nuno Miguel da Silva Moreira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Prof. Doutor Paulo José Cerqueira Gomes da Costa

Co-orientador: Prof. Doutor José Alexandre de Carvalho Gonçalves

31 de Julho de 2014

© Nuno Moreira, 2014

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES 2013/2014

A Dissertação intitulada

"Localização de Robôs Autónomos em Ambiente Industrial"

foi aprovada em provas realizadas em 24-07-2014

o júri

Presidente Professor Doutor Aníbal Castilho Coimbra de Matos
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor José Luís Sousa de Magalhães Lima
Professor Adjunto do Departamento de Engenharia Eletrotécnica da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Bragança

Professor Doutor Paulo José Cerqueira Gomes da Costa
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Nuno Miguel da Silva Moreira

Faculdade de Engenharia da Universidade do Porto

Resumo

Este documento foca um algoritmo de localização para ambiente estruturado de um robô omnidirecional, fundindo a odometria com dados de um laser, por forma a obter um sistema de localização preciso com o mínimo de intervenção possível no ambiente em que o robô é inserido.

A localização em robótica móvel é um problema chave que permite aos robôs serem totalmente autónomos. Para que o robô saiba o que fazer no instante de tempo seguinte, necessita de saber a sua posição e orientação no mundo em que se insere, em cada instante de tempo. Para tal, o robô recorre a medidas relativas e absolutas provenientes dos sensores nele existentes, e assim deve ser capaz de determinar com a máxima precisão possível a sua localização. Para tal são necessários algoritmos que combinem essa informação, de forma a estimar a localização do robô.

Por forma a satisfazer a componente de localização relativa, é calculada a odometria através da informação proveniente dos *encoders* existentes nos motores e adotou-se o algoritmo Perfect Match como forma de localização absoluta. Para este fim foi necessário incorporar um *laser range finder* no robô, permitindo uma localização sem alteração do meio envolvente. A fusão sensorial das posições obtidas é conseguida com recurso a um Filtro de *Kalman* Estendido. Os resultados obtidos são satisfatórios, obteve-se um erro diminuto, adequado ao tamanho reduzido do campo da competição.

Desenvolveu-se um robô que se enquadra no desafio *Robot@Factory*, seguindo as regras da competição, tornando-a o desafio do trabalho desenvolvido.

Foi utilizado um ambiente de simulação, o *SimTwo* (simulador oficial da competição), por forma a facilitar o teste dos algoritmos desenvolvidos e reduzir a fadiga mecânica do sistema.

Abstract

This document focuses on a location algorithm to a structured environment of an omnidirectional robot, fusing odometry with a laser data in order to obtain a precise localization system with the least possible interference in the environment in which the robot is inserted.

In mobile robotics, location is a key problem that allows robots to be completely autonomous. For the robot to know what to do next, it needs to know its position and orientation in the world in which it operates, on the fly. For this end, the robot uses relative and absolute measures from existing sensors in it, and so should be able to determine as accurately as possible its location. To this end, algorithms which combine the information in order to estimate the location of the robot The location.

In order to satisfy the component of relative location, odometry is calculated based on the information from the existing encoders on the motors and adopted the *Perfect Match* algorithm as a form of absolute location. To this end it was necessary to incorporate a laser range finder in the robot, enabling a location without changing the surrounding environment. The sensor fusion of the obtained positions is achieved by using an Extended Kalman Filter. The results obtained are satisfactory, yielded a small error, appropriate to the small size of the field of competition.

A prototype robot that fits the challenge *Robot@Factory* was developed, following the rules of the competition, making it the challenge of the developed work.

A simulation environment was used, *SimTwo* (official competition simulator) in order to facilitate testing of the developed algorithms and reduce the mechanical stress of the system.

Agradecimentos

Ao Prof. Doutor Paulo Costa por toda a ajuda prestada ao longo da realização deste trabalho. Estou também grato ao Prof. Doutor José Alexandre Gonçalves pela disponibilidade em acompanhar e auxiliar a concretização da dissertação.

À equipa do laboratório de robótica, em especial ao Sr. Jorge Barbosa pela ajuda durante o período em que realizei a minha dissertação na FEUP.

A todos os meus amigos com quem cresci e aprendi ao longo do meu percurso pessoal e académico. Um especial agradecimento ao Hui, Peng, Tiago, Jorge, Diogo por estarem sempre presentes. Ao Daniel (ID) por toda a paciência demonstrada e ajuda prestada. Ao Nuno Carvalho pela amizade incondicional e ao Rui Dantas e Tony Sá pelo apoio e boa disposição sempre presentes.

À minha namorada Diana Lima por acreditar em mim, pela amizade e apoio em todos os momentos e pela paciência que demonstrou para comigo ao longo do nosso percurso.

Agradeço aos meus pais, irmãos e avós o apoio, formação e motivação dados ao longo da minha vida, que contribuíram de forma crucial para os meus êxitos pessoais e académicos. Ao José Carlos Fernandes pelo apoio e disponibilidade constantes.

Nuno Miguel da Silva Moreira

"It is our attitude at the beginning of a difficult undertaking which more than anything else, will determine the outcome."

William James

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

AGV	*Automated Guided Vehicle*
CAD	*Computer-Aided Design*
DC	*Direct current*
EKF	*Extended Kalman Filter*
IMU	*Inertial Measurement Unit*
LED	*Light Emitting Diode*
RAM	*Random-acess Memory*
RFID	*Radio Frequency Identification*
UDP	*User Datagram Protocol*

Capítulo 1

Introdução

Neste documento é apresentado todo o trabalho efetuado na unidade curricular Dissertação, projeto de final de curso no âmbito do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto.

Na primeira secção do presente documento será apresentado o Enquadramento e Motivação (1.1) para o desenvolvimento da presente dissertação, seguida dos Objetivos (1.2) que se pretendem atingir. Por fim, está presente a Estrutura do documento (1.3).

1.1 Enquadramento e Motivação

A indústria é um mundo de elevada competitividade e em permanente evolução, necessitando de uma constante procura de soluções competitivas e económicas para os demais sistemas que a constituem. Atualmente a logística das linhas de produção consiste em tapetes e condutas de transporte de configuração fixa, implicando custos e tempos de paragem elevados aquando a necessidade de nova reconfiguração.

De forma a colmatar estas restrições são adotados sistemas de transporte mais flexíveis, tais como empilhadoras e transportadoras. No entanto, o manuseamento destes veículos implica a contratação de mão de obra especializada, que nem sempre é eficiente. Distrações por parte do condutor e controlo desproporcional da velocidade do veículo são fatores que potenciam acidentes, podendo originar tempos de baixa de trabalhadores, bem como a danificação do bem transportado.

Começam assim a ser adotados os AGV. Estes veículos são bastante versáteis, permitindo uma fácil e rápida reconfiguração das linhas de produção de um ambiente industrial. No entanto a versatilidade e correto comportamento destes sistema depende bastante do sistema de localização e navegação nestes presente. Estes sistemas permitem a correta identificação do robô no seu mundo, bem como de possíveis obstáculos durante o seu trajeto, possibilitando um percurso eficiente e seguro entre dois pontos.

Esta dissertação enquadra-se no âmbito da competição *Robot@Factory* do *Robótica2014*.

1.2 Objetivos

A presente dissertação tem como objetivo o desenvolvimento de um sistema de localização de AGVs, com o mínimo de intervenção no ambiente. De forma a possibilitar a implementação deste sistema, e por forma a participar na competição *Robot@Factory*, é ainda necessário, em parceria com outros elementos presentes na equipa, projetar e construir um robô que cumpra os regulamentos internos da competição, servindo desta forma como protótipo de implementação desta dissertação.

1.3 Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais cinco capítulos.

No capítulo 2, é descrito o estado da arte e são apresentados trabalhos relacionados com este tema.

No capítulo 3, é feita um enquadramento geral da indústria no tema da dissertação, bem como uma apresentação da competição *Robot@Factory* e a associação da mesma ao tema em que este trabalho se insere.

No capítulo 4 são descritas as plataformas de implementação e validação da dissertação. Inicia-se com a descrição e justificação dos sensores utilizados, seguindo-se do projeto em CAD e concretização do mesmo resultante numa plataforma física de implementação. Por fim é referido o simulador onde se efetuaram todos os testes de simulação do protótipo.

No capítulo 5 são expostos os algoritmos e metodologias utilizadas constituintes no decorrer de todo o projeto. É feita inicialmente uma caracterização da dinâmica do robô, necessária à localização relativa. Posteriormente é especificada a localização absoluta, concluindo o capítulo com o algoritmo de fusão sensorial Filtro de *Kalman* Estendido.

No capítulo 6 são apresentados e debatidos todos os resultados práticos.

Por fim, no capítulo 7 são apresentadas as conclusões finais e sugestões de trabalho futuro.

Capítulo 2

Revisão Bibliográfica

Neste capítulo serão apresentadas diversas metodologias e topologias para a construção, configuração e controlo de um robô apto a participar na competição *Robot@Factory*, no festival *Robótica2014*. Inicialmente procedeu-se à análise e estudo das topologias de tração e deslocamento mais usuais (2.1), seguido dos possíveis métodos de localização (2.2). Posteriormente fez-se o estudo dos métodos mais comuns de perceção do mundo (2.3) e por fim apresenta-se um estudo das metodologias mais atuais para efetuar a fusão sensorial por forma a corrigir a posição do robô (5.3.2).

2.1 Locomoção

Neste capítulo são especificados diversos tipos de locomoção utilizados nos robôs na área de transporte industrial, mais especificamente em competições enquadradas nesta área.

2.1.1 *Ackerman Steering*

Esta geometria tem como característica o facto de todas as rodas terem os seus eixos dispostos como um raio de um círculo, tendo um ponto central em comum. Como as rodas traseiras são fixas, este ponto central tem de ser definido a partir de uma linha estendida que atravessa o eixo traseiro. De forma a intersetar os eixos das rodas dianteiras, esta linha requer que a roda dianteira interior, aquando de uma mudança de direção, tenha um ângulo de viragem superior ao da roda dianteira exterior. A Figura 2.1 descreve esta topologia.

A grande vantagem deste sistema é o facto das forças laterais serem passivamente neutralizadas através das restrições de deslizamento, reduzindo o binário requerido pelo motor [6].

Facilmente se percebe que, tal como se verifica nos veículos automóveis, esta configuração implica um raio de curvatura mínima sempre que haja a necessidade de uma mudança de direção na trajetória do veículo, dificultando as manobras do mesmo.

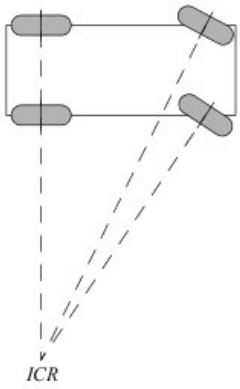


Figura 2.1: Estrutura de um sistema baseado em *Ackerman Steering*

2.1.2 Tração Diferencial

Robôs com tração diferencial são usados extensivamente na área da robótica, devido ao seu movimento ser de fácil programação e controlo. O movimento destes é baseado em duas rodas acionadas separadamente, colocadas em ambos os lados do corpo do robô, cujos veios passam pelo mesmo eixo [7]. Estes sistemas podem, assim, alterar a direção da sua trajetória variando a velocidade de rotação de cada uma das suas rodas e, por conseguinte, não necessita de um movimento de direção adicional.

Devido à sua estrutura, possível de ser vista na Figura 2.2, esta configuração não permite deslocamentos no sentido do eixo de rotação das rodas.

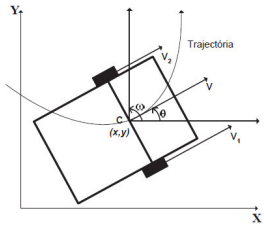


Figura 2.2: Estrutura de um sistema de tração diferencial

Tem a mais-valia de poder rodar sobre si mesmo, não apresentando a desvantagem da configuração anterior. No entanto, necessita de efetuar manobras aquando a mudança de direção, constituindo assim uma solução não ótima.

2.1.3 Tração Omnidirecional

Por forma a colmatar as restrições de um robô de tração diferencial, cada vez mais são implementadas trações omnidirecionais. Esta configuração tem como principal característica o descolamento em todas as direções, sendo maioritariamente implementadas em competições robóticas, onde o desempenho dos robôs é um fator crítico e decisivo para o seu sucesso [8].

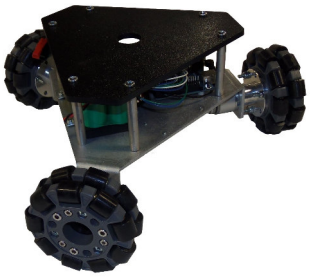


Figura 2.3: Robô omnidirecional de três rodas

As configurações tradicionais neste tipo de tração passam pelo uso de três ou quatro rodas. Embora os sistemas baseados em quatro rodas apresentem melhores características relativamente à aderência, aceleração e velocidades máximas [8], a adoção de um sistema de três rodas são de mais fácil controlo e projeto mecânico. Tal característica verifica-se uma vez que para a definição de um plano, serem suficientes apenas três pontos, sendo necessária a adoção de uma suspensão aquando a implementação de quatro rodas. Para a competição em que o objeto desta dissertação se insere, esta topologia verifica-se ser bastante viável, não sendo necessário a ponderação de um sistema de quatro rodas. Um exemplo da configuração utilizada pode ser visto na Figura 2.3.

As rodas necessárias a este tipo de deslocamento têm características especiais que as distinguem de uma roda convencional (Figura 2.4). No sentido do eixo de rotação da mesma, esta roda apresenta pouco atrito, permitindo desta forma o deslizamento segundo este sentido.

Figura 2.4: Roda omnidirecional

2.2 Localização

A localização de robôs é atualmente um tema extensamente estudado e explorado, existindo no mercado uma grande variedade de soluções. A escolha da solução mais indicada depende de uma grande diversidade de fatores, tais como as condições do local, a flexibilidade requerida e os custos associados à implementação e manutenção do sistema.

A diferença entre os dois tipos de medições está relacionada com o referencial onde estas se efetuam. Em medições absolutas, a medida é feita em relação à origem do mundo onde o sistema robótico se encontra. Por outro lado, medidas relativas são feitas em referência a um ponto, neste caso, o referencial do robô. Estas são também caracterizadas por estarem associadas à dinâmica do robô.

2.2.1 Medidas Absolutas

2.2.1.1 Sistema filoguiado

Sistemas filoguiados baseiam-se, como representado na Figura 2.5, no seguimento, por parte do robô, de um fio enterrado no chão [9]. Este fio é percorrido por uma corrente elétrica sinusoidal, criando desta forma um campo magnético. O campo criado é detetado por uma antena estrategicamente colocada no robô móvel, permitindo ao robô uma correção contínua da sua trajetória, por forma a que este não se perca. No entanto estes sistemas estão limitados ao uso a velocidades controladas, de forma a que o sistema robótico nunca deixe de sentir o efeito do campo magnético [10] [11].

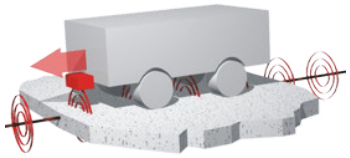


Figura 2.5: Sistema filoguiado [1]

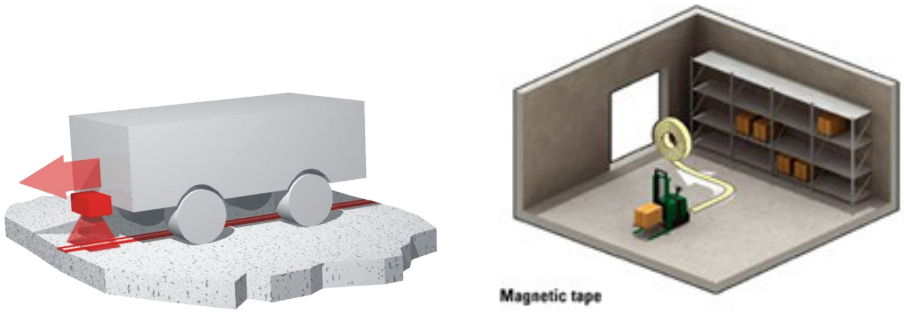
2.2.1.2 Sistema de faixas

Nos sistemas baseados em faixas, as trajetórias são definidas por fita magnética ou linhas coloridas pintadas no chão [11], como representado na Figura 2.6a e Figura 2.6b. A deteção da linha é feita por meio de um sensor, que deteta o campo magnético da fita ou a mudança de gradiente. Este sistema é bastante semelhante ao filoguiado, no entanto possibilita uma reconfiguração dos percursos bastante mais rápida, acarretando custos também mais reduzidos.

No entanto, em ambientes muito movimentados, este sistema revela desvantagens ao nível da durabilidade das marcas implementadas, uma vez que as fitas danificam-se mais facilmente e as linhas pintadas estão expostas a maior sujidade, potencialmente inibindo o seu seguimento por parte do robô [10, 9, 11, 2].

2.2.1.3 Sistema baseado em marcadores

Uma solução ao problema de definição de trajetórias dinâmicas é a implementação de sistemas baseados em marcadores embutidos ou sobre o chão, representado na Figura 2.7. Estes marcadores



(a) Seguimento de linha [1] (b) Seguimento de fita magnética [2]

Figura 2.6: Sistema de faixas

são geralmente magnéticos, podendo também serem marcas especificas no chão [10, 2]. Como estes marcadores se encontram espaçados entre eles, geralmente a uma distância mínima de 8 metros [12], é necessário complementar a sua localização com métodos de localização relativa, geralmente odometria e giroscópios [12, 10], permitindo ao robô ter uma boa noção do seu deslocamento e variações de direção entre marcadores.

Nesta área é ainda proposta a localização baseada em sistemas RFID [13, 14]. Este sistema baseia-se na interação entre um leitor presente no robô e *tags* dispostas no ambiente onde o sistema robótico se irá deslocar.

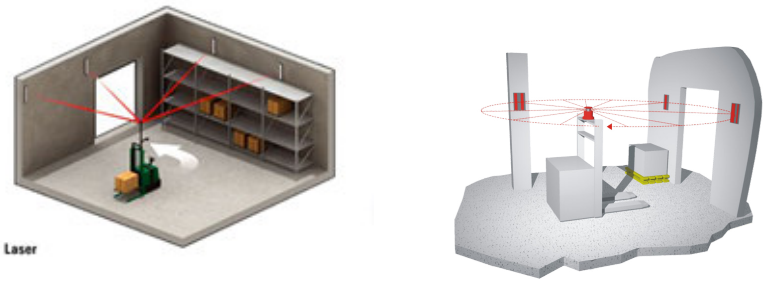
Comparativamente com os métodos de navegação já apresentados, este método permite uma reconfiguração do meio rápida e simples [2]. Em contrapartida, o equipamento necessário é mais caro do que nos sistemas anteriores e a sua precisão é de inferior qualidade [11]. Nos sistemas RFID, segundo [14], a precisão varia consoante o número de *tags* existentes, bem como a qualidade e modos de funcionamento dos mesmo. Verifica-se também um atraso nas leituras, aumentando significativamente o erro.

Figura 2.7: Sistema baseado em marcadores [2]

2.2.1.4 Sistema de trilateração e triangulação

Dentro das trajetórias flexíveis, outro método bastante adotado na indústria é navegação baseada em triangulação e/ou trilateração. Este método consiste na deteção da localização do robô

através de postes ou faróis refletores, colocados em postes ou paredes, e no uso de um *laser* rotativo, que permite a deteção destes refletores (ver Figura 2.8a e Figura 2.8b) [9, 11].



(a) Exemplo de sistema de triângulação [2]

(b) Exemplo de sistema de triângulação [1]

Figura 2.8: Sistema baseado em trilateração e triangulação

De forma a efetuar a sua localização, é necessário que o laser consiga detetar pelo menos três refletores. Sendo a posição de cada refletor constante e um dado adquirido, o laser permite obter a distância do robô a cada um dos refletores, bem como a sua orientação relativamente aos mesmos, permitindo uma localização eficaz e precisa do sistema robótico [11, 9].

Comparativamente os métodos até agora apresentados, este é o método mais eficaz aquando a localização do robô, bem como o mais flexível aquando a reconfiguração do ambiente industrial onde é inserido. No entanto é uma solução bastante cara, e que requer um conhecimento prévio do meio circundante, por forma a colocar estrategicamente os refletores, garantindo assim que, em todos os instantes, o robô deteta um mínimo de três [2, 11].

2.2.1.5 Sistema baseado em visão artificial

Sistemas robóticos baseados em visão artificial tem vindo a despertar grande interesse na comunidade cientifica, levando ao aparecimento de diversas metodologias para a localização destes sistemas.

Biswas et. al. [15] apresenta uma solução de localização em ambientes interiores baseada em câmeras de profundidade. A nuvem de pontos, de dimensão $\mathbb{R}^3$, capturada é primeiramente filtrada, por metodos de *threshold*, por forma a reduzir o peso computacional requerido para o processamento da informação e, posteriormente, esta é transposta para planos $2D$, onde é efetuado um *matching* com um mapa conhecido pelo robô.

Lee et. al. [16] sugere um sistema de localização baseado em marcadores de referência, recorrendo a uma câmara *web* para detetar estes marcadores. Nos marcadores encontra-se uma letra maiúscula ou um triângulo, indicando a direção do mesmo. Verifica-se assim ser um sistema de baixo custo de implementação. Quando o marcador, de tamanho conhecido, é visualizado pelo robô, a relação posicional entre o robô e o marcador pode ser calculada. Recorrendo a transformadas de distância é possível reconhecer os caracteres presentes nos marcadores. Verificou-se

que com quatro triângulos direcionais e dez letras foi possível um nível de reconhecimentos dos mesmo na ordem dos 98.87%, tornando este um sistema de localização viável para ambientes interiores.

Cucchiara et al. [17] sugere a utilização de técnicas de estereoscopia. Primeiramente são pintadas linhas paralelas ou perpendiculares aos eixos coordenados, possibilitando a correção da orientação do robô. De seguida, a câmara deteta as imagens e assim torna-se possível calcular a distância às mesmas. Desta forma consegue-se corrigir a posição do veiculo robótico ao longo do seu trajeto.

Uma opção bastante viável, principalmente para robôs de reduzida dimensão é apresentada em [18]. Segundo esta metodologia, um laser lê de forma constante a informação do mundo envolvente ao robô, e posteriormente essa informação é comparada com um mapa existente na sua memória, por forma a aumentar a informação disponível a cada momento. Aquando a impossibilidade de haver esta comparação entre mapas, recorre-se à otimização por enxame de partículas (*Particle Swarm Optimization* - PSO), mantendo uma boa previsão da localização do robô quando este se encontra perdido. Esta metodologia é recomendada quando o número de sensores é limitado e a sua precisão é menor.

Lauer et al. [19] sugere uma solução, o *Perfect Match*, baseada na minimização do erro existente na correspondência entre um mapa existente na memória do robô com as medições feitas por uma câmara. A minimização do erro é feita recorrendo ao algoritmo *resilient backpropagation*. Esta solução é bastante implementada em sistemas de localização em robôs a competir em futebol robótico, dada a sua elevada precisão e reduzido tempo de processamento.

2.2.1.6 Metodologias mais adotadas

Em [3], *Schulze et al.* apresentam um gráfico estatístico (Figura 2.9) onde discriminam as tecnologias mais adotadas por fabricantes de AGVs europeus para efetuar a localização dos seus robôs. É de notar que a metodologia mais adotada é a triangulação por laser 2.2.1.4, seguido de sistemas filoguiados 2.2.1.1 e de seguimento de faixas 2.2.1.2.

Estes factos justificam-se dado ao seu reduzido custo de implementação, comparativamente a outros sistemas à sua eficácia.

2.2.2 Medidas Relativas

2.2.2.1 Odometria

A odometria é um dos métodos mais utilizados para estimar a posição de um robô. Esta proporciona uma boa precisão em curto prazo, é barata de implementar, e permite taxas de amostragem muito altas. Uma vez que a odometria consiste na integração incremental do movimento das rodas ao longo do tempo, o erro associado a esta medição é desta forma acumulado, originando grandes erros na estimação da posição do robô. Esta está também limitada ao uso em superfícies planas [20] e o seu cálculo é dificultado pelo formato da roda omnidirecional.

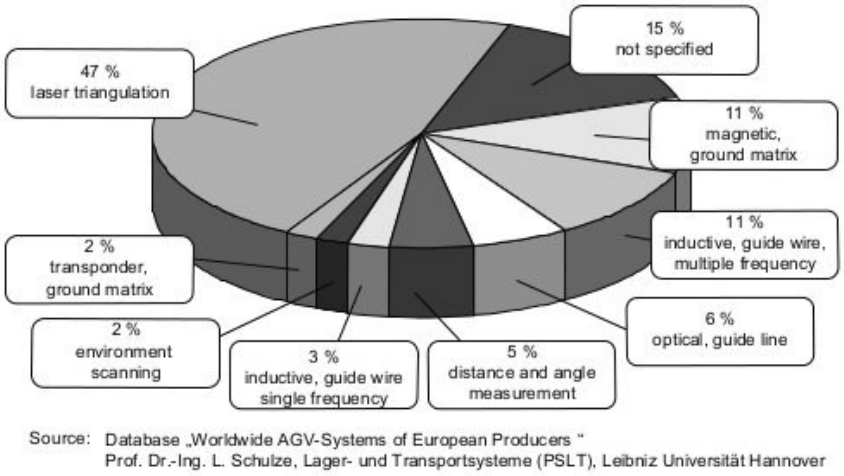


Figura 2.9: Tecnologias mais adotadas em 2006 [3].

Apesar dos seus inconvenientes, considera-se a odometria uma importante parte do sistema de navegação, tendo esta um papel importante aquando a fusão sensorial com medidas de posicionamento absolutas, tornam a estimação da posição mais viável.

2.2.2.2 Sensores Inerciais

Outra forma de ter conhecimento da posição de um robô, recorrendo a medidas relativas, é com recursos a IMUs. Um IMU é um dispositivo eletrónico que mede a velocidade, orientação e forças que atuam sobre um corpo, sendo este o componente principal de sistemas de navegação baseados em forças inerciais. Os principais componentes de um IMU são giroscópios e acelerómetros.

Zmuda et. al. [20] salienta os problemas associados a este tipo de medições. O fator mais problemático verifica-se ser as derrapagens a que o robô se encontra sujeito, sendo o erro por estas induzido acentuado aquando longos e contínuos períodos de descolamento. Verificou também que, a ocorrência de movimentos bruscos, pode causar o sensor a levar alguns segundos a voltar aos seus valores baixos.

Uma grande desvantagem deste tipo de medidas, comparativamente à odometria, em ambientes interiores prende-se com o facto de os IMU basearem as suas medições no campo magnético terrestre. Desta forma, os materiais ferromagnéticos presentes na estrutura dos edifícios e na construção do robô (motores, estrutura,...) influencia grandemente as medições por este obtidas.

2.3 Perceção do Mundo

Este capítulo, baseado em [6], expõe diversas metodologias de processamento da imagem adquirida por alguns dos sensores absolutos, mais especificamente lasers e câmaras. Primeiramente são abordados em singular, descrevendo os princípios base por de trás do seu funcionamento, e numa última secção é feita a análise comparativa dos diversos métodos.

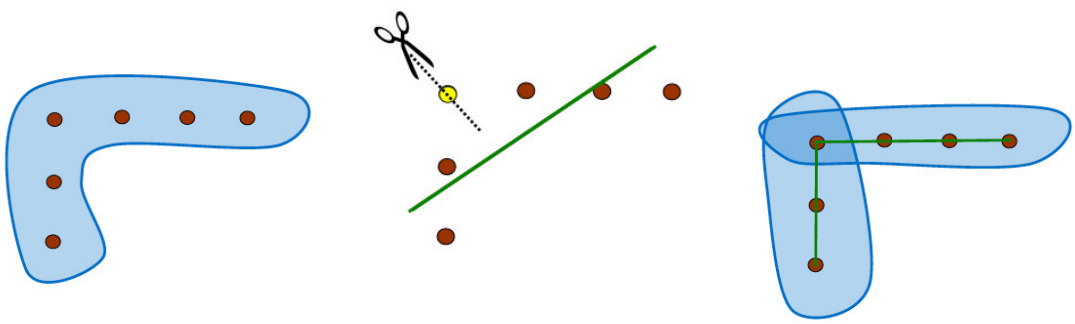
2.3.1 Split-and-Merge

Split-and-Merge é um algoritmo bastante utilizado na área da visão computacional. Consiste num algoritmo recursivo de fusão e cisão, que permite determinar as retas que melhor representam um conjunto de pontos. Como o nome indica, este algoritmo é dividido em duas partes: separação (*split*) e junção (*merge*).

A separação de um conjunto de pontos em vários é iniciada com a determinação da reta que melhor os aproxima. Comparando a distância desses pontos à reta, e comparando essa distância com um *threshold* previamente estipulado, identificam-se os pontos mais distantes. A divisão em subconjuntos é feita pela direita e pela esquerda desses pontos mais distantes.

Quando já estiverem todos os subconjuntos definidos, procede-se à união das retas. Se dois segmentos consecutivos estão suficientemente perto/colineares, obtém-se a linha comum e encontra-se o ponto mais distante. Se a distância for menor que um *threshold* estipulado, funde-se ambos os segmentos.

A Figura 2.10 representa o principais passos que caracterizam esta metodologia.



(a) Conjunto de pontos inicial a aplicar algoritmo

(b) Determinação de reta e determinação de ponto mais distante

(c) Separação de conjuntos

Figura 2.10: Representação do algoritmo Split-and-Merge [4]

2.3.1.1 Iterative End-Point-Fit

O *Iterative End-Point-Fit* é um algoritmo muito parecido com o *Split-and-Merge*. Nesta metodologia simplesmente se une os pontos finais do conjunto, formando assim uma reta. Após a determinação do ponto mais distante a essa reta, unem-se ambos os pontos que formam a reta ao ponto mais distante a esta, formando assim duas retas. Este processo iterativo decorre para todas as novas retas, enquanto houver pontos a unir.

A Figura 2.11 representa um exemplo de aplicação deste algoritmo.

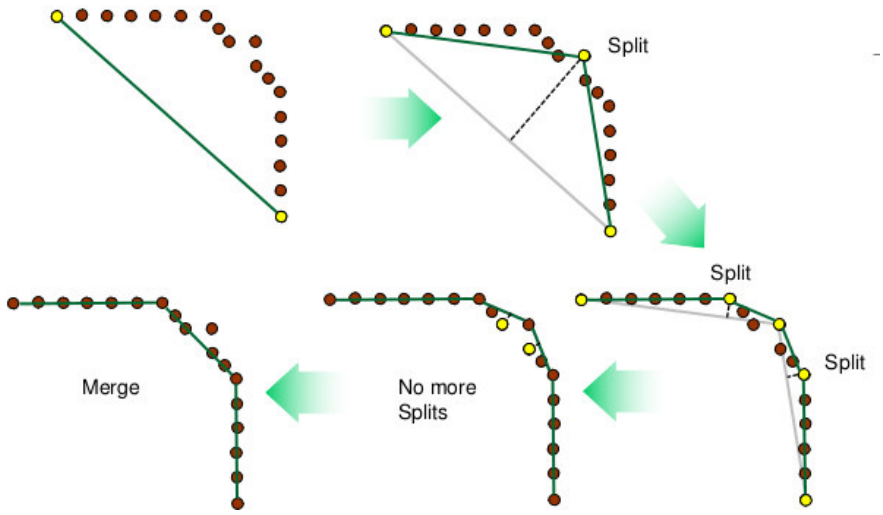


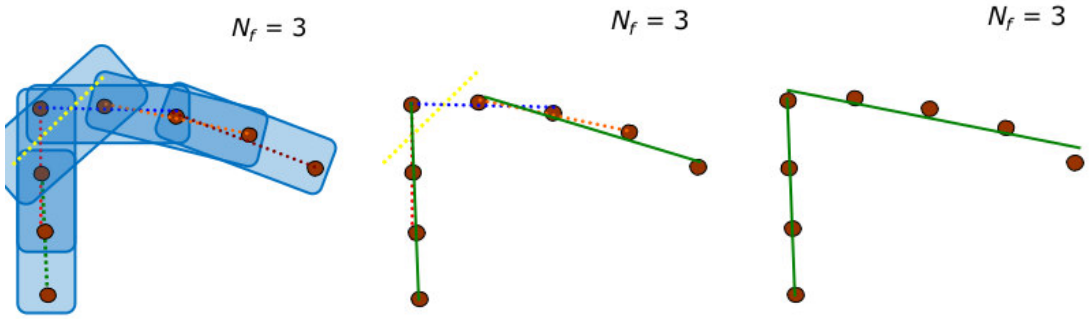
Figura 2.11: Representação do algoritmo *Iterative End-Point-Fit* [4]

2.3.2 Line-Regression

Line-Regression é um outro algoritmo de segmentação de imagem. Com uma abordagem relativamente simples, este algoritmo consiste na definição de uma janela de tamanho N_f, que aproxima cada N_f pontos consecutivos por uma reta.

Cria-se um vetor de valores, onde cada elemento é a soma das distâncias de Mahalanobis entre três janelas consecutivas. Definindo um *threshold*, para cada elemento do vetor menor que esse valor de *threshold*, cria-se uma nova linha (Figura 2.12). Para cada nova linha criada é necessário atualizar os parâmetros das linhas existentes.

Verifica-se que para esta metodologia, quanto menor as distâncias percorridas em cada iteração, maior será a fidelidade da imagem final.



(a) Enquadramento da janela nos pontos

(b) Iteração intermédia das distâncias de Mahalanobis

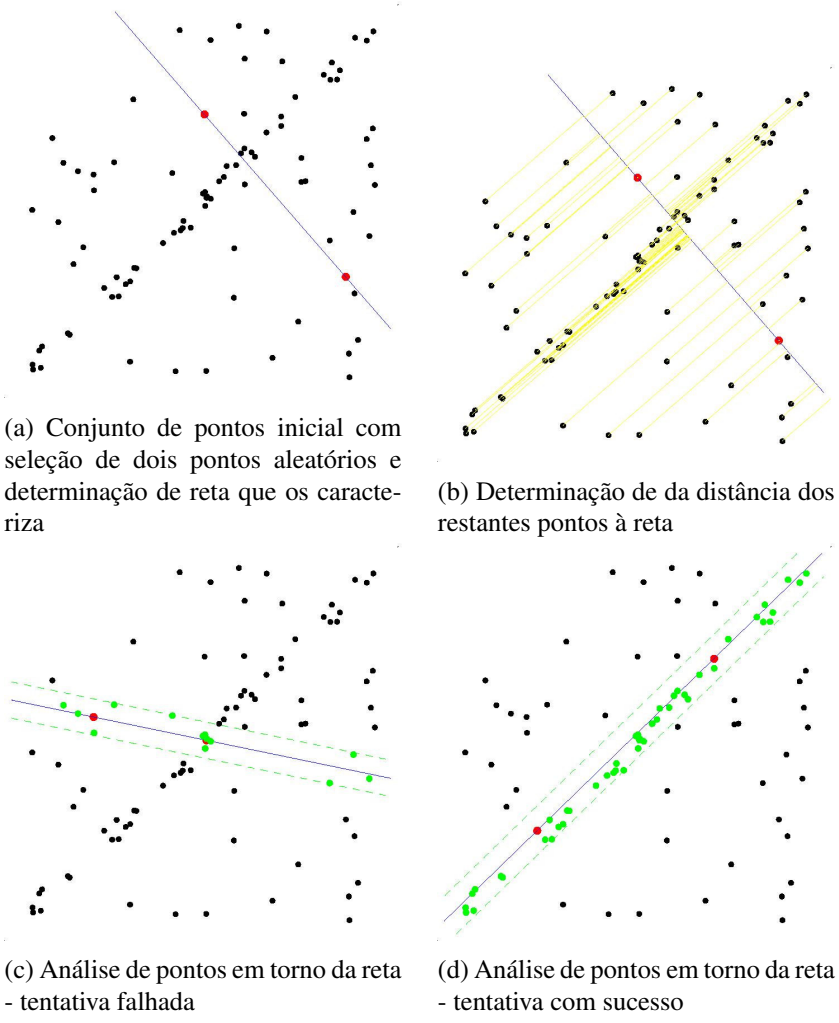
(c) Resultado final das várias iterações

Figura 2.12: Representação do algoritmo *Line-Regression* [4]

2.3.3 RANSAC

RANSAC, abreviatura de *random sample consensus* (consenso de amostra aleatória), é um algoritmo de segmentação iterativo e não determinístico. Este método é especialmente robusto na presença de discrepâncias entre medidas lidas e a informação já presente no robô.

Dado um conjunto de pontos P e um *threshold* t, são selecionados de forma aleatória dois pontos do conjunto P (Figura 2.13a). É traçada uma linha que atravessa estes pontos e é determinada a distância de todo o restante conjunto de pontos a esta linha (Figura 2.13b). Se dentro desse limite t se encontrarem mais do que n pontos, então é feita uma regressão linear desses pontos e definida a reta. Caso contrário são definidos dois novos pontos aleatórios e o processo repete-se (Figura 2.13c e 2.13d).



(a) Conjunto de pontos inicial com seleção de dois pontos aleatórios e determinação de reta que os caracteriza

(b) Determinação de da distância dos restantes pontos à reta

(c) Análise de pontos em torno da reta - tentativa falhada

(d) Análise de pontos em torno da reta - tentativa com sucesso

Figura 2.13: Representação do algoritmo RANSAC [4]

2.3.4 Transformada de Hough

A Transformada de *Hough* é uma técnica matemática que realiza a deteção de formas geométricas em imagens digitais. No âmbito desta dissertação, o interesse do uso desta metodologia encontra-se na deteção de retas e formas geométricas formadas por retas.

Todas as retas são descritas pela equação [2.1].

$$y = a * x + b \tag{2.1}$$

Estes parâmetros, a e b, podem ser representados como um só ponto num espaço de parâmetros gerados por estes dois parâmetros. No entanto, quando se representam retas que tendem para a posição vertical, os valores de a e b tendem para infinito, tornando este espaço de parâmetros infinitamente grande [5]. Opta-se assim por descrever a reta, não em função de um declive e de uma interseção, mas por meio de uma distância e de um ângulo.

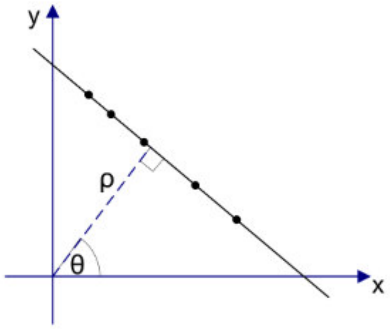


Figura 2.14: Representação da reta de acordo com a sua geometria [5]

Sendo θ o ângulo que a reta faz com a sua normal (eixo dos xx) e ρ a distância entre a origem do referencial (como demonstrado na Figura 2.14), a reta é agora definida pela equação [2.2].

$$xcos(\theta) + ysin(\theta) = \rho \tag{2.2}$$

Ao contrário do que acontecia com os parâmetros a e b, $\theta \in [0, \pi]$ e $\rho \in [-D, D]$, onde D é a diagonal da imagem. No entanto é possível continuar a representar estes valores num plano parametrizado $\theta - \rho$, também denominado por espaço de *Hough* [5, 21].

Considerando um *pixel* numa imagem, com posição (x, y), uma infinidade de linhas podem pode passar por ele. Por meio da equação [2.2], este *pixel* é representado no espaço de Hough como uma sinusoide, única e característica deste *pixel*. Efetuando o mesmo procedimento para os restantes *pixels* constituintes da imagem, verifica-se que todos os *pixels* constituintes de uma linha se vão cruzar num ponto do espaço de Hough. Este ponto representa a linha no espaço da imagem.

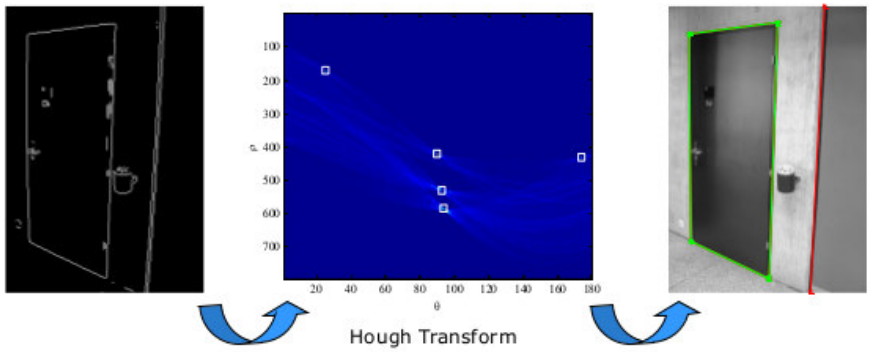


Figura 2.15: Representação das transformações segundo a transformada de Hough [6]

O resultado da transformada de *Hough* é guardado numa matriz denominada de "acumulador de *arrays*", com dimensão dos valores θ pelos valores de ρ [6].

2.3.5 Comparação de Algoritmos

As características que melhor definem os algoritmos acima apresentados, encontram-se explicitas na tabela 2.1.

Tabela 2.1: Comparação de algoritmos para extração de linhas

	Complexidade	Velocidade[Hz]	Positivos Falsos	Precisão
Split-and-Merge	$N \cdot \log N$	1500	10%	+++
Line-Regression	$N \cdot N_f$	400	10%	+++
RANSAC	$S \cdot N \cdot N_{tentativas}$	30	30%	++++
Transformada de Hough	$S \cdot N \cdot N_C + S \cdot N_R \cdot N_C$	10	30%	++++

Onde:

- N: Número de pontos varridos
- S: Número de segmentos da linha extraída
- N_f: Tamanho da janela para o algoritmo *Line-Regression* (2.3.2)
- $N_{tentativas}$: Número de tentativas para RANSAC (2.3.3)
- N_C, N_R: Número de colunas e linhas para o acumulador de *arrays* na transformada de Hough (2.3.4)

Mais esclarecimentos relativamente à comparação entre algoritmos podem ser consultados em [6]. Os autores evidenciam, no entanto, que em sistemas de localização e mapeamento baseados em lasers $2D$, o algoritmo *Split-and-Merge* é a melhor solução. Esta escolha baseia-se no seu reduzido número de positivos falsos e elevada eficiência.

2.4 Fusão Sensorial

A fusão sensorial refere-se ao uso da informação proveniente de múltiplos sensores para auxiliar na localização do sistema robótico. Para isto são aplicadas técnicas na criação de um sistema que forneça uma base de conhecimento que seja mais precisa, mais robusta e maior que as bases individuais de cada sensor.

2.4.1 Markov

A abordagem de Markov para o problema da localização assenta-se na base de, para todos os instantes, existir uma densidade de probabilidade para cada ponto possível de o robô se encontrar. Para esta metodologia é necessário dividir o mapa em células, contendo cada célula a probabilidade de o robô se encontrar na mesma. Esta densidade probabilística advém da informação proveniente dos sensores incorporados no veículo sendo necessário, para todos os instantes, um novo cálculo da densidade de probabilidades para cada célula do mapa [22, 23]. O algoritmo completo é descrito na Figura 2.16.

for each location *x* **do**
  generate the initial belief $Bel(x_0)$ // initialize
end for
forever do
  if a new motion data u_k is received **do**
    for each location *x* **do** // apply motion signal
      $\overline{Bel}(x_k) = \sum_{[1]} p(x_k \mid x_{k-1}, u_{k-1}) Bel(x_{k-1})$
    end for
  if a new perceptual data s_k is received **do**
    $\eta \leftarrow 0$
    for each location *x* **do** // apply sensory signal
      $A \leftarrow 0$
      for each grid *c* **do**
        $A = A + p(s \mid x_k, c)\, p(c)$
      end for
      $Bel^{+}(x_k) = A\overline{Bel}(x_k)$
      $\eta = \eta + Bel^{+}(x_k)$
    end for
    for each location *x* **do** // normalization
      $Bel(x_k) = \eta^{-1} Bel^{+}(x_k)$
    end for
  end if
end forever

Figura 2.16: Algoritmo Localização baseada em Markov

A localização baseada em Markov permite que o robô não tenha um ponto fixo para começar o seu movimento, permitindo também recuperar quando este é transportado por outrem para uma nova posição [24].

Em contrapartida, e devido ao facto de a cada iteração ser necessário calcular a densidade de probabilidade em cada célula, este algoritmo é extremamente pesado computacionalmente. Revela-se também ser necessário achar um bom compromisso entre o tamanho das células e o peso computacional suportado. Células pequenas são um fator determinante na boa localização da posição do robô, no entanto requer um aumento do nível de processamento.

2.4.2 Filtro de Partículas

Filtro de Partículas, ou Monte Carlo Sequencial, são algoritmos de estimação de densidade, que estimam a função densidade de probabilidade do estado posterior do sistema. Para tal, usa um conjunto de partículas, com pesos atribuídos de forma aleatória. Esses pesos representam a probabilidade dessa partícula ser amostrada a partir da função densidade de probabilidade.

Como exposto por Gouveia [22], as partículas são espalhadas pelo mapa ao longo das possíveis posições do robô e, com o decorrer das iterações, procura reduzir a disparidade entre pesos. Assim as partículas com pesos insignificantes voltam a ser projetadas no mapa, mas desta vez em torno das com maior peso. Por isso esta etapa é denominada por "reamostragem".

Verifica-se, assim, uma convergência da maioria das partículas para o local de maior probabilidade para a posição do robô. Um elevado número de partículas assegura de forma quase certa uma convergência para o valor mais preciso. As equações que descrevem a metodologia podem ser consultadas em [25].

O filtro de partículas pode ser aplicado a sistemas não lineares. Tem também como vantagem o facto de o estado inicial e distribuição do ruído poderem ser não gaussianas. No entanto, é um filtro computacionalmente pesado, não funcionando bem quando aplicado a sistemas de grande dimensão.

2.4.3 Filtro de Kalman Estendido

O EKF é outra ferramenta para resolver problemas de estimação não lineares. Esta metodologia baseia-se no Filtro de Kalman, filtro que estima o estado de um sistema linear para um dado instante, baseando-se no estado do sistema no instante anterior e nas medidas mais atuais. O EKF é um filtro de Kalman que lineariza as dinâmicas não lineares do filtro em torno das estimativas de estado anteriores [25].

As equações do EKF dividem-se em dois grupos: previsão e correção. As equações de previsão projetam no tempo o estado do sistema e a covariância do erro da estimação, prevendo a sua posição no instante seguinte. As equações de correção, juntamente com as medições obtidas, fazem a realimentação do sistema, obtendo-se o estado à *posteriori*. Estas equações encontram-se especificadas na Figura 2.17.

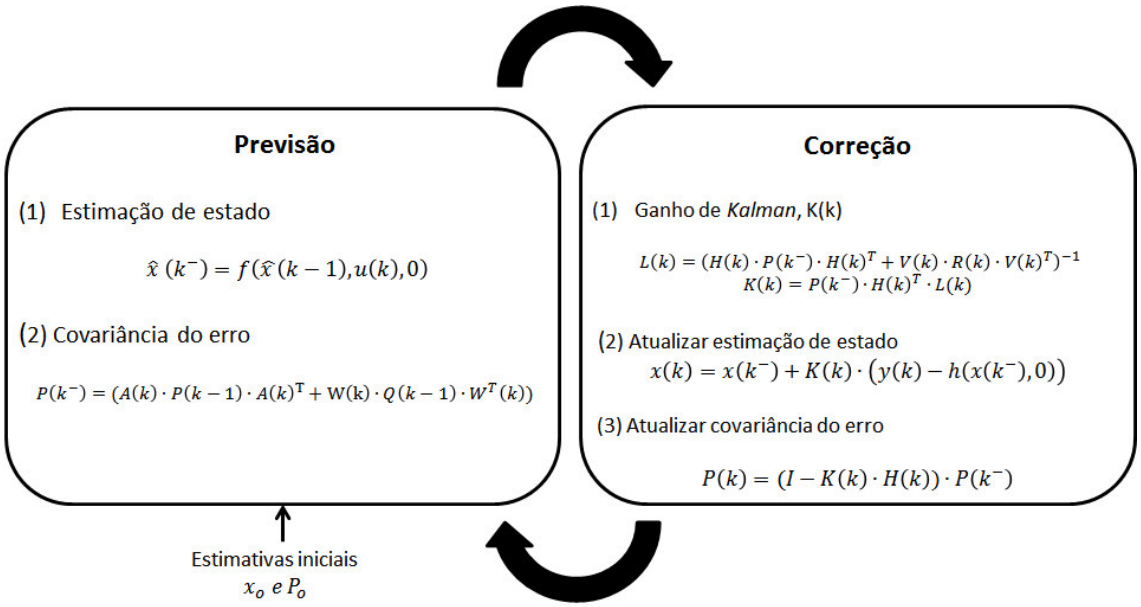


Figura 2.17: Algoritmo do Filtro de Kalman Estendido

A grande desvantagem deste método relativamente aos anteriormente apresentados prende-se com o facto de ser dependente de ruídos com uma distribuição Gaussiana. No entanto, segundo [26], o ruído de um sensor pode ser caracterizado por uma distribuição normal. Este método restringe também o robô a começar o seu movimento sempre de um ponto do mapa conhecido.

Após uma comparação entre o Filtro de Partículas e o EKF [25] concluiu-se que, aquando a existência de ruído Gaussiano, o EKF verificou-se ser mais eficaz, com uma exigência computacional mais reduzida.

Capítulo 3

Descrição do Problema

No presente capítulo é feito um estudo mais aprofundado da importância de veículos autoguiados em ambiente industrial (3.1), bem como também é feito o enquadramento da competição *Robot@Factory* neste âmbito e as vantagens que esta oferece ao nível de investigação nesta temática (3.2).

3.1 AGVs na indústria

Nos dias que correm verifica-se um constante aumento da competitividade na indústria, requerendo desta forma que estas tenham que se otimizar, reduzindo tempos de baixa das suas operações. Por forma a colmatar esta problemática, tem havido um crescente interesse em automatizar a indústria, desde os processos de fabrico ao transporte de mercadoria e materiais.

Na área de transporte de mercadorias, a solução adotada com o intuito de substituir empilhadoras e outros meios de transporte guiados por operários, é a adoção de AGVs (Figura 3.1). Uma grande vantagem destes sistemas em relação a outros equipamentos de movimentação é que os AGVs podem manter um nível de serviço muito constante e com grande cadência, já que não fazem paragens nem desvios que não estejam programados, excetuando para carregar a bateria, ao contrário do que se verifica aquando a utilização de uma empilhadora por parte de um operário. Analogamente, o risco de acidente de trabalho no transporte de materiais é reduzido, dado o sistema robótico não estar sujeito a distrações e tempos de resposta potencialmente mais lentos, inerentes a um humano.

Os AGVs têm como função transportar paletes e/ou materiais de/para posições programadas em áreas de produção e armazéns e normalmente o seu uso está associado a uma elevada capacidade de carga. Aquando a sua utilização em linhas de montagem, os AGVs podem, para além de transportar material entre postos de trabalho, como podem também servir de posto de trabalho, desde que devidamente preparados para tal efeito [3].

Estes são orientados por um único operário, que estipula as tarefas e correspondentes prioridades a serem realizadas por estes sistemas.

Figura 3.1: Exemplo de um AGV em ambiente industrial

3.2 Robot@Factory

O *Robot@Factory* é uma competição que procura recriar um problema inspirado nos desafios que um robô autónomo terá de enfrentar durante a sua utilização numa fábrica. Esta fábrica é constituída por um armazém de aprovisionamento, um armazém de produto final e oito máquinas de processamento, como se encontra representado na Figura 3.2.

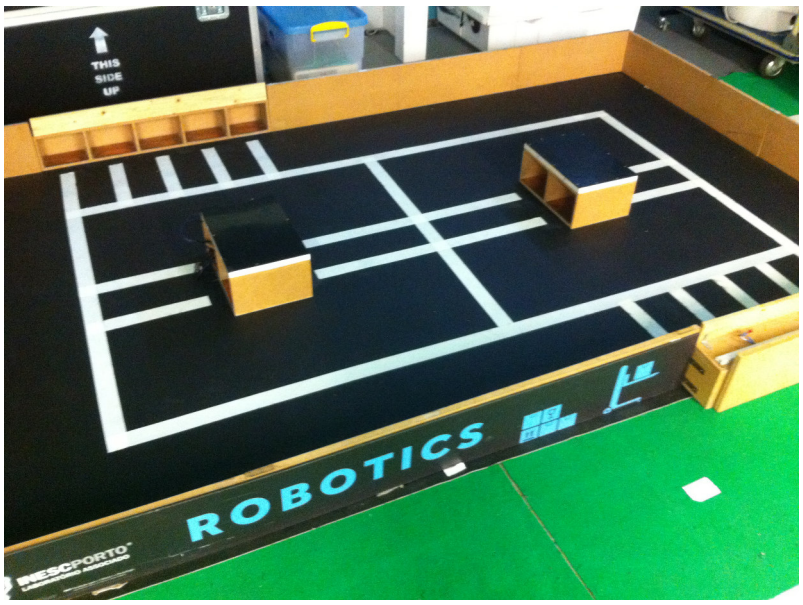


Figura 3.2: Campo da competição *Robot@Factory*

A tarefa dos robôs projetados para participar consiste em transportar caixas (Figura 3.3), diferenciadas por LED's de cores diferentes, entre armazéns e máquinas. Para tal é necessário que estes possuam capacidades de deteção de obstáculos, planeamento de trajetórias e um preciso sistema de localização e posicionamento.

Figura 3.3: Caixa a transportar da competição *Robot@Factory*

A competição é constituída por 3 mangas, com um grau de dificuldade acrescido comparativamente à anterior. Na primeira manga, a de menor grau de dificuldade, apenas é necessário transportar as caixas presentes no armazém de entrada para o armazém de saída. Para a segunda manga estão presentes no armazém de entrada dois tipos de caixa diferente, sendo que um desses tipos tem que ser maquinado por uma das máquinas antes de ser guardado no armazém de produto final. Para finalizar, na última manga encontram-se 3 tipos de peça no armazém de aprovisionamento, sendo que o terceiro tipo de peça necessita de ser maquinado em ambos os tipos de máquina existentes. Ganha a competição a equipa que conseguir transportar o maior número de caixas, no menor tempo possível.

O *Robot@Factory* permite a aprendizagem e investigação na área da robótica móvel, participando na competição robôs com simples sistemas de locomoção, localização e planeamento de trajetórias, até robôs altamente tecnológicos, com soluções inovadoras não usadas na indústria, mas eficazes no âmbito em que se inserem.

3.3 Metodologias adotadas

Depois de estudadas várias soluções para abordar o problema associado a esta dissertação em 2, e tendo em conta os objetivos da presente dissertação, optou-se pelo recurso à odometria como forma de localização relativa e a implementação do algoritmo *Perfect Match* como solução para a

localização absoluta. Por fim, a informação adquirida por estas duas vias é fundida recorrendo ao Filtro de *Kalman* Estendido.

A odometria foi um dos métodos adotados dado o seu bom comportamento, a curto prazo, em superfícies planas, fácil implementação e não afetação por parte de eventuais vigas metálicas inerentes a qualquer infraestrutura, tal como referido em 2.2.2.2 na utilização de IMUs. Para colmatar as limitações da odometria, o *Perfect Match*, utilizando dados do laser, foi a escolha feita como forma de localização absoluta dada a sua elevada precisão. Apesar de ser mais pesado computacionalmente do que outras soluções apresentadas, é possível limitar o número de iterações de correção que este efetua e por conseguinte reduzir o tempo de processamento. Esta metodologia permite também uma livre circulação por todo o campo da competição, não limitando a componente de planeamento de trajetórias. Por fim, o recurso ao Filtro de *Kalman* Estendido justifica-se pelo facto de ser computacionalmente leve, e a restrição de ser necessário conhecer a sua posição inicial não ser problemática, uma vez que esta é sempre conhecida aquando o início das provas.

Capítulo 4

Plataforma Implementada

O conteúdo da presente dissertação tem como objetivo ser implementado num robô apto a competir no *Robot@Factory* (3.2), no festival nacional de robótica. Para tal foi também necessário projetar um robô competitivo.

No presente capítulo é apresentada a plataforma de implementação e validação desta dissertação. É feita uma descrição do sistema físico de testes (4.1) e posteriormente uma descrição do sistema de simulação utilizado (4.2).

4.1 Robô Projetado

Tendo em consideração as possibilidades apresentadas na revisão bibliográfica, e as capacidades que este necessita para ser competitivo, optou-se por adotar um robô com tração omnidirecional de três rodas, recorrendo à odometria como forma de obter a localização relativa. Adotou-se também um *Laser Range Finder* por forma a obter a localização absoluta do sistema robótico. Estas características permitem uma precisa localização do robô, e um sistema de locomoção competitivo, permitindo ao robô movimentos de translação e rotação simultâneos.

Todo o processo de projeto e montagem do sistema foi levado a cabo por toda a equipa a competir.

4.1.1 Especificações técnicas dos sensores

4.1.1.1 Encoders dos motores

Com o objetivo de saber a velocidade de cada uma das rodas, foram utilizados *encoders* incrementais. Os *encoders* utilizados (Figura 4.1) são parte integrante dos motores escolhidos (*37Dx52L* da *Pololu*), facilitando assim o seu uso.

Com 360 pulsos por volta, o *encoder* tem também como característica o facto de ser de duas fases, permitindo diferenciar o sentido de rotação do motor. Sendo este de duas fases, possui dois sensores *Hall*, cada um tendo como sinal de saída uma onda quadrada. Esta característica permite

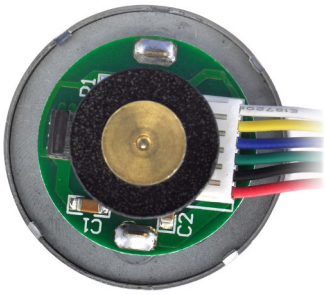


Figura 4.1: Encoder incorporado nos motores DC.

um melhor controlo da velocidade de rotação de cada motor, uma vez que são contabilizados o dobro dos pulsos.

4.1.1.2 *Laser Range Finder*

Por forma a efetuar as medições absolutas necessárias para uma boa localização do robô, efetuou-se uma escolha entre dois *rangefinders* distintos: *URG-04LX* da *HOKUYO* e o *Lidar* incorporado no aspirador robótico *Neato XV-11*.

O *Lidar* da *HOKUYO* [27] é bastante utilizado no âmbito da robótica móvel. Tem como principais características uma resolução angular de 0.36^o, um ângulo de leitura de 240^o, frequência de aquisição de 10Hz e efetua leituras na gama $0.06 - 4.1$ metros, revelando-se ser uma opção viável para sistemas de localização. Por sua vez, o sistema da *Neato* [28, 29, 30] possui uma resolução angular menos elevada (aproximadamente 1^o), frequência de aquisição de 5Hz e um ângulo de leitura de 360^o. Este possui também um alcance de $0.06 - 5$ metros.

Figura 4.2: *Lidar URG-04LX* da *HOKUYO*

Tendo em conta o contexto em que o robô está inserido, a diferença de resoluções angulares e alcances não são muito relevantes, uma vez que se trata de um campo de reduzidas dimensões e não existe a possibilidade de aparecimento de obstáculos novos.

A decisão sobre qual sistema de localização utilizar baseou-se em outro fator bastante importante em qualquer sistema: o custo da solução. O sistema da *HOKUYO* custa aproximadamente

850€, enquanto que o sistema da *Neato* tem um custo aproximado de 80€. Avaliando assim ambas as possibilidades, optou-se pelo uso do *LIDAR* proveniente do aspirador robótico *Neato XV-11*.

Figura 4.3: *Lidar* do *Neato XV-11*

É necessário avaliar o impacto que este *rangefinder* tem no sistema robótico projetado. Dada as pequenas dimensões do robô, os parâmetros críticos a serem avaliados são o peso e dimensões. Verificou-se que o peso deste sistema de localização tem pouco impacto no robô e que as suas dimensões não comprometem a eficiência do sistema.

O maior desafio revelou-se ser a comunicação e alimentação via porta *USB*. Uma vez que o laser funciona a $3.3V$ e o motor necessita de $3V$, e uma vez que as portas *USB* de um computador fornecem $5V$, foi necessário recorrer a um *Arduino DUE*. A escolha por este microcontrolador baseou-se no facto de as suas saídas analógicas serem de $3.3V$, valor de tensão ideal para alimentar o laser. Desta forma, as medições efetuadas são transmitidas pelo *Arduino* para o computador.

Estando o laser a funcionar e a comunicar, é necessário estudar a qualidade das medições efetuadas pelo *Lidar*. Para tal colocou-se um objeto perpendicular em frente do laser e efetuaram-se 128 medições, para cada uma das 30 posições conhecidas entre os 250 e os 3250 mm. Assumiu-se como leitura feita a média dos 128 valores lidos para cada distância.

De forma a garantir que a luz ambiente e reflexões da mesma nos materiais não é um fator relevante para a qualidade das medições, realizou-se um teste de calibração do *Lidar* no seu ambiente normal de utilização. Como é possível verificar na tabela 4.1, bem como Figura 4.4, os resultados ao teste em ambiente luminoso (círculos laranja) não são coincidentes com os valores da distância correspondente (quadrados azuis).

Deste modo, foi necessário passar os valores medidos por uma função, com o objetivo de os aproximar da distância a que correspondem. Esta função foi obtida com recurso à aplicação *solver* da folha de cálculo do *software LibreOffice*, tendo esta como objetivo minimizar o somatório do erro dos valores medidos em relação às distâncias. A função de calibração que se pretendeu ter foi do formato de 4.1.

Tabela 4.1: Distância, valores medidos do *LIDAR* e valores calibrados

Distância (m)	**Valor Medido (m)**	**Valor Calibrado (m)**
0.250	0.282999	0.250010
0.350	0.382476	0.351377
0.450	0.479000	0.449237
0.550	0.579023	0.550130
0.650	0.678539	0.649989
0.750	0.779218	0.750486
0.850	0.878906	0.849468
0.950	0.980695	0.949998
1.050	1.081671	1.049187
1.150	1.184968	1.150101
1.250	1.286789	1.249024
1.350	1.391148	1.349848
1.450	1.494875	1.449494
1.550	1.602273	1.552072
1.650	1.704929	1.649553
1.750	1.814367	1.752864
1.850	1.919257	1.851292
1.950	2.025070	1.949999
2.050	2.136335	2.053159
2.150	2.239820	2.148520
2.250	2.350593	2.249975
2.350	2.460109	2.349644
2.450	2.571937	2.450767
2.550	2.688679	2.555632
2.650	2.801328	2.656141
2.750	2.903312	2.746560
2.850	3.024289	2.853109
2.950	3.135109	2.950038
3.050	3.232687	3.034850
3.250	3.450914	3.222716

$$y = ax^2 + bx + c \tag{4.1}$$

Nesta equação, y representa os valores calibrados e x os valores de distância lidos, sendo que:

$$\begin{cases} a = -0.0262756318238165 \\ b = 1.03649062495971 \\ c = -0.0412124119851335 \end{cases} \tag{4.2}$$

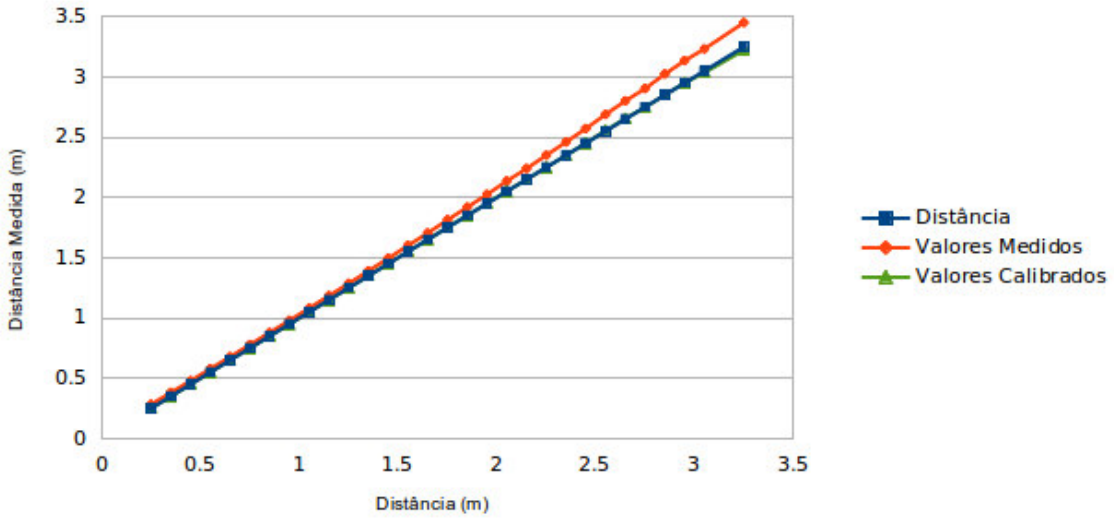


Figura 4.4: Distâncias medidas e função de correção

Aplicando a função 4.1, com os parâmetros apresentados em 4.2, às medições obtidas, são calculadas uma melhor aproximação das mesmas aos valores ideais. Esses valores são apresentados na tabela 4.1 e a sua representação encontra-se na Figura 4.4 como triângulos verdes. É notória a aproximação que os valores lidos têm relativamente à distâncias reais.

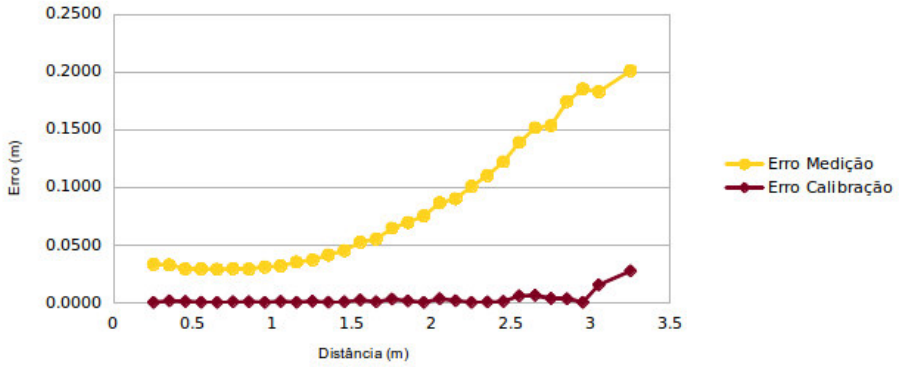
Com o objetivo de validar o modelo, fez-se um estudo comparativo das medições lidas com as posteriormente calibradas. O resultado pode ser verificado na Figura 4.5. Em 4.5a é possível constatar o erro das leituras numa escala linear, permitindo visualizar com clareza a qualidade dos valores obtidos antes e depois da calibração. Posteriormente, como visualizado na Figura 4.5b, optou-se pelo estudo destes valores em escala logarítmica, com o intuito de analisar de forma mais clara a variação dos valores do erro após a calibração. É agora mais notória, desprezando os valores mais baixos de erro, a evolução que estes valores têm relativamente aos medidos.

Com o interesse de analisar o erro das leituras calibradas relativamente à distância real, produziu-se o gráfico presente na Figura 4.6.

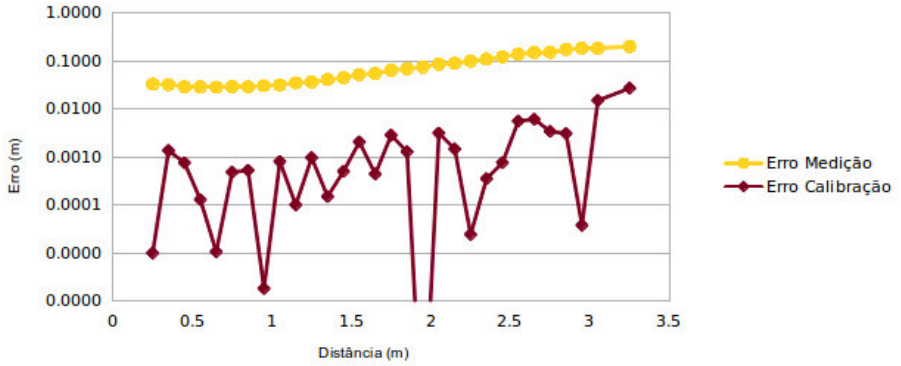
Verifica-se, em todos os gráficos apresentados, um aumento significativo do erro, mesmo após a calibração, para distâncias acima dos 3 metros. No entanto, dado que o campo onde o robô está inserido não tem mais de 3 metros de comprimento, este erro não é considerado relevante.

Outra característica importante a analisar é o desvio padrão das leituras. Na Figura 4.7 é possível ver o comportamento deste em relação às distâncias lidas (pontos cor de laranja). Por forma a estudar o contraste entre o uso da equação de calibração e os valores inalterados, passaram-se as 128 leituras por distância no filtro de calibração e representou-se o desvio padrão dessa população como os quadrados verdes, presentes na mesma Figura. É notória a qualidade das medições, principalmente para distâncias elevadas, reforçando os resultados obtidos nos pontos anteriores.

Uma análise do desvio padrão depois de calibradas as leituras, numa escala logarítmica (Figura



(a) Comparação de erros em escala linear



(b) Comparação de erros em escala logarítmica

Figura 4.5: Comparação dos erros de medição e dos erros da calibração

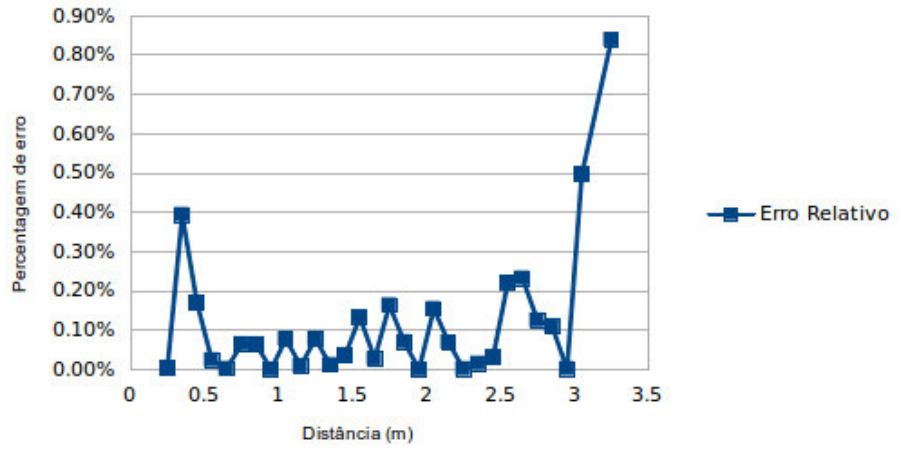


Figura 4.6: Erro relativo da calibração em relação à distância

4.8), permitiu descrever a equação que o define numa equação de primeira ordem, apresentada em 4.3.

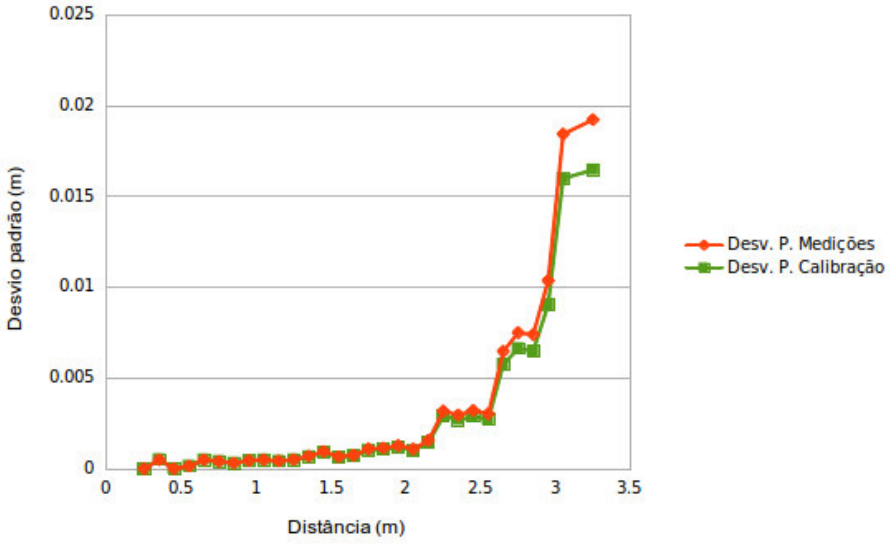


Figura 4.7: Desvio padrão das leituras feitas e das leituras calibradas

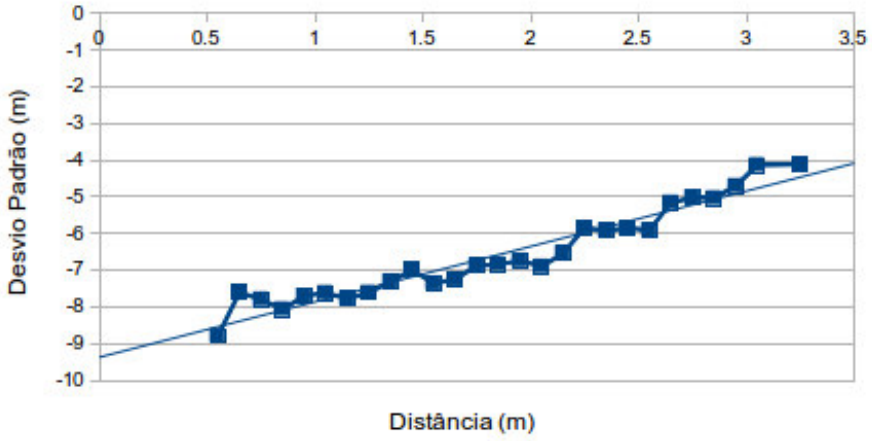


Figura 4.8: Desvio padrão das leituras calibradas em escala logarítmica

$$\sigma(x) = 1.510731986x - 9.3715156568 \tag{4.3}$$

Transpondo a equação 4.3 para um formato linear, obtém-se a equação 4.4, que descreve a evolução do desvio padrão com as distâncias lidas.

$$\sigma(x) = e^{1.510731986x - 9.3715156568} \tag{4.4}$$

4.1.1.3 Câmara de Vídeo

Por forma a ser possível identificar a cor do LED das caixas, foi necessário adotar uma câmara de vídeo. Foi escolhida a *PlayStation Eye*, Figura 4.9a, uma câmara digital para a *PlayStation 3*. A escolha deste sistema prendeu-se ao facto da *PlayStation Eye* ser uma câmara com conexão USB barata e popular, sendo desta forma de acesso fácil no mercado e com diversos *softwares* implementados para recurso em diferentes âmbitos. Por outro lado, esta é também bastante modular, tornando-se compacta e fácil de acoplar ao robô, quando desprovida do invólucro, tal como visível na Figura 4.9b.

(a) Câmara de vídeo *PlayStation Eye*

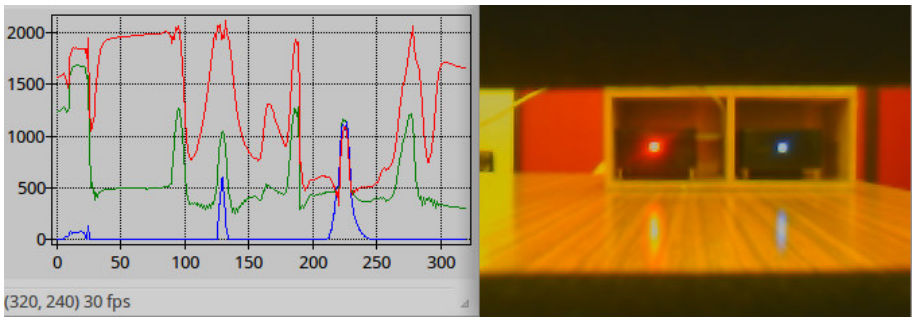
(b) Câmara de vídeo *PlayStation Eye* desmontada

Figura 4.9: *PlayStation Eye* - montada e desmontada

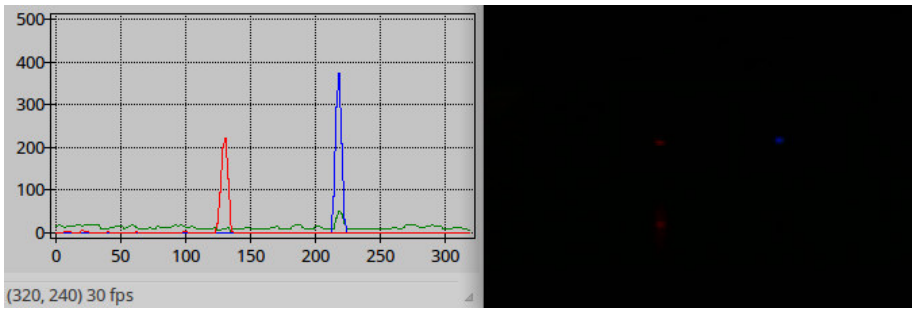
A câmara foi estrategicamente colocada ao nível dos LEDs aquando a sua presença nos armazéns ou fábricas. Isto permite uma melhor filtragem das componentes RGB (*red*, *green* e *blue*), uma vez que, independentemente da distância entre o robô e as caixas, a luz dos LEDs encontra-se sem apresentada a meio da altura da imagem captada. Desta forma é possível restringir a identificação da componente RGB da imagem a uma área bem menor, melhorando significativamente os resultados obtidos.

Devido ao ruído luminoso e ao restante conteúdo desnecessário da imagem, foi necessário filtrar a captura por forma a ser possível identificar a cor dos LED's. Nas figuras 4.11a, 4.10a e 4.12a é possivel ver a imagem que a câmara de vídeo recebe e os seus correspondentes valores de vermelho, verde e azul. É possível também constatar que se torna impercetível, pelos valores RGB, detetar que cor se encontra exibida pelos LED's das caixas.

Procedeu-se à manipulação da imagem, regulando valores de contraste, brilho, ganho, etc. Os resultados obtidos estão presentes nas figuras 4.11b, 4.10b e 4.12b. Nas mesmas figuras é agora visível dois máximos locais, representando o nível das cores vermelha, verde e azul presentes nos LED's. É de salientar que, aquando a deteção da componente azul, é inerente a existência de componente verde. No entanto, a quantidade desta componente não é suficientemente elevada para ser confundida com o azul.

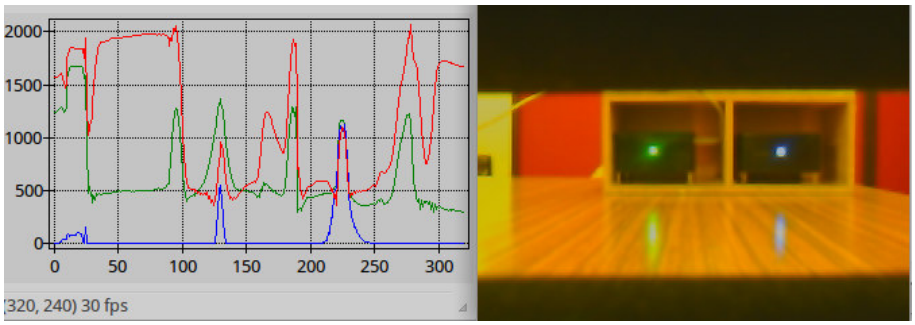


(a) Captura de imagem de máquina com 2 caixas: vermelha e azul

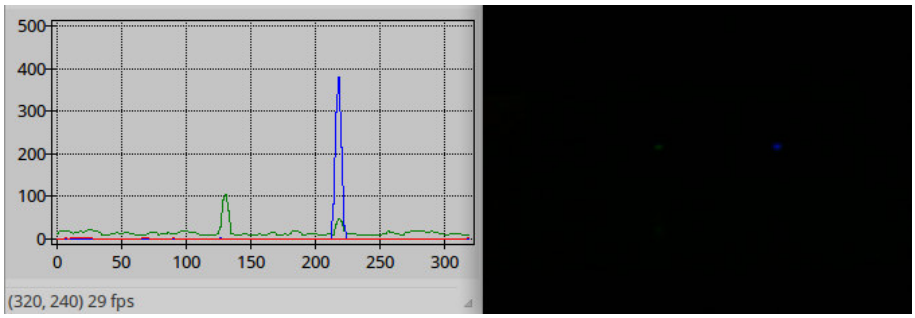


(b) Filtro de imagem de máquina com 2 caixas: vermelha e azul

Figura 4.10: Comparação das componentes RGB da imagem com caixas vermelha e azul

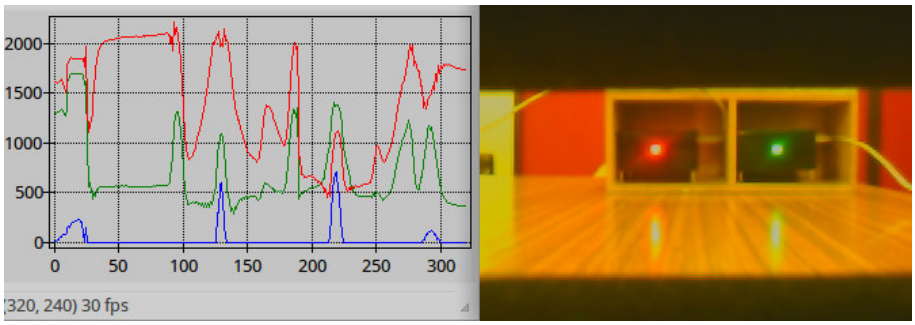


(a) Captura de imagem de máquina com 2 caixas: verde e azul

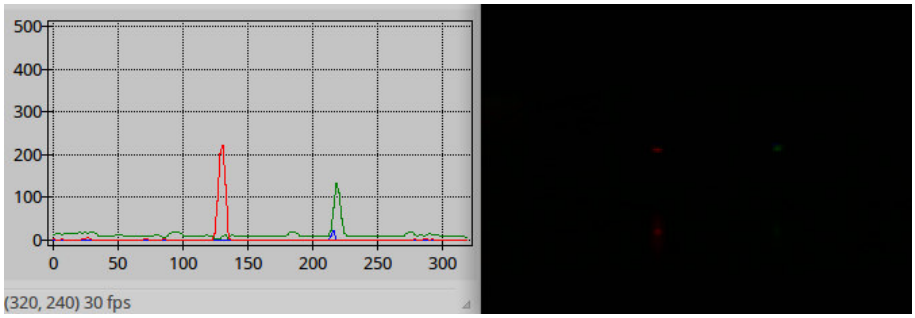


(b) Filtro de imagem de máquina com 2 caixas: verde e azul

Figura 4.11: Comparação das componentes RGB da imagem com caixas verde e azul



(a) Captura de imagem de máquina com 2 caixas: vermelha e verde



(b) Filtro de imagem de máquina com 2 caixas: vermelha e verde

Figura 4.12: Comparação das componentes RGB da imagem com caixas vermelha e verde

4.1.2 Projeto CAD

Com o objetivo de facilitar a montagem do robô desenhou-se, com auxílio do *software SolidWorks*, uma representação da sua estrutura. Este desenho, à escala, contém todos os elementos constituintes do robô, com a exceção de fios, *drives* dos motores DC e câmara de vídeo.

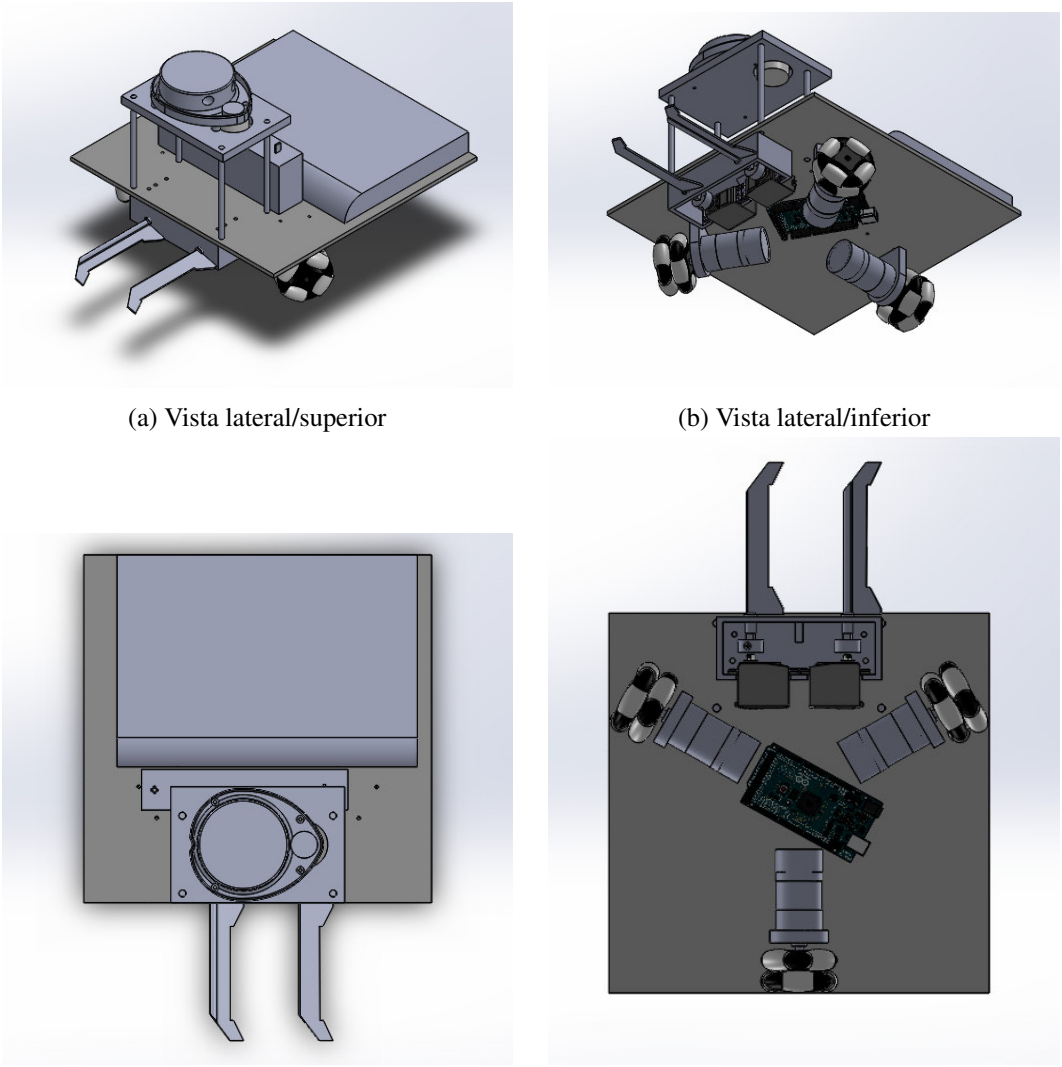
A grande finalidade desta componente é um correto dimensionamento de componentes a serem impressos ou maquinados, por forma a evitar constantes modificações e danificações na estrutura. Aquando a eventual necessidade de reproduzir uma réplica do robô ou da substituição de alguma peça danificada, a existência destes dimensionamentos facilitarão também o processo.

Os modelos desenhados encontram-se na Figura 4.13.

4.1.3 Robô prototipado

Após o projeto em 4.1.2, procedeu-se à construção do robô. O resultado, como é possível ser visto na Figura 4.14, é um robô pequeno e robusto, com todos os seus componentes estrategicamente posicionados por forma a obter o máximo de manobrabilidade, segurança no transporte e informação possível do meio que o envolve.

Na Figura 4.16 e 4.15 estão apresentadas, sob a forma de diagrama de blocos, a arquitetura funcional do robô físico e simulado, respetivamente. Como blocos azuis encontram-se representadas as camadas pré-existentes, tais como o laser e câmara adotados (e correspondentes *firmwares* de aquisição), o programa de controlo dos motores DC e servomotores (*Rover5*) e um *joystick*



(a) Vista lateral/superior

(b) Vista lateral/inferior

(c) Vista superior

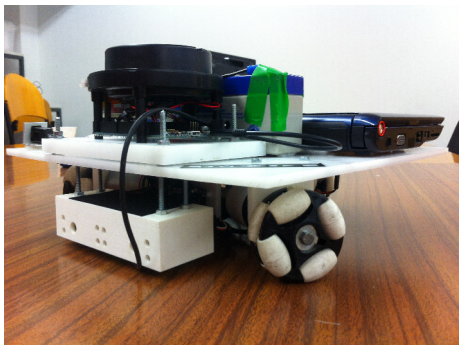
(d) Vista inferior

Figura 4.13: Desenho CAD do robô projetado

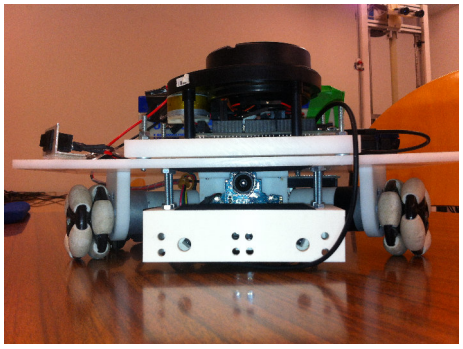
para controlo manual do robô. Aquando a ligação dos programas de localização e planeamento de trajetórias ao simulador *SimTwo*, as funções das camadas do *Robô - Arduino*, *Laser* e *Rover5* são realizadas pelo simulador. As velocidades de cada roda e medições do laser são então enviadas pelo simulador. Representados a verde, amarelo e cor de laranja encontram-se os blocos de controlo associados às dissertações dos elementos da equipa *FeupFactory*, estando a presente dissertação representada a cor de laranja. É também apresentado na figura 4.17 o diagrama representativo da ligação do hardware existente no robô.

Dado tratar-se de um sistema em tempo real, as comunicações entre as demais componentes são efetuadas maioritariamente via pacotes UDP, sendo apenas efetuada a ligação ao *hardware* via USB.

(a) Vista superior



(b) Vista diagonal



(c) Vista frontal

Figura 4.14: Robô projetado

4.2 Simulador

Para que o trabalho ao longo da dissertação pudesse ser testado e validado, sem a constante necessidade de recorrer à máquina física projetada, criou-se um robô no simulador *SimTwo*, para satisfazer essa necessidade.

O *SimTwo* é um sistema de simulação onde é possível implementar vários tipos de robôs, desde manipuladores, quadrúpedes, humanóides e qualquer tipo de robô terrestre que possa ser descrito como um conjunto articulações e rodas, sejam elas clássicas ou omnidirecionais. O realismo do simulador é conseguido uma vez que, a cada componente do robô, estão associadas características físicas tais como: forma, massa, momentos de inércia, atritos, entre outros.

O ambiente da competição já se encontrava implementado, foi apenas necessário criar o robô (Figura 4.19), o mais fiel ao modelo físico possível, tendo em consideração os tamanhos, pesos e posições das diferentes componentes. O ambiente de simulação e respetivo robô modelado pode ser visto nas Figuras 4.18 e 4.19.

Este ambiente de simulação permitiu uma constante validação de algoritmos sem a necessidade de se estar ligado à máquina física. Isto revelou-se ser uma mais valia, uma vez que o projeto do robô revelou-se ser mais desafiante do que inicialmente planeado. Por várias vezes este necessitou de estar parado para ser ajustado, quer a nível de código de baixo nível, quer a nível da sua estrutura física.

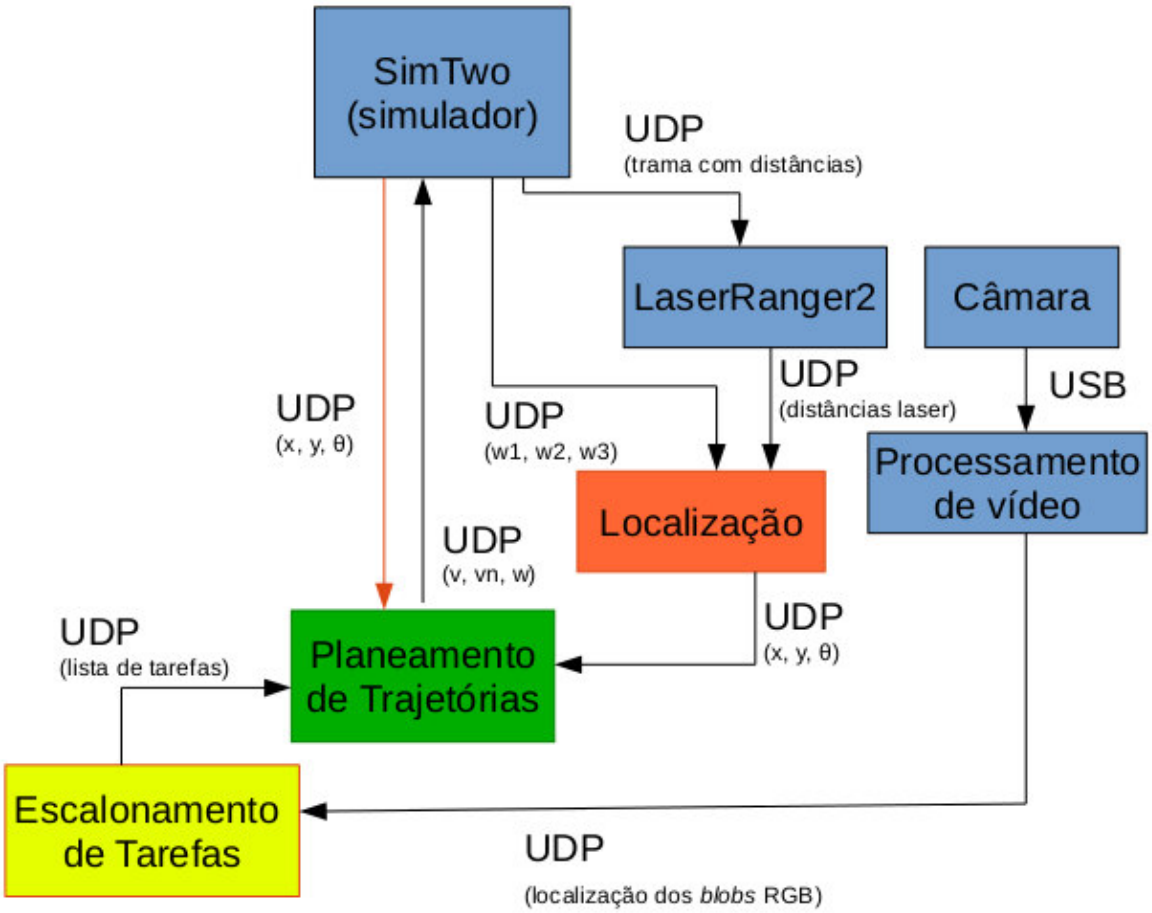


Figura 4.15: Diagrama de blocos representativo da arquitetura funcional do robô simulado

Informação detalhada sobre o ambiente de simulação pode ser encontrada em [31].

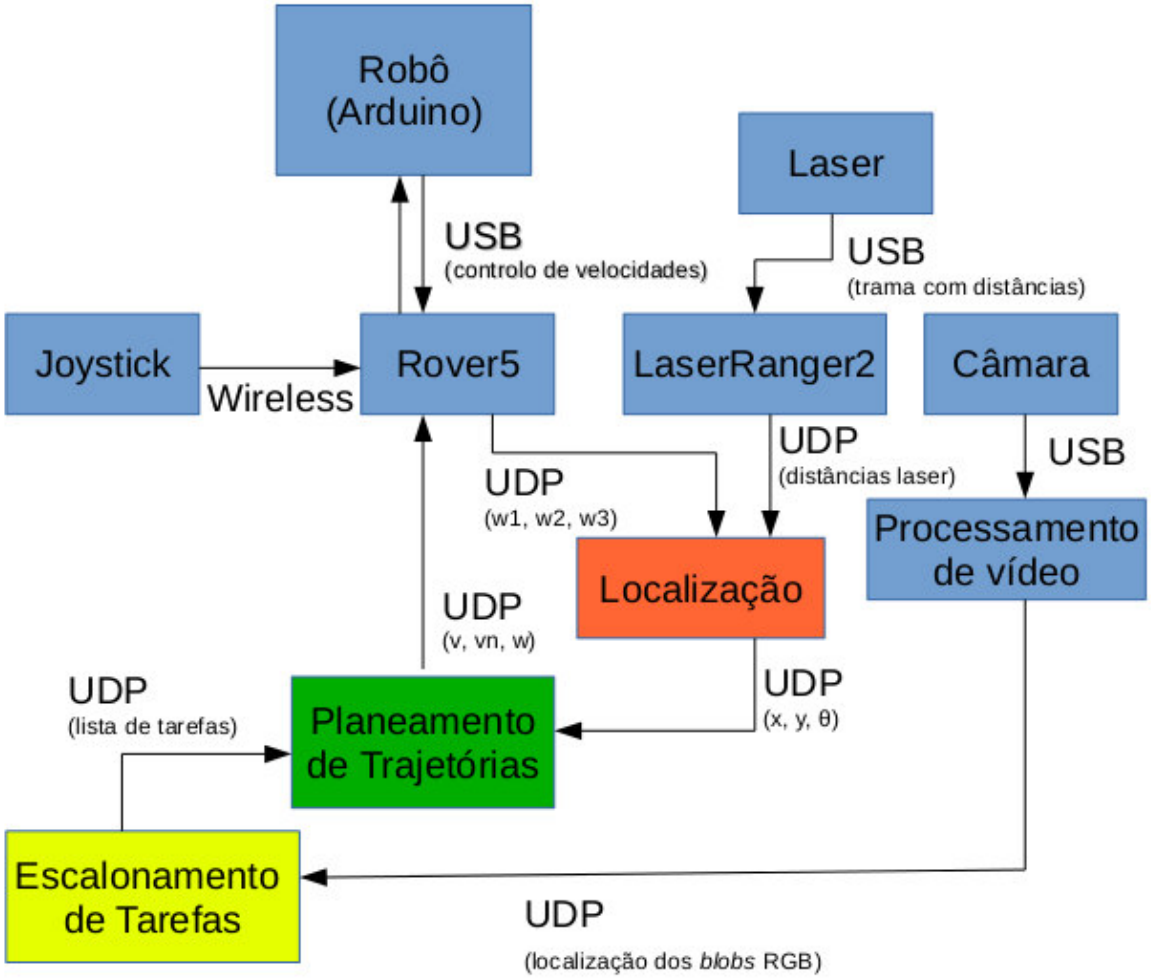


Figura 4.16: Diagrama de blocos representativo da arquitetura funcional do robô prototipado

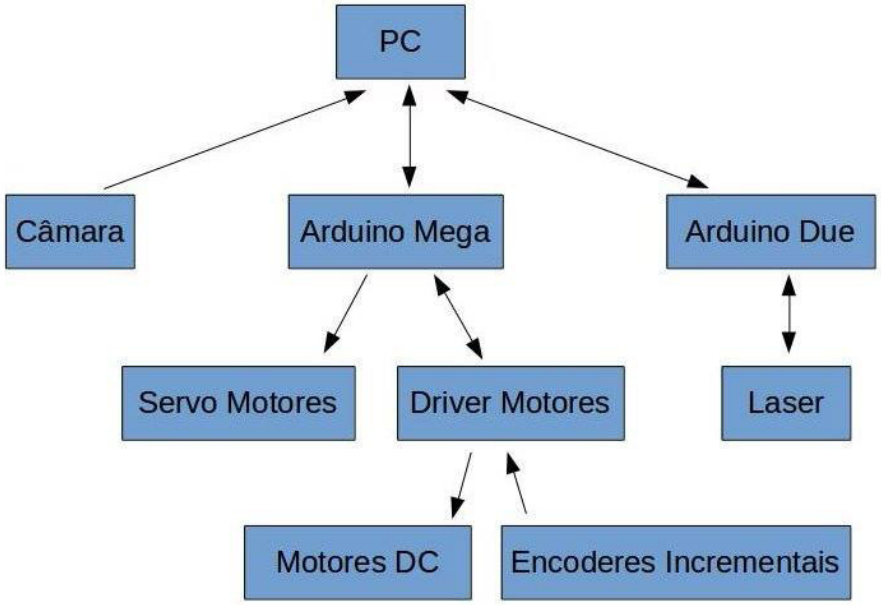


Figura 4.17: Diagrama de blocos representativo das ligações entre *hardware* do sistema

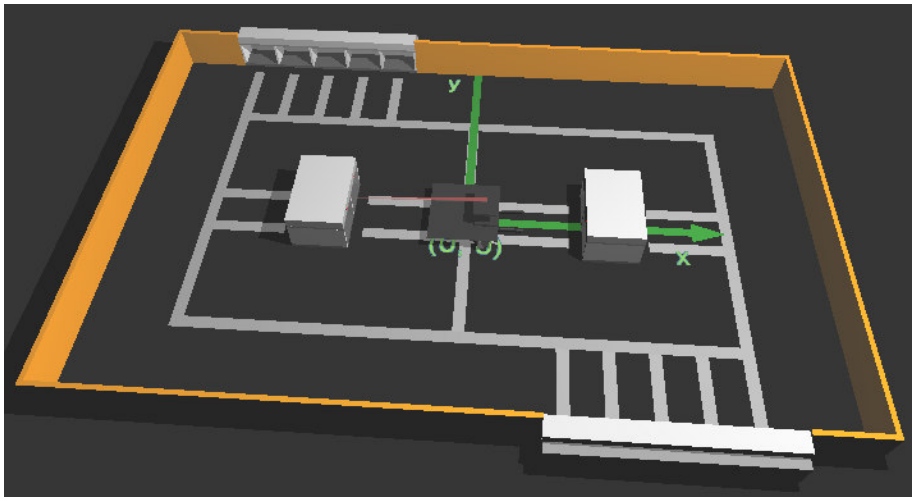


Figura 4.18: Ambiente de simulação representativo da competição

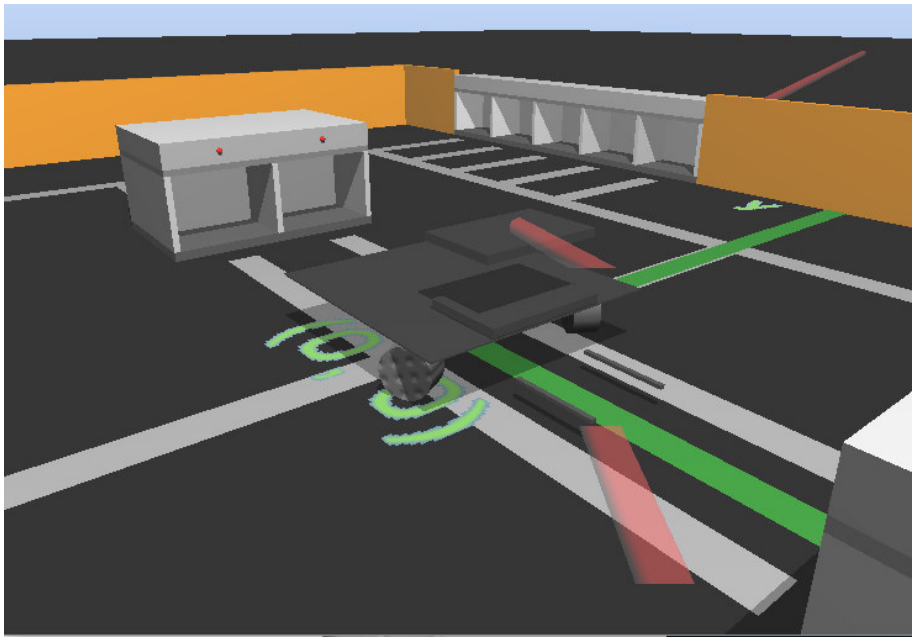


Figura 4.19: Simulação do robô projetado em ambiente de competição

Capítulo 5

Sistema de Localização

Neste capítulo são apresentados os diversos tipos de localização implementados, bem como descritas todas as equações e características do robô que tornam possível e influenciam a localização do mesmo, tendo em consideração o projeto no capítulo 3.

Primeiramente são apresentadas as equações que definem o movimento do robô, necessárias à sua localização relativa (5.1.1).

Posteriormente é apresentado o algoritmo utilizado para efetuar a localização absoluta do robô, o *Perfect Match* (5.2.2). É também feita uma breve referência ao *software* responsável pela aquisição da informação (5.2.1).

Por fim é apresentado o filtro de *Kalman* estendido, usado para efetuar a fusão sensorial entre as medições da odometria e as do laser. Este encontra-se dividido em duas secções: a previsão de estado (5.3.1) e a correção de estado (5.3.2). Em cada uma das secções apresentadas é feita um estudo das equações que as caracterizam.

5.1 Localização Relativa

5.1.1 Dinâmica do sistema

Como forma de obter a localização relativa do robô, adotou-se a odometria. Para tal é necessário, primeiramente, fazer um estudo das características da dinâmica do mesmo.

Como o robô construido é um robô omnidirecional de três rodas, com rodas do tipo *swedish* 90^o *wheel* (apresentada na Figura 2.4 e Figura 5.1), é possível descrever a sua dinâmica segundo as equações [5.1] e [5.2] [6].

$$\begin{bmatrix} sin(\alpha+\beta+\gamma) & -cos(\alpha+\beta+\gamma) & -l*cos(\beta+\gamma) \end{bmatrix} R(\theta)\dot{\xi}_I - r\dot{\varphi}cos(\gamma) = 0 \tag{5.1}$$

$$\begin{bmatrix} cos(\alpha+\beta+\gamma) & sin(\alpha+\beta+\gamma) & l*sin(\beta+\gamma) \end{bmatrix} R(\theta)\dot{\xi}_I - r\dot{\varphi}sin(\gamma) - r_{sw}\dot{\varphi}_{sw} = 0 \tag{5.2}$$

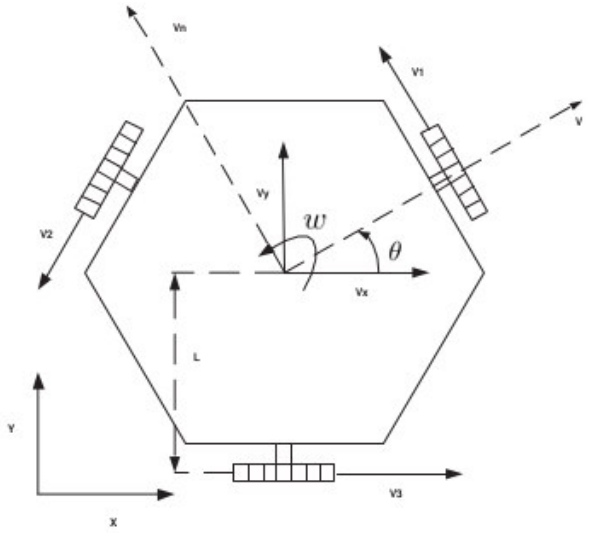


Figura 5.1: Geometria de um robô omnidirecional de três rodas [7]

Onde:

- r - raio da roda
- l - distância entre cada roda e o centro do robô
- γ - ângulo entre o plano da roda e o eixo de rotação das rodas pequenas
- β - ângulo do plano da roda relativamente ao chassis.
- θ - ângulo que o eixo xx do robô faz com o eixo xx do mundo
- α - ângulo entre as diversas rodas
- φ - posição angular da roda
- $\dot{\xi}$ - velocidade do robô em relação ao referencial mundo [5.3]
- $R(\theta)$ - matriz de rotação do referencial do robô para o referencial do mundo [5.4]

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} & \dot{y} & \dot{\theta} \end{bmatrix}^T \tag{5.3}$$

$$R(\theta) = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

Como se trata de uma *swedish wheel* de $\frac{\pi}{2}$ *rad*, $\gamma = 0$ *rad*, logo as equações [5.1] e [5.2] podem ser simplificadas, como demonstrado pelas equações [5.5] e [5.6].

$$\begin{bmatrix} sin(\alpha+\beta) & -cos(\alpha+\beta) & -l*cos(\beta) \end{bmatrix} R(\theta)\dot{\xi}_I - r\dot{\varphi} = 0 \tag{5.5}$$

$$\begin{bmatrix} cos(\alpha+\beta) & sin(\alpha+\beta) & l*sin(\beta) \end{bmatrix} R(\theta)\dot{\xi}_I - r_{sw}\dot{\varphi}_{sw} = 0 \tag{5.6}$$

Dado que todas as rodas do robô estão tangentes ao seu corpo, $\beta = 0$ *rad*, reduzindo-se as equações [5.5] e [5.6] a [5.7] e [5.8].

$$\begin{bmatrix} sin(\alpha) & -cos(\alpha) & -l \end{bmatrix} R(\theta)\dot{\xi}_I - r\dot{\varphi} = 0 \tag{5.7}$$

$$\begin{bmatrix} cos(\alpha) & sin(\alpha) & 0 \end{bmatrix} R(\theta)\dot{\xi}_I - r_{sw}\dot{\varphi}_{sw} = 0 \tag{5.8}$$

Verifica-se também que o as três rodas estão desfasadas entre si $\frac{2\pi}{3}$ *rad*, levando α a assumir valores como:

$$\begin{cases} \alpha_1 = \frac{\pi}{3}\ rad \\ \alpha_2 = \pi\ rad \\ \alpha_3 = -\frac{\pi}{3}\ rad \end{cases} \tag{5.9}$$

Obtém-se assim as equações finais ([5.10][5.11]) que definem o movimento deste sistema robótico.

$$\begin{bmatrix} sin(\frac{\pi}{3}) & -cos(\frac{\pi}{3}) & -l \\ 0 & 1 & -l \\ sin(-\frac{\pi}{3}) & -cos(-\frac{\pi}{3}) & -l \end{bmatrix} R(\theta)\dot{\xi}_I - r\dot{\varphi} = 0 \tag{5.10}$$

$$\begin{bmatrix} cos(\frac{\pi}{3}) & sin(\frac{\pi}{3}) & 0 \\ -1 & 0 & 0 \\ cos(-\frac{\pi}{3}) & sin(-\frac{\pi}{3}) & 0 \end{bmatrix} R(\theta)\dot{\xi}_I - r_{sw}\dot{\varphi}_{sw} = 0 \tag{5.11}$$

Por forma a obter a configuração do sistema, o recurso a [5.10] verifica-se ser suficiente. O recurso às restrições de deslizamento (definidas em [5.11]) permite avaliar a manobrabilidade, complementando assim a estimação da posição do robô [6].

Desenvolvendo [5.10] e simplificando os resultados, obtém-se uma equação que relaciona a velocidades V_x, V_y e W no referencial do mundo com as velocidades lineares de cada roda.

$$\dot{\xi}_I = R(\theta)^{-1} \begin{bmatrix} sin(\frac{\pi}{3}) & -cos(\frac{\pi}{3}) & -l \\ 0 & 1 & -l \\ sin(-\frac{\pi}{3}) & -cos(-\frac{\pi}{3}) & -l \end{bmatrix}^{-1} \cdot r\dot{\varphi} \tag{5.12}$$

$$\dot{\xi}_I = R(\theta)^{-1} \begin{bmatrix} \frac{\sqrt{3}}{3} & 0 & -\frac{\sqrt{3}}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3*l} & -\frac{1}{3*l} & -\frac{1}{3*l} \end{bmatrix} \cdot r\dot{\varphi} \tag{5.13}$$

$$\therefore$$

$$\begin{bmatrix} V_x \\ V_y \\ W \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}*cos(\theta)}{3} + \frac{sin(\theta)}{3} & -\frac{2*sin(\theta)}{3} & -\frac{\sqrt{3}*cos(\theta)}{3} + \frac{sin(\theta)}{3} \\ \frac{\sqrt{3}*sin(\theta)}{3} - \frac{cos(\theta)}{3} & \frac{2*cos(\theta)}{3} & -\frac{\sqrt{3}*sin(\theta)}{3} - \frac{cos(\theta)}{3} \\ -\frac{1}{3*l} & -\frac{1}{3*l} & -\frac{1}{3*l} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \tag{5.14}$$

A estimativa da posição e orientação do robô pode ser calculada, aplicando-se uma aproximação de primeira ordem [7], como exemplificado nas equações [5.15],[5.16] e [5.17],

$$x(k) = x(k-1) + V_x T \tag{5.15}$$

$$y(k) = y(k-1) + V_y T \tag{5.16}$$

$$\theta(k) = \theta(k-1) + \omega T \tag{5.17}$$

onde T é o tempo de amostragem.

A velocidade linear de cada roda, durante um período de amostragem, é determinada a partir do número de impulsos gerados pelo correspondente *encoder*. Definindo o deslocamento entre cada impulso e o multiplicando-o pelo número de impulsos contados entre amostragens, obtém-se a velocidade angular de cada roda (ω_1, ω_2 e ω_3). Cada uma das velocidades lineares é obtida resolvendo 5.18

$$V_i = \omega_i * r_{r_i},\ i = 1,\ 2,\ 3 \tag{5.18}$$

sendo r_{r_i} o raio de cada uma das rodas.

5.2 Localização Absoluta

5.2.1 Interface de Comunicação

Por forma a adquirir as várias distâncias lidas pelo *Lidar*, recebidas via porta série, foi necessário recorrer ao um programa auxiliar. Este programa lê a porta série do computador, filtra o pacote de dados recebido e envia um novo pacote, constituído exclusivamente por um cabeçalho e as medições efetuadas, via UDP, para o programa de localização do robô.

O programa foi fornecido pelo laboratório onde a dissertação foi desenvolvida.

5.2.2 Perfect Match

O *Perfect Match* é um algoritmo com uma abordagem ao problema da localização baseada no erro associado a cada medição, minimizando-o. É uma metodologia caracterizada por ser de elevada eficiência e de requerer baixo poder computacional [19].

5.2.2.1 Mapa

Para ser possível a localização utilizando o algoritmo *Perfect Match* é necessário o robô conhecer o mapa do local por onde se vai movimentar. Este mapa, mapa do campo da competição *Robot@Factory*, é conhecido, o que permite uma fácil representação do mesmo.

O mapa criado para interpretação do robô é uma matriz onde se encontram representadas as paredes e máquinas do campo, com uma resolução de $1cm$. Esta representação encontra-se apresentada na Figura 5.2.

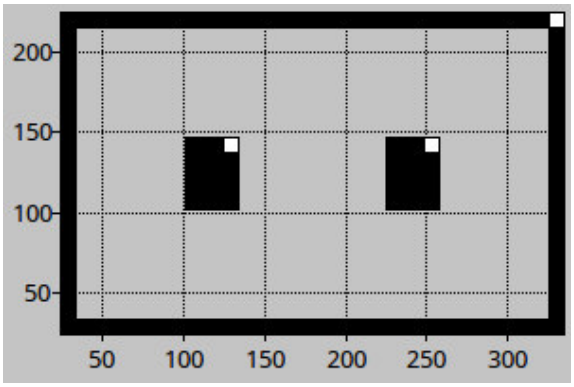


Figura 5.2: Representação do mapa do campo para interpretação do robô

5.2.2.2 Mapeamento das leituras

Primeiramente é necessário enquadrar cada uma das medições feitas pelo *range finder* no mapa onde o robô se encontra. Seja $s_1 ... s_n$ o vetor de medições, relativas ao robô, proveniente da interface de comunicação com o laser (Figura 5.3), e α o ângulo dessa medição relativa à

orientação do robô. É possível então determinar a posição de cada ponto lido, em relação ao sistema robótico no mundo (ξ_{xy}) , segundo a equação [5.19].

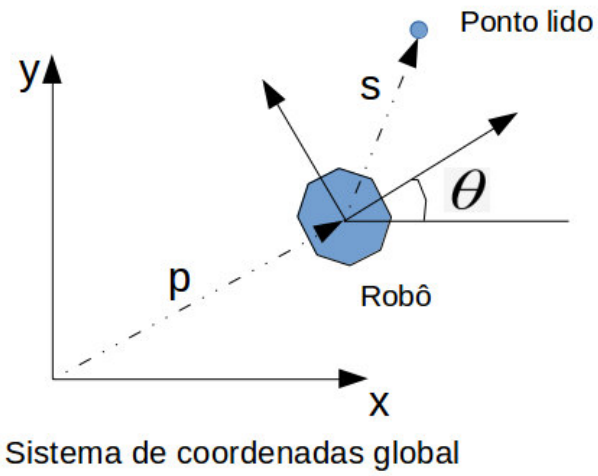


Figura 5.3: Representação de robô em referencial do mundo e de ponto lido relativo a robô

$$P_{l_i} = \xi_{xy} + \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \cdot s_i \begin{bmatrix} cos(\alpha) \\ sin(\alpha) \end{bmatrix} \tag{5.19}$$

Obtém-se desta forma as coordenadas x e y, no mundo, de cada medição feita pelo *range finder*.

Um exemplo do mapeamento das leituras é exibido na Figura 5.4. Neste exemplo encontra-se representado o mapeamento dos pontos quando o robô se encontra na posição $x = 0$ e $y = 0$ (centro do campo).

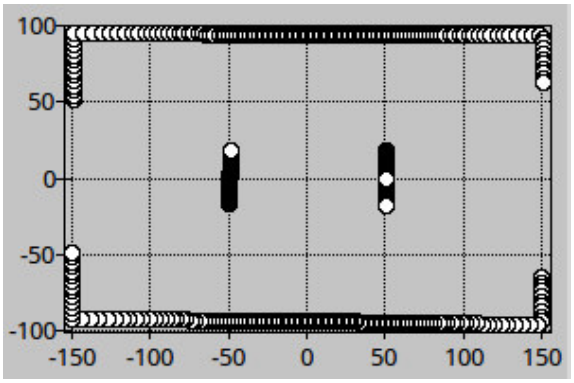


Figura 5.4: Pontos mapeados quando robô se encontra em $(0,0)$

Dado os referenciais das figuras 5.4 e 5.2 não serem os mesmos, visto o ponto (0,0) do mapa

representativo não ser o mesmo da mapa da projeção de pontos, foi necessário aplicar um *offset* aos pontos representados em 5.4, por forma a coincidirem com os apresentados no mapa.

5.2.2.3 Minimização do erro

Como já referido, a estimação da posição do robô, feita segundo o *Perfect Match*, baseia-se na minimização do erro entre o mapa do campo e o mapeamento feito. Recorrendo a um mapa de distâncias, é possível definir o erro para cada um dos pontos lidos.

Um mapa de distâncias é uma matriz que contém em cada célula um valor numérico representativo da distância ao objeto mais próximo. Por forma a obter o máximo de precisão nos valores calculados, recorreu-se a uma máscara $3x3$ [5.20], que faz o varrimento por todo o mapa. No final do varrimento, todos os valores da matriz são divididos por 2.

$$\begin{bmatrix} 3 & 2 & 3 \\ 2 & 0 & 2 \\ 3 & 2 & 3 \end{bmatrix} \tag{5.20}$$

A grande vantagem desta metodologia em relação a outras, tais como a métrica *city block* ou *chessboard*, é o facto de esta permitir uma aproximação viável das distâncias nas diagonais do ponto a ser avaliado. Segundo o teorema de Pitágoras, a distância entre um ponto em $P_{l-1,c-1}$ e $P_{l,c}$ será $\sqrt{1^2+1^2} = 1.4$. Aplicando a metodologia referida, este valor será igual a 1.5, tendo apenas um erro de 1mm, considerando-se ser uma boa aproximação. O resultado da aplicação desta máscara encontra-se representado na Figura 5.5.

Em seguida é necessário sobrepor as duas matrizes e avaliar o erro de cada leitura. Considera-se como o erro da leitura o valor presente na célula da matriz de distâncias na qual o ponto P_i pousar.

O função do erro é determinar o quão desalinhados estão os dois mapas. Quanto maior o erro, menos coincidentes estes são. Segundo [19], a melhor função para avaliar este tipo de erros é apresentada em [5.21].

$$erro = 1 - \frac{c^2}{c^2 + e^2} \tag{5.21}$$

Esta função de erro tem como vantagem em relação à do erro quadrático médio o facto de ser mais robusta aquando a ocorrência de medidas discrepantes. Desta forma, se um determinado erro de medição for muito elevado, não vai afetar de forma significativa a correção efetuada.

Por forma a minimizar o erro calculado, como sugerido por [19], recorreu-se ao algoritmo *resilient backpropagation* (RPROP) [32, 33] . Este algoritmo tem em conta apenas o sinal da derivada parcial sobre todos os pontos, e atua de forma independente sobre o erro de cada ponto.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0														0
0														0
0						0	0	0						0
0						0	0	0						0
0						0	0	0						0
0														0
0														0
0														0
0						0	0	0						0
0						0	0	0						0
0						0	0	0						0
0														0
0														0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(a) Exemplo de mapa

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	2	2	2	1.5	1	1	1	1.5	2	2	2	1	0
0	1	2	3	2	1	0	0	0	1	2	3	2	1	0
0	1	2	3	2	1	0	0	0	1	2	3	2	1	0
0	1	2	3	2	1	0	0	0	1	2	3	2	1	0
0	1	2	3	2.5	1.5	1	1	1	1.5	2.5	3	2	1	0
0	1	2	3	3	2.5	2	2	2	2.5	3	3	2	1	0
0	1	2	3	2.5	1.5	1	1	1	1.5	2.5	3	2	1	0
0	1	2	3	2	1	0	0	0	1	2	3	2	1	0
0	1	2	3	2	1	0	0	0	1	2	3	2	1	0
0	1	2	3	2	1	0	0	0	1	2	3	2	1	0
0	1	2	2	2	1.5	1	1	1	1.5	2	2	2	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b) Exemplo de mapa de distâncias

Figura 5.5: Mapa de distâncias obtido a partir de mapa pré-definido

Dado que a estimação da posição do robô é feita tanto para x, y e θ, é necessário calcular o gradiente para cada uma destas variáveis. O sinal do gradiente calculado tem como função indicar a direção da atualização do erro.

Primeiramente é necessário definir um valor de atualização, Δ_{ij}, que define magnitude do andamento que o erro vai ter. Consoante o sinal do somatório dos gradientes segundo determinada direção, o valor da correção segundo essa mesma direção vai variar segundo 5.22.

$$\Delta w_{ij} = \begin{cases} -\Delta_{ij}, & \textit{se} \text{ somatório } \textit{de gradiente} > 0 \\ \Delta_{ij}, & \textit{se} \text{ somatório } \textit{de gradiente} < 0 \\ 0, & \textit{se outros valores} \end{cases} \tag{5.22}$$

Posteriormente atualiza-se Δ_{ij}, consoante o sinal do gradiente se mantenha constante ou mude. A mudança do sinal do gradiente indica que a última correção foi demasiado grande, levando o algoritmo a passar pelo objeto. Caso isto se verifique, o valor de atualização é multiplicado por uma constante de valor inferior a 1, η^-, de forma a diminuir o avanço e se conseguir aproximar do mínimo local. Contrariamente, se o valor do gradiente mantiver o seu sinal, Δ_{ij} é multiplicado por uma constante superior a 1, η^+, de forma a ocorrer uma convergência mais rápida. No caso de mudança de sinal do gradiente, [32, 33] sugerem que não haja atualização de Δw_{ij}. Isto é conseguido igualando o valor do gradiente anterior a zero.

Os valores de Δw são somados e divididos pelo número total de pontos avaliados, obtendo desta forma a média do deslocamento que o robô terá que efetuar segundo x, y e θ. Este processo é repetido até se verificarem critérios de paragem previamente estipulados. O número de interações do algoritmo RPROP, o erro total e a variação do erro total foram os critérios adotados.

O pseudocódigo do algoritmo de minimização do erro é descrito de seguida:

Algoritmo: RPROP

$\forall i,j : \Delta_{ij}(t) = \Delta_0$
$\forall i,j : \frac{\partial E}{\partial w_{ij}}(t-1) = 0$
Repetir até convergir:
  Calcular gradiente $\frac{\partial E}{\partial w}$
  Para todos os pesos e direções {
    Se ($\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) > 0$) então {
      $\Delta_{ij}(t)$ = mínimo($\Delta_{ij}(t-1) * \eta^+, \Delta_{max}$)
      $\Delta w_{ij}(t)$ = -sinal($\frac{\partial E}{\partial w_{ij}}(t)$)$*\Delta_{ij}(t)$
      $w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$
      $\frac{\partial E}{\partial w_{ij}}(t-1) = \frac{\partial E}{\partial w_{ij}}(t)$
    }
    Se não, se ($\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) < 0$) então {
      $\Delta_{ij}(t)$ = máximo($\Delta_{ij}(t-1) * \eta^-, \Delta_{min}$)
      $\frac{\partial E}{\partial w_{ij}}(t-1) = 0$
    }
    Se não, se ($\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) = 0$) então {
      $\Delta w_{ij}(t)$ = -sinal($\frac{\partial E}{\partial w_{ij}}(t)$)$*\Delta_{ij}(t)$
      $w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$
      $\frac{\partial E}{\partial w_{ij}}(t-1) = \frac{\partial E}{\partial w_{ij}}(t)$
    }
  }

Modificações ao algoritmo RPROP

Por forma a simplificar a implementação do algoritmo, foram efetuadas umas mudanças ao mesmo. Estas alterações estão relacionadas com a atualização do peso na célula a ser analisada, onde $\Delta w_{ij}(t)$ passou a ser o salto que o ponto dá segundo a coordenada a ser analisada. Desta forma, se o ponto analisado se encontrar a uma elevada distância do mínimo local, este é transposto $\Delta w_{ij}(t)$ células na sua direção.

5.2.2.4 Camada de seleção de pontos

A fim de obter a melhor localização possível, são filtrados os valores de distância lidos e ignorados os que induzem um erro elevado. Essa filtração é feita pelo software de aquisição do laser que, juntamente com a trama de valores, envia uma *flag* para cada um desses valores, assinalando se é uma medição sem qualidade. Caso isto se verifique, esses valores são considerados como nulos, não exercendo influência sobre o ajuste de posição efetuado pelo algoritmo acima descrito.

5.3 Fusão sensorial

Equação de estado:

$$\frac{dX(t)}{dt} = f(X(t),\, u(t),\, t) \tag{5.23}$$

onde $u(t)$ são os parâmetros de entrada que, neste caso particular, é composto pelas velocidades lineares do ponto de contacto com a superfície de cada roda.

5.3.1 Previsão

5.3.1.1 Estimação de estado

A estimação do estado no instante t_k requer o conhecimento do estado em t_{k-1} e é feita por integração numérica, como demonstrado em 5.15, 5.16 e 5.17.

5.3.1.2 Propagação da covariância

Por forma a calcular a propagação da covariância, é necessário ter definida as equações que definem a transição do estado. Este sistema de equações, definido em 5.14, deve ser linearizado em torno de $X(t) = X(t_k)$, $u(t) = u(t_k)$ e $t = t_k$, resultando em:

$$A^*(t_k) = \begin{bmatrix} 0 & 0 & (\frac{\sqrt{3}*cos(\theta)}{3} + \frac{sin(\theta)}{3})V_1 - (\frac{2*sin(\theta)}{3})V_2 + (\frac{-\sqrt{3}*cos(\theta)}{3} + \frac{sin(\theta)}{3})V_3 \\ 0 & 0 & (\frac{\sqrt{3}*cos(\theta)}{3} + \frac{sin(\theta)}{3})V_1 + (\frac{2*cos(\theta)}{3})V_2 - (\frac{\sqrt{3}*sin(\theta)}{3} + \frac{cos(\theta)}{3})V_3 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.24}$$

A matriz transição de estado (ϕ) é então a apresentada em 5.25:

$$\phi^*(t_k) = \begin{bmatrix} 1 & 0 & (\frac{\sqrt{3}*cos(\theta)}{3} + \frac{sin(\theta)}{3})V_1T - (\frac{2*sin(\theta)}{3})V_2T + (\frac{-\sqrt{3}*cos(\theta)}{3} + \frac{sin(\theta)}{3})V_3T \\ 0 & 1 & (\frac{\sqrt{3}*cos(\theta)}{3} + \frac{sin(\theta)}{3})V_1T + (\frac{2*cos(\theta)}{3})V_2T - (\frac{\sqrt{3}*sin(\theta)}{3} + \frac{cos(\theta)}{3})V_3T \\ 0 & 0 & 1 \end{bmatrix} \tag{5.25}$$

A propagação da covariância, $P(t_k^-)$, é calculada através da equação 5.26,

$$P(t_k^-) = \phi^*(t_k)P(t_{k-1})\phi^*(t_k)^T + Q(t_k) \tag{5.26}$$

onde $Q(t_k)$ define a covariância do erro. Esta matriz estipula o rigor das medições feitas pela odometria.

5.3.2 Correção

5.3.2.1 Ganho do Filtro de Kalman

O ganho do Filtro de *Kalman* tem como objetivo pesar quais os valores mais confiáveis, entre as diversas fontes de medidas. Este é calculado pela equação que se segue:

$$K(t_k) = P(t_k^-)H(t_k)^T[H(t_k)P(t_k^-)H(t_k)^T + R(t_k)]^{-1} \tag{5.27}$$

Onde $R(t_k)$ representa a covariância do erro das medições. Esta matriz, à semelhança de $Q(t_k)$, define o rigor das medições feitas, neste caso das do laser.

Outro aspeto a salientar é o facto de, como as medidas efetuadas pelo laser são processadas fora do Filtro de *Kalman* Estendido, isto é, as medições que entram nos parâmetros do filtro são coincidentes com o referencial do mundo, a matriz $H(t_k)$ é a matriz identidade. Desta forma, a equação que define o ganho do filtro é reduzida à equação 5.28.

$$K(t_k) = P(t_k^-)[P(t_k^-) + R(t_k)]^{-1} \tag{5.28}$$

5.3.2.2 Atualização da estimação de estado

A atualização do estado é feita recorrendo a equação 5.29.

$$X(t_k) = X(t_k^-) + K(t_k) * [z(t_k) - X(t_k^-)] \tag{5.29}$$

Onde $z(t_k)$ representa o estado calculado obtido através das medições do laser.

5.3.2.3 Atualização da covariância

O último passo do cálculo do Filtro de *Kalman* Estendido é a atualização da covariância, necessário para as iterações seguintes do algoritmo.

$$P(t_k) = [I - K(t_k)H(t_k)]P(t_{k-1}) \tag{5.30}$$

Como já verificado, a matriz $H(t_k)$ é a matriz identidade, simplificando 5.30 a 5.31.

$$P(t_k) = [I - K(t_k)]P(t_{k-1}) \tag{5.31}$$

Capítulo 6

Resultados

Este capítulo é dedicado à apresentação dos resultados obtidos nas plataformas experimentais descritas no capítulo 4. O capítulo inicia-se com a validação dos algoritmos no ambiente de simulação, na secção 6.2, posteriormente são apresentados os resultados obtidos aquando a implementação na plataforma física de testes, secção 6.3. Ao longo do capítulo são também descritos os vários testes efetuados para validar os algoritmos.

Por forma a efetuar os testes, foi utilizado o programa de planeamento de trajetórias de um dos elementos integrados na equipa do *Robot@Factory*.

6.1 Parametrização de variáveis

Aquando a implementação dos algoritmos, é necessária a parametrização de variáveis, por forma a que o controlo definido funcione conforme o estipulado. Na tabela 6.1 estão definidas as variáveis necessárias e correspondentes valores adotados. Para o cálculo da localização relativa, é necessário conhecer o raio de cada uma das rodas (r) e a distância entre o centro do robô e cada roda (l). Por sua vez, a estipulação de parâmetros para o cálculo da localização absoluta, recorreu-se a valores sugeridos por [19], bem como valores estipulados de forma empírica.

6.2 Resultados em ambiente de simulação

6.2.1 Validação da odometria

Dada a natureza ideal dos parâmetros associados ao simulador, é expectável a ausência de erro e ruído nas medições obtidas pelos sensores. Assim, ao contrário do que se verifica no ambiente físico de implementação, a localização baseada exclusivamente na odometria é viável.

Por forma a validar os resultados obtidos, recorreu-se a uma ferramenta disponível no *SimTwo*, que permite a descrição ao longo do tempo de variáveis, sobre a forma de um gráfico. Assim, representando os valores de x, y e θ que o simulador considera os reais, e comparando-os com os que os algoritmos de localização calcula, é possível avaliar a discrepância de valores e a qualidade dos mesmos.

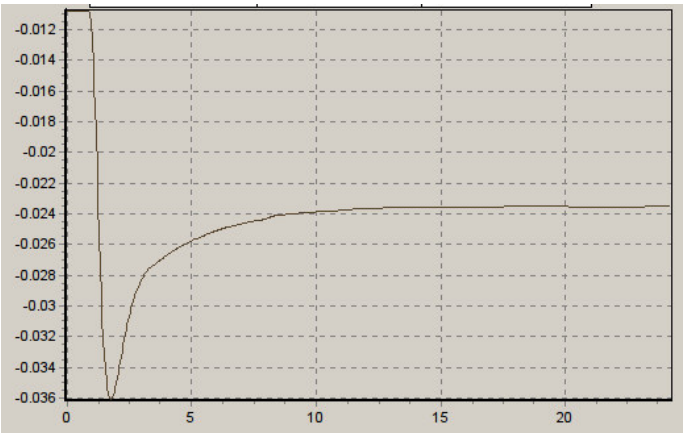
Tabela 6.1: Parametrização de variáveis

Variável	**Valor**
Localização Relativa	
r	0.03 m
l	0.15 m
Localização Absoluta	
$erro_{mnimo}$	0.005
Δ_0	0.05
Δ_{max}	30
Δ_{min}	0.01
η^+	1.2
η^-	0.5
Iterações	10

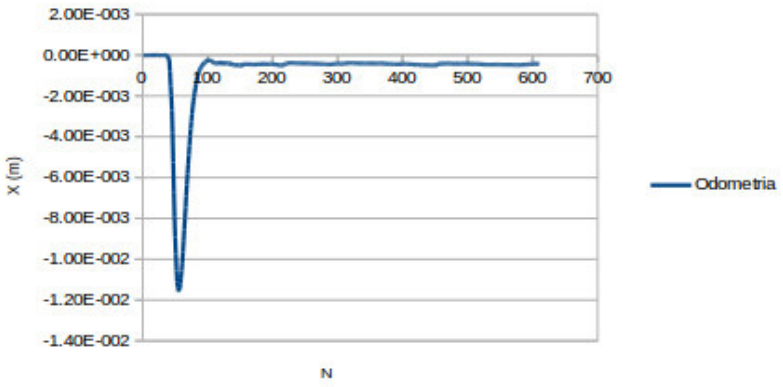
Nas figuras 6.1, 6.2, 6.3 e 6.4 são feitas as comparações das três variáveis, bem como a representação do trajeto do robô no mapa. Excertos dos valores que permitiram a representação gráfica da localização calculada são apresentados na tabela 6.2.

Tabela 6.2: Coordenadas (x, y, θ) obtidos por localização relativa

x (m)	y (m)	θ (º)
-0.000002	-0.499999	-0.000168
-0.000005	-0.500001	-0.000170
⋮	⋮	⋮
-0.001847	0.503738	55.859750
-0.001637	0.504633	56.406509
⋮	⋮	⋮
-0.000427	0.500479	89.983352
-0.000429	0.500480	89.983418

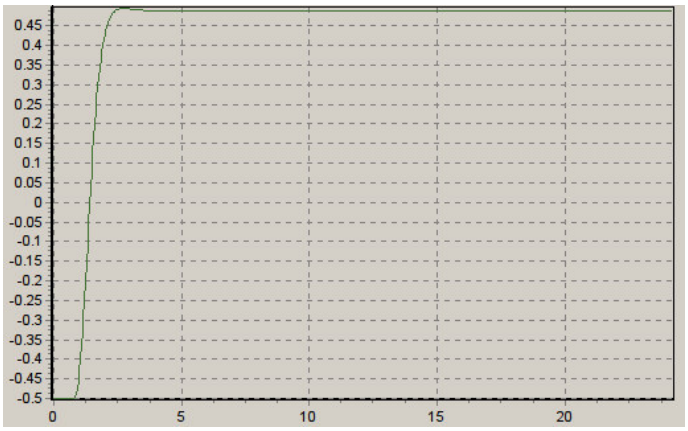


(a) Variação da posição real em x do robô (eixo xx: tempo (s), eixo yy: deslocamento (dm))

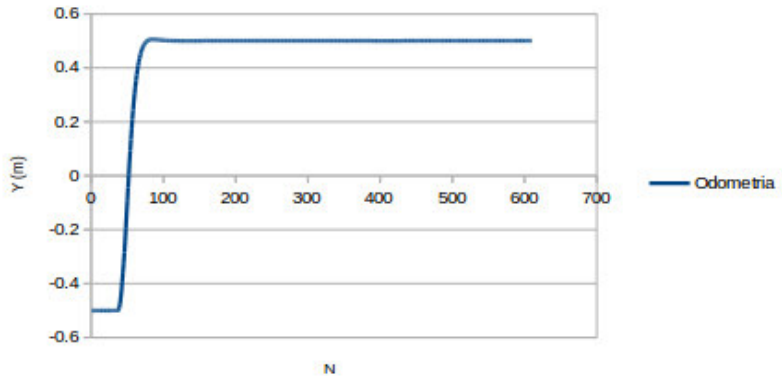


(b) Variação da posição em x do robô segundo algoritmo *Perfect Match* (eixo xx: iterações, eixo yy: deslocamento (m))

Figura 6.1: Localização baseada em laser em simulação, variação segundo x

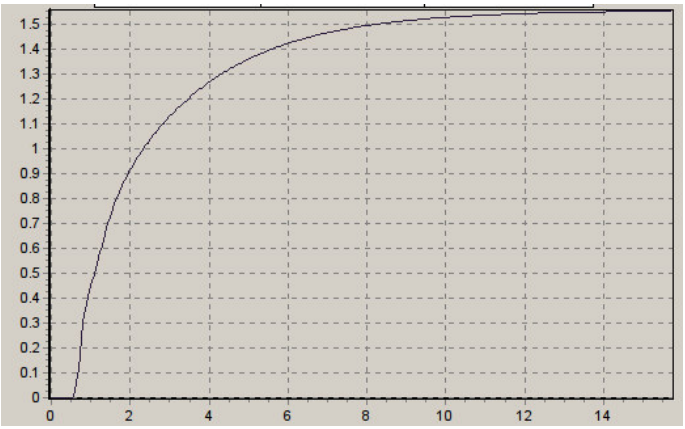


(a) Variação da posição real em *y* do robô (eixo *xx*: tempo (s), eixo *yy*: deslocamento (m))

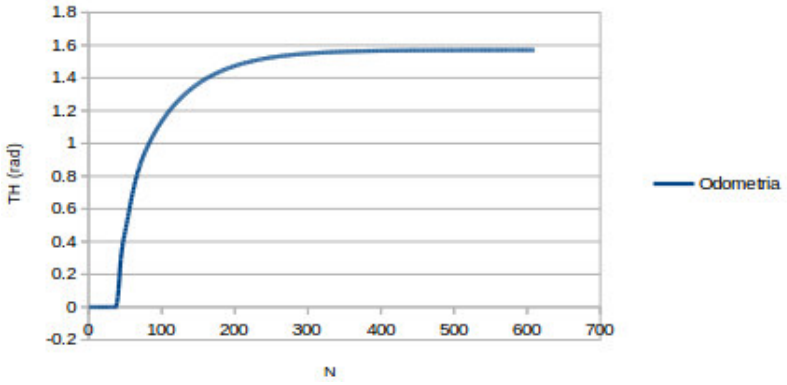


(b) Variação da posição em *y* do robô segundo algoritmo *Perfect Match* (eixo *xx*: iterações, eixo *yy*: deslocamento (m))

Figura 6.2: Localização baseada em laser em simulação, variação segundo *y*



(a) Variação da posição real em θ do robô (eixo *xx*: tempo (s), eixo *yy*: rotação (rad))



(b) Variação da posição em θ (em radianos) do robô segundo algoritmo *Perfect Match* (eixo *xx*: iterações, eixo *yy*: deslocamento (m))

Figura 6.3: Localização baseada em laser em simulação, variação segundo θ

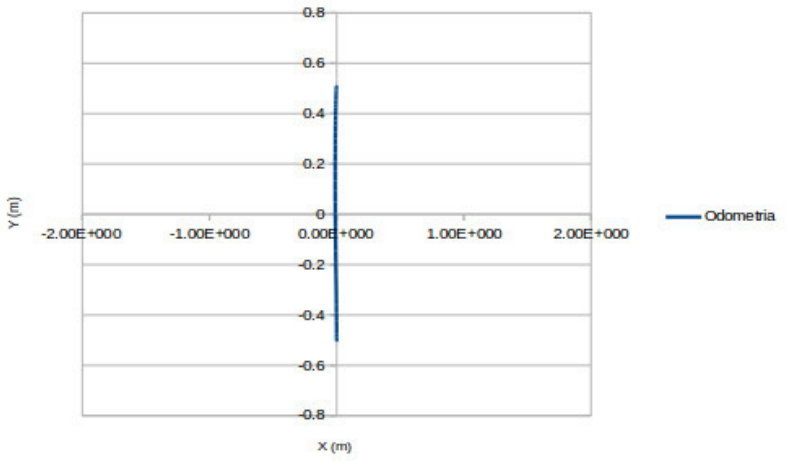


Figura 6.4: Representação do deslocamento do robô no mapa, em simulação (eixo *xx*: deslocamento (m), eixo *yy*: deslocamento (m))

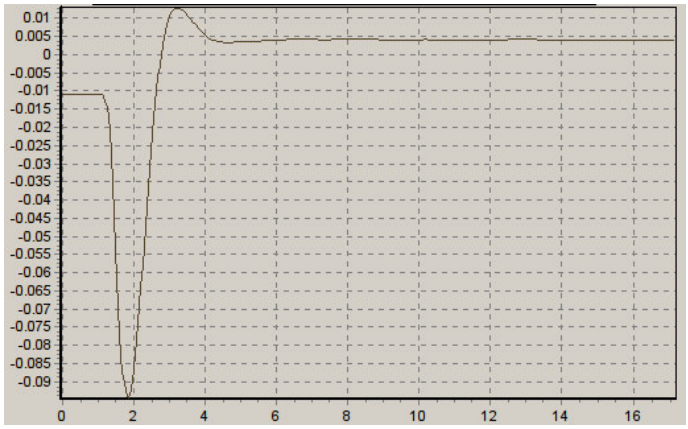
6.2.2 Validação do Perfect Match

Como referido anteriormente, é expectável a ausência de erro e ruído nas medições obtidas pelos sensores, neste caso, o laser.

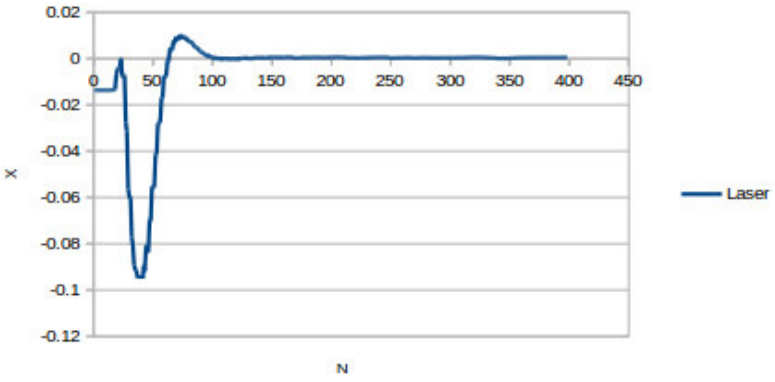
Repetiu-se o teste anteriormente efetuado, teste que consiste no movimento do robô desde o ponto $x = 0$ m, $y = -0.5$ m e $\theta = 0^{o}$ até ao ponto $x = 0$ m, $y = 0.5$ m e $\theta = 90^{o}$. O resultado deste teste é visível nas figuras 6.5, 6.6, 6.7, 6.8 e correspondente tabela 6.3.

Tabela 6.3: Coordenadas (x, y, θ) obtidos por localização absoluta

x (m)	y (m)	θ (^{o})
-0.013671	-0.493749	0.112094
-0.013668	-0.493700	0.109372
⋮	⋮	⋮
0.008897	0.506998	66.300127
0.009139	0.501260	67.243519
⋮	⋮	⋮
0.000422	0.499709	89.902542
0.000424	0.499697	89.904980

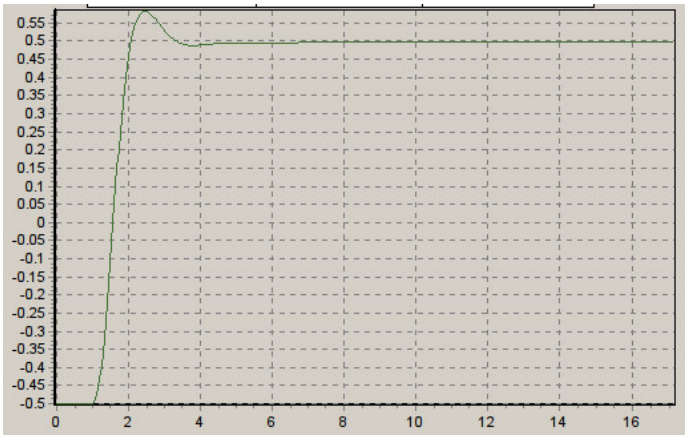


(a) Variação da posição real em x do robô (eixo xx: tempo (s), eixo yy: deslocamento (m))

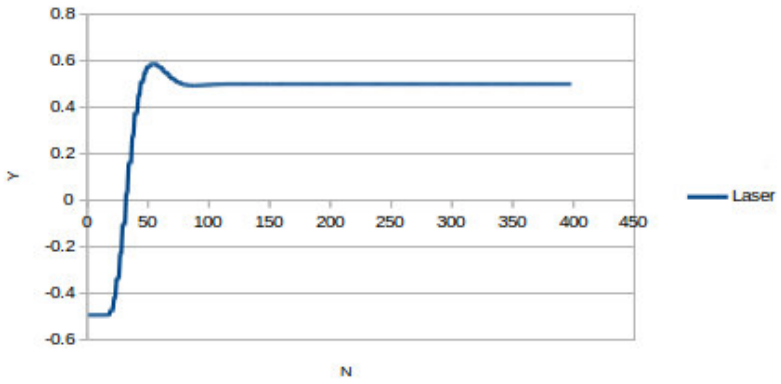


(b) Variação da posição em x do robô segundo algoritmo *Perfect Match* (eixo xx: iterações, eixo yy: deslocamento (m))

Figura 6.5: Localização baseada em laser, variação segundo x.

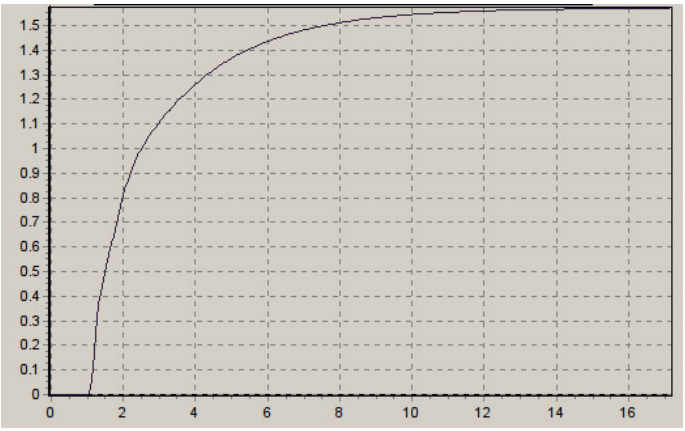


(a) Variação da posição real em *y* do robô (eixo *xx*: tempo (s), eixo *yy*: deslocamento (m))

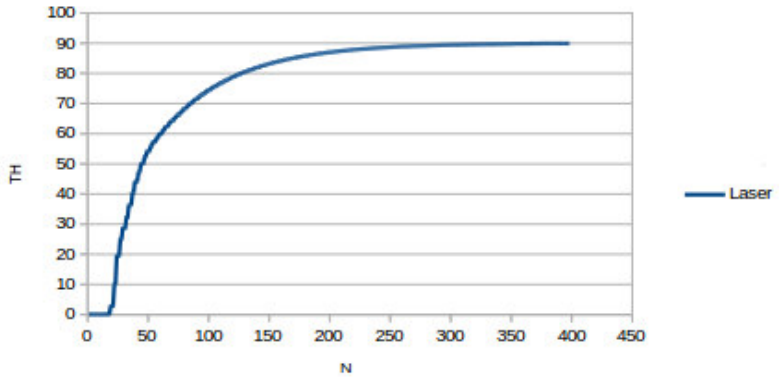


(b) Variação da posição em *y* do robô segundo algoritmo *Perfect Match* (eixo *xx*: iterações, eixo *yy*: deslocamento (m)).

Figura 6.6: Localização baseada em laser, variação segundo *y*



(a) Variação da posição real em θ do robô (eixo *xx*: tempo (s), eixo *yy*: rotação (rad))



(b) Variação da posição em θ (em graus) do robô segundo algoritmo *Perfect Match* (eixo *xx*: iterações, eixo *yy*: deslocamento (m))

Figura 6.7: Localização baseada em laser, variação segundo θ.

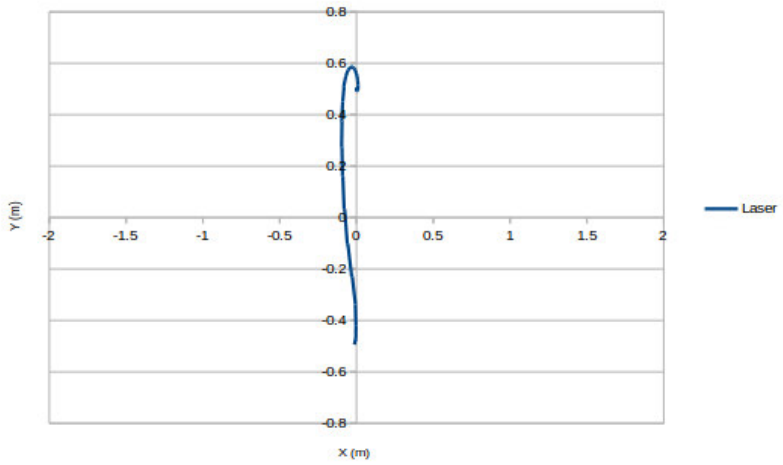


Figura 6.8: Representação do deslocamento do robô no mapa (eixo *xx*: deslocamento (m), eixo *yy*: deslocamento (m))

É bastante percetível que a localização absoluta, realizada através do *Perfect Match*, é muito precisa e viável, em ambiente de simulação.

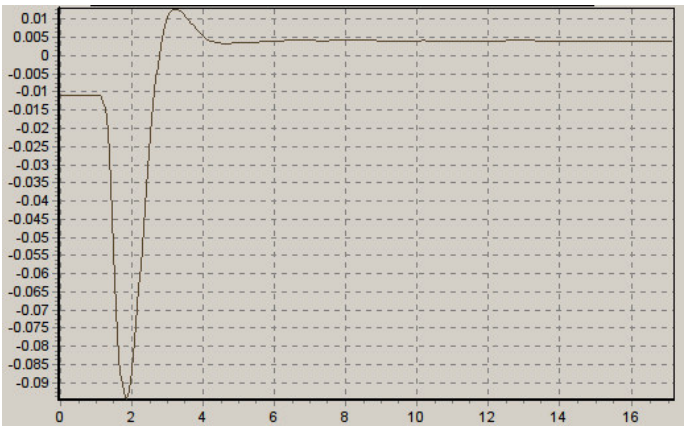
6.2.3 Validação do Filtro de Kalman Estendido

Após a validação da localização relativa e da localização absoluta, procedeu-se à validação do algoritmo de fusão sensorial. No entanto, dado o facto de, em ambiente de simulação, ambos os tipos de localização referidos serem bastante eficazes e precisos, não é possível verificar melhorias na localização do robô. É, no entanto, possível verificar se a implementação se encontra correta, analisando se a fusão sensorial não adicionou erros à localização.

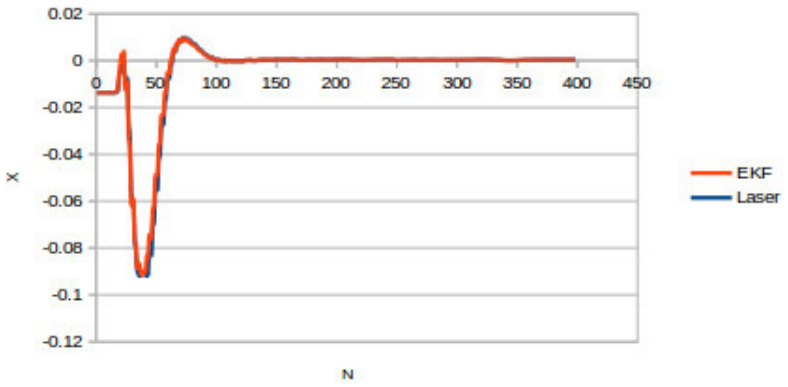
Os resultados são expostos na tabela 6.4 e nas figuras 6.9, 6.10, 6.11 e 6.12, representados pela linha cor de laranja. É também comparado o resultado obtido pela localização absoluta, representado nas mesmas figuras pela linha azul.

Tabela 6.4: Coordenadas (x, y, θ) obtidos por aplicação do Filtro de *Kalman* Estendido

x (m)	y (m)	θ (º)
-0.013675	-0.493749	0.112074
-0.013674	-0.493701	0.109344
⋮	⋮	⋮
0.007600	0.500106	67.691854
0.007592	0.496187	68.601777
⋮	⋮	⋮
0.000420	0.499706	89.904129
0.000428	0.499695	89.906981

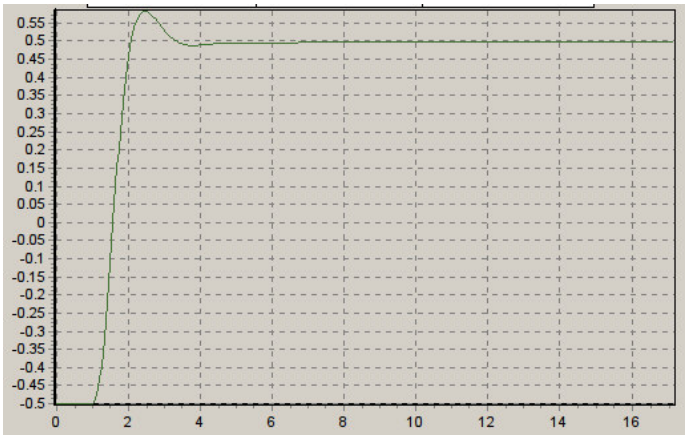


(a) Variação da posição real em x do robô (eixo xx: tempo (s), eixo yy: deslocamento (m))

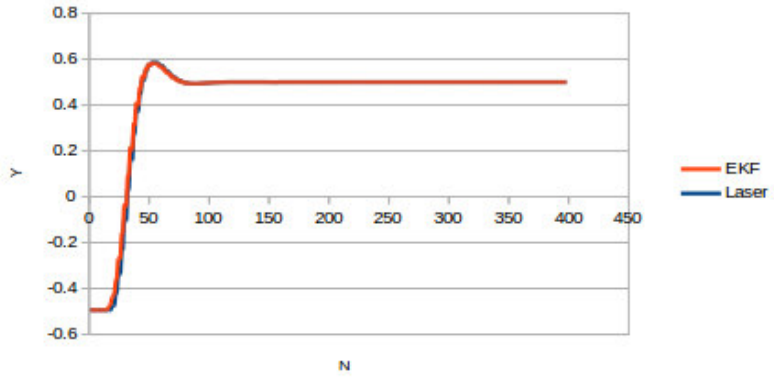


(b) Variação da posição em x do robô segundo algoritmo *Perfect Match* (eixo xx: iterações, eixo yy: deslocamento (m))

Figura 6.9: Localização baseada em laser, variação segundo x

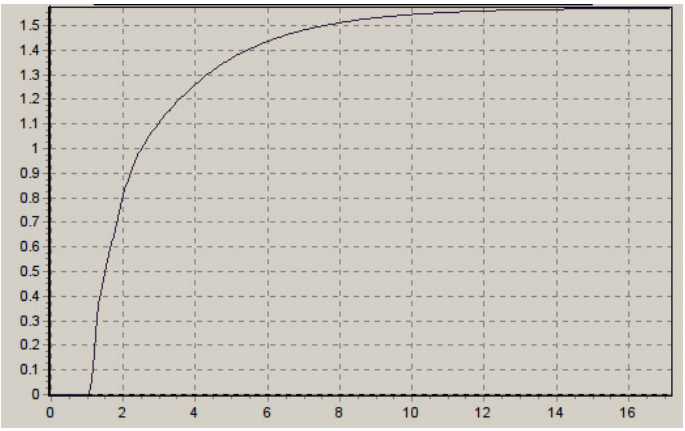


(a) Variação da posição real em *y* do robô (eixo *xx*: tempo (s), eixo *yy*: deslocamento (m))

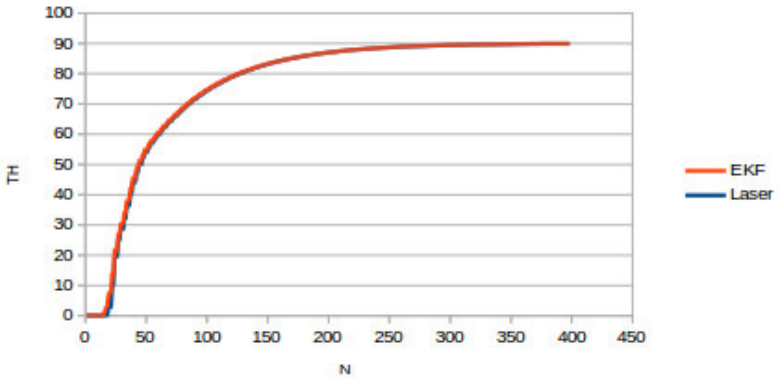


(b) Variação da posição em *y* do robô segundo algoritmo *Perfect Match* (eixo *xx*: iterações, eixo *yy*: deslocamento (m))

Figura 6.10: Localização baseada em laser, variação segundo *y*



(a) Variação da posição real em θ do robô (eixo *xx*: tempo (s), eixo *yy*: rotação (rad))



(b) Variação da posição em θ (em graus) do robô segundo algoritmo *Perfect Match* (eixo *xx*: iterações, eixo *yy*: deslocamento (m))

Figura 6.11: Localização baseada em laser, variação segundo θ

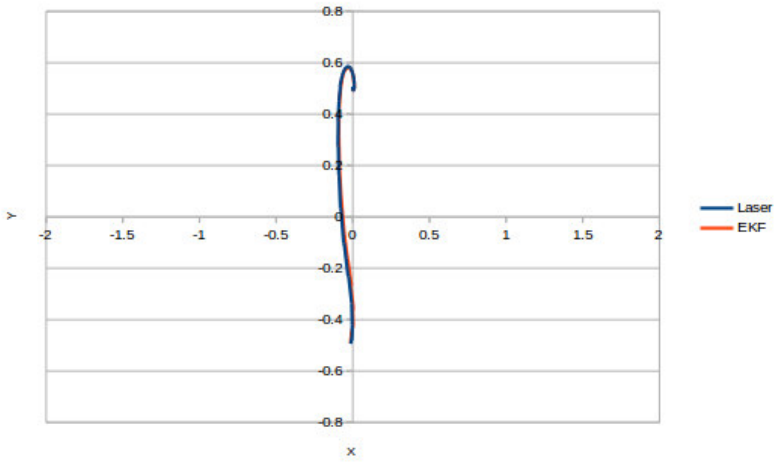


Figura 6.12: Representação do deslocamento do robô no mapa (eixo *xx*: deslocamento (m), eixo *yy*: deslocamento (m))

Verificou-se desta forma que a localização aplicando a fusão sensorial continuou viável.

6.3 Resultados no robô real

Nesta secção são apresentados e debatidos os resultados provenientes dos testes efetuado no robô prototipado. Para além de testes de movimentos de rotação e de translação, o teste final de validação será segundo uma trajetória que tem como objetivo o movimento do robô por várias partes do campo, validando a localização em toda a extensão do campo.

Esta trajetória, de 3.2 metros, encontra-se representada na Figura 6.13, sendo uma sequência de retas com a seguinte ordem: A - B - C - D - E.

Neste teste, o robô começa na posição $(-1.13, -0.5, 90^o)$ (em coordenadas no formato (x, y, θ)) e dirige-se para a posição $(1.13, -0.5, 0^o)$, definida como o ponto B. Ao chegar perto de B, este começa a dirigir-se para o ponto C, caracterizado pelas coordenadas $(1.13, 0.5, 90^o)$ e de seguida dirige-se até $(0, 0.5, 270^o)$ (ponto D). Para terminar, segue para o ponto E, nas coordenadas $(0, -0.5, 180^o)$, onde estabiliza.

Todos os testes foram efetuados num computador com um processador *Intel Core* $i5-460M$ (2.53GHz, 3MB L3 cache) e com 4GB de memória RAM.

6.3.1 Validação da odometria

Para uma correta validação da odometria no robô prototipado, foi necessária uma calibração da mesma. Esta calibração consistiu no ajuste dos parâmetros l e r. Esta calibração torna-se necessária uma vez que, por muito idênticas que as rodas sejam, nunca são bem iguais, nem a distância a que estas se encontram do centro do robô.

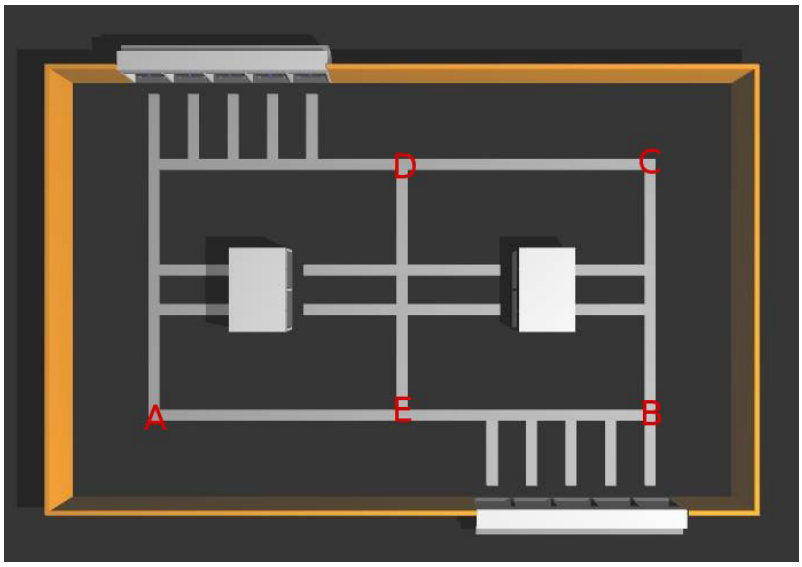


Figura 6.13: Pontos de referência da trajetória para teste e validação de algoritmos

Por forma a ajustar estes parâmetros, fez-se o robô efetuar dois movimentos distintos: movimentos rotacionais sobre si mesmo, e movimentos translacionais sobre uma linha definida. O teste dos movimentos rotacionais consistiu em fazer o robô rodar sobre si mesmo, para uma série de θs diferentes, analisando o erro e ajustando o valor de l até o erro ser quase nulo. Analogamente, procedeu-se a um ajuste dos valores do raio das rodas por forma a aproximar a trajetória do robô o máximo possível da estipulada. No entanto, uma vez que odometria consiste na integração incremental do movimento das rodas ao longo do tempo, o erro associado a esta medição é desta forma acumulado, revelando haver um maior desvio da sua rota quanto maior for a sua distância ao seu ponto inicial. Os novos valores de para estas variáveis são apresentados na tabela 6.5.

Tabela 6.5: Novos parâmetros calibrados para uso de odometria

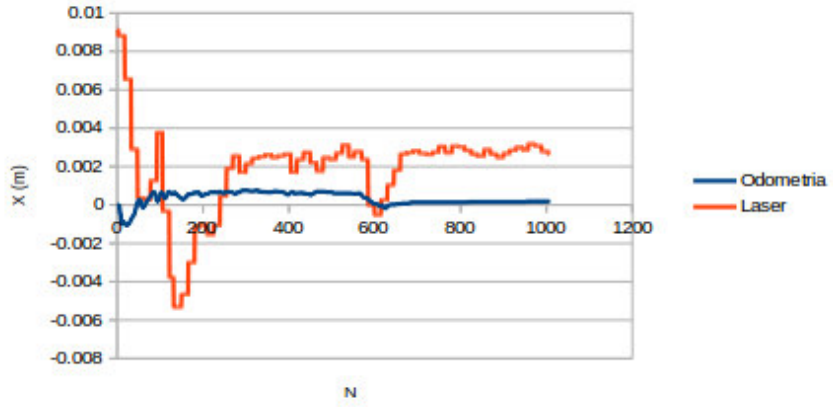
Nova parametrização	
l	0.111 m
r	0.032 m

Após os ajustes feitos, foram registados um exemplo de uma rotação e de uma translação. Estes dados foram comparados com os fornecidos pelo algoritmo de *Perfect Match*. Apesar da localização absoluta estar também sujeita a erros, estes não são da natureza dos da odometria, sendo desta forma uma aproximação mais exata das trajetórias realizadas.

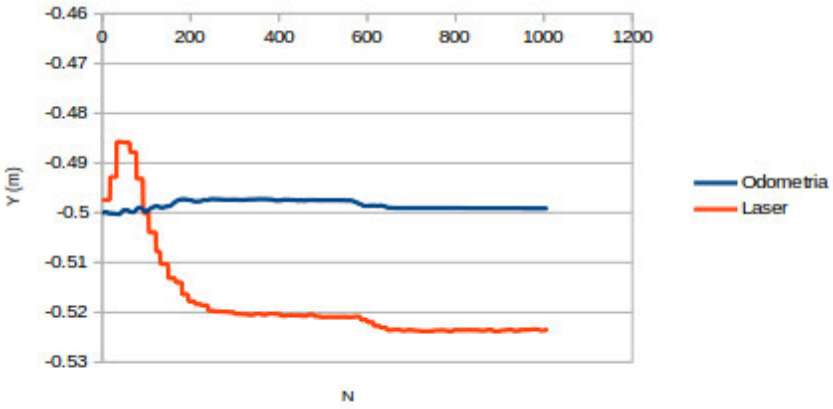
Nos gráficos que se seguem, exibidos na Figura 6.14, é possível verificar a evolução das coordenadas x, y e θ ao longo das várias iterações do programa (N), no teste do movimento rotacional do robô. Está também presente na tabela 6.6 excertos dos pontos usados para o desenho dos gráficos. Este teste consiste na rotação desde o ponto $x = 0$m, $y = -0.5$m e $\theta = 0^{o}$ até $x = 0$m, $y = -0.5$m e $\theta = 90^{o}$.

Tabela 6.6: Coordenadas (x, y, θ) obtidas em movimento rotacional por aplicação por cálculo da odometria e comparação com resultados de *Perfect Match*

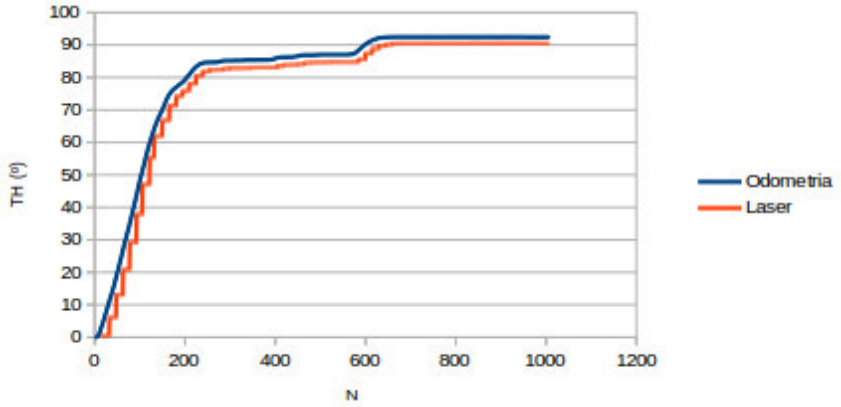
Odometria			Laser		
x (m)	y (m)	θ (º)	x (m)	y (m)	θ (º)
0	-0.5	0	0.009127	-0.497458	-0.208739
-0.000234	-0.499864	0.069877	0.008821	-0.497431	-0.193105
⋮	⋮	⋮	⋮	⋮	⋮
0.000648	-0.498972	65.457653	-0.005324	-0.510322	61.924940
0.000616	-0.498941	65.820204	-0.005324	-0.510322	61.924940
⋮	⋮	⋮	⋮	⋮	⋮
0.000179	-0.4991226	92.452869	0.002863	-0.523555	90.507956
0.000188	-0.499137	92.448482	0.002659	-0.523461	90.505709



(a) Evolução da coordenada x ao longo das iterações N



(b) Evolução da coordenada y ao longo das iterações N



(c) Evolução da coordenada θ ao longo das iterações N

Figura 6.14: Odometria: Representação da evolução da coordenadas ao longo do movimento rotacional

É notório o erro entre o calculado pelas medições do laser e pela odometria, no entanto este não é muito elevado. Parte do erro entre as duas formas de localização deve-se também ao erro da posição inicial do robô. No entanto, ao longo do movimento das rodas é visível um acrescento do erro. Relativamente à posição final, segundo o algoritmo de localização absoluta, o robô estará desfasado 0.64 centímetros segundo x, 2.6 centímetros segundo y e 0.72º segundo θ.

Em seguida são apresentados, na Figura 6.16, o teste ao movimento translacional. Este teste consiste na translação segundo a reta definida pelo ponto inicial $x = 0$m, $y = -0.5$m e $\theta = 90^o$ até $x = 0$m, $y = 0.5$m e $\theta = 90^o$. É também possível ver no gráfico da Figura 6.15 a representação deste movimento no campo. Está também presente na tabela 6.7 excertos dos pontos usados para o desenho dos gráficos.

Tabela 6.7: Coordenadas (x, y, θ) obtidas em movimento translacional por aplicação por cálculo da odometria e comparação com resultados de *Perfect Match*

Odometria			Laser		
x (m)	y (m)	θ (º)	x (m)	y (m)	θ (º)
0	-0.5	90	-0.005180	-0.495534	89.627661
0.00011	-0.499809	89.943220	-0.004756	-0.495577	89.625844
⋮	⋮	⋮	⋮	⋮	⋮
-0.000114	0.026426	90.293447	-0.035344	0.131511	88.731998
0.000051	0.068365	90.123302	-0.036527	0.179974	88.434256
⋮	⋮	⋮	⋮	⋮	⋮
0.000708	0.504779	89.830241	-0.024310	0.730227	84.805226
0.000708	0.504779	89.830241	-0.024308	0.730063	84.776717

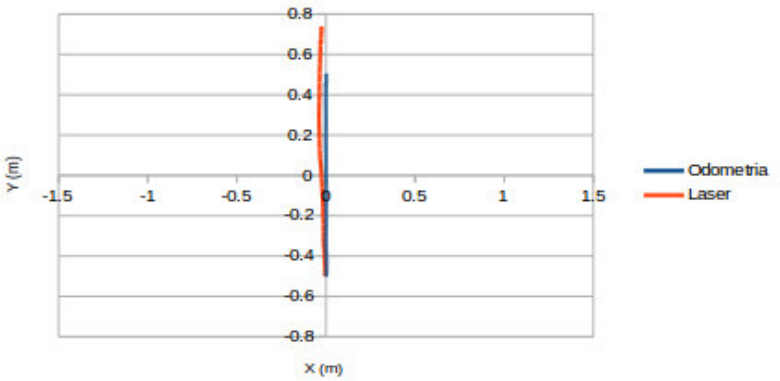
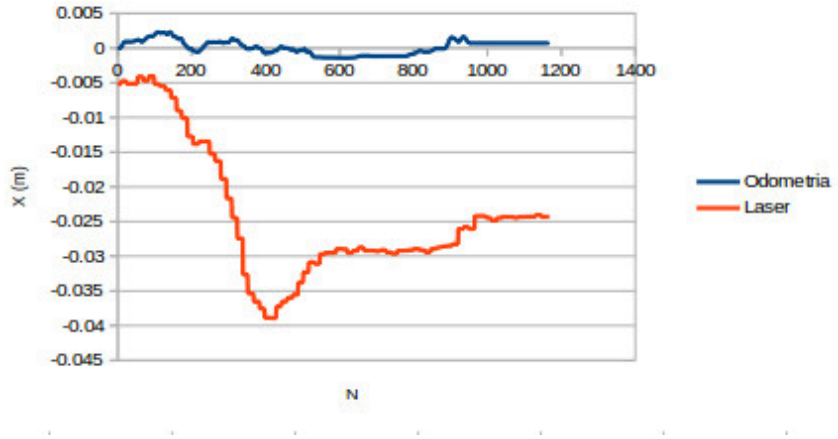
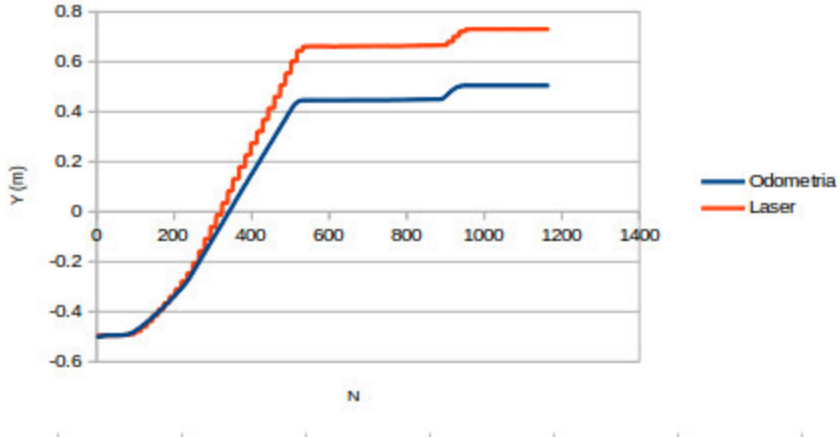


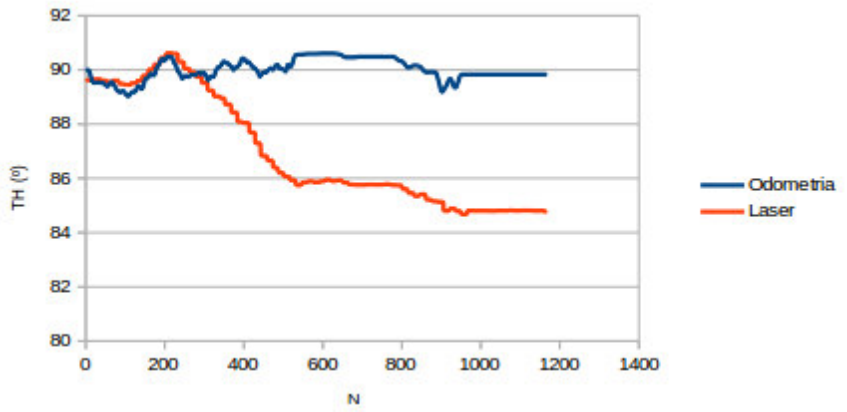
Figura 6.15: Odometria: Representação do deslocamento do robô no mapa



(a) Evolução da coordenada x ao longo das iterações N



(b) Evolução da coordenada y ao longo das iterações N



(c) Evolução da coordenada θ ao longo das iterações N

Figura 6.16: Odometria: Representação da evolução da coordenadas ao longo do movimento translacional

Encontra-se bem descrito o erro associado a este movimento. Relativamente à posição final, e recorrendo ao mesmo método de comparação, o robô estará desfasado 1.9 centímetros segundo x, 23.45 centímetros segundo y e 4.85º segundo θ.

Por fim validou-se a odometria na trajetória definida em 6.13. Fez-se, no entanto, dois testes baseados nesta trajetória: um só com movimentos translacionais e rotação no fim de todo o percurso (teste 1), e o segundo com movimentos de rotação e translação ao mesmo tempo (teste 2). O resultado é visível nas figuras 6.17 e 6.18 e também na tabela 6.8 (excertos dos pontos usados para o desenho dos gráficos, onde cada conjunto de dois pontos da tabela representa, aproximadamente, a transição de reta, representando os dois primeiros o ponto A e os dois últimos o ponto E) é referente ao primeiro destes testes.

Tabela 6.8: Coordenadas (x, y, θ) obtidas em teste final por aplicação por cálculo da odometria e comparação com resultados de *Perfect Match* - teste 1

Odometria			Laser		
x (m)	y (m)	θ (º)	x (m)	y (m)	θ (º)
-1.13	-0.51	90	-1.127408	-0.514247	89.198828
-1.129729	-0.509530	89.860201	-1.127408	-0.514247	89.198828
⋮	⋮	⋮	⋮	⋮	⋮
1.132776	-0.491959	84.414263	0.989508	-0.466456	86.275165
1.133759	-0.490720	84.356107	0.989508	-0.466456	86.275165
⋮	⋮	⋮	⋮	⋮	⋮
1.127537	0.503607	86.879874	1.031954	0.310327	79.746017
1.125905	0.507259	87.194399	1.036073	0.388244	79.857451
⋮	⋮	⋮	⋮	⋮	⋮
-0.007986	0.500691	75.232227	0.150079	0.502880	70.707873
-0.012853	0.500336	75.028337	0.066622	0.507331	71.050974
⋮	⋮	⋮	⋮	⋮	⋮
-0.001381	-0.498053	179.650344	0.016424	-0.419788	153.080371
-0.001381	-0.498053	179.650344	0.016713	-0.420092	153.111435

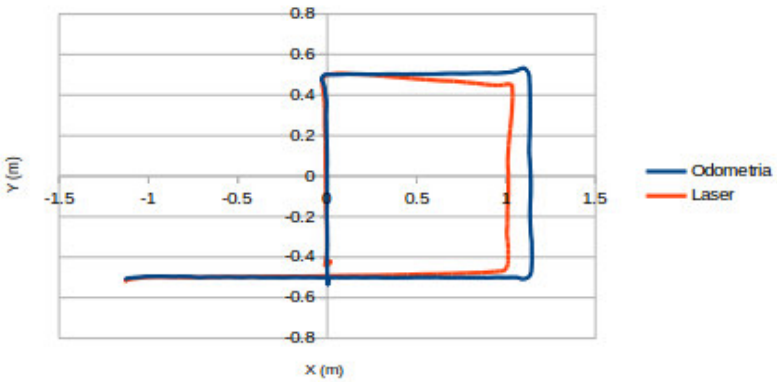
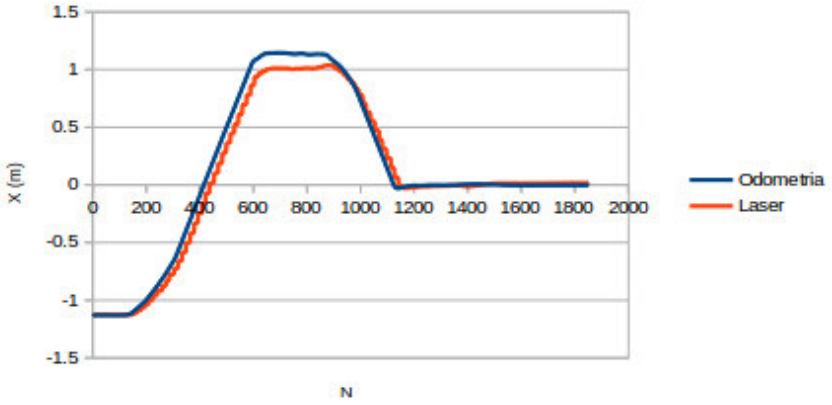
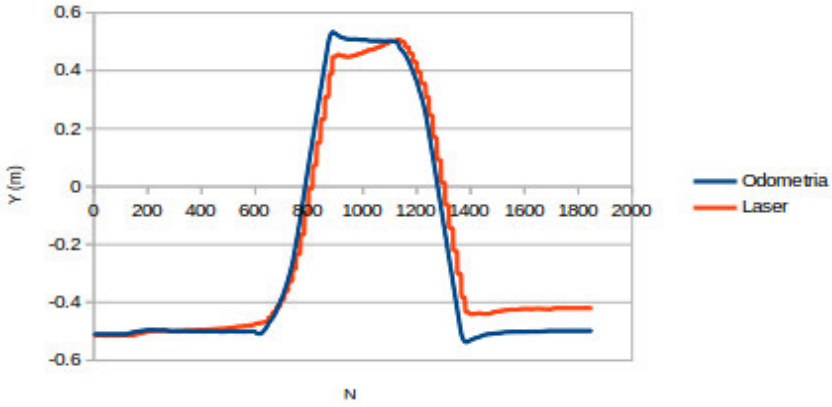


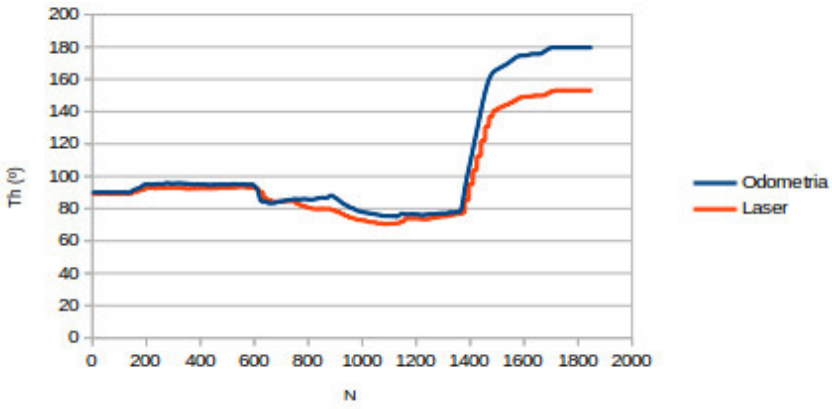
Figura 6.17: Odometria: Pontos de referência da trajetória para teste de validação de algoritmos - teste 1



(a) Evolução da coordenada x ao longo das iterações N



(b) Evolução da coordenada y ao longo das iterações N



(c) Evolução da coordenada θ ao longo das iterações N

Figura 6.18: Odometria: Representação da evolução da coordenadas ao longo do movimento do teste de validação de algoritmos - teste 1

Na posição final, robô está desfasado 1.67 centímetros segundo x, 7.99 centímetros segundo y e 26.88º segundo θ do que é estipulado.

Nas figuras 6.19 e 6.20 e também na tabela 6.9 é possível analisar o resultado ao teste, com movimentos translacionais e rotacionais simultâneos.

Tabela 6.9: Coordenadas (x, y, θ) obtidas em teste final por aplicação por cálculo da odometria e comparação com resultados de *Perfect Match* - teste 2

Odometria			Laser		
x (m)	y (m)	θ (º)	x (m)	y (m)	θ (º)
-1.13	-0.5	90	-1.124221	-0.505765	89.291370
-1.129187	-0.499883	90.104784	-1.124221	-0.505765	89.291370
⋮	⋮	⋮	⋮	⋮	⋮
1.137812	-0.502331	0.022914	1.001882	-0.309204	12.414256
1.137811	-0.500909	-0.344088	1.002381	-0.306322	11.755495
⋮	⋮	⋮	⋮	⋮	⋮
1.110314	0.461280	72.908308	0.809812	0.449363	85.788979
1.108980	0.467886	74.044158	0.809812	0.449363	85.788979
⋮	⋮	⋮	⋮	⋮	⋮
-0.005902	0.506588	-86.879040	-0.074082	0.065321	-24.548895
-0.006728	0.504957	-87.350827	-0.071171	0.059854	-25.735972
⋮	⋮	⋮	⋮	⋮	⋮
-0.000645	-0.501447	-178.888871	0.557064	-0.595657	170.083568
-0.000645	-0.501447	-178.888871	0.556751	-0.595482	170.086723

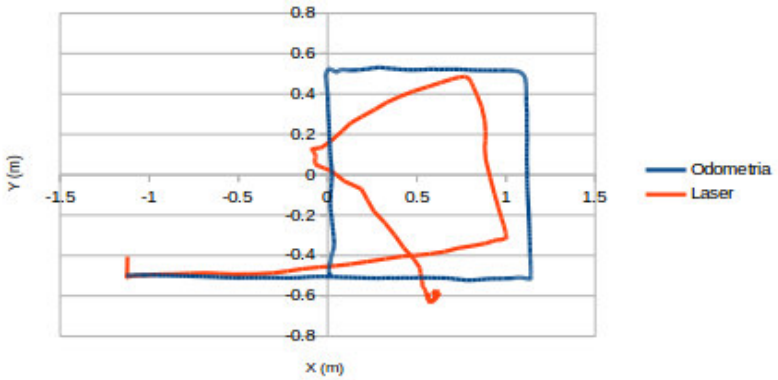
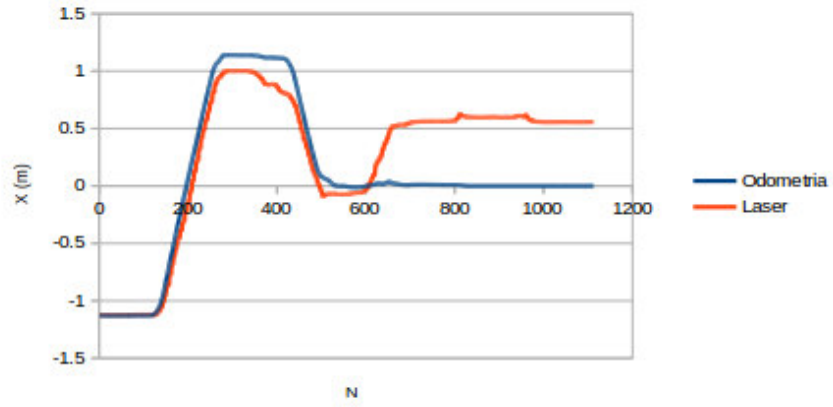
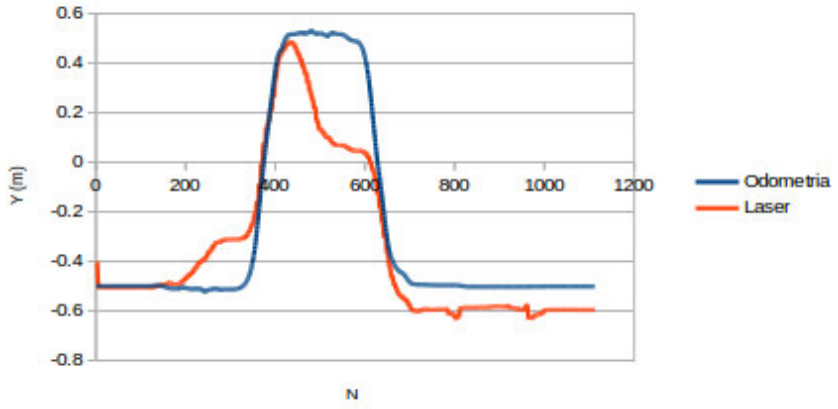


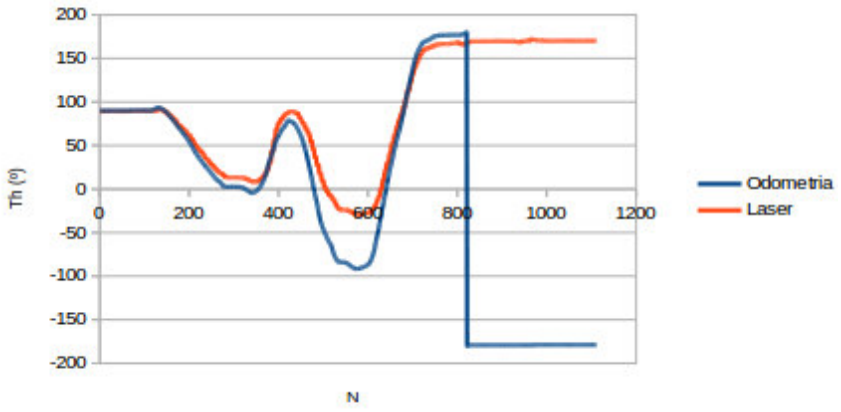
Figura 6.19: Odometria: Pontos de referência da trajetória para teste de validação de algoritmos - teste 2



(a) Evolução da coordenada x ao longo das iterações N



(b) Evolução da coordenada y ao longo das iterações N



(c) Evolução da coordenada θ ao longo das iterações N

Figura 6.20: Odometria: Representação da evolução da coordenadas ao longo do movimento do teste de validação de algoritmos - teste 2

Na posição final, robô está desfasado 55.6 centímetros segundo x, 9.5 centímetros segundo y e 9.91º segundo θ do que é estipulado. O erro aumentou substancialmente em relação ao teste

anterior, sendo isto uma consequência do controlo simultâneo das três rodas do robô, ao invés do que aconteceu no teste anterior, onde só se controlavam duas a cada instante.

Considerando que se trata da localização exclusivamente baseada na odometria, considera-se que está calibrada e apta a ser inserida como parâmetro de entrada no Filtro de *Kalman* Estendido, ajustando os valores que a localização absoluta providenciar.

Ao longo dos testes efetuados, verificou-se que o tempo de amostragem das velocidades de cada roda é aproximadamente 40ms. A distribuição dos valores de amostragem de um dos testes feitos está representada em 6.21.

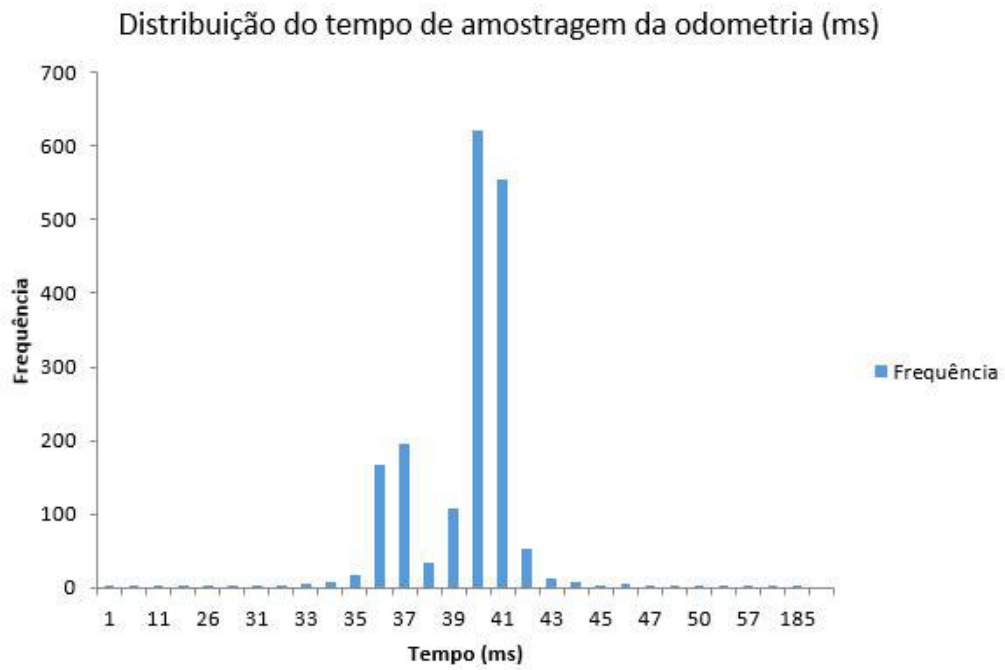


Figura 6.21: Distribuição dos tempos de amostragem da odometria

6.3.2 Validação do Perfect Match

Dada a frequência de aquisição do laser utilizado, 5Hz, a localização baseada exclusivamente no algoritmo *Perfect Match* tem influência no trajeto efetuado pelo robô, na medida em que dado este só atualiza o robô do seu estado a cada, aproximadamente, 200ms. Isto tem o efeito negativo de o robô durante esse intervalo de tempo achar que se encontra no estado anterior, já afastado do real, provocando o controlo dos motores a reagir de forma menos correta e causando alguns *overshoots* ao longo da trajetória. Este efeito pode verificado ao longo dos dois testes efetuados (teste 1 e teste 2). Nas figuras 6.22 e 6.23 e tabela 6.10 encontram-se expostos os resultados ao teste 1, nas figuras 6.24 e 6.25, bem como na tabela 6.11 encontram-se os resultados ao teste 2 onde, tal como anteriormente, cada conjunto de dois pontos representa a mudança de reta na trajetória de testes.

Tabela 6.10: Coordenadas (x, y, θ) obtidas em teste final por aplicação por cálculo de *Perfect Match* - teste 1

Laser		
x (m)	y (m)	θ (°)
-1.137152	-0.545454	88.232432
-1.137152	-0.545454	88.232432
⋮	⋮	⋮
1.129064	-0.377543	88.619526
1.130316	-0.333995	88.752264
⋮	⋮	⋮
1.125800	0.483562	87.876907
1.123682	0.520920	88.016639
⋮	⋮	⋮
-0.031301	0.487957	83.500422
-0.000793	0.486974	84.504473
⋮	⋮	⋮
0.002613	-0.496172	-179.762123
0.002951	-0.495348	-179.721798

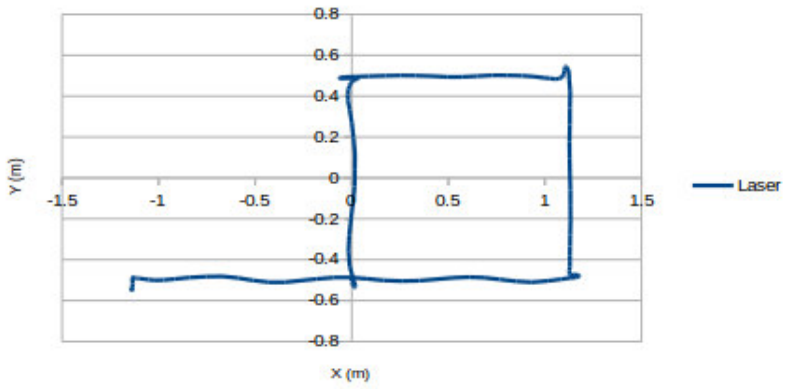
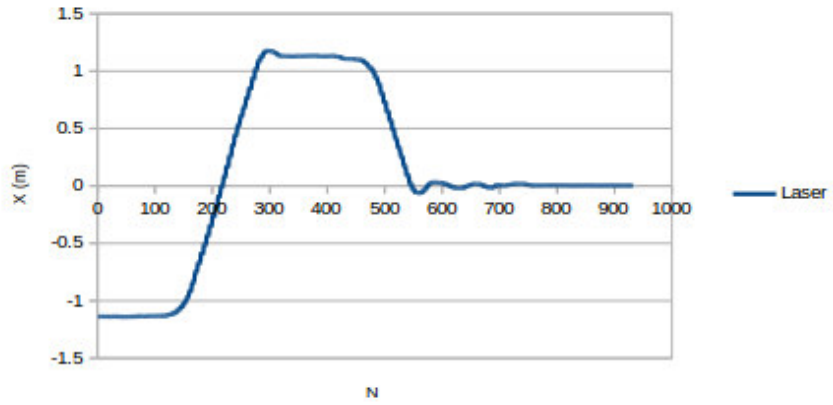
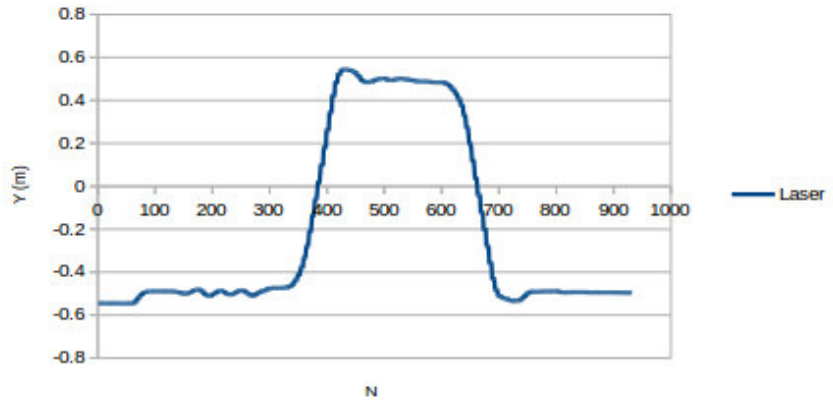


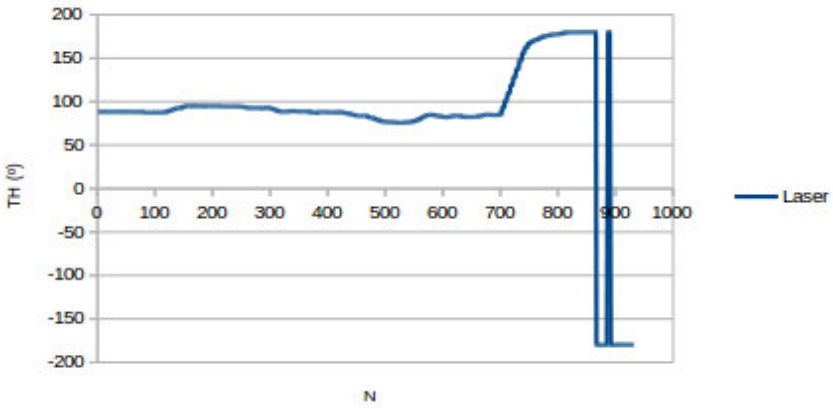
Figura 6.22: *Perfect Match:* Pontos de referência da trajetória para teste de validação de algoritmos - teste 1



(a) Evolução da coordenada x ao longo das iterações N



(b) Evolução da coordenada y ao longo das iterações N



(c) Evolução da coordenada θ ao longo das iterações N

Figura 6.23: *Perfect Match:* Representação da evolução da coordenadas ao longo do movimento do teste de validação de algoritmos - teste 1

Relativamente ao ponto final da trajetória, é possível verificar que o erro é bem menor do que obtido graças à odometria. Desta forma é possível constatar que o erro segundo a variável x é de $2.8mm$, segundo y é de $4.1mm$ e apenas 0.25^{o} de erro em θ. Constata-se que, apesar dos *overshoots*

verificados, a localização é mais precisa do que a da odometria. Este facto era expectável, dado ao maior tempo de aquisição para a implementação e maior precisão da localização absoluta.

Tabela 6.11: Coordenadas (x, y, θ) obtidas em teste final por aplicação por cálculo de *Perfect Match* - teste 2

Laser		
x (m)	y (m)	θ (^{o})
-1.113585	-0.495805	89.287231
-1.113899	-0.495722	89.306301
⋮	⋮	⋮
1.136990	-0.493003	3.014319
1.136896	-0.484342	1.144346
⋮	⋮	⋮
1.103940	0.485268	55.229248
1.091368	0.522700	63.661092
⋮	⋮	⋮
-0.005033	0.564918	-86.351083
0.002847	0.556454	-91.959932
⋮	⋮	⋮
-0.001432	-0.499917	-178.304069
-0.001279	-0.500107	-178.324409

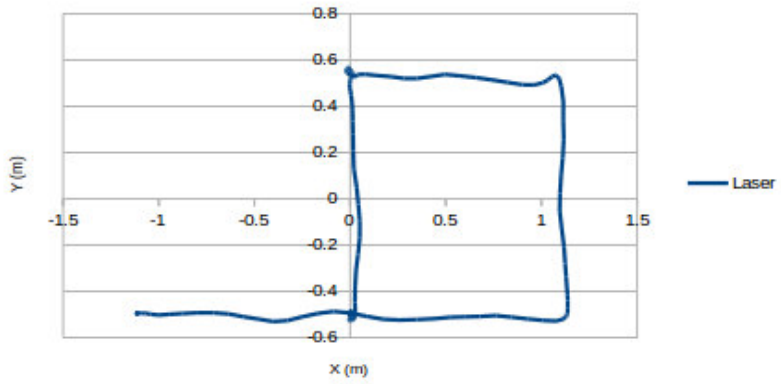
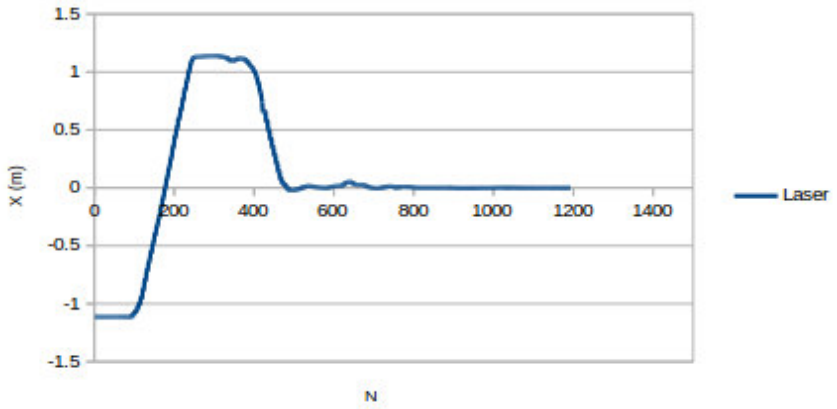
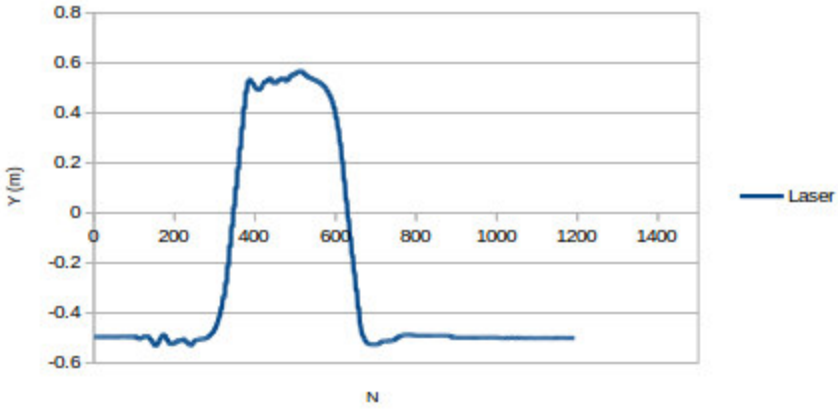


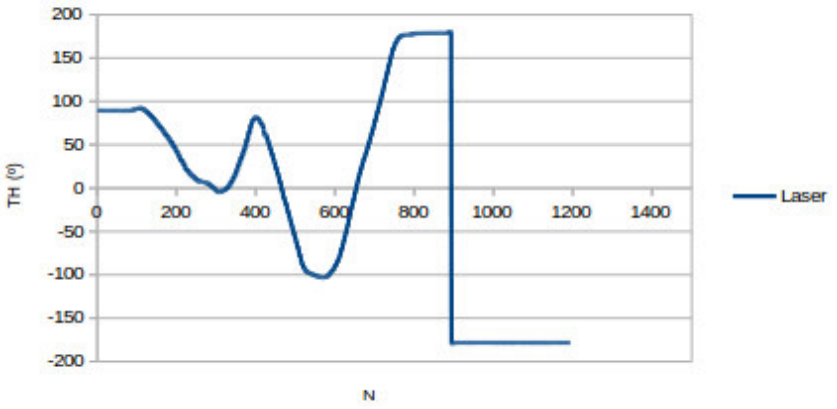
Figura 6.24: *Perfect Match:* Pontos de referência da trajetória para teste de validação de algoritmos - teste 2



(a) Evolução da coordenada x ao longo das iterações N



(b) Evolução da coordenada y ao longo das iterações N



(c) Evolução da coordenada θ ao longo das iterações N

Figura 6.25: *Perfect Match:* Representação da evolução da coordenadas ao longo do movimento do teste de validação de algoritmos - teste 2

Verifica-se que o erro segundo a variável x é de $1.3mm$, segundo y é de $0.1mm$ e apenas 1.68^{o} de erro em θ.

Comparativamente aos resultados obtidos pela odometria, e tal como já era expectável, o erro da localização absoluta é bastante inferior à da localização relativa.

Durante os testes foram registados os tempos de amostragem e processamento associados à utilização deste algoritmo. Quanto ao tempo de amostragem médio, de facto comprovou-se o referenciado pelos fabricantes do sistema do laser, sendo este de 199.329*ms*. Tal como foi feito para a odometria, na Figura 6.26 encontra-se representada a distribuição dos tempos de amostragem do laser, registados ao longo de um dos testes.

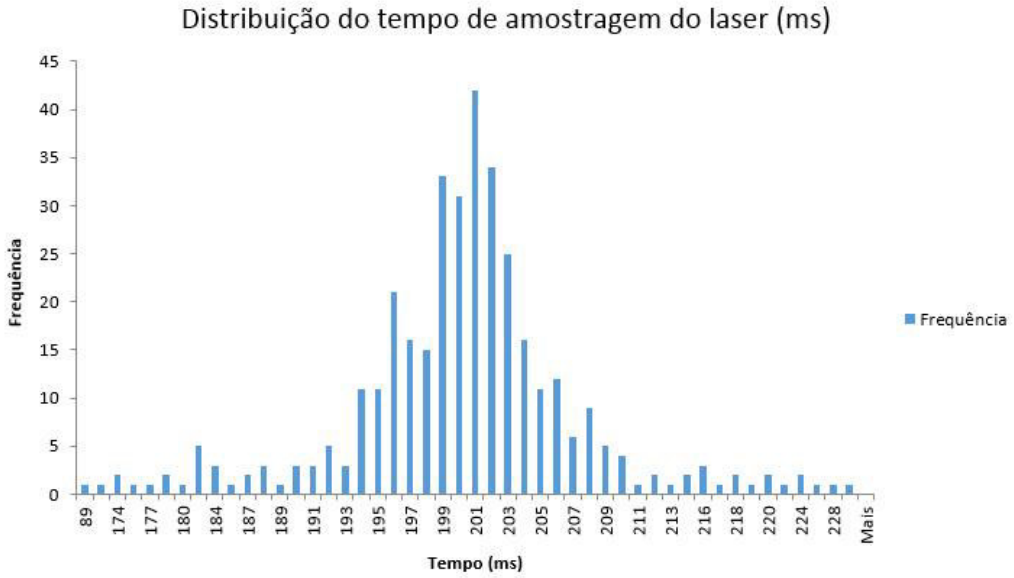


Figura 6.26: Distribuição dos tempos de amostragem do laser

6.3.3 Validação do Filtro de Kalman Estendido

O objetivo deste teste foi validar o algoritmo e avaliar o efeito do EKF na localização do robô, analisando o seu comportamento com diversos graus de confiança para ambas as variáveis de entrada do algoritmo.

Os testes efetuados no percurso referido em 6.3, e estudou-se a influência de diversos valores nas matrizes de covariância R e Q.

$$R = \begin{bmatrix} R_{11} & 0 & 0 \\ 0 & R_{22} & 0 \\ 0 & 0 & R_{33} \end{bmatrix} \tag{6.1}$$

$$Q = \begin{bmatrix} Q_{11} & 0 & 0 \\ 0 & Q_{22} & 0 \\ 0 & 0 & Q_{33} \end{bmatrix} \tag{6.2}$$

Primeiramente efetuou-se o teste, que se vai denominar por *teste 2A* com os seguintes valores de covariâncias: $R_{11} = R_{22} = R_{33} = 1$ e $Q_{11} = Q_{22} = 1000$ e $Q_{33} = 100000$. O motivo pela qual o valor de Q_{33} é superior aos restantes valores da diagonal da mesma matriz está relacionado com a influência que a variável θ tem nas restantes componentes da posição, e o facto de essa variável ser muito errónea. Como é de fácil interpretação, aquando a fusão sensorial entre a odometria e a localização absoluta, o EKF terá a odometria como variável menos viável.

Tabela 6.12: Coordenadas (x, y, θ) obtidas em teste final por aplicação por cálculo de EKF: teste 2A

Fusão Sensorial		
x (m)	y (m)	θ (^{o})
-1.124026	-0.503323	89.757154
-1.123910	-0.503404	89.680293
⋮	⋮	⋮
1.139280	-0.500652	3.968828
1.139168	-0.499031	3.549433
⋮	⋮	⋮
1.087946	0.503055	69.122412
1.083818	0.506575	70.135670
⋮	⋮	⋮
0.002116	0.499641	-87.680779
0.001361	0.498202	-88.100174
⋮	⋮	⋮
-0.005349	-0.502192	-179.875096
-0.005300	-0.502124	-179.871911

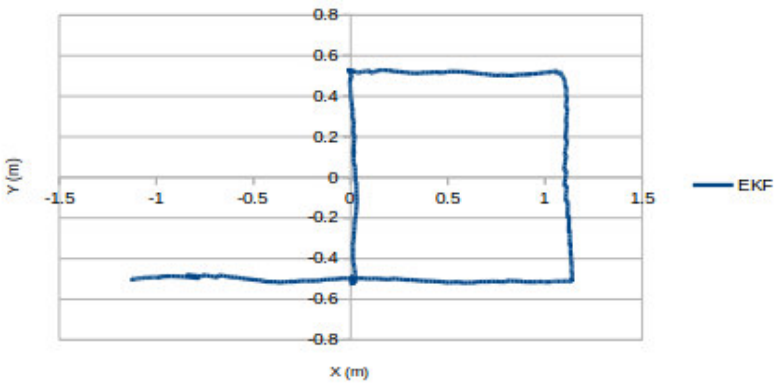
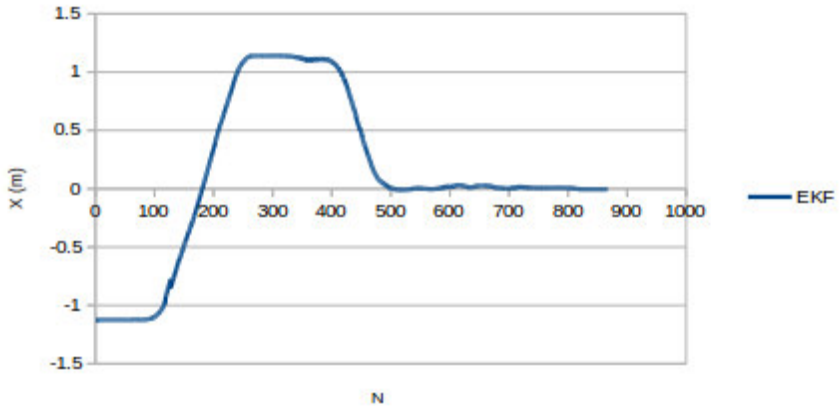
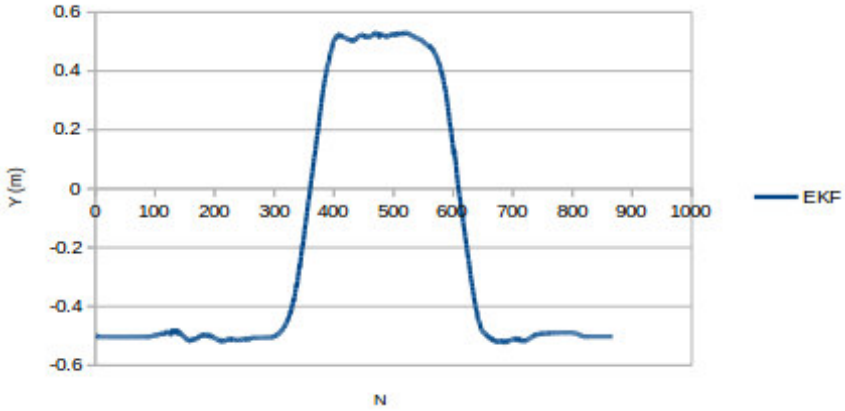


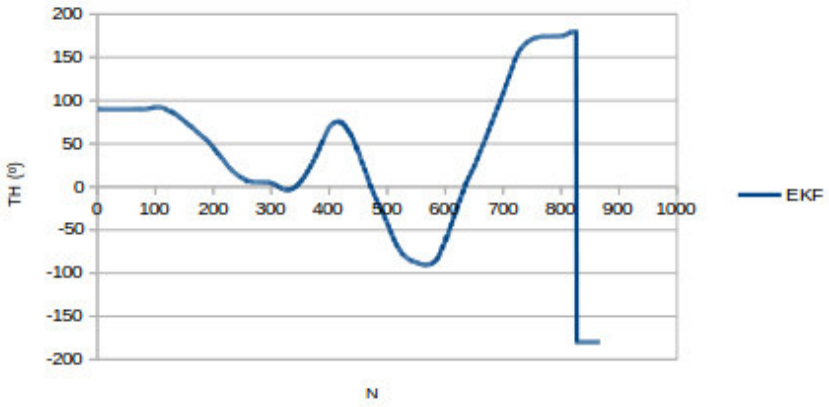
Figura 6.27: EKF: Pontos de referência da trajetória para teste de validação de algoritmos - teste 2A



(a) Evolução da coordenada x ao longo das iterações N



(b) Evolução da coordenada y ao longo das iterações N



(c) Evolução da coordenada θ ao longo das iterações N

Figura 6.28: EKF: Representação da evolução da coordenadas ao longo do movimento do teste de validação de algoritmos - teste 2A

É notória a tendência ondulada do movimento do robô, característico da localização baseada no laser, tal como se pode verificar em 6.3.2. Isto verifica-se dado ao grau de viabilidade associado

à localização absoluta neste teste.

Em seguida efetuou-se outro teste, por forma a verificar o comportamento da localização baseada em EKF, onde a componente do laser é bastante mais errónea do que a da odometria. Para tal adotaram-se os seguintes valores de covariâncias: $R_{11} = R_{22} = R_{33} = 1000$ e $Q_{11} = Q_{22} = 10$ e $Q_{33} = 100$. O seguinte teste é denominado por *teste 2B*.

Tabela 6.13: Coordenadas (x, y, θ) obtidas em teste final por aplicação por cálculo de EKF: teste 2B

Fusão Sensorial			Laser		
x (m)	y (m)	θ (º)	x (m)	y (m)	θ (º)
-1.11	-0.5	90.0	-1.102715	-0.483919	88.579999
-1.109789	-0.499533	89.941293	-1.102619	-0.484056	88.556565
⋮	⋮	⋮	⋮	⋮	⋮
1.131785	-0.492948	4.900343	1.129991	-0.478369	5.782615
1.131761	-0.492678	4.830401	1.130758	-0.477975	5.563156
⋮	⋮	⋮	⋮	⋮	⋮
1.109813	0.474926	69.795041	1.090655	0.482255	68.897027
1.108788	0.480864	71.350286	1.083540	0.500016	72.767886
⋮	⋮	⋮	⋮	⋮	⋮
0.003001	0.474264	-69.579969	0.016965	0.440781	-69.949142
0.002483	0.471192	-70.383916	0.014091	0.414985	-72.416726
⋮	⋮	⋮	⋮	⋮	⋮
0.000910	-0.498609	179.137412	0.001291	-0.497982	179.038150
0.000921	-0.498590	179.132193	0.001701	-0.497701	179.081577

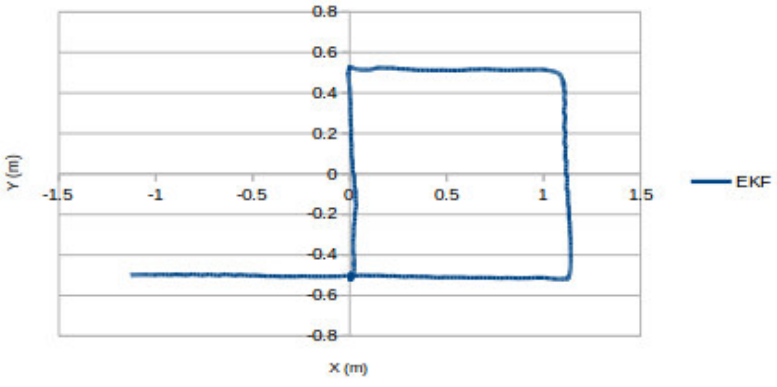
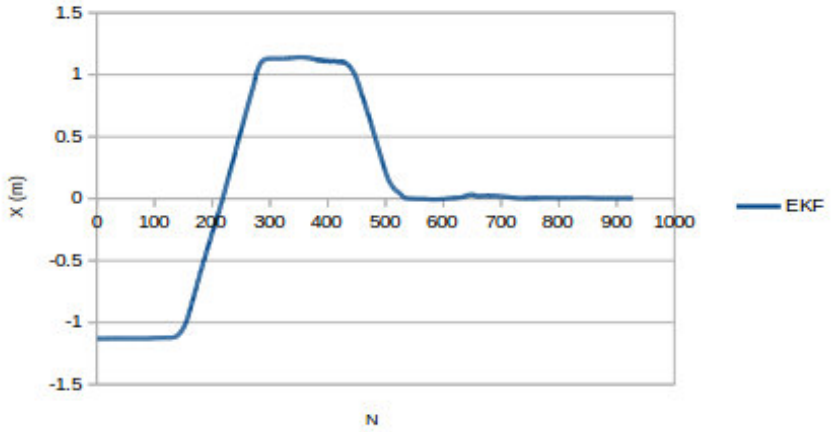
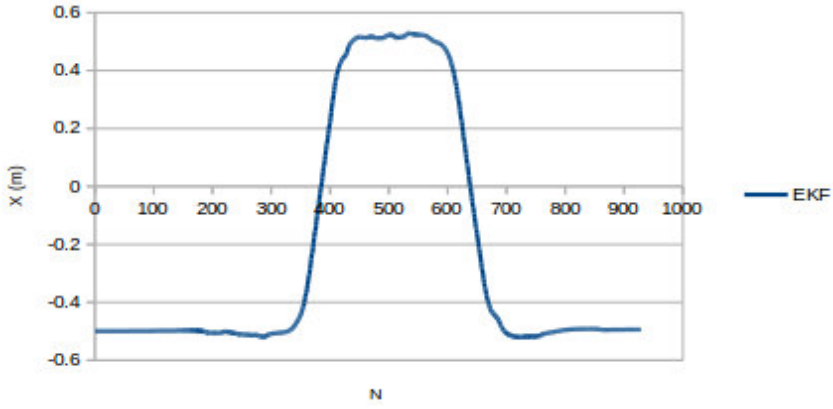


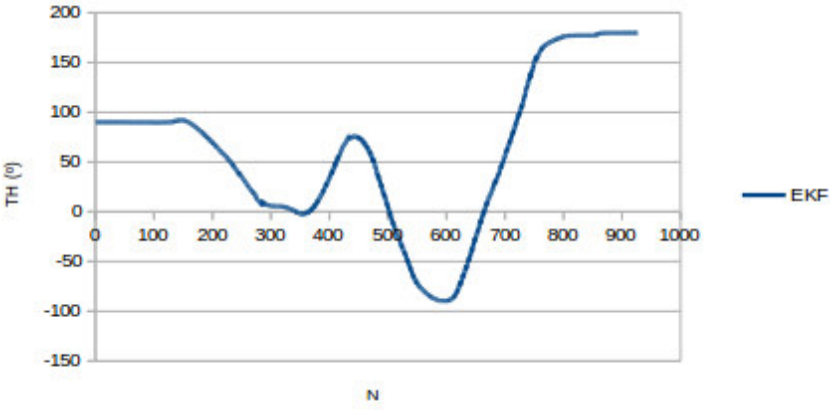
Figura 6.29: EKF: Pontos de referência da trajetória para teste de validação de algoritmos - teste 2B



(a) Evolução da coordenada x ao longo das iterações N



(b) Evolução da coordenada y ao longo das iterações N



(c) Evolução da coordenada θ ao longo das iterações N

Figura 6.30: EKF: Representação da evolução da coordenadas ao longo do movimento do teste de validação de algoritmos - teste 2B

Como era de prever, a trajetória agora representada tem um grau de semelhança com a apresentada em 6.18 e 6.17. No entanto, dado o carácter pouco preciso da localização relativa, fez-se a

comparação do gráfico apresentado em 6.29 com uma localização a parte feita através do algoritmo *Perfect Match*. O resultado é apresentado em

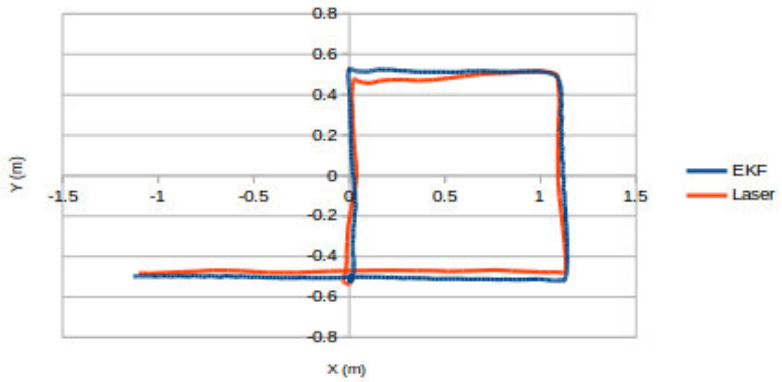


Figura 6.31: EKF: Comparação entre trajetória para teste de validação de algoritmos obtida por EKF com a de laser- teste 2B

Como é possível verificar, e assumindo o sistema de localização absoluta adotado é bastante preciso, é bem visível a discrepância entre o que é o resultado do Filtro de *Kalman* Estendido com estes valores de R e Q com o que é considerado um valor mais próximo do real.

Ao longo de vários testes, o valor para as matrizes R e Q que se verificou ser mais eficaz para a estimação do estado foi $R_{11} = R_{22} = R_{33} = 100$ e $Q_{11} = Q_{22} = 1000$ e $Q_{33} = 100000$ (*teste 2C*).

Tabela 6.14: Coordenadas (x, y, θ) obtidas em teste final por aplicação por cálculo de EKF: teste 2C

Fusão Sensorial		
x (m)	y (m)	θ (°)
-1.13	-0.5	90.0
-1.129721	-0.500125	90.001940
⋮	⋮	⋮
1.139814	-0.490726	1.509256
1.139520	-0.488499	1.037470
⋮	⋮	⋮
1.088737	0.513470	68.791945
1.084085	0.514395	69.421166
⋮	⋮	⋮
-0.006526	0.492391	-89.801844
-0.006892	0.490690	-90.361123
⋮	⋮	⋮
-0.002680	-0.498924	179.848929
-0.002622	-0.498958	179.866479

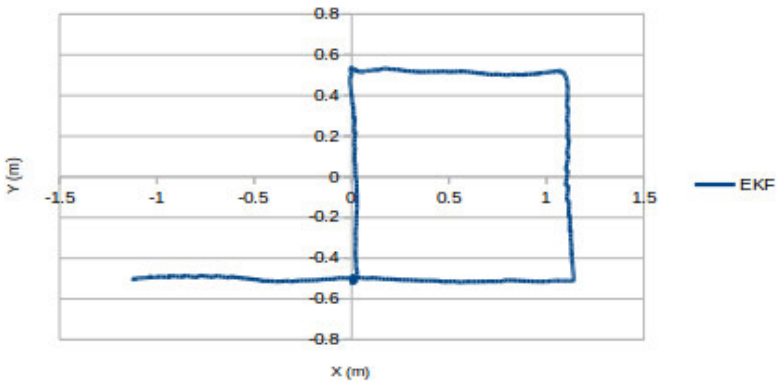
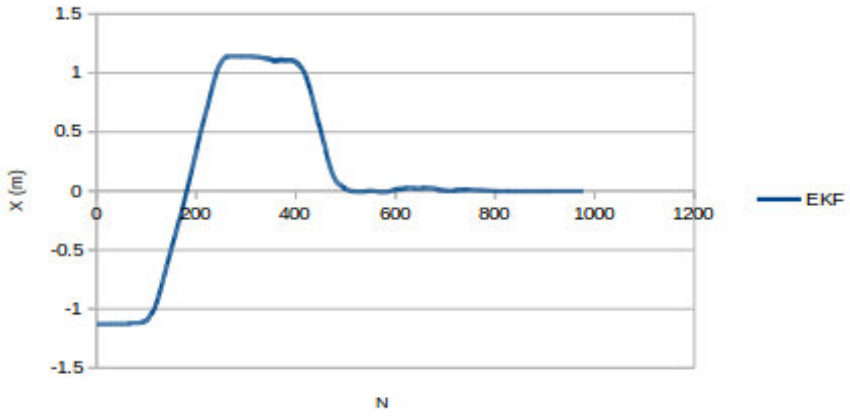
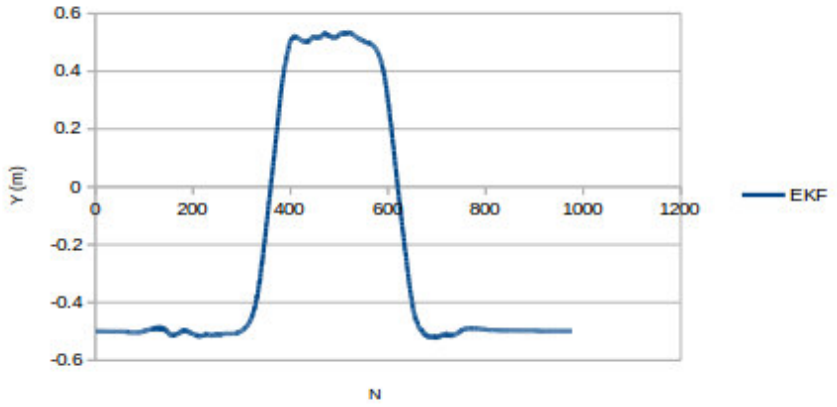


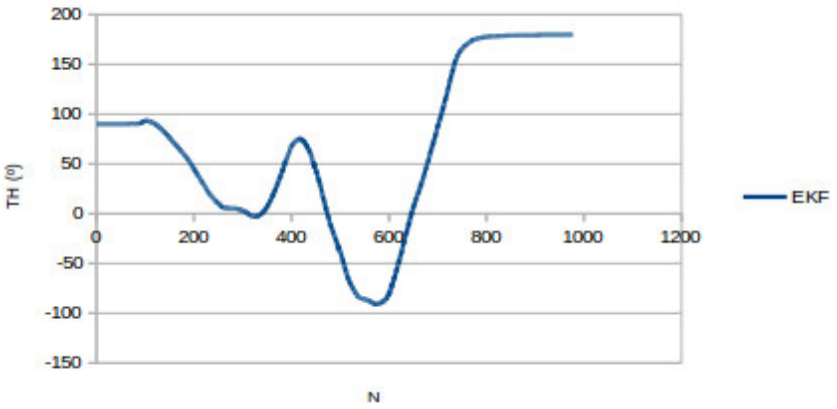
Figura 6.32: EKF: Pontos de referência da trajetória para teste de validação de algoritmos - teste 2C



(a) Evolução da coordenada x ao longo das iterações N



(b) Evolução da coordenada y ao longo das iterações N



(c) Evolução da coordenada θ ao longo das iterações N

Figura 6.33: EKF: Representação da evolução da coordenadas ao longo do movimento do teste de validação de algoritmos - teste 2C

O erro ao longo da trajetória, como se verifica na tabela 6.14, o erro nas posições onde a trajetória muda de direção é bastante reduzido e verifica-se a existência de pequenas correções

efetuadas pelos valores da odometria, que a curto prazo são bastante precisos.

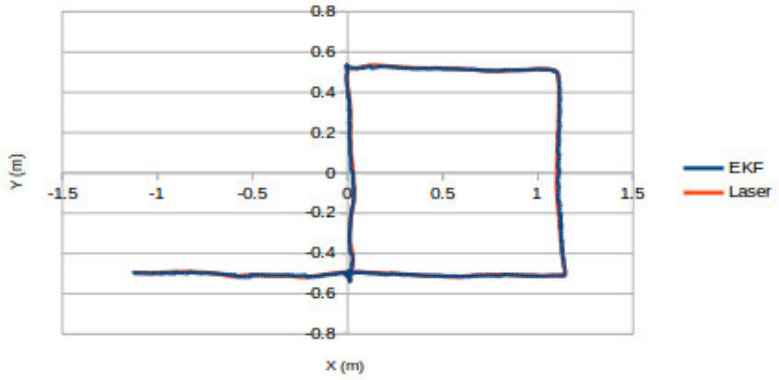
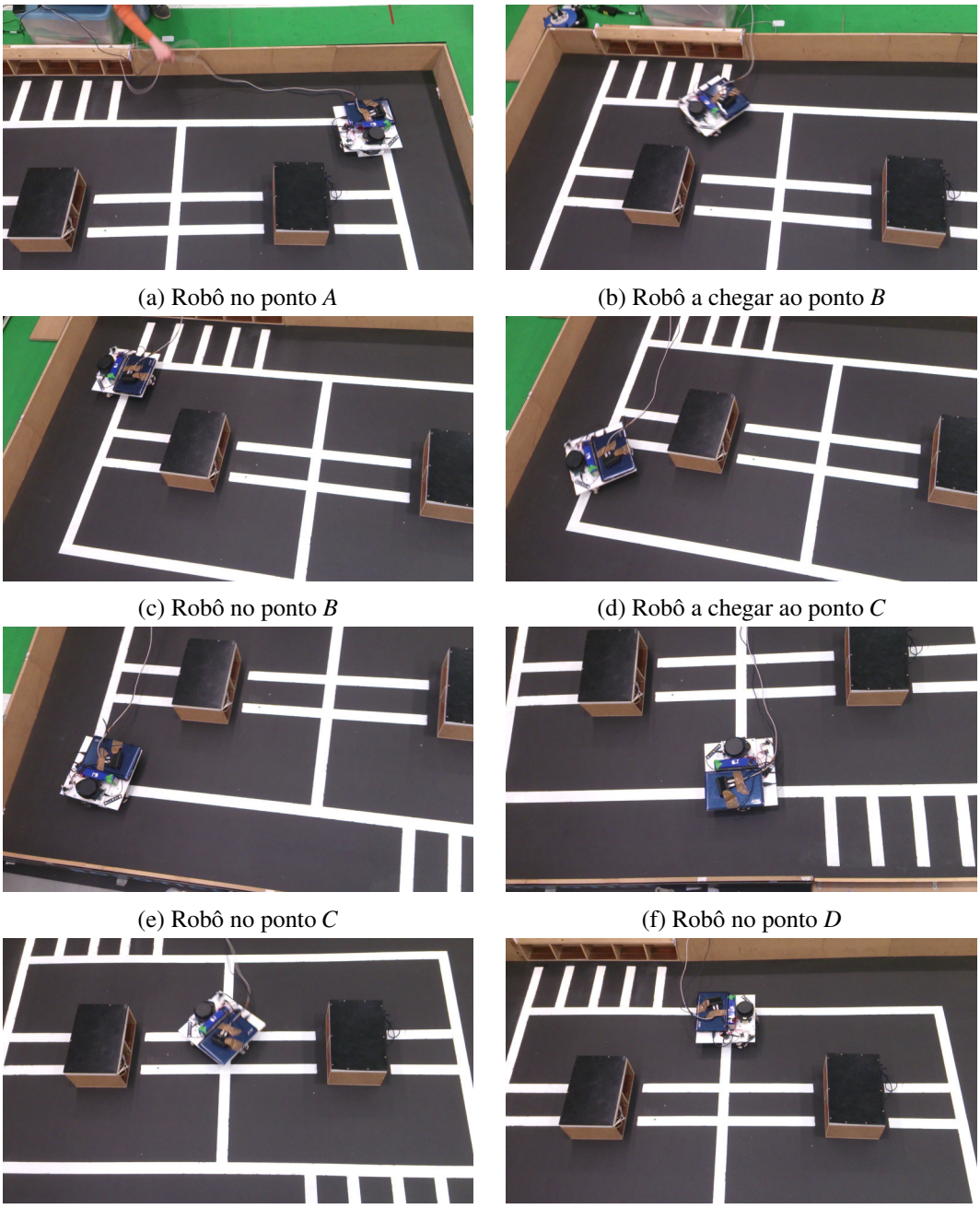


Figura 6.34: EKF: Comparação da trajetória para teste de validação de algoritmos com localização baseada em laser - teste 2C

Na Figura que se segue encontram-se *frames* de um vídeo no qual se registou o movimento do robô ao longo da trajetória de testes, com os parâmetros apresentados para o *teste 2C*. Nestas imagens, o circulo preto (laser) representa a frente do sistema robótico, permitindo uma perceção da orientação do mesmo ao longo do trajeto. Este movimento comprova o apresentado nas figuras 6.33 e 6.32.



(a) Robô no ponto *A*

(b) Robô a chegar ao ponto *B*

(c) Robô no ponto *B*

(d) Robô a chegar ao ponto *C*

(e) Robô no ponto *C*

(f) Robô no ponto *D*

(g) Robô a chegar ao ponto *E*

(h) Robô no ponto *E*

Figura 6.35: EKF: Representação da posição do robô - teste 2C

Capítulo 7

Conclusões e Trabalho Futuro

Neste capítulo são apresentadas as principais conclusões desta dissertação bem como descritas as perspetivas de desenvolvimentos futuros que podem surgir na sua sequência.

7.1 Satisfação dos Objectivos

Face ao resultado final desta dissertação, podemos concluir que os objetivos inicialmente definidos foram alcançados.

Foi desenvolvido um sistema de localização robusto, capaz de localizar o robô sem a necessidade de efetuar alterações no meio envolvente. Esta característica permite uma reconfiguração do meio em que o sistema robótico se insere, sem restrições. Houve uma atenção especial em reduzir o poder computacional exigido pela aplicação que executa os algoritmos de localização absoluta.

Complementarmente, foi apresentado um método de localização relativa baseada no deslocamento de cada uma das rodas do robô. Apesar de sujeita a elevados erros, ficou provada a sua capacidade de prover uma estimativa da localização do robô móvel. A utilização de um Filtro de *Kalman* Estendido permitiu fundir informação de diversas medidas e melhorar a estimativa da pose do robô. Com este método poderão ser acrescentadas novas soluções de localização que venham a ser estudadas.

No decorrer de todo o projeto, foi projetado e prototipado um sistema robótico que permitiu o teste e validação dos algoritmos no ambiente em que este está inserido e apto a participar na competição *Robot@Factory* de forma competitiva.

7.2 Trabalho Futuro

Com vista à continuidade deste projeto são apresentadas de seguida algumas sugestões de trabalho futuro.

Em relação ao sistema baseado em visão, propõe-se um sistema para evitar colisões contra obstáculos, através da identificação de erros sistemáticos nas medições do laser, por forma a evitar que o robô vá de encontro a caixas caídas ou obstáculos propositadamente colocados no meio do

campo. Esta funcionalidade permitirá à camada de planeamento de trajetórias mais informação, permitindo-lhe um cálculo atempado e ponderado da melhor trajetória a seguir.

Propõe-se igualmente como trabalho futuro, a calibração da odometria recorrendo a um método mais completo e a adoção das funções de erro para um melhor uso das matrizes de covariância aplicadas no Filtro de *Kalman* Estendido.

Sugere-se a comparação do algoritmo de localização absoluta com um sistema de localização baseado em balizas e a avaliação da eficácia de ambos.

Por fim, sugere-se a melhoria da interface com o utilizador da aplicação.

Anexo A

Ficheiro XML do SimTwo

A.1 Ficheiro de descrição do robô omnidirecional

```
<?xml version="1.0" ?>

<robot>
  <defines>

  <!-- Robot Dimensions -->
    <const name='RobotWidth' value='0.30'/>
    <const name='RobotLength' value='0.30'/>
    <const name='RobotThickness' value='0.002'/>
    <const name='RobotHeight' value='0.11'/>
    <const name='RobotMass' value='3'/>


    <const name='WheelToCenter' value='RobotWidth/2'/>
    <const name='MotorPosX' value='0.05'/>
    <const name='CenterMotorToFront' value='RobotLength/2 - MotorPosX'/>
    <const name='CasterToBack' value='0.07'/>

    <const name='ForkWidth' value='0.06'/>
    <const name='ForkLength' value='0.1'/>
    <const name='ForkThickness' value='0.002'/>
    <const name='ForkMass' value='0.1'/>

    <const name='ForkMin' value='0.005'/>
    <const name='ForkMax' value='0.05'/>
```

```
<!-- Motor Contants -->
  <const name='MotorDiameter' value='0.028'/>
  <const name='MotorLength' value='0.075'/>
  <const name='MotorMass' value='0.027'/>

<!-- Wheel Contants -->
  <const name='WheelDiameter' value='0.098'/>
  <const name='WheelThickness' value='0.028'/>
  <const name='WheelMass' value='0.5'/>

<!--Caster Contants -->
  <const name='CasterWheelDiameter' value='0.05'/>
  <const name='CasterWheelThickness' value='0.028'/>
  <const name='CasterToWheel' value='0.05'/>
  <const name='CasterWheelMass' value='0.2'/>
  <const name='CasterMass' value='0.2'/>


<!--Calculated Measures  -->
  <const name='BracketHeight' value='RobotHeight-RobotThickness
   -(WheelDiameter/2+MotorDiameter/2)'/>
  <const name='CasterPosY' value='-RobotLength/2
   -(RobotLength/2-CenterMotorToFront)+CasterToBack'/>
  <const name='CasterToBase' value='RobotHeight
   -RobotThickness-CasterWheelDiameter/2'/>

</defines>


<solids>

  <cuboid>
    <ID value='BasePlate'/>
    <mass value='RobotMass'/>
    <size x='RobotLength' y='RobotWidth' z='RobotThickness'/>
    <pos x='-0.01' y='0' z='0.07'/>
    <rot_deg x='0' y='0' z='0'/>
    <color_rgb r='20' g='20' b='20'/>
  </cuboid>
  <cuboid>
```

```
      <ID value='BaseLaser'/>
      <mass value='RobotMass/3+0.2'/>
      <size x='0.1' y='0.15' z='0.01'/>
      <pos x='0.09' y='0' z='0.09'/>
      <rot_deg x='0' y='0' z='0'/>
      <color_rgb r='20' g='20' b='20'/>
    </cuboid>

    <cuboid>
      <ID value='ForkE'/>
      <mass value='ForkMass'/>
      <size x='ForkLength' y='ForkWidth/10' z='ForkThickness*2'/>
      <pos x='RobotLength/2 + ForkLength/2' y='RobotWidth/6' z='0.025'/>
      <rot_deg x='0' y='0' z='0'/>
      <color_rgb r='40' g='40' b='40'/>
    </cuboid>
    <cuboid>
      <ID value='ForkD'/>
      <mass value='ForkMass'/>
      <size x='ForkLength' y='ForkWidth/10' z='ForkThickness*2'/>
      <pos x='RobotLength/2 + ForkLength/2' y='-RobotWidth/6'
       z='0.025'/>
      <rot_deg x='0' y='0' z='0'/>
      <color_rgb r='40' g='40' b='40'/>
    </cuboid>
  </solids>
  <shells>
    <cuboid>
      <ID value='BaseLaserShell'/>
      <size x='0.1' y='0.15' z='0.01'/>
      <pos x='0.09' y='0' z='0.08'/>
      <rot_deg x='0' y='0' z='0'/>
      <color_rgb r='20' g='20' b='20'/>
    </cuboid>


  </shells>

  <wheels>
                <default>
```

```
                <omni/>
                <pos x='0' y='0.1' z='0'/>
                <tyre mass="0.13" radius="0.03" width="0.033"
                 centerdist="0.144"/>
                <axis angle="0"/>
                <motor ri="2.853" ki="0.0289" vmax="12" imax="2"
                 active="1"/>
                <gear ratio="51/10"/>
                <friction bv="0" fc="0.00283" coulomblimit="1e-2"/>
                <encoder ppr="216" mean="0" stdev="0"/>
                <controller mode="pidspeed" kp="0.2" ki="0" kd="0.01"
                kf="0.05" active="1" period="10"/>
                <color_rgb r="128" g="0" b="128"/>
                </default>

                <wheel>
                <axis angle="300"/>
                </wheel>

                <wheel>
                <axis angle="60"/>
                </wheel>
                <wheel>
        <axis angle="180"/>
        </wheel>
        </wheels>
  <shells>

  </shells>

  <articulations>

<default>
  <motor ri='3.6' li='4.88e-3' ki='0.01025' vmax='12' imax='1'
   active='1'/>
  <gear ratio='0'/>
  <friction bv='2.63e-5' fc='3.558e-4' />
  <encoder ppr='400' mean='0' stdev='0'/>
<controller mode='pidspeed' kp='1' ki='0' kd='0' kf='0.05' active='1'
 period='10'/>
```

```
</default>


 <joint>
    <ID value='ForksE'/>
    <connect B1='ForkE' B2='BasePlate'/>
    <pos x='(RobotLength/2 + ForkLength/2)/2' y='RobotWidth/6'
     z='0.025'/>
    <axis x='1' y='0' z='0'/>
    <!--limits Min='0' Max='ForkMax-ForkMin'/-->
    <type value='Hinge'/>
  </joint>

  <joint>
    <ID value='ForksD'/>
    <connect B1='ForkD' B2='BasePlate'/>
    <pos x='(RobotLength/2 + ForkLength/2)/2' y='-RobotWidth/6'
     z='0.025'/>
    <axis x='1' y='0' z='0'/>
    <!--limits Min='0' Max='ForkMax-ForkMin'/-->
    <type value='Hinge'/>
  </joint>

  <joint>
    <ID value='PesoLaser'/>
    <connect B1='BaseLaser' B2='BasePlate'/>
    <pos x='0.1' y='0' z='0.08'/>
    <axis x='0' y='0' z='1'/>
    <limits Min='-0.012' Max='-0.011'/>
    <type value='Slider'/>
  </joint>


   <joint>
    <ID value='Forkssss'/>
    <connect B1='BackFork' B2='BasePlate'/>
    <pos x='CenterMotorToFront + ForkWidth/2' y='0'
     z='0.005+(RobotHeight-ForkMIn-RobotThickness)/2'/>
    <axis x='0' y='0' z='1'/>
    <!--<limits Min='0' Max='ForkMax-ForkMin'/>    -->
```

```
    <type value='Slider'/>
    <gear ratio='1000'/>
    <friction bv='5e-3' fc='3.558e-4' />
    <controller mode='pidposition' kp='100' ki='0.2' kd='0'
     kf='0.05' active='1' period='10'/>
  </joint>


  </articulations>

  <defines>
    <!-- Sensor "dimensions" -->
    <const name='IRSharpOffsetX' value='-0.03'/>
    <const name='IRSharpOffsetY' value='-0.1'/>
    <const name='IRSharpOffsetZ' value='-0.02'/>

    <const name='LineSensorOffsetX' value='0.05'/>
    <const name='LineSensorYSpace' value='0.015'/>
  </defines>

  <sensors>

    <ranger2d>
          <ID value='ranger2d'/>
          <period value='0.1'/>
          <beam length='4' initial_width='0.01' final_width='0.015'/>
         <!-- <pos x='1' y='0' z='0.49'/> -->
          <pos x='0.1' y='0' z='0.09'/>
          <rot_deg x='0' y='0' z='180'/>
          <tag value='00'/>
          <beam angle='360' rays='360'/>
          <noise stdev='0.0' stdev_p='0' offset='0' gain='1'/>
          <color_rgb r='255' g='0' b='0'/>
    </ranger2d>

  </sensors>

</robot>
```

Referências

[1] Goetting. Sensorik zurFahrzeugautomatisierung. 2013.

[2] Transbotics. URL: http://www.transbotics.com/learning–center/guidanc, Visitado em Janeiro de 2014.

[3] Lothar Schulze, Sebastian Behling, e Stefan Buhrs. Automated Guided Vehicles Systems: a Driver for Increased Business Performance. 2008.

[4] Roland Siegwart, Margarita Chli, Martin Rufli, e Davide Scaramuzza. Slides de Apoio à Unidade Curricular de Autonomous Mobile Robots - Robot Vision III, ETH. 2012/2013.

[5] Jeppe Jensen. Hough Transform for Straight Lines. 2007.

[6] Roland Siegwart, Illah R. Nourbakhsh, e Davide Scaramuzza. *Introduction to Autonomous Mobile Robots, 2nd Edition*.

[7] José Gonçalves. Controlo de robots omnidireccionais. 2005.

[8] Hélder Oliveira. Análise do Desempenho e da Dinâmica de Robôs Omnidireccionais de Três e Quatro Rodas. 2007.

[9] Eduardo Santos. Logística baseada em AGVs. 2013.

[10] André Oliveira. Localização de um AGV em Ambiente Industrial Através de um Laser de Segurança e Marcadores Artificiais. 2013.

[11] David Lima. *Localização Absoluta de Robôs Móveis em Ambientes Industriais*. Tese de doutoramento, 2010.

[12] AGVSystems. URL: http://www.agvsystems.com/navigation-landing, Visitado em Janeiro de 2014.

[13] Thomas Kämpke, Boris Kluge, Erwin Prassler, e Matthias Strobel. Robot Position Estimation on a RFID-Tagged Smart Floor. 2008.

[14] Wei Wei e Kevin Curran. Indoor Robot Localisation with Active RFID . 2012.

[15] Joydeep Biswas e Manuela Veloso. Depth Camera Based Indoor Mobile Robot Localization and Navigation. 2012.

[16] Jeisung Lee, Chang-Ho Hyun, e Mignon Park. A Vision-Based Automated Guided Vehicle System with Marker Recognition for Indoor Use. 2013.

[17] Rita Cucchiara, Emanuele Perini, e Giuliano Pistoni. Efficient Stereo Vision for Obstacle Detection and AGV Navigation. 2007.

[18] Andry M. Pinto, António P. Moreira, e Paulo G. Costa. A Localization Method Based on Map-Matching and Particle Swarm Optimization. *Journal of Intelligent & Robotic Systems*, Dezembro 2013.

[19] Martin Lauer, Sascha Lange, e Martin Riedmiller. Calculating the Perfect Match: An Efficient and Accurate Approach for Robot Self-localization. 2006.

[20] Michael A. Zmuda, Aleksandr Elesev, e Yu T. Morton. Robot Localization Using RF and Inertial Sensors. 2008.

[21] Richard O. Duda e Peter E. Hart. Use of the Hough Transformation to Detect Lines and Curves in Pictures. 1972.

[22] Manuel Gouveia. Estudo e Implementação de um Algoritmo de Localização Naseado na Correspondência de Mapas. 2008.

[23] Dieter Fox, Wolfram Burgard, e Sebastian Thrun. Markov Localization for Mobile Robots in Dynamic Enviroments. 1999.

[24] António Moreira. Slides de Apoio à Unidade Curricular de Sistemas Robóticos Autonomos, FEUP. 2013/2014.

[25] Manya Afonso. Particle Filter and Extended Kalman FIlter for Nonlinear Estimation: A Comparative Study. 2008.

[26] José Gonçalves. *Modelação e simulação realista de sistemas no domínio da robótica móvel*. Tese de doutoramento, 2009.

[27] Hokuyo. URL: http://tinyurl.com/qbnksu8, Visitado em Maio de 2014.

[28] Sparkfun. URL: https://www.sparkfun.com/news/490, Visitado em Maio de 2014.

[29] ROS. URL: http://www.ros.org/news/2011/02/neato-xv-11-laser-driver.html, Visitado em Maio de 2014.

[30] XV11Hacking. URL: http://xv11hacking.wikispaces.com/LIDAR+Mechanical+Info, Visitado em Maio de 2014.

[31] Paulo Costa. URL: http://paginas.fe.up.pt/ paco/wiki/index.php?n=Main.SimTwo, Visitado em Janeiro de 2014.

[32] Martin Riedmiller. Rprop - Description and Implementation Details. 1994.

[33] Martin Riedmiller e Heinrich Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. 1993.