"The Cleaning Robot Project"

Aplicação do Filtro de Partículas como sistema de fusão de informação

Gustavo Gama da Rocha Pimentel

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
sob orientação de
Professor Doutor Armando Jorge Miranda de Sousa
e do Professor Doutor Paulo José Cerqueira Gomes da Costa

_____

O Presidente do Jurí,
Professor Doutor António Miguel Pontes Pimenta Monteiro

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

Março de 2008

Apoios

Este projecto foi realizado com o apoio de:

Departamento de Engenharia Electrotecnia e de Computadores da Faculdade de Engenharia da Universidade do Porto

Institute of Systems and Robotics - Porto

Futebol Robótico da Faculdade de Engenharia da Universidade do Porto

Gostaria de agradecer:

Aos meus pais e família pelo apoio constante,
aos quais peço desculpa pelas tantas preocupações
dadas ao longo deste projecto

À minha querida e grande companheira Laura
que tanto me acompanhou e apoiou em todos os momentos
com a sua infinita paciência e sabedoria

Aos meus amigos que sempre me
acompanharam ao longo deste projecto

Fernando Pedro Vieira Freitas Pinto,
colega e amigo de inúmeros projectos
e de longas noites a fio a trabalhar neste projecto

José Alberto da Cunha Barros,
ao apoio constante e amizade

Pedro da Silva Machado,
pela dedicação, motivação e paixão pela robótica

Aos meus orientadores
pela oportunidade, confiança e dedicação em mim neste projecto
Armando Sousa e Paulo Costa

Aqui fica o meu grande obrigado

Gustavo Pimentel

"A verdadeira grandeza de um homem reside na consciência de um propósito honesto na vida, alicerçado numa estimativa justa de sua pessoa e de tudo o mais; num frequente auto exame, numa firme obediência às regras por ele tidas como certas, sem perturbar-se com o que os outros possam vir a pensar ou dizer, ou com fazerem elas, ou não, aquilo que ele pensa, diz e faz."

(Marco Aurélio)

A possibilidade de realizarmos um sonho
é o que torna nossa vida interessante.

(Paulo Coelho)

Resumo

A robótica tem sido uma área de grande desenvolvimento e crescimento económico na sua vertente industrial, militar e civil. Os robôs tem demonstrado serem ferramentas de elevada importância nesta era moderna, sem os quais muito dificilmente poderíamos igualar a qualidade de vida que hoje possuímos.

A autonomia é um factor importantíssimo no desenvolvimento de robôs dotados de uma maior inteligência. Permitindo assim que estes consigam executar funções com a mesma destreza que nós, seres humanos, assumimos como inato. Neste passo gigantesco que é a autonomia para um robô, estabelece-se a base da auto-localização, antes de um robô realizar qualquer tipo de função é necessário que este conheça primeiro onde se localiza, passando assim a outras fases de controlo que permitam executar as mais diversas tarefas.

A fusão de informação é então integrada na auto-localização, sendo usados Filtros de Kalman e Filtros de Partículas, com um único objectivo de fundir a informação proveniente de diferentes sensores para produzir informação relevante que permite a localização completa para os robôs móveis x, y e θ. Neste documento é descrito a aplicação do Filtro de Partículas na auto-localização de um robô autônomo de limpeza que opera num meio semi-estruturado e dinâmico.

Abstract

The robotics have been an area of great economic development and growth in the industrial, military and civilian strand. The robots have shown to be tools of high importance in the modern era, beeing hard to match the quality of life that we have today without them.

Autonomy is a major factor in the development of robots endowed with greater intelligence, allowing them to perform the innate functions of the humans with the same dexterity.

In this gigantic step of autonomy to a robot, there is the basis of self-location, that gives the robot the possibility to know his locations before performing any function, and reach higher stages of control to perform the most diverse tasks.

The data fusion is then incorporated into self-location, by the Kalman and Particles filters, with the purpose of merging different information from sensors to produce a single information of relevant importance that the location allows for the complete mobile robots x, y and θ. This document is the portrait of the application of the filter particles in the self-location of an autonomous robot for cleaning operates in a semi-structured and dynamic enviroment.

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1

Introdução

1.1 Motivação

Desde os primórdios dos tempos que o ser humano tem mostrado uma capacidade inata para imaginar e desenvolver ferramentas cujo o intuito é de facilitar as diversas actividades da vida quotidiana. Com o avançar galopante da tecnologia, o ser humano rompe frequentemente os limites impostos, não havendo qualquer limitação para a nossa capacidade de criar/inovar.

Os robôs autônomos de serviço têm vindo a crescer de importância à medida que a Ciência e Tecnologia evolui e permite a construção de verdadeiros assistentes para o ser humano facilitando a execução de diversas tarefas.

Actualmente existem diversos robôs de serviço aliados sobretudo ao uso doméstico sendo um deles o "*Roomba*" (robô aspirador), o "*PatrolBot*" (um robô de vigilância, actualmente em funcionamento na sede das nações unidas).


Figura 1.1: Robô aspirador "Roomba" à esquerda,
Robô vigilante "PatrolBot" à direita

As projecções de crescimento desta área da robótica são elevadas e as expectativas grandes, de tal modo, que se gera grande interesse nesta área, sendo o mercado dos robôs de serviço uma área em franco crescimento atingindo actualmente os 5,4 biliões de dólares, esperando alcançar já em 2010 os 17,1 biliões de dólares [1].

1.2 Contexto

Este trabalho surge na sequência de uma grande paixão por parte do autor pela robótica em particular e de um ponto de vista geral pela automação industrial.

O presente trabalho insere-se no prolongamento do projecto do "*CleanRob*" – Robô autônomo de limpeza do DEEC. Este projecto foi iniciado em 2003 e porém este encontrava-se numa fase muito embrionária, sendo necessário o desenvolvimento e implementação da maior parte do trabalho que agora se apresenta.

No momento da atribuição deste projecto, o estado da plataforma robótica encontrava-se bastante debilitada e pouco desenvolvida, tendo a mecânica construída, contudo com bastantes problemas e inúmeras deficiências ao nível da robustez e eficiência de funcionamento. Este e muitos outros problemas foram sendo resolvidos ao longo da duração deste projecto, à medida que se identificava a fonte dos diversos problemas. O programa de comando anterior encontrava-se incompleto apresentando inúmeros problemas no que respeita a programação e controlo do robô, para além de um ponto crítico, este programa de controlo estava desenvolvido numa ferramenta de desenvolvimento antiga e sem qualquer suporte, o que traria problemas futuros. Tornando estes problemas num desafio aliciante os quais não foi possível resistir.

O principal objectivo deste projecto é estudar e implementar um

sistema de auto-localização com base no Filtro de Partículas que permitisse o funcionamento de robôs autônomos em ambientes dinâmicos semi-estruturados numa aplicação real. Num caso mais concreto de um robô de limpeza que consiga deslocar-se pelos corredores do Departamento de Engenharia Electrotécnica e de Computadores, realizando operações de limpeza.

1.3 Contribuições

Esta dissertação apresenta as seguintes contribuições de relevo:

- Documentação de todo o trabalho relevante realizado no projecto "*CleanRob*" – Robô autônomo de limpeza.
- Análise teórica de um método de fusão de informação com base no Filtro Bayesiano - Filtro de Partículas.
- Análise da implementação prática do Filtro de Partículas com diversos sensores.
- Análise de diversos sensores integrantes ao "*CleanRob*".
- Documentação de ferramentas de análise e desenvolvimento ao projecto "*CleanRob*".

1.4 Estrutura

Este documento aborda em primeiro lugar o projecto de sistemas robóticos autônomos.

De seguida aborda-se o problema da localização, incluindo sistemas e métodos associados.

Segue-se a teoria necessária aos trabalhos apresentados. É abordada a modelização e calibração de diversos sensores e apresenta-se o Filtro de Partículas e sua implementação no robô sendo validado em vários testes dinâmicos reais.

No final são apresentadas conclusões e propostas de melhoramentos futuros.

Capítulo 2

Sistemas Robóticos

2.1 Introdução

Um robô é frequentemente visto como um trabalhador artificial que executa tarefas tradicionalmente realizadas por humanos tendo como base planos de acção detalhados. Estes planos de acção nesta era industrial moderna assumem-se sob a forma de programas computarizados. O controlo e a sensorização são usados primordialmente para garantir a execução precisa destes planos. A programação do robô é realizada por um engenheiro e/ou um operador através de várias interfaces, incluindo interfaces de grande interactividade, nos casos mais avançados.

Desde o inicio da era robótica, a ideia de robôs autônomos tem fascinado investigadores. Estes têm a potencialidade de aprender e programar-se automaticamente, desenvolvendo estratégias executando acções com base em informação sensorial adquirida pelo próprio robô. Robôs autônomos poderão assim aproximar-se a uma versão inteligente semelhante às acções tomadas por operadores humanos.

Em particular, as operações executadas em meios não ideais, ou

em ambientes naturais requerem manutenção, construção, reparação ou supervisão de tarefas, requerem um certo grau de autonomia do robô. Devido a um elevado custo de antecipação de tal agentes de "inteligência", a sua aplicação inicial foi restringida a áreas onde uma operação humana é dispendiosa e/ou perigosa como em missões subaquáticas/espaciais ou até mesmo em ambiente radioactivos. De facto é bastante lógico que tais robôs, quando viáveis sob ponto de vista económico e tecnológico, consigam fazer parte de um vasto leque de aplicações fora do âmbito clássico associado ao robô que tem sido alvo de tema de entretenimento de tantas fantasias de ficção cientifica e facto científicos.

Para os investigadores o desafio encontra-se em permitir que a máquina consiga imitar de uma forma autónoma o comportamento, a percepção e a capacidade cognitiva que os humanos e animais possuem de forma inata. Paralelamente tem o intuito de analisar o sucesso desta interacção em tempo real associada a uma certa incerteza num ambiente dinâmico.

Uma abordagem clássica ao problema de um robô autônomo tem sido desenvolvida desde os anos cinquenta numa relação simbiotica com a área de inteligência artificial. A ideia base é que a inteligência do robô adquira um modelo abstracto do ambiente no qual é suposto operar. Nesta abordagem o robô é estruturado à medida que a corrente de informação flui pelo sistema, abarcando desde os processos de sensibilidade à sensorização, passando no final por uma representação central do mundo (na maioria das vezes simbólica). Esta representação é derivada do processo de informação proveniente da sensorização, a um estágio de planeamento em que é tratada e analisada para que possa ser transmitida a uma camada de controladores cuja função é a execução actual do plano de acção. Nestes sistemas de comportamento a resposta surge da interpolação do planeamento com os objectivos a serem cumpridos e do modelo particular do mundo construído a partir da informação da sensorização.

O ponto alto deste método é a abordagem num grau elevado que foi teoricamente idealizado. O ponto fraco mantém-se na fragilidade imposta pelo sucesso prático dos robôs baseados na ideia utópica de se manterem autônomos em ambientes dinâmicos e não-ideais. Por exemplo, é bastante difícil (se não impossível) obter-se uma representação do ambiente por um modelo simbólico ou métrica precisa através de informação sensorial com ruído. Em adição, a construção de tais representações tem um custo computacional bastante alto.

2.2 Sistema Embebido

Segundo referido pelo autor [2] designa-se por sistema embebido o conjunto de sistemas embarcado no robô que lida com as interfaces robô-ambiente e processamento dos respectivos sinais. Este sistema lida com a conversão de energia de e para o exterior incluindo sensores, actuadores e respectivos circuitos de acondicionamento. A aquisição de sinal dos sensores faz parte deste sistema bem assim como o seu pré-processamento, eventual conversão e eventual disponibilização adequada, por exemplo, numa rede interna ao robô. Também a geração de sinais adequados aos actuadores é da responsabilidade deste importante subsistema do robô.

2.3 Navegação

As perguntas fundamentais da navegação de um sistema robótico móvel são:

"**Onde estou?**", "**Para onde vou?**" e "**Como chegar lá?**"

Estas questões não são de reposta simples quando se pretende um robô autônomo. "Onde estou?" implica medidas face ao ambiente exterior que pode ser dinâmico. Saber para onde se deve deslocar o robô para melhor cumprir a sua tarefa pode também ter solução não trivial. Seguir numa certa direcção pode não garantir chegar a certo ponto mesmo que ele fique nessa direcção pois o caminho pode estar bloqueado.

Complexidade adicional aparece se se tiver em atenção considerações de segurança tal como quando o robô se desloca em ambiente dinâmico. Especial cuidado é necessário se o espaço é partilhado com seres humanos.

A navegação envolve diversas capacidades como por exemplo o contorno de obstáculos e actualização dinâmica do mapa actual. As técnicas mais utilizadas para navegação são:

- Utilização de "mapas de estradas" de segmentos utilizáveis;
- Decomposição em sub-blocos do mapa actualizando blocos com informação ocupado/livre mais recente;

Campos de Atracção/Repulsão (obstáculos repelem o robô que é atraído pelos pontos onde pretende chegar).

A navegação pode ser considerada como a plataforma intermédia que faz cumprir a ordem de chegar a um certo destino. O nível de autonomia que se pretende dar a esta camada intermédia de decisão é muito variável conforme a aplicação. Pode acontecer que um dado problema de navegação não tenha solução perante dado nível de autonomia concedida a este sistema. Neste caso novas decisões de nível superior são necessárias para que o robô atinja os seus objectivos.

Considerando que é possível uma localização satisfatória para o problema em causa, o problema seguinte prende-se em fazer com que o robô se desloque para cumprir os seus objectivos. É então necessário saber onde o robô está, para onde deve seguir e como o conseguir, isto é, é necessário que o robô navegue através do caminho escolhido até ao seu destino.

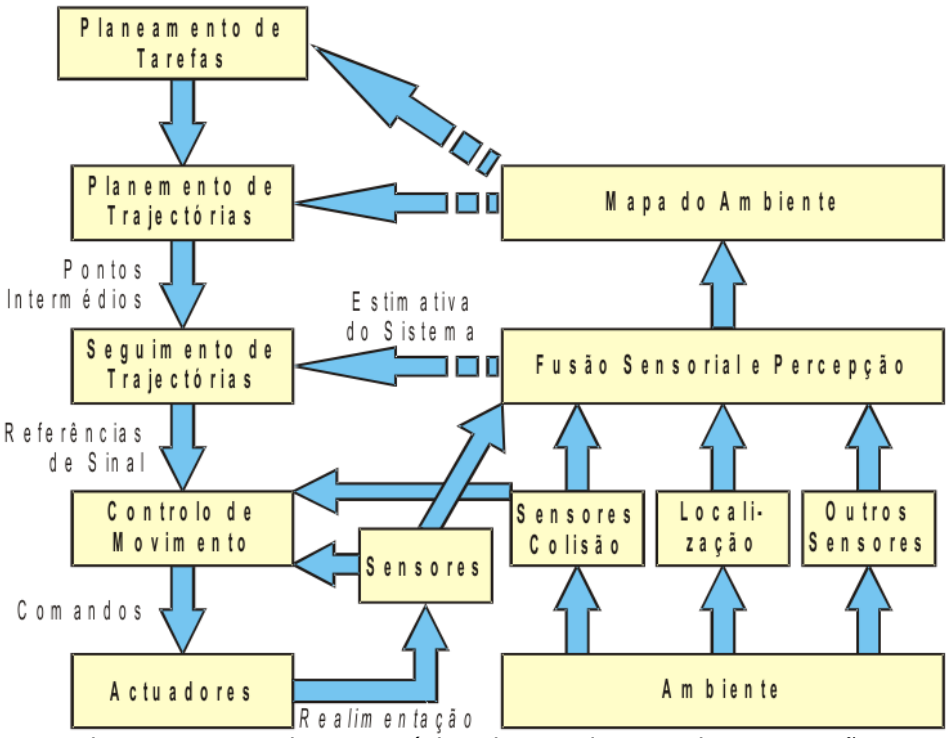


Figura 2.1: Arquitectura típica de um sistema de navegação

A arquitectura típica de um sistema de navegação pode ser decomposta tal como mostra a figura 2.1. No nível superior de decisão encontra-se geralmente o planeamento de tarefas que gera comandos para o sistema de planeamento de trajectórias. O sistema de contorno de obstáculos actua muitas vezes sobre o nível de controlo de movimento.

Sistemas mais complexos podem fornecer a informação de obstáculos aos níveis superiores de decisão. A gestão do desvio de

obstáculos complicados em sistemas mais evoluídos pode exigir o traçado de novas trajectórias.

Sistemas com este tipo de arquitectura permitem fechar a malha de comando a vários níveis o que permite por sua vez:

- Contorno e desvio de obstáculos – a baixo nível;
- Fusão de dados para criar uma estimativa global do sistema – a nível intermédio;
- Actualização do mapa corrente – a nível superior.

Tipicamente a navegação acaba por ser abordada como sendo o seguimento de referências intermédias até alcançar destinos finais onde são atribuídas novas tarefas, sendo todo este tema analisado em pormenor pelo autor [2] .

2.4 Decisão, Autonomia e Cooperação

O autor [2] define que a arquitectura do "*software*" de nível mais elevado é crítica para o desempenho satisfatório do robô que se pretende projectar. Chame-se ao "*software*" com mais elevado nível de abstracção o "*software*" de decisão. Tal como ilustrado na figura 2.2, o módulo de decisão toma as suas decisões com base em informações do mundo real fornecidas pelo bloco de entradas do robô e muda esse mesmo estado do mundo por intermédio das saídas do sistema. No bloco das entradas cabem subsistemas de sensores e respectivo tratamento de sinais enquanto que ao bloco das saídas dizem respeito os actuadores e respectivo acondicionamento.

O bloco de entradas deve ser entendido como um bloco que reúne a geração de todos os dados necessários à produção de decisões pelo respectivo bloco e não apenas como um conjunto de sensores. Este bloco incluirá possivelmente informação relativa à fusão de informação de sensores e do estado estimado do robô.

O bloco de saídas reúne todos os sub-sistemas que implementam as decisões do módulo superior. Este bloco incluirá possivelmente malhas de controlo em tempo contínuo e a geração de sinais de referência e comando sobre os actuadores.

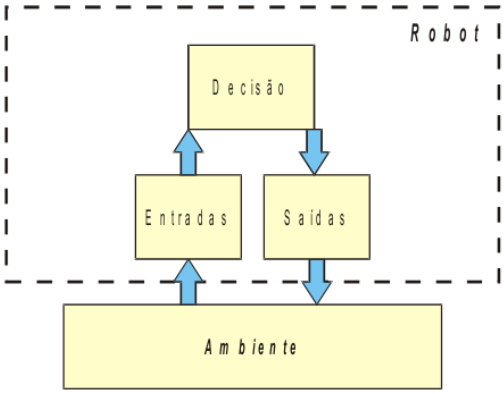


Figura 2.2: Blocos de entras, saídas e decisão dentro do robô e respectivo relacionamento com o ambiente exterior

O funcionamento dos vários blocos do robô é claramente paralelo e síncrono com a realidade, havendo assim diversas restrições de tempo real.

O "*software*" de decisão deve então basear-se em diversos outros blocos de "*software*" de nível inferior para criar o comportamento de alto nível desejado para o robô.

Diversas considerações são importantes relativamente à arquitectura do "*software*" de alto nível do robô para que este seja útil e adequado ao problema em causa:

- Divisão de tarefas/modularidade/suporte para paralelismo;
- Adequação tecnológica à plataforma real (ao nível dos sensores, sensores configuráveis, actuadores, plataformas computacionais, comunicações, etc.);
- Adequação ao meio envolvente e às funcionalidade pretendidas;
- Robustez do "*software*" (mesmo em situação de avaria parcial);
- Qualidade e tipo de ferramentas existentes para a implementação;
- Capacidades de configuração durante a execução;
- Avaliação do desempenho – as metas temporais relativas ao comportamento em Tempo Real são atingíveis ? Os dados para a tomada eficaz de decisões estão disponíveis ? A estrutura permite flexibilidade para todas as tarefas a realizar serem "bem realizadas"?

As considerações mais importantes que o "*software*" de decisão necessita de ter em conta são:

- Como guardar informação acerca do seu próprio estado;
- Como guardar informação acerca do ambiente exterior;
- Como tomar decisões.

As abordagens mais frequentes à decisão em sistemas robóticos são:

- Puramente reactiva – sem planeamento a prazo, para cada condição há uma reacção programada, sem estado interno do sistema;
- Hierárquico – controlo e planeamento utilizando diferentes níveis de abstracção; a informação do mundo é utilizada para gerar sequências de comandos a cumprir;
- Controlo híbrido – combina as duas abordagens anteriores num sistema hierárquico onde o nível inferior é reactivo e o nível superior produz sequências de acções;
- Comportamental – incluem partes reactivas mas utilizam representações complexas do mundo; utiliza-se por vezes coordenação, competição e concorrência para decidir a lista de acções a tomar.

Em termos de implementação, uma questão relevante é como incluir na arquitectura do robô a existência de várias tarefas para diversos subsistemas do robô pois pode existir a necessidade de gerir a coordenação/conflito entre diversos subsistemas. Como exemplo, considere-se um robô autônomo de transporte onde um braço do robô deve depositar um objecto numa caixa vazia. Este sistema hipotético utilizaria uma câmara que deve ser orientada para o local a depositar o objecto. Se este local não estiver vazio, o robô e a câmara devem receber ordens de alto nível para o robô se deslocar e visualizar a caixa seguinte até aparecer um local vazio. A visão deve depois fornecer dados relativamente à manobra de descarga do objecto que o braço do robô vai realizar.

O suporte para acções em caso de falha de um elemento do robô deve também ser previsto a nível de arquitectura de "*software*". Poderá ser importante que o próprio robô tome medidas para limitar os danos e pode ser interessante ele conseguir algum tipo de auto diagnóstico da sua falha.

A nível de implementação, grande parte dos robôs utiliza computadores embarcados no robô para implementação das camadas

superiores de decisão em sistemas robóticos. São amplamente utilizados os Sistemas Operativos genéricos “*Windows*” e “*Linux*”. Menor número de aplicações utiliza sistemas operativos mais adequados a desempenho em Tempo Real tal como “*Tornado*”, “*QNX*”, “*LinxOS”*, “*VxWorks*”, “*Aperios*” e ainda diversas variantes Tempo Real do “*Linux*”. Projectos que necessitem de segurança adicional podem ainda utilizar autómatos programáveis para tarefas críticas, à custa de um preço acrescido.

2.4.1 Tentativa de Indiferença ao "*hardware*"

Segundo o autor [2], o "*software*" de alto nível em robótica deve caminhar em direcção à independência ao "*hardware*". O encapsulamento de tarefas deve ser efectuado de tal modo que esta afirmação seja levada em consideração.

Dado o elevado ritmo do progresso tecnológico na área dos sistemas robóticos, é, em geral, interessante criar algum tipo abstracção ao "*hardware*", criando assim uma camada de abstracção ao "*hardware*" – *(HAL)*[1]. Desta forma, a troca de "*hardware*" por outro equivalente pode ser conseguida sem grande esforço. Dado o elevado nível de interdependência das tecnologias envolvidas na robótica tal pode não ser conseguido na totalidade mas o conceito, mesmo tomado de forma parcial, é ainda interessante.

Uma solução possível é modularizar o "*software*" em duas grandes partes. Uma parte inclui a sensorização e a actuação, enquanto que a outra parte compreende a decisão. A troca de dados entre as partes deve utilizar sempre interfaces e linguagens bem definidas. Sempre que possível, a troca de dados deve ser conseguida de forma independente da implementação. Pretende-se assim limitar as dependências do "*hardware*" ao módulo da sensorização e actuação. A modularização proposta apresenta as vantagens adicionais de passar a existir um ponto importante de "*debugging*" e de permitir que as metades do sistema sejam simulados separadamente.

2.4.2 Fusão de Informação

A fusão de informação é pode ser classificada em três possíveis casos:

- Sensores complementares – sem conflito;
- Sensores competitivos – com o objectivo de obter redundância;
- Sensores cooperativos – cooperam para gerar informação, por: ex

1 HAL – Hardware Abstraction Layer, camada de abstracção da componente física

visão “*stereo*”.

Um dos principais objectivos da fusão de informação é cruzamento de informação proveniente de diversos sensores com características diferentes e gerar informação útil a partir da combinação da informação parcial fornecida por estes. É um processo que permite obter elevados níveis de qualidade da informação, permite o uso de sensores com um custo económico reduzido, compensando na redundância adicional da informação destes, assegurando assim decisões correctas nas presença de dados errados, desde que para tal exista uma certa razão de quantidade de dado válidos e correctos.

A caracterização de cada um dos sensores é então essencial:

- Precisão, exactidão e sensibilidade;
- Resposta dinâmica;
- Caracterização da eventualidade de leituras erradas;
- Caracterização do comportamento sob avarias.

2.5 “Interface” e Comunicação com o Exterior

A comunicação é uma área vital que tem vindo a acompanhar o ser humano ao longo dos tempos, o mesmo se assemelha a um robô autônomo que necessita de um meio de comunicação. Possibilitando deste modo criar uma rede de informação com outros robôs que permitirá desde da elaboração de estratégias de cooperação na execução de tarefas passando por avisos a um agente supervisor.

Os recentes avanços nas tecnologias das redes sem fios permitem manter de forma económica comunicações com plataformas móveis fazendo uso de estruturas já existentes, tal como a rede “*wireless*” de um “*campus*” universitário.

2.6 Ferramentas de Ajuda ao Desenvolvimento

A utilização de um cabo de comunicações especial para ajuda ao desenvolvimento era muito frequente no passado. Este cabo de

comunicações denominado de “cordão umbilical” permite descarregar uma grande quantidade de informação para fora do robô para melhorar o desenvolvimento. Nesta situação, os dados seriam recolhidos e armazenados ao longo da missão e depois descarregados através do dito cordão umbilical. Com o aumento da largura de banda disponível os modernos sistemas de comunicações sem fios, esta noção deixou de ser utilizada tão frequentemente.

Um simulador para o sistema é uma ferramenta muito importante para poder testar a programação de alto nível antes de embarcar o "*software*" a testar. É também importante uma maneira de registar dados importantes para o sistema que está a ser desenvolvido e idealmente o ficheiro de registos “*log-file”* deveria poder ser executado de novo até que o ciclo de desenvolvimento esteja concluído.

Para além das ferramentas de ajuda ao projecto do sistema, o autor [2], refere que será interessante prever ferramentas de auxílio ao “*debugging*” já a níveis funcionais e a diversos níveis de abstracção. Os simuladores de conjunto do sistema são ajudas preciosas ao desenvolvimento. Também os sistemas de armazenamento dos históricos são importantes para consulta dos detalhes dos motivos que levaram tal movimento a acontecer. A possibilidade de consulta remota deste tipo de dados é útil mas muitas vezes impraticável por motivos tecnológicos da quantidade de dados a transferir através de comunicações sem fios. Esses dados podem ser transferidos facilmente pelo “cordão umbilical”.

2.7 Conclusões

Neste capítulo foram listadas abordagens muito genéricas à localização e soluções concretas de localização trabalhadas por outros autores. Esta área é motivo de intensa pesquisa cientifico-tecnológica. As soluções dependem fortemente da aplicação específica em causa, em particular do o nível de estrutura do ambiente, dentro ou fora de portas, das características do mapa disponível e das dimensões em causa para o movimento.

Contudo esta área conceptual é bastante vasta, merecendo uma larga discussão sobre todos os factores intervenientes no planeamento de um sistema robótico, infelizmente não é possível neste documento debater todos os aspectos importantes, tendo sido apenas referenciados aos dos factores que se enquadram mais no objectivo principal desta tese.

Capítulo 3

Modelo do Sonar

3.1 Introdução

Figura 3.1: Sonar Devantech SRF08

O sonar é um dispositivo capaz de emitir ondas de ultra-sons e captar o eco reflectido por um objecto, sabendo assim a distância a que este se encontra, através da contabilização do tempo entre a emissão do som e a recepção do eco. Este dispositivo é muito utilizado na navegação, dado ao seu potencial de detecção de obstáculos, porém o seu uso está mais orientado para um perfil marítimo dado à propagação do som na água ser muito superior (atingindo facilmente os 1450m/s) à propagação do mesmo no ar (rondando os 346m/s a 25ºC).

Esta capacidade de detecção de objectos é utilizada na natureza, nomeadamente por golfinhos e morcegos, no entanto nenhum dispositivo criado até à data consegue igualar as capacidades destes no que respeita à eco localização.

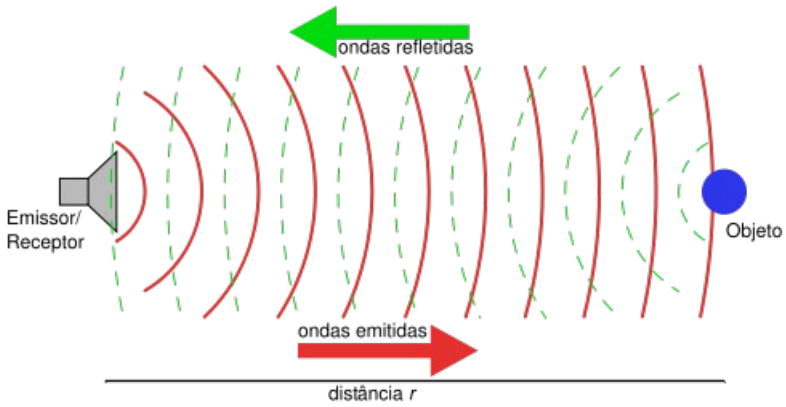


Figura 3.2: Esquema simplificado do sonar

O "*CleanRob*" possui um conjunto de seis sonares SRF08 (figura 3.1). O objectivo principal destes sensores foi o de tentar capturar uma "fotografia" do mundo que rodeia o robô à custa de sensores de baixo custo. Para que a fusão de informação conseguisse gerar uma posição ou um conjunto de posições prováveis para a localização do robô. Como referido anteriormente pretende-se compensar a imprecisão dos sensores através do uso de um elevado número de sensores de baixo custo. A conjugação da informação redundante fornecida pelos sensores, leva-nos a atingir uma solução correcta.

Contudo não é suficiente ter um conjunto considerável de sensores de baixo custo, é necessário conseguir determinar o quanto imprecisos estes são, quais as suas limitações e problemas.

Na figura 3.3, temos uma representação da direcionalidade do eco do sonar, sendo apresentado uma abertura de cerca de 110º.

Figura 3.3: Dispersão e ganho da onda de ultra-som (40Khz) do sonar SRF08

3.1.1 Modelo

Para a obtenção do modelo do sonar conduziu-se a duas experiências, num ambiente controlado, num espaço amplo e estático, sem que houvesse interferência de objectos nas mediações do sonar com a excepção do objecto alvo de testes. Foi posicionado o sonar a cerca de 40cm do piso, numa posição paralela à superfície.

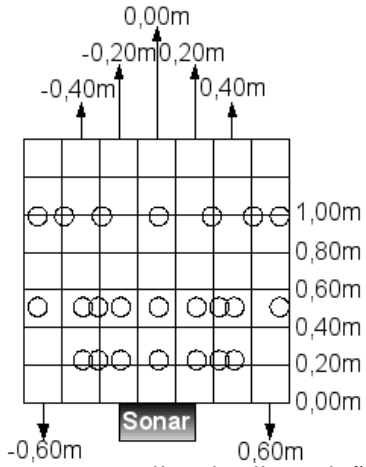


Figura 3.4: Grelha da disposição do objecto cilíndrico de testes em relação ao sonar

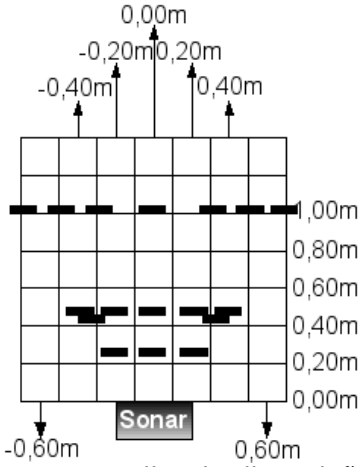


Figura 3.5: Grelha da disposição do objecto plano de testes em relação ao sonar

Numa primeira fase foram registadas cerca de 380 medidas a um objecto cilíndrico de acrílico com um diâmetro de 10cm por uma altura aproximada de 50cm posicionado em diferentes distâncias do sonar (figura 3.4). Na segunda fase foram registadas cerca de 340 medidas a um objecto com uma superfície lisa de madeira contraplacada com uma altura de 50cm e uma largura de 10cm, sendo este colocado com também a diferentes distâncias, conforme a figura 3.5 demonstra.

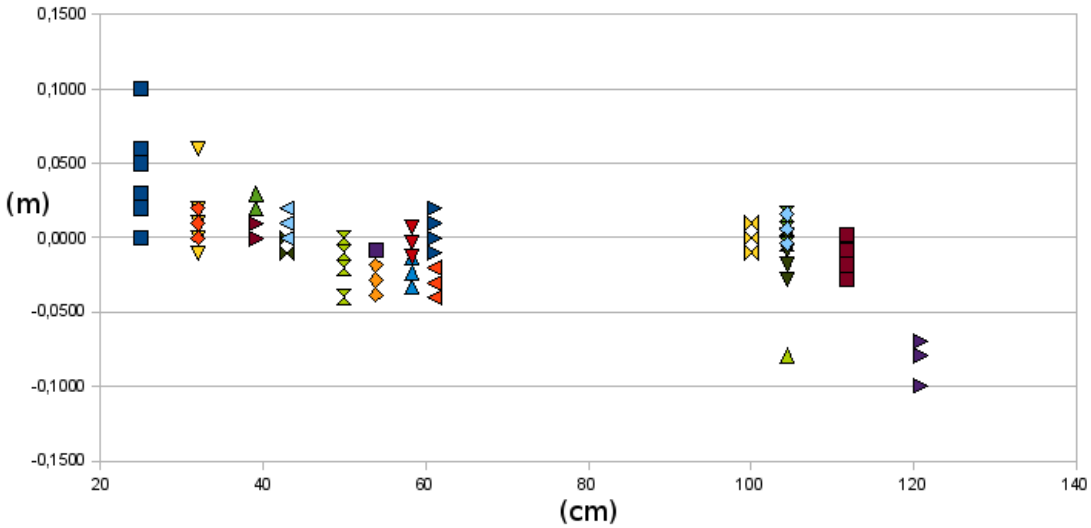


Figura 3.6: Representação gráfica dos erros absolutos em relação à distância no teste com um cilindro de acrílico

Nas figuras 3.6 e 3.7, encontram-se gráficos que representam os erros absolutos em relação às distâncias no qual foram efectuadas as medições. Na experiência realizada com o cilindro, observa-se que os piores resultados resultam quando o objecto está na proximidade do sonar, embora existam resultados fracos quando este se situa a cerca de 1,20m, ao qual já se deve a reflexões do eco proveniente da reflexão no piso.

Contudo é seguro afirmar que para uma gama de valores entre 0cm a 120cm o erro situa-se nos 10cm, por outras palavras existe um erro percentual de 8,3%, o que torna claro que este sensor não é o ideal para medidas de distâncias a paredes quando comparados com outros sensores igualmente de baixo custo.

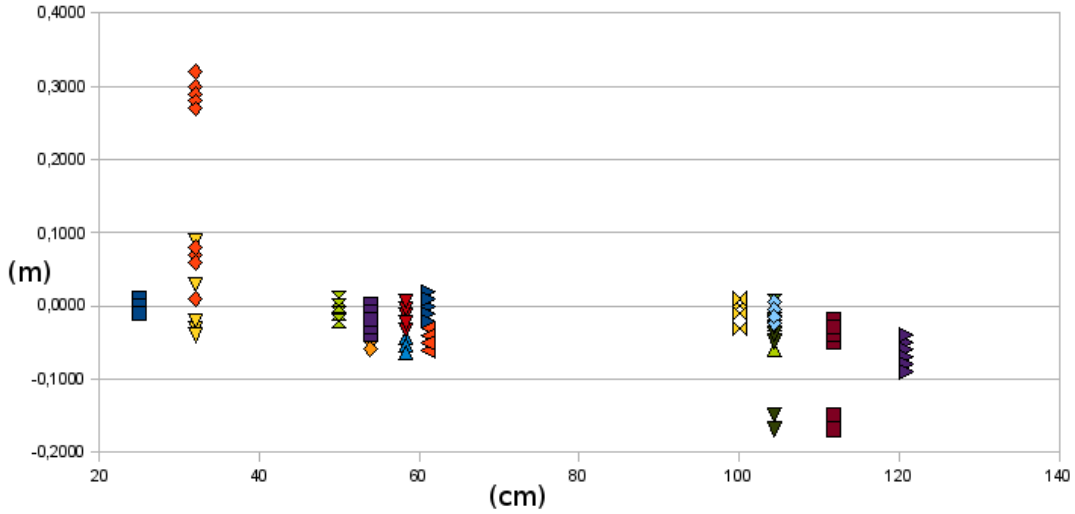


Figura 3.7: Representação gráfica dos erros absolutos em relação à distância no teste com uma placa de contra-placado

Quando analisado os erros correspondentes ao teste com a placa de contra-placado, a análise demonstra a mesma evolução, com a excepção de algumas medições em que a placa reflecte os ecos recebidos para uma direcção diferente da original, originando assim o efeito de deflexão. Não considerando as medidas sujeitas a este tipo de efeito, o erro é semelhante à experiência anterior, havendo um erro acentuado quando o objecto está próximo e estando limitado a um erro de 10 cm na gama de valores entre 0cm a 120cm.

Modelo do sonares	
[m / m]	**Rho**
Média	0,0088
Desvio padrão	0,0579

Tabela 1: Modelo do erro da distância do sonar SRF08

3.2 Conclusões

Este tipo de sonar comunica por interface "*I2C*" e possibilita uma certa variedade de opções a nível de configuração do alcance e ganho, contudo constatou-se que mesmo provocando um ganho nulo através da sua configuração, este tipo de sonar ainda possuía uma sensibilidade razoável, tornando-se indesejável no disparo dos seis sonares em conjunto já que origina interferência na captação de ecos entre eles, implicando assim medidas incorrectas. Por outro lado o seu grande ângulo de dispersão do eco, torna-o ideal para a detecção de obstáculos, que o deixam à mercê de irregularidades no solo a uma curta distância, dando origem a medidas incorrectas.

Uma outra desvantagem do uso de sonares é o tempo de aquisição de uma medida, para um alcance de 6m o tempo de aquisição é de 65ms, para ir de encontro com a abordagem de já referida de inúmeros sensores e dado à interferência entre estes é necessário, criar uma relação de compromisso, reduzindo o alcance destes, bem como desenvolver métodos que permitam o disparo numa sequência alternada de forma a que não haja interferências entre os ecos de cada sonar na tentativa de reduzir o tempo de aquisição.

Face a estas desvantagens todas, porque é que se utilizará este tipo de sensor? Como referido anteriormente este é um sensor ideal para a detecção de obstáculos, podendo ser usado pelo “*path planning*” na decisão de alteração de rotas para situações anómalas, porém como o objectivo da fusão de informação é a aquisição de todo o tipo de informação boa e menos boa de forma a combinar uma informação útil.

Capítulo 4

Estratégias de Auto-localização

4.1 Introdução

Os robôs móveis tem vindo a mostrar um significativo potencial na execução de tarefas que são demasiado simples e rotineiras, perigosas ou difíceis ou até com um custo demasiado elevado para o ser humano realizá-las, pelo que o espectro de operações realizadas por robôs tem vindo a largar de dia para dia. Desde das operações mais rotineiras como operações de busca e salvamento, reconhecimento aéreo ou de navios afundados, podemos adicionar operações como supervisão de tráfico aéreo, localização de vitimas, navegação de áreas contaminadas com resíduos perigosos ou radiação, vigilância de edifícios como estádios, aeroportos, estações de comboio ou metropolitano entre muitos outros.

Na maior parte dos casos, não é possível a manutenção ou intervenção humana quer devido à distância envolvida ou devido a problemas de comunicações via rádio ou por cabo que possam existir. Um outro problema reside de não estar disponível informação relativa ao meio em que o robô se deslocará, como por exemplo as ruínas de um edifício.

Devido a estes factores é necessário que o robô seja autônomo, sem que seja necessário recorrer ao controlo, supervisão, ajuda humana. Tais controlos requerem um algoritmo de "*SLAM*"[2].

Estes algoritmos referem-se ao problema em questão de um robô deslocando-se num ambiente cujo o mapa não é disponibilizado. Tais algoritmos usam o ruído inerente às medidas que o robô obtém, como hodometria, medidas de distâncias a objectos e sinais de "*GPS*"[3] se disponíveis e constrói uma representação real do mundo (processos concorrentes). Esta capacidade de construir mapas permite ao robô operar com o mínimo de manutenção e adaptação originando uma vantagem quando o cenário é alterado.

Este tipo de método alarga o espectro de utilização de robôs. Tem vindo a ser culminados esforços no uso destas técnicas e no de robôs com o intuito de construírem modelos precisos 3D de minas e estruturas subterrâneas, permitindo assim uma maior precisão no processo de abrir túneis e evitando acidentes de desmoronamento de terras devido a mapas incorrectos.

Após ou enquanto os mapas são criados, existe a necessidade de uma estimação precisa da "*pose*"[4] do robô. Factor imperativo para qualquer tipo de planeamento de trajectórias e algoritmos de navegação. O grau de dificuldade pode ser elevado ou baixo conforme o tipo, quantidade e precisão de sensores que o robô consegue embarcar.

Existem vários graus de auto-localização. Os mais básicos e comuns são os baseados em monitorização de posição, onde a posição inicial é conhecida até um certo ponto e em que o problema consiste em compensar erros de hodometria do robô como deslizamentos, resolução e outros.

Numa abordagem mais abrangente, temos a localização global, onde o robô não conhece a sua localização inicial e portanto tem que descobrir por si mesmo.

Num grau de dificuldade maior, é o caso do rapto[5] do robô, Neste caso, o robô deverá reconhecer esta situação e iniciar procedimentos de forma a conseguir descobrir a sua nova posição.

E sendo o caso mais complexo de todos, o caso de um robô inserido num ambiente dinâmico, existindo movimentação de pessoas, alterações físicas do ambiente. Estes tipo de características de um

2 Simultaneous Localization And Mapping
3 Global Positioning System
4 Designação dada a uma colecção de informação que define a posição exacta do robô. Num robô móvel terrestre é definida por $X=[x,y,\theta]^T$
5 Esta designação origina do teste que normalmente se realiza

ambiente dinâmico, levam ao robô confundir pessoas por exemplo com paredes que supostamente não deveriam estar presentes originando assim o robô actuar numa situação de rapto quando este na realidade não existiu.

A maioria dos métodos, como Filtros de Kalman, usam distribuições condicionadas por medidas de sensores, tirando vantagens das restrições por parte da incerteza inicial da distribuição gaussiana, permitindo assim a aplicabilidade da monitorização de problemas de posição. Contudo estas mesmas restrições tornam impraticável numa localização global.

Quer a restrição do uso de distribuições gaussianas, ou a localização global ou o do caso do rapto são casos resolvidos pelo Filtro de Partículas que lida com diferentes distribuições gaussianas e não-linearidades de modo a adequarem-se melhor ao problema em questão [3].

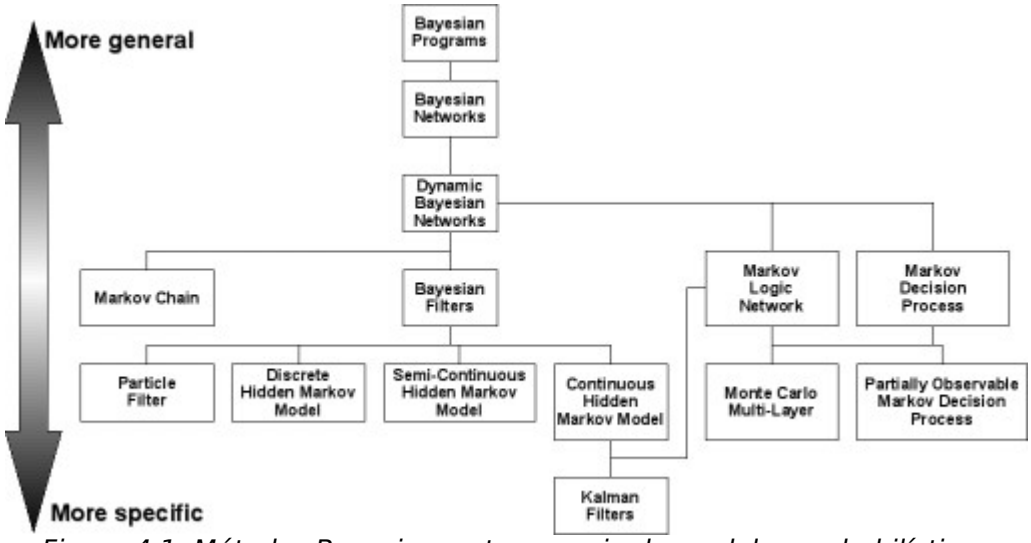


Figura 4.1: Métodos Bayesianos: taxonomia de modelos probabilísticos

Na figura 4.1, encontra-se uma representação esquemática dos diversos métodos existentes e respectiva relação entre os mesmos.

4.2 Modelo do Sistema em Espaços de Estados

Um sistema discreto é usualmente descrito como um modelo em espaços de estados. Este consiste em duas equações principais que representam a evolução do sistema. A primeira equação (4.1) descreve como o sistema evolui face ao estado actual e a uma entrada.

$$X_k = AX_{k-1} + BU_{k-1} \tag{4.1}$$

X_k é o modelo do estado no instante k

A representa como o sistema evolui por si mesmo

B representa como as entradas afectam o estado

U_{k-1} é a entrada do sistema no instante $k-1$

em que

$$X_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} \tag{4.2}$$

e

$$U_{k-1} = \begin{bmatrix} v_{x\,k-1} \\ v_{y\,k-1} \\ w_{k-1} \end{bmatrix} = \begin{bmatrix} v_{k-1} * \cos(\theta_{k-1}) \\ v_{k-1} * \sin(\theta_{k-1}) \\ w_{k-1} \end{bmatrix} \tag{4.3}$$

Contudo a equação 4.1 não contempla os casos mais realistas. Para tal é necessário adicionar um terceiro membro à equação que significará o ruído. De forma a que o problema seja tratável, o ruído deverá ser nulo, invariante no tempo e seguindo uma distribuição gaussiana, pelo que teremos em combinação com a equação 4.1:

$$X_k = AX_{k-1} + BU_{k-1} + W_{k-1} \tag{4.4}$$

W_{k-1} representa o ruído no instante $k-1$

A segunda equação (4.5) é a formula de estimação de medidas. Esta permite calcular/prever as medidas obtidas pelo robô num determinado estado, tomando em conta o estado do sistema.

$$Z_k = HX_k + V_k \tag{4.5}$$

Z_k são as medidas no instante k

X_k é o modelo do estado no instante k

H é a matriz das medidas

V_k é o ruído das medidas no instante k

4.3 Filtro Bayesiano

4.3.1 Introdução

Para controlar o comportamento de um sistema é necessário conhecer o estado de certas variáveis conhecidas como variáveis de estado. Por vezes essas variáveis não são acessíveis ou não podem ser medidas, pelo que é necessário estimá-las a partir das entradas e saídas do sistema. Isto pode rapidamente tornar-se numa tarefa complexa devido ao sistema ser conduzido por outras acções que não as previamente conhecidas ou previsíveis. A dedução bayesiana é um método estocástico que usa evidências ou observações para construir probabilidades sobre uma certa hipótese.

Com o evoluir do tempo e estando submetidas às diversas observações as probabilidades irão convergir, havendo probabilidades mais elevadas e outras mais baixas. No início de cada hipótese é dado uma estimação numérica que reflecte a confiança desta. A esta confiança é normalmente associada a uma função densidade de probabilidade. Então esta função densidade de probabilidade é constantemente e recursivamente actualizadas, por dois procedimentos, de acordo com as entradas dadas e a informação recolhida.

Normalmente, o sistema não possui qualquer ideia do estado inicial e por isso a probabilidade é normalmente uniforme distribuída pelo universo de possíveis estados. Contudo se um estado é conhecido, mesmo que só até um certo ponto, o sistema poderá ser inicializado com qualquer outra distribuição que melhor se adequa à informação disponível.

A função densidade de probabilidade representa um universo de possíveis estados, com base em entradas anteriores, estados e observações. Neste processo o próximo estado depende apenas do estado presente e de entradas, não tendo dependência directa a estados ou a entradas anteriores. Isto significa que o sistema é isento de memória, isto é, sistema não se lembra dos estados anteriores, apenas sabe o estado actual e é com base neste que será determinado estado futuro. Isto é um sistema de primeira ordem do processo de Markov.

$$p(x)=p(\vec{x_k}|Y_{k-1}) \tag{4.6}$$

$\vec{x_k}$ é o estado actual no instante (k)

Y_{k-1} informação da acção e observação anteriores

$p(x)$ é a distribuição de probabilidade do estado x

O algoritmo Bayesiano actualiza recursivamente a função densidade de probabilidade por intermédio de dois processos, a previsão e actualização como esquematizado na figura 4.2.

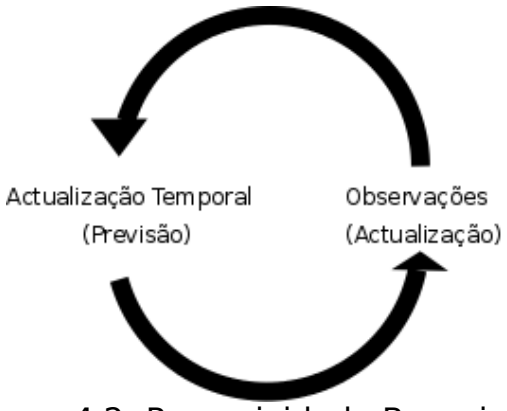


Figura 4.2: Recursividade Bayesiana

4.3.2 Actualização temporal na Filtro Bayesiano

É uma função periódica responsável pela previsão em cada iteração de um novo estado baseada na última acção realizada e no estado actual. Nesta fase de previsão é adicionado ruído, normalmente gaussiano, à entrada do sistema, resultando numa distribuição de probabilidade que reflecte todo o conjunto provável inerente à realização da acção. Tal distribuição representa o ruído introduzido pelo processo de movimentação, por exemplo o deslizamento lateral do robô por assimetria de peso ou o deslizamento do ponto de contacto da roda com o piso. Estes tipos de ruído provocam que nas diversas fases de previsão, o conhecimento preciso da posição do robô, seja cada vez menor, aumentando o número de posições possíveis para o mesmo, isto reflecte-se no movimento de translação e de rotação, como se pode observar através das figuras 4.3 e 4.4.

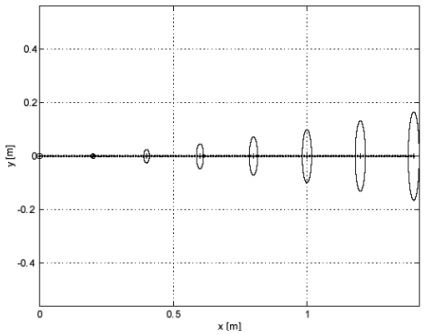


Figura 4.3: Exemplo da propagação de erros de translação e de deslizamento na hodometria

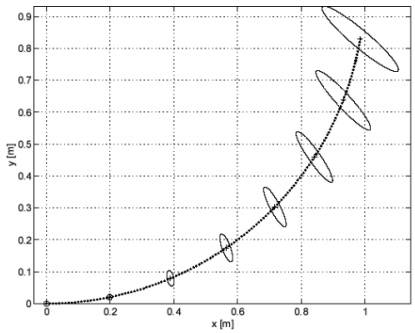


Figura 4.4: Exemplo da propagação de erros de rotação na hodometria

Podemos então chegar à equação da previsão aplicando algumas transformações à equação 4.6.

Começando com:

$$p(\vec{x}_k \cap Y_{k-1}) = \int_{\vec{x_{k-1}}} p(\vec{x}_k \cap \vec{x_{k-1}} \cap Y_{k-1})\, d\vec{x_{k-1}} \tag{4.7}$$

e aplicando a regra de Bayes $p(A \cap B) = p(A) * p(B|A) = p(B) * p(A|B)$,

$$p(\vec{x}_k \cap Y_{k-1}) = \int_{\vec{x_{k-1}}} p(\vec{x}_k | \vec{x_{k-1}} \cap Y_{k-1}) * p(\vec{x_{k-1}} \cap Y_{k-1})\, d\vec{x_{k-1}} \tag{4.8}$$

aplicando novamente a regra de Bayes ao segundo membro,

$$p(\vec{x}_k \cap Y_{k-1}) = \int_{\vec{x_{k-1}}} p(\vec{x}_k | \vec{x_{k-1}} \cap Y_{k-1}) * p(\vec{x_{k-1}} | Y_{k-1}) * p(Y_{k-1})\, d\vec{x_{k-1}} \tag{4.9}$$

agora é possível retirar o terceiro membro do integral dado que é independente de $\vec{x_{k-1}}$

$$\frac{p(\vec{x}_k \cap Y_{k-1})}{p(Y_{k-1})} = \int_{\vec{x_{k-1}}} p(\vec{x}_k | \vec{x_{k-1}} \cap Y_{k-1}) * p(\vec{x_{k-1}} | Y_{k-1})\, d\vec{x_{k-1}} \tag{4.10}$$

agora, aplicando o teorema de Markov ($p(\vec{x}_k | \vec{x_{k-1}}, Y_{k-1}) = p(\vec{x}_k | \vec{x_{k-1}})$), obtemos

$$p(\vec{x}_k \cap Y_{k-1}) = \int_{\vec{x_{k-1}}} p(\vec{x}_k | \vec{x_{k-1}}) * p(\vec{x_{k-1}} | Y_{k-1})\, d\vec{x_{k-1}} \tag{4.11}$$

que é a base para o processo de previsão. Desta forma cada estado é periodicamente propagado a atingir uma nuvem de novos possíveis estados. O erro neste processo caracteriza-se por um acumulativo dado que a previsão é baseada na "*pose*" actual que por sua vez fora prevista com um determinado erro. Ao fim de algum tempo de previsões o erro torna-se excessivamente elevado, que arruína qualquer previsão. Desde modo é necessário incluir algum tipo de medida global de posicionamento, semelhante à função de um farol para um navio junto à costa, que possibilite a este descobrir as suas

coordenadas através da triangulação e assim actualizar a sua previsão da posição, concentrando a nuvem de possíveis estados numa esperada área menor e mais precisa. Isto é conseguido através das observações.

4.3.3 Actualização das observações no Filtro Bayesiano

O processo de actualização de observações normalmente não é periódico como o processo de previsão. De facto pode ser executado várias vezes numa iteração ou até nem ser executado. Este procedimento é responsável pela actualização dos estados originados pela previsão, reunindo quaisquer informação obtida. Por exemplo, robôs moveis tem diversos sensores de posicionamento ou de movimentação tais como "*encoders*", acelerómetros, câmaras, sensores de distâncias por IR, sonares de forma a obter uma posição absoluta ou relativa. Esta informação irá ser fundida com a probabilidade dos estados previamente originados pela previsão em ordem a uma função densidade de probabilidade final para a iteração.

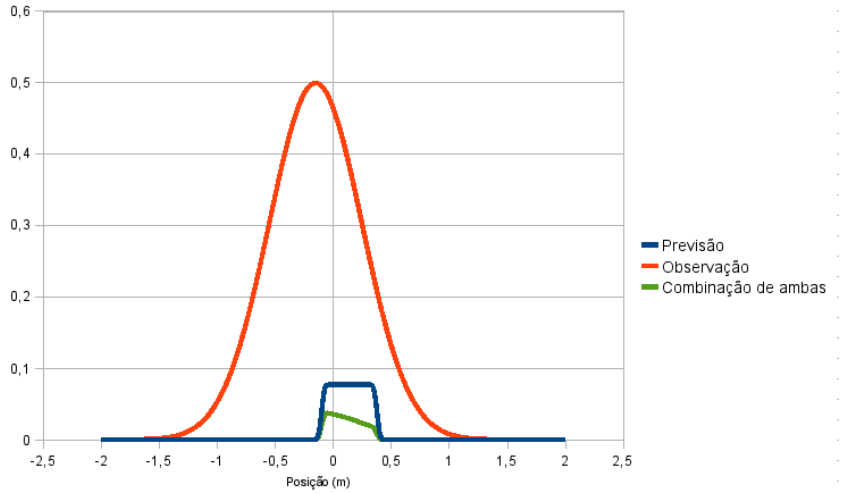


Figura 4.5: Exemplo gráfico da fusão de informação a 1D

Podemos então obter a equação da actualização das observações $p(\vec{x}_k|Y_k)$ através da equação previsão usando $\vec{y}_k$.

$$p(\vec{x}_k|\vec{y}_k \cap Y_{k-1}) = \frac{p(\vec{x}_k \cap \vec{y}_k \cap Y_{k-1})}{p(\vec{y}_k \cap Y_{k-1})} \tag{4.12}$$

Dado que

$$p(\vec{x}_k \cap \vec{y}_k \cap Y_{k-1}) = p(\vec{y}_k|\vec{x}_k \cap Y_{k-1}) * p(\vec{x}_k \cap Y_{k-1}) \tag{4.13}$$

e

$$p(\vec{y}_k \cap Y_{k-1}) = p(\vec{y}_k|Y_{k-1}) * p(Y_{k-1}) \tag{4.14}$$

obtendo-se:

$$p(\vec{x}_k|Y_k)=\frac{p(\vec{y}_k|\vec{x}_k\cap Y_{k-1})*p(\vec{x}_k\cap Y_{k-1})}{p(\vec{y}_k|Y_{k-1})*p(Y_{k-1})} \quad (4.15)$$

$$p(\vec{x}_k|Y_k)=\frac{p(\vec{y}_k|\vec{x}_k\cap Y_{k-1})*p(\vec{x}_k\cap Y_{k-1})}{p(\vec{y}_k\cap Y_{k-1})} \quad (4.16)$$

A informação contida em Y_{k-1} já se encontra condensada em $\vec{x}_k$ e por isso

$$p(\vec{y}_k|\vec{x}_k\cap Y_{k-1})=p(\vec{y}_k|\vec{x}_k) \quad (4.17)$$

$$p(\vec{x}_k\cap Y_k)=\frac{p(\vec{y}_k|\vec{x}_k)*p(\vec{x}_k\cap Y_{k-1})}{p(\vec{y}_k\cap Y_{k-1})} \quad (4.18)$$

Esta é a equação para a actualização das observações em que o denominador actua como um factor de normalização.

Estes filtros são muito robustos e facilmente adaptáveis a uma grande variedade de problemas e sensores. Uma vez mais, podemos alterar as distribuições de densidade de forma a descrever melhor o sensor, realimentação ou ruído de processos.

É um método muito preciso, quase exacto, porém o custo computacional é enorme, tornando-se impraticável para o uso de robôs móveis).

De forma viabilizar esta solução a solução é discretizar a distribuição e usar amostras invés de um espaço continuo.[3]

4.4 Filtro de Partículas

4.4.1 Introdução

O Filtro de Partículas é uma aproximação da implementação discreta do Filtro Bayesiano. Nos filtros de partículas, a função densidade de probabilidade é representada por um conjunto de objectos com um determinado peso, denominados de partículas, invés de uma distribuição continua como o Filtro Bayesiano. Quanto mais partículas se recorrer, mais o sub-espaço aproximará a um espaço contínuo e consequentemente se obterá uma maior aproximação, porém com um custo computacional maior.[3]

Cada vez mais em inúmeras aplicações, tem sido importante incluir elementos não-lineares e não-gaussianos de modo a modelizar

correctamente um sistema físico. Por tal é fundamental processar a informação de um modo contínuo, à medida que esta é disponibilizada. O filtro de partículas tem sido alvo de aplicação em diversas áreas tal como reconhecimento de voz, seguimento de objectos em processamento de imagem, fusão de voz e vídeo para reconhecimento de palavras e até no campo da medicina no que diz respeito à previsão da evolução do cancro.

O principal objectivo do filtro de partículas é criar uma representação do estado do mundo, através de inúmeras partículas que polvilham o mesmo, podendo esta representação assumir distribuições não-gaussianas e potencialmente multi-modais. Através de acções e observações estas partículas evoluem modificando a representação do mundo. Desta forma a evolução passa pela extinção, nascimento ou sobrevivência das mesmas, possibilitando neste último caso a possibilidade de se gerarem diversas cópias das mesmas, conforme a sua contribuição na representação do mundo, como forma de propagar a sua relevância no mundo.[4]

Cada partícula é definida por 4.19 no instante de tempo $t=k$ de uma população 4.20 com M elementos. A partícula então representa uma variável de interesse definida por 4.21 (definição de "*pose*" de um robô) e a sua contribuição w na representação global do mundo ou muitas das vezes também definido por "*belive*".

$$S_j^k=[X_j^k, w_j^k]^T \tag{4.19}$$

$$S_j^k : j=1 \ldots M \tag{4.20}$$

$$X^k=[x, y, \theta]^T \tag{4.21}$$

O filtro de partículas é um método recursivo por natureza e opera em duas fases denominadas de previsão e actualização. Assim, após uma acção física, todas as partículas são afectadas pela mesma por um modelo matemático da acção e é introduzido ruído aleatório de forma a simular uma situação mais próxima da realidade. Esta fase é chamada de previsão ("*prediction*").

Então cada partícula é reavaliada com base nas observações obtidas através da última sensorização disponível, sendo recompensadas as partículas que vão de encontro com a "visão" do mundo fornecida pela sensorização e penalizadas as que pouco tem em comum através do factor w. Esta fase é denominada de actualização ("*update*").

Contudo, devido à recursividade do filtro de partículas, existirão inúmeras situações em que certas partículas atingirão uma contribuição ou "*belive*" (w) infinitesimais e outras partículas assumindo valores

bastante elevados, pelo que é necessário introduzir um processo de reamostragem (“*resampling”).* O processo de reamostragem tem como finalidade permitir as partículas que não atinjam uma contribuição miníma, um sorteio que tem como objectivo a decisão a sua extinção ou sobrevivência no mundo.

Em contrapartida às partículas que tenham uma grande contribuição na representação do mundo poderão ser criadas n cópias idênticas. Cada uma destas cópias evoluirá de forma diferente ao longo do tempo, devido à modelização das acções e consecutivamente à adição de ruído, tendo o processo de reamostragem a função de propagar ou extinguir a continuação da partícula no mundo, modificando assim representação do mundo como descrito anteriormente.

Todo este processo, fluirá de forma a convergir a representação do mundo com a realidade, sendo porém é necessário estimar a variável de interesse. Para o processo de estimação existem três métodos descritos em [4] sendo estes:

- Melhor Partícula - “*Best Particle*”

$$\bar{X}_s = X_s^{max} \tag{4.22}$$

- Média Ponderada - “*Weighted Mean*”

$$\bar{X}_s = \sum_{j=1}^{M} X_s^j * w_j \tag{4.23}$$

- Média Robusta - “*Robust Mean*”

$$\bar{X}_s = \sum_{j=1}^{L} X_s^j * w_j : |X_s^j - X_s^{max}| \leq \epsilon \tag{4.24}$$

Cada um destes métodos possui as respectivas vantagens e desvantagens sendo o primeiro o método de estimação mais rápido, mas com um erro maior de estimação, o segundo é um método mais correcto de realizar uma estimação, mas com um custo computacional maior que o anterior e com a possibilidade de criar uma estimação correspondente a uma zona inválida[6]. Por último, o terceiro método possibilita realizar a junção do melhor dos dois mundos, um menor erro na estimação dado que as partículas tenderam aglomerar-se numa "*pose*", porém leva um custo computacional elevado, não estando livre da desvantagem inerente ao segundo método referente à estimação de

6 Zona inválida significa uma área em que é fisicamente impossível para o robô estar posicionado quer por limitação física do mesmo ou por limitação do mapa, nestas situações enquadram-se as possibilidade do robô se situar dentro de uma parede ou numa abertura no piso.

uma "*pose*" numa área inválida.

Um exemplo típico do filtro de partículas é o caso de um robô equipado com um detector de portas e possui informação de hodometria, que se desloca unicamente um eixo paralelo à parede que têm três portas. Dado que o filtro de partículas requer que o ambiente seja conhecido é necessário fornecer-lhe o mapa correspondente (figura 4.6).

Inicialmente todas as partículas tem um peso igualmente distribuído, pelo que o robô desconhece a sua posição e o algoritmo de "*path planning*" ordena que este avance na esperança de obter mais informações numa posição diferente.

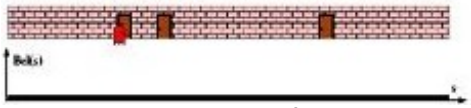


Figura 4.6: Exemplo nº1 do Filtro de Partículas - Ambiente unidimensional

Após o robô ter-se deslocado, o sensor detectou a existência de uma porta, porém como a maior parte dos sensores este não consegue informar qual das portas detectou esse tipo de informação cabe ao filtro de partículas descobrir. Dado que o filtro de partículas possuí o mapa do ambiente no qual o robô se movimenta, este através da actualização recompensa as partículas que possuem informação que colaborem com a informação obtida através da observação e penalizando as outras. Este tipo de decisão provocará como constatado na figura 4.7, o aparecimento de três zonas com igual probabilidade por outras palavras, o filtro de partículas estima que o robô poderá estar posicionado em três localizações diferentes, mas sem mais informação é neste momento o melhor que o filtro pode estimar.

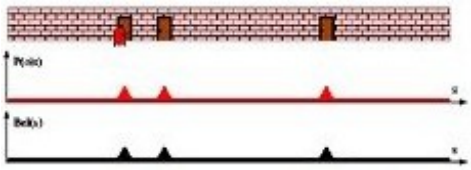


Figura 4.7: Exemplo nº2 do Filtro de Partículas - Ambiente unidimensional

Sem melhor informação o algoritmo de "*path planning*", decide retomar a decisão tomada no momento anterior, continuar a deslocar o robô de forma a obter mais informação do meio que o rodeia. Neste momento as partículas também são deslocadas de acordo com o modelo da hodometria, porém como esta informação não é precisa as partículas dispersam também com o objectivo de simular o comportamento da acção realizada, como observamos para figura 4.8.

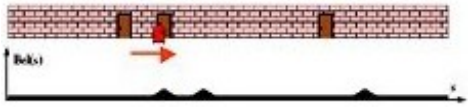


Figura 4.8: Exemplo nº3 do Filtro de Partículas - Ambiente unidimensional

Após o deslocamento do robô, o sensor detectou outra porta, mais uma vez não conseguindo fornecer mais nenhuma informação extra, devido à sua natureza limitada, porém o filtro de partículas numa nova fase de actualização já consegue estimar a posição possível do robô, combinando o conhecimento anterior com o actual, ou seja, através da modificação dos pesos das partículas a quando da detecção da primeira porta, com a informação fornecida pela hodometria do quanto o robô se deslocou e a obtenção de uma nova informação (detecção de outra porta), consegue-se recompensar as partículas que vão de encontro com este tipo de observações e acções, pelo que verifica-se um conjunto de partículas com probabilidades bastante elevadas na posição relativa à segunda porta como representado na figura 4.9. Neste momento podemos considerar que o robô conhece a sua posição.

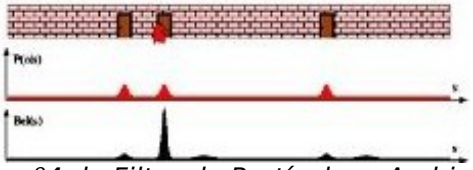


Figura 4.9: Exemplo nº4 do Filtro de Partículas - Ambiente unidimensional

A partir do ponto que o robô conhece a sua posição relativamente ao mapa, o algoritmo de "*path planning*", decide agora percorrer o resto do corredor. Neste momento dado que não existe informação sensorial o filtro de partículas fica reduzido apenas à previsão actualizando as partículas de acordo com a acção que o robô realiza. Não obstante, à medida que não vai existindo informação sensorial e consequentemente não é realizada observações as partículas irão divergindo de forma a tentar cobrir todas as possibilidades existentes de localização do robô, tornando assim cada vez menos exacta a posição deste, como verificado pela figura 4.10

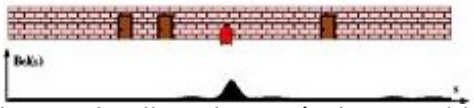


Figura 4.10: Exemplo nº5 do Filtro de Partículas - Ambiente unidimensional

4.4.2 “*Prediction*”

De modo a simular de forma adequada uma acção desempenhada no mundo físico é necessário modelizar matematicamente a acção desempenhada. No caso de um robô móvel, esta acção será a sua movimentação e o modelo será regido por equações de movimento linear. Estas equações tem como base o movimento físico, porém as suas entradas provêm de informação adquirida por sensores,

nomeadamente “*encoders*”, em que estão associados a um erro de quantificação[7]. Os erros determinísticos podem ser eliminados por calibração do sistema, porém os erros não determinísticos tem que ser descritos por modelos de erros que sempre levarão a estimativas de posição incertas.

Existem muitas abordagens para a minimização do impacto deste tipo de erro [5] [6], a maior parte usa a adição de um modelo gaussiano do ruído para a movimentação.

Para simplificar este problema dividiremos a movimentação em duas paramétricas independentes, cada uma sendo é responsável por um dos movimentos, definiremos uma paramétrica pela translação parcial definida por $\delta\rho_k$ e outra paramétrica responsável pela rotação parcial definida por $\delta\theta_k$, ambos nos instante de tempo $t=k$.

$$X_{k-1}=[x_{k-1}, y_{k-1}, \theta_{k-1}]^T \tag{4.25}$$

Assumindo que a "*pose*" inicial definida em 4.25 e que os dados obtidos pela hodometria (Ox_k, Oy_k) fornecem uma estimação das distâncias absolutas percorridas em cada uma das componentes do sistema de eixos, poderemos então definir:

$$\begin{cases} \Delta x_k = Ox_k - Ox_{k-1} \\ \Delta y_k = Oy_k - Oy_{k-1} \\ \Delta\theta_k = arctan2(\Delta y/\Delta x) \end{cases} \tag{4.26}$$

Desenvolvendo um pouco mais, obtemos então as componentes da translação parcial 4.27 e rotação parcial 4.28:

$$\delta\rho_k = \sqrt{(\Delta x_k)^2 + (\Delta y_k)^2} \tag{4.27}$$

$$\delta\theta_k = \Delta\theta_k - \theta_{k-1} \tag{4.28}$$

Reunindo as equações 4.27 e 4.28, obtemos a matriz que define o movimento linear do robô 4.29, que usaremos para modelizar a acção que este desempenhará na sua movimentação prevendo para cada partícula existente na representação da função densidade de probabilidade. No entanto ainda é necessário adicionar o modelo do ruído como descrito anteriormente, pelo que a modelização mais adequada à realidade será dada pela matriz 4.30.

7 Os principais fontes dos erros de quantificação estão associados à resolução limitada durante a integração (incremento de tempo, resolução medida), rodas desalinhadas (distância entre rodas variável), rodas com diâmetros diferentes, variação do ponto de contacto com o terreno e a roda, contacto desigual com o chão (deslizamento).

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + \delta\rho_k * \cos(\theta_k) \\ y_{k-1} + \delta\rho_k * \sin(\theta_k) \\ \theta_{k-1} + \delta\theta_k \end{bmatrix} \tag{4.29}$$

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + (\delta\rho_k + randN(\delta\rho_k * M_{trs}, \delta\rho_k * \sigma_{trs})) * \cos(\theta_k) \\ y_{k-1} + (\delta\rho_k + randN(\delta\rho_k * M_{trs}, \delta\rho_k * \sigma_{trs})) * \sin(\theta_k) \\ \theta_{k-1} + \delta\theta_k + randN(\delta\theta_k * M_{rot}, \delta\theta_k * \sigma_{rot}) + randN(\delta\rho_k * M_{drft}, \delta\rho_k * \sigma_{drft}) \end{bmatrix} \tag{4.30}$$

A função "*randN*" é uma função de geração de números aleatórios com base na distribuição normal gaussiana. Esta foi escolhida para a modelação do ruído proveniente da hodometria dado que é a mais adequada face aos resultados experimentais obtidos aquando da calibração desta. Para a modelização de qualquer outra acção é conveniente usar uma distribuição adequada, (não sendo de carácter obrigatório o uso da distribuição normal gaussiana) dado a não existir qualquer restrição por parte do filtro de partículas neste aspecto.

Na matriz 4.30, existem três modelos matemáticos de ruído proveniente da hodometria, sendo este categorizados nas seguintes formas:

- Incerteza proveniente da rotação

$$randN(\delta\theta_k * M_{rot}, \delta\theta_k * \sigma_{rot}) \tag{4.31}$$

A função 4.31 caracteriza o desvio e respectivos erros existentes aquando da rotação do robô em torno do seu centro, numa relação estrita com o deslocamento parcial $\delta\theta_k$ em que $t=k$. M_{rot} é um erro sistemático que teoricamente seria possível eliminar, mas face às dificuldades inerentes à calibração da hodometria, só é possível minimizá-lo, enquanto σ_{rot} é o desvio padrão que caracteriza a dispersão de valores prováveis em torno de M_{rot}.

- Incerteza proveniente da translação

$$randN(\delta\rho_k * M_{trs}, \delta\rho_k * \sigma_{trs}) \tag{4.32}$$

A função 4.32 caracteriza o desvio e respectivos erros existentes aquando da translação do robô, dado que a hodometria assume-se como tendo erros acumulativos tornando-se fácil prever a sua evolução. Porém tal com os erros de rotação, M_{trs} é um erro sistemático que é difícil de eliminar na prática, restando apenas a hipótese de minimizá-lo.

- Incerteza proveniente de deslizamentos

$$randN(\delta\rho_k * M_{drft}, \delta\rho_k * \sigma_{drft}) \quad (4.33)$$

A função 4.33 caracteriza o desvio e respectivos erros existentes a aquando da translação do robô afectando a orientação do mesmo. Esta função torna-se então a relação entre as duas paramétricas de movimento, não tendo qualquer sentido a nível físico dado que as duas paramétricas são independentes.

O autor [4] refere que seria aconselhável efectuar vários passos intermédios (K passos) no caso da modelização do deslizamento, sendo uma translação de 1m deveria ser constituída por exemplo 5 translações de 20cm cada em ordem a modelizar melhor o deslizamento.

O valor de K deveria ser escolhido de forma a manter a relação de compromisso de um peso computacional baixo com uma maior precisão na estimação do deslizamento. Para efectuar este método de melhor estimação do deslizamento, o autor [4] sugere ainda que os desvios padrões sejam afectados da seguinte forma:

$$\sigma'_{trs} = \sigma_{trs} * \sqrt{(K)} \quad (4.34)$$

$$\sigma'_{drft} = \sigma_{drft} * \sqrt{(K/2)} \quad (4.35)$$

e o deslocamento parcial

$$\delta\rho'_k = \rho_k / K \quad (4.36)$$

4.4.3 "*Update*"

"*Todas as vezes que empregarmos Matemática a fim de estudar alguns fenómenos de observação, deveremos essencialmente começar por construir um modelo matemático (determinístico ou probabilístico) para esses fenómenos. Inevitavelmente, o modelo deve simplificar as coisas e certos pormenores devem ser desprezados. O bom resultado do modelo depende de que os pormenores desprezados sejam ou não realmente sem importância na elucidação do fenómeno estudado. A resolução do problema matemático pode estar correcta e, não obstante, estar em grande discordância com os dados observados, simplesmente porque as hipóteses básicas feitas não sejam confirmadas. Geralmente é bastante difícil afirmar com certeza se um modelo matemático especificado é ou não adequado, antes que alguns dados de observação sejam obtidos. A fim de verificar a validade de um modelo, deveremos deduzir um certo número de consequências do nosso modelo e, a seguir, comparar esses resultados previstos com observações.*" [7]

Após a execução de uma acção, esta é normalmente seguida da fase de actualização ou "*update*". A função correspondente resulta na

aquisição de informação sensorial em que é combinada de forma a recompensar ou punir as partículas que estejam de acordo ou não com a informação sensorial existente no momento.

Quando se têm apenas uma observação ou um conjunto de observações, em que todas têm a mesma ordem de grandeza inseridas no mesmo contexto, é relativamente fácil realizar a actualização. Porém o grande problema impõem-se quando se utiliza diversas observações com naturezas diferentes, surgindo as seguintes questões: como combiná-las de forma a que todas possibilitem o uso de boa informação e de uma forma fácil? Como integrar dados de uma câmara com dados de sonares ou de bússola?

A esta questão a resposta por parte do filtro de partículas é dada através do recurso ao mundo das probabilidades, combinando assim toda a informação de um modo probabilístico e estabelecendo uma base comum. Aproveitando o facto das informações provenientes de cada observação serem independentes das outras é possível combiná-las, analisando cada uma conforme o contexto em que se insere, transformando assim a grande quantidade de informação em pequenas porções, possibilitando uma resolução simples e individual e por fim combinando-a por forma a ser transmitir uma aproximação matemática próxima da realidade.

Dada à abordagem ao mundo das probabilidades, é necessário ter em conta as propriedades que se regem.

Num um espaço amostral Ω e uma $\sigma-álgebra\ associada\ \Lambda$, uma função de probabilidade é uma função com domínio Λ (e com contradomínio $[0,1]$) que verifica:

- $0 \leq p(A) \leq 1,\ \forall A \in \Lambda$ (4.37)
- $p(\Omega) = 1$ (4.38)
- Se $A_1, A_2, \ldots A_n \in \Lambda$ e forem mutuamente exclusivos, então
  $p(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} p(A_i)$ (4.39)

Propriedades básicas

- $p(\emptyset) = 0$ (4.40)
- $p(\overline{A}) = 1 - p(A)$ (4.41)
- $p(A \cup B) = p(A) + p(B) - p(A \cap B)$ (4.42)
- Se $A \subset B$, então $p(A) \leq P(B)$ (4.43)

- $p(A \cap B) = p(A) * p(B|A)$, $p(A) \neq 0$ (4.44)
- $p(A \cap B) = p(A) * p(B)$ (Acontecimentos Independentes) (4.45)

- Para qualquer partição $B_1, B_2, \ldots, B_n$ (i.e., $B_i \cap B_j = \emptyset$ e $\bigcup_{i=1}^{n} B_i = \Omega$), $p(A) = \sum_{i=1}^{n} p(A \cap B_i)$ (4.46)
- Para quaisquer eventos $A_1, A_2, \ldots A_n$ temos $p(\bigcup_{i=1}^{n} A_i) \leq \sum_{i=1}^{n} p(A_i)$ (4.47)
- $p(B_i|A) = \frac{p(A|B_i) * p(B_i)}{\sum_{j=1}^{k} p(A|B_j) * p(B_j)}$, $i = 1, 2, \ldots, k$ (4.48)

Com o apoio a estas propriedades é possível então realizar a integração dos diferentes tratamentos de dados relativos a cada uma das observações realizadas com diferentes sensores. Este pormenor é a principal vantagem da fusão de informação. Sendo umas das grandes vantagem do filtro de partículas conseguir integrar informação de modelos matemáticos com base em diferentes distribuições de modo a obter-se uma aproximação superior à realidade.

Dado que a informação adquirida por cada sensor, não tem qualquer relação com os restantes sensores, poderemos afirmar então que a informação fornecida por cada um sensores é independente das informações dadas pelos restantes. Esta premissa é um ponto fundamental, permitindo-nos então combinar a probabilidade gerada pela informação do sensor com as probabilidades geradas pelo mesmo processo relativo aos restantes sensores recorrendo para tal à equação 4.45.

4.4.4 "*Resampling*"

Este procedimento só é necessário devido ao facto do Filtro de Partículas ser uma discretização do Filtro Bayesiano. Esta discretização provoca a existência de uma situação potencialmente perigosa, sendo caracterizada pelo esgotamento da população de partículas, implicando que o método não funcione.

Para evitar o esgotamento da população de partículas, devendo-se sobretudo a uma redução significativa da sua contribuição provocada pelas observações, é necessário utilizar uma operação de reamostragem que garante a multiplicação das partículas cujo factor de confiança é elevado e por sua vez elimina partículas com factores de

confiança muito reduzidos, permitindo a criação de novas partículas no mundo. Estas novas partículas possibilitam a convergência do método caso exista o caso do rapto do robô.

4.5 Filtro de Kalman

Um outro método mais comum de fusão de informação é o Filtro de Kalman introduzido em 1960 por R. E. Kalman. Como método matemático, assegura uma estimação óptima, se todos os processos forem lineares e o ruído associado for do tipo Gaussiano com média nula. A característica de estimação óptima é conseguida devido à minimização do erro quadrático.

A utilização do Filtro de Kalman na estimação de posição com características geométricas tem provado ser uma técnica bastante poderosa, contendo muitas das características desejáveis para este tipo de problemas. Entre elas encontramos: o funcionamento com uma representação minimalista do ambiente em que se encontra inserido, uma exactidão para as medidas sem limite, assim como uma implementação considerada computacionalmente leve. Este método consegue realizar uma estimação a partir de medidas ou estados ruidosos, mas o método consegue ainda alterar o estado do sistema de forma a reflectir não só as novas medidas como os ruídos existentes, realizando para tal uma actualização dinâmica do sistema.

Este método apresenta ainda uma extensão para sistemas não-lineares, o Filtro de Kalman Extendido, aumentando assim o universo de utilização deste método que anteriormente só poderia ser utilizado em sistemas lineares. Esta extensão encontra-se restringida, já que não e capaz de garantir a convergência nem que a solução seja óptima, garantindo somente que será a melhor solução que o filtro linear é capaz de determinar.

Actualmente decorre um estudo por parte de um colega sobre a aplicação do filtro de Kalman Extendido sobre a mesma plataforma robótica, com o intuito futuro realizar uma interligação entre os dois filtros, o Filtro de Kalman Extendido e o Filtro de Partículas.

4.6 Conclusões

O Filtro de Partículas como método de fusão de informação é um método muito eficaz sendo uma aproximação discreta do Filtro Bayesiano, suportando a integração de inúmeros sensores com naturezas diferentes e com a vantagem de permitir uma auto-localização bastante eficaz através do uso de vários sensores com um custo económico reduzido e com bastantes limitações. Contudo no limite, a eficácia deste método resume-se à quantidade de sensores envolvidos. Para sensores com baixa qualidade é necessário compensar com este factor com a quantidade, obtendo assim uma forma de compensação. Um outro ponto forte deste método é o uso de modelos não gaussianos e com médias não-nulas que poderão caracterizar melhor cada tipo de sensor, obtendo assim uma melhor aproximação à realidade, porém como referido anteriormente limitada à qualidade do sensor. Um dos pontos fracos deste método é o seu forte peso computacional, limitando o uso em plataformas robóticas cujo o poder de processamento é limitado ou reduzido.

Em resumo o Filtro de Partículas possuiu as seguintes vantagens:

- Capacidades de localização num determinado mapa
- Integração de diferentes modelos de distribuições com médias não-nulas
- Robustez
- Precisão

Enquanto o Filtro de Kalman possui as seguintes vantagens:

- Eficiência computacional
- Precisão
- Capacidades de seguimento de posições

Capítulo 5

Trabalho Desenvolvido

5.1 O robô de Limpeza

Figura 5.1: Fotografia do "CleanRob"

O "*CleanRob*" foi-me apresentado no inicio de Março de 2007, no âmbito de um projecto de final de curso em parceria com um colega de curso, com os apoios do ISR-Porto e do Departamento de Electrotecnia e Computadores da Faculdade de Engenharia da Universidade do Porto.

Embora não tenha tido a possibilidade de ter presenciado e/ou

acompanhado o início deste projecto, o trabalho desenvolvido até então estava numa fase muito embrionária, existindo apenas a estrutura mecânica, a estrutura de potência e a unidade de processamento.

Porém devido a uma deterioração do equipamento muito provavelmente devido ao tempo que o projecto esteve parado, foi necessário proceder a inúmeras modificações, que eram necessárias quer a nível de performance, fiabilidade, reforço, adaptação a novas potencialidades que requereram tempo e recursos que não estavam planeados inicialmente. Estando neste momento o robô com um nível de desenvolvimento bastante grande e apto a realizar funções que anteriormente não era possível.

Actualmente a plataforma robótica encontra-se estável, possibilitando missões. Possui um aspirador mecânico como acessório adicional que permite completar o objectivo para o qual foi desenvolvido.

5.1.1 Sistema Eléctrico

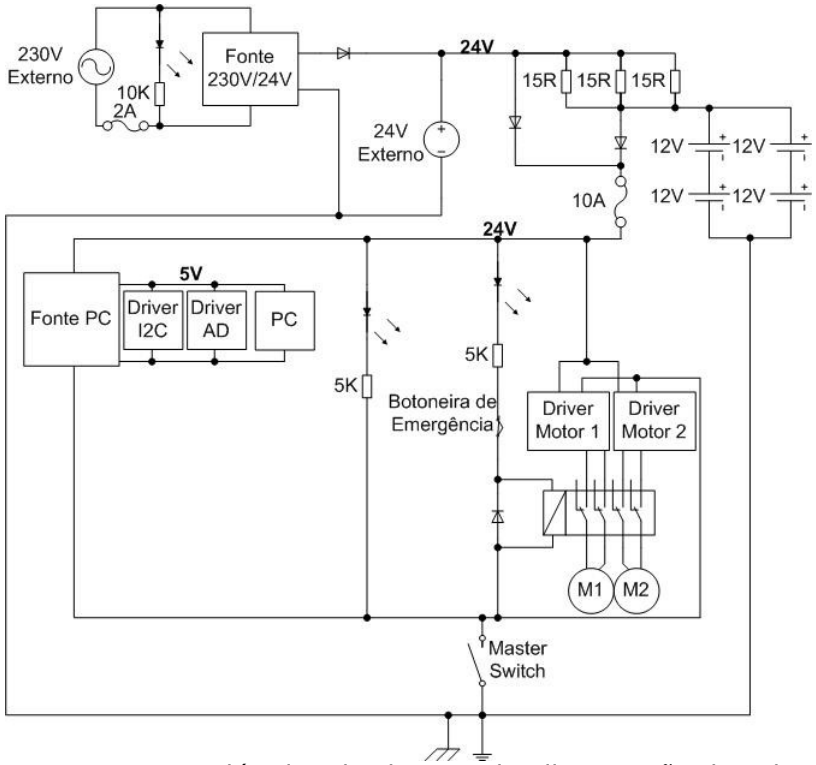


Figura 5.2: Esquema eléctrico do sistema de alimentação do "CleanRob"

Remodelação completa da cablagem de potência do robô, devido a existir ligações partidas, mal soldadas e na maior parte das situações presas apenas por fita isoladora.

- Projectou-se e implementou-se um novo painel que permitisse facilidade de manutenção, robustez e melhor estética.

- Inclui-se um relê permitindo assim um corte mecânico da potência fornecida aos motores, ao invés do anterior sistema que dependia do poder de corte do interruptor de emergência.

- Aproveitou-se a alimentação de 5V da fonte de alimentação comutada do PC, devido a esta demonstrar-se mais estável para alimentação dos "*drivers"* de comunicação "*I2C*" e da aquisição de sinal.

- Implementou-se uma nova forma de fornecimento de potência ao robô, através da incorporação de uma fonte de 230V para 24V dentro do robô, alimentando e aproveitando assim todo o sistema de potência já desenvolvido.

5.1.2 Sistema Mecânico

Embora na proposta deste projecto, estivesse a inclusão de um robô, previamente estruturado com sensores (sonares, câmara, acelerómetros, giroscópio, "*encoders*"), ao longo deste projecto foi necessário realizar inúmeras modificações à sua estrutura.

Estas modificações foram muito extensivas e de carácter obrigatório devido aos seguintes factores:

- Os sonares não possuíam qualquer suporte que oferecesse estabilidade, pelo que foi necessário projectar e implementar uma solução que garantisse a possibilidade de ajuste de posição, bem como um isolamento eléctrico da estrutura do robô.

- O meio de comunicação dos sonares ("*I2C"*) por meio de uma fita "*flat cable*", encontrava-se em más condições, pelo que procedeu-se à sua substituição.

- Os painéis laterais que protegem o interior da estrutura não continham qualquer abertura para que os sonares pudessem medir as distâncias sem que fossem limitados pelo painel de acrílico, pelo que projectou-se e implementou-se uma solução que permitisse o funcionamento dos sonares sem comprometer a protecção da estrutura do robô.

- Inclusão de diversos conectores aéreos para fácil manutenção.

- Modificação da alimentação do "*driver*" de comunicação com os sonares, através do recurso à alimentação da fonte comutada do computador.

- Fez-se uma actualização ao sistema de baterias permitindo deste

modo a inclusão de até 6 "*packs*" de baterias de chumbo (cada "*pack*" é constituído por duas baterias de chumbo de 12 V cada). Estando neste momento o robô equipado com 2 "*packs*" de baterias.

- Fortaleceu-se o “pescoço” do robô que se encontrava instável e projectou-se novos suportes para as câmara “*firewire*” que se encontravam desajustados às novas necessidades.

- Projectou-se e implementou-se um sistema sistema para a segunda câmara (neste momento estando a ser utilizada num outro projecto do Professor Doutor Paulo Costa) que possibilita visualizar o meio que rodeia o robô, através de um espelho em olho de boi com um suporte que providencia um fundo negro para facilitar o processamento de visão. O antigo suporte encontrava-se em estado de degradação bastante acentuado.

- Alteração do encaixe das rodas, no sistema antigo as rodas estavam demasiado próximas à estrutura do robô, o que provocava atrito devido à fricção das rodas com a estrutura a quando do movimento do robô.

- Projecto e implementação de um suporte para a antena “*wireless*” com o fim de interligar o robô à rede “*wireless* eduroam”.

- Modificação da estrutura do robô, de modo a possibilitar ventilação do equipamento dentro da estrutura. Com a inclusão de arrefecimento por um sistema mecânico e outras aberturas para exaustão de ar quente.

- Reconstrução do “*Daisy Chain*”, devido às ligações anteriores estarem danificadas.

- Modificação da protecção em acrílico do computador do interior da estrutura de modo a possibilitar a inclusão de um botão para o arranque do computador e a adição de um suporte fixo para uma futura “*pen-bluetooth*”.

- Reforço geral da estrutura com o uso de “cotovelos de ligação”.

- Isolamento de todos os equipamentos electrónicos do robô com a estrutura deste através de anilhas de borracha, fita isoladora, anilhas de plástico.

- Melhoramento estético do robô

- Melhoramentos na arquitectura mecânica do robô para melhorar

acessibilidade e modularidade.

- Foi implementado e desenvolvido uma unidade extra que poderá ser copulado ao robô que permite aspirar aproveitando o movimento deste.

- Foi implementado e desenvolvido pára-choques para o robô, protegendo assim as rodas do mesmo no caso de impacto.

5.1.3 Sistema de Locomoção

Embora, a estrutura do "*CleanRob*", estivesse previamente montada numa configuração do tipo de locomoção diferencial (Figura 2.1), este tipo de decisão mostrou claramente ser foi uma opção vantajosa dado ao tipo de robô de serviço.

A locomoção diferencial apresenta uma estrutura mais robusta, estável e com um custo economicamente mais baixo, sendo simples de implementar e manter. Além de permitir diferentes adaptações como por exemplo do atrelado aspirador.

5.1.4 Sistema de "*Drivers*" e Sensorização

"Drivers"

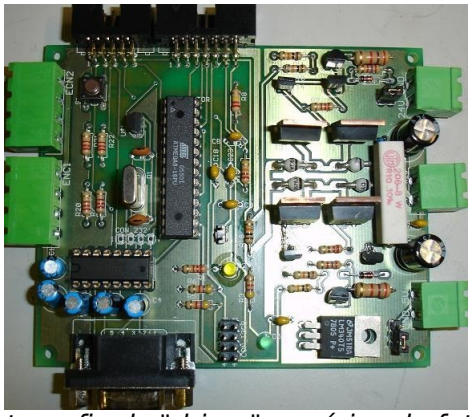


Figura 5.3: Fotografia do "driver" genérico do futebol robótico

O "*CleanRob*" possui 4 "*drivers*", estando estes divididos nas seguintes funções:

- "*Driver*" 1 – Destinado ao controlo do motor da direita
- "*Driver*" 2 – Destinado ao controlo do motor da esquerda
- "*Driver*" 3 – Destinado à comunicação pelo protocolo "*I2C*"
- "*Driver*" 4 – Destinado à aquisição de sinal proveniente de

sensores analógicos

Os “*drivers*” 1, 2 e 3 foram desenvolvidos pelo grupo do futebol robótico da FEUP, nomeadamente pelo Professor Doutor Paulo Costa. Foram concebidos com o intuito de serem o máximo genéricos possíveis, possibilitando apenas com a troca de "*firmware*" a alteração da função destes. Foram estudados estes “*drivers*” devido a problemas de fiabilidade herdados do futebol robótico .

Nos “*drivers*” 1 e 2 responsáveis pelo controlo dos motores, para além da electrónica de potência necessária para o controlo dos motores, incorporam também um microcontrolador que gera sinais em PWM para o controlo dos semi-condutores de potência. Neste microcontrolador está implementado um PID responsável pelo controlo em malha fechada da velocidade dos motores. A comunicação com estes “*drivers*” e a unidade de processamento principal é realizada através de um protocolo de comunicação denominado “*Daisy-Chain*” sobre o meio de comunicação RS-232.

Enquanto no "*driver*" 3 este comunica directamente com a unidade principal de processamento. A função deste "*driver*" é interligar dois sistemas separados, ocupando-se de tarefas de comunicação, recebendo comandos da unidade principal e encarregando-se de executar todos os procedimentos necessários de forma a comunicar com os dispositivos pendurados no barramento de “*I2C*”.

"*Encoders*"

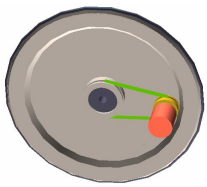


Figura 5.4: Vista em perspectiva do sistema de polias

Os "*encoders*" incrementais são elementos fundamentais no cálculo da hodometria de um robô, praticamente todos robôs móveis terrestres fazem uso destes devido ao seu custo económico reduzido e facilidade de implementação, existindo mesmo módulos constituídos que trazem "*encoders*" acoplados ao eixo do motor. Porém neste projecto a forma de implementação dos "*encoders*" é realizada por intermédio de um sistema de polias visualizado na figura 5.4. O "*encoder*" possui uma polia de reduzida dimensão com um sulco que ladeia todo o seu perímetro. Este sulco tem a finalidade de conduzir uma correia sob uma pequena tensão não permitindo que esta salte. Por sua vez o eixo do motor está copulado a uma roda metálica que transmite a força tracção ao piso, originando o deslocamento do robô. Esta roda metálica possui na sua própria estrutura uma polia esculpida com um sulco ao qual está ligada a correia que passa pela polia do

encoder, desta forma conseguindo transmitir a mesma velocidade tangencial.

A velocidade tangencial é definida pela equação 5.1, dado que a velocidade a velocidade tangencial no ponto de contacto da roda com o piso é idêntica à velocidade tangencial na polia esculpida na roda definiremos que $v_{\perp\ roda}=v_{\perp\ polia}$, o que implica que a correia associada à polia da roda e à polia do "*encoder*" tenha a mesma velocidade, pelo que está implícito que a polia do "*encoder*" tenha a mesma velocidade tangencial.

$$v_{\perp}=w*r \tag{5.1}$$

O "*encoder*" utilizado tem uma resolução de 200 "*ticks*" ou incrementos por volta, será o mesmo que dizer que a cada volta da roda do robô o "*encoder*" terá assinalado $\Delta_{ticks}=K_1*K_2*200$, sendo que K_1 é uma relação entre polias (equação 5.2) e K_2 é uma relação entre a roda do robô e a polia esculpida nesta (equação 5.3).

$$K_1=\frac{P_{polia\ roda}}{P_{polia\ encoder}} \tag{5.2}$$

$$K_2=\frac{P_{roda}}{P_{polia\ roda}} \tag{5.3}$$

Podemos então calcular a velocidade tangencial de cada uma das rodas dentro de um intervalo de tempo definido por Δt (normalmente associado ao ciclo de controlo) recorrendo à equação 5.4 e através da geometria do robô diferencial, representada na figura 5.5, obtemos a velocidade e a velocidade angular do robô (equação 5.5, dado que num neste tipo de geometria diferencial a componente normal da velocidade é nula, $V_n=0$, tornando assim a velocidade igual à velocidade tangencial.

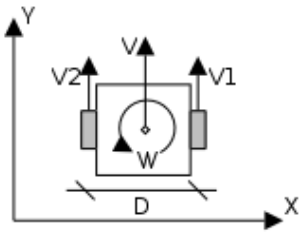


Figura 5.5: Robô diferencial

$$v_{\perp\ roda}=\frac{\Delta_{ticks}*K_1*K_2*200}{\Delta t*10^{-3}} \tag{5.4}$$

$$V = \frac{V_{\perp 1} + V_{\perp 2}}{2}$$
$$W = \frac{V_{\perp 1} - V_{\perp 2}}{D} \quad (5.5)$$

As projecções da velocidade do robô em cada eixo de coordenadas define-se por:

$$V_x = V * \cos(\theta_{k+1})$$
$$V_y = V * \sin(\theta_{k+1}) \quad (5.6)$$

Após estes simples cálculos é possível então estimar a posição do robô como constatado em 5.7.

$$x_{k+1} = x_k + V_x * \Delta t$$
$$y_{k+1} = y_k + V_y * \Delta t \quad (5.7)$$
$$\theta_{k+1} = \theta_k + W * \Delta t$$

Contudo a hodometria não é o único objectivo do uso dos "*encoders*", estes também fornecem informação aos PID implementados nos "*drivers*" de controlo de velocidade dos motores, sendo que estes sensores fornece uma realimentação em velocidade dos motores, sendo detalhado este aspecto mais adiante.

Sonares

Figura 5.6: Sonar Devantech SRF08

O robô está equipado com 6 sonares Devantech SRF08 (figura 5.6 e 5.7) cujo interface de comunicação é através do protocolo "*I2C*" e tendo como característica física um ângulo de abertura na ordem dos 110 graus e uma gama de medição de 3cm até 6m (figura 5.8). Para tal foi desenvolvido "*firmware*" para um "*driver*" genérico fornecido pela equipa de futebol robótico da FEUP, que possibilitasse a interligação entre o computador e os sonares através de uma simples comunicação RS-232.

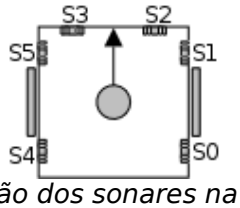


Figura 5.7: Disposição dos sonares na geometria do robô

Figura 5.8: Dispersão e ganho da onda de ultra-som (40Khz) do sonar SRF08

O "*firmware*" desenvolvido permite a recepção de mensagens simples relativas ao modo de disparo dos sonares, bem como a configuração de valores dos mesmos. A implementação consistiu na criação de diversos modos de funcionamento, aos quais passo a citar:

- "*BroadCast*"
- Manual
- Sequencial
- Sequencial em grupos
- Modificação de alcance
- Modificação de ganho
- Modificação de endereço de identificação "*I2C*"

O modo de broadcast, permite o disparo de todos os sonares pendurados no barramento de "*I2C*". Esta possibilidade tem uma vantagem bastante apelativa, devido ao facto de permitir obter uma "imagem" do mundo que rodeia o robô, por outras palavras, as medidas obtidas dos sonares face à geometria destes no robô, possibilitariam um conjunto de medidas do mesmo instante de tempo de todos os obstáculos à volta deste. Contudo este modo possui uma grande desvantagem, devido à limitação inerentes à geometria do robô, não é possível manter uma distância requerida de modo a que não haja interferência entre os sonares pelos seus ecos afectando assim as medidas obtidas.

O modo sequencial permite o disparo dos sonares numa certa sequência pré-definida com base nos identificadores destes, tendo uma desvantagem bastante acentuada devido ao seu longo tempo de aquisição das medidas de todos os sonares. Face a esta desvantagem desenvolveu-se o modo sequencial em grupo, que dispara simultâneamente um grupo constituído por quatro sonares, havendo quatro grupos disponíveis para configuração. Este modo permite o disparo de dois grupos, o primeiro definido pelos sonares com identificadores ímpares e o segundo pelos complementares, ou seja os sonares com identificadores pares. Este tipo de ciclo tem um tempo de aquisição de medidas na ordem de 60 ms e dado estes tipos de grupos foram escolhidos com base na disposição dos sonares na estrutura do robô, consegue-se então evitar/minimizar a interferência entre sonares.

As definições do ganho, alcance e sequências, podem ser redefinidas "*online*" e permanecem guardadas dado que são programadas em memória "*EEPRom*"[8].

Este tipo de "*firmware*", foi desenvolvido de modo a colmatar uma falha que existia no que diz respeito a comunicação por "*I2C*", com tempos de aquisição de medida bastante reduzidos.

O grande potencial dos sonares é a detecção de obstáculos, devido à sua grande abertura de eco. Contudo para o uso no cálculo de ângulos do robô a paredes torna-se bastante impreciso quando as paredes possuem saliências, tornando-se em última análise impraticável o uso destes com este fim.

câmara

Figura 5.9: DBK 21F04 - câmara "firewire" da ImagingSource

Neste projecto foi utilizada apenas uma câmara "*firewire*" da "*ImagingSource*", modelo DBK 21F04. Este tipo de câmara possui um sensor "*CCD*" que transforma fotões em electrões. Em particular, na câmara utilizada, a saída não é em formato "*RGB*", formato este que define para cada "*pixel*" de uma imagem existem três componentes, conforme a cor representada estas três componentes tem valores diferentes de acordo com a importância dessa mesma componente na cor representada. Cada componente é escalonada numa gama de

8 EEPRom – ***E****lectrically-****E****rasable* ***P****rogrammable* ***R****ead-****O****nly* ***M****emory* é uma memória que permite ser apagada e programada electricamente. Pode ser lida um numero ilimitado de vezes, mas só pode ser apagada e programada um número limitado de vezes, que variam entre as 100.000 e 1 milhão.

valores definida entre 0 e 255, sendo 255 a influência máxima que essa componente representa na cor. Porém a câmara como referido anteriormente não permite a saída de dados no formato "*RGB*", mas sim em formato "*Bayer*" (figura 5.10). Isto deve-se ao facto de que por razões económicas a câmara apenas possuir um sensor "*CCD*" em vez de três, uma para cada componente de cor.

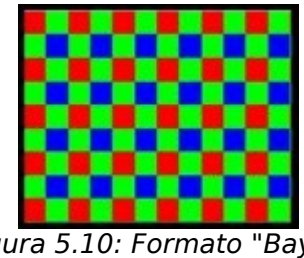


Figura 5.10: Formato "Bayer"

Assim, cada "*pixel*" possui apenas uma das componentes de cor, "*RED*" ou "*GREEN*" ou "*BLUE*". As restantes componentes de cor do "*pixel*" são calculadas no "*software*" por interpolação das componentes dos "*pixeis*" vizinhos.

Deste modo é necessário realizar operações de processamento de imagem, que permitem aproximarem-se de uma operação de transformação de formato "*Bayer*" para formato "*RGB*", estas operações funcionam por interpolação de valores entre "*pixeis*" vizinhos, por isso não será uma função de transformação exacta, mas aproximada.

Figura 5.11: Código de barras no formato Interleaved 2 de 5

O processamento de imagem da câmara, foi realizada pelo professor Doutor Paulo Costa, numa ferramenta de desenvolvimento denominada de "*Kylix3*" utilizada no futebol robótico. Este processamento de imagem usa a câmara com um sensor que possibilita a detecção e identificação de códigos de barras no formato "*Interleaved 2 de 5*" (figura 5.11). No entanto, usando o código de barras como um objecto conhecido é possível usar as linhas que o definem como referências.

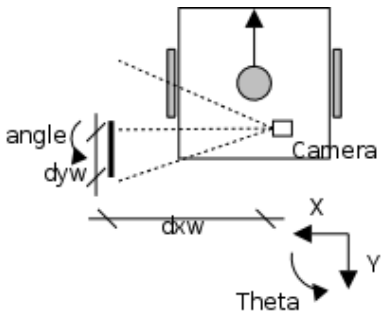


Figura 5.12: Caracterização dos dados fornecidos pela câmara

Desta forma, através da inclinação das linhas dada pela perspectiva calcular o ângulo da câmara em relação ao código de barras e dado que o estes marcadores tem um tamanho conhecido é possível determinar as distâncias relativas da câmara aos mesmos. Este programa de processamento comunica com a aplicação de controlo através de sockets, demonstrando outra potencialidade inerente ao desenvolvimento deste projecto, sendo a capacidade de interacção com diferentes programas, desenvolvidos em diferentes ferramentas aproveitando ao máximo as vantagens de cada uma delas.

Na figura 5.13 é caracterizado no espaço os dados que a câmara fornece (dxw , dyw , $angle$, para além destes dados é recebido o número do código de barras detectado e o número de linhas verticais que foi possível interceptar com o código de barras visível na figura 5.14 a cor verde).

Contudo está fora do contexto desta dissertação uma justificação mais prolongada sobre o funcionamento do algoritmo que permite a obtenção destes parâmetros. Deste modo a abordagem face a este tipo de sensor, está orientada para uma explicação dos dados que este sensor fornece, bem como o uso destes num panorama geral relativo ao Filtro de Partículas.

Figura 5.13: Imagem da janela principal do programa de processamento de marcadores códigos de barras desenvolvido pelo Professor Doutor Paulo Costa

Este tipo de sensor requer o uso de marcadores distribuídos pelo ambiente sob a forma de códigos de barras, tornando o ambiente em que o robô opera num ambiente semi-estruturado.

Medidores de distâncias por infra-vermelhos - "Sharps"

Figura 5.14: Sharp GP2Y0A02YK0F - Medidor de distâncias por infra-vermelhos

O robô tem equipado um conjunto de seis medidores de distâncias por infra-vermelhos (Sharp GP2Y0A02YK0F), com gama de medição que se situa entre os 20cm e os 150cm, tendo a mesma disposição no robô que os sonares.

A vantagem destes sensores é a sua direcionalidade, o que torna-o perfeito para o cálculo de ângulos do robô a paredes, no entanto a sua principal desvantagem é o seu curto alcance. Para minimizar esta desvantagem existe a possibilidade da integração simultânea de um outro modelo na tentativa de compensar esta desvantagem (Sharp GP2Y0A700K0F) cujo a gama de medidas situa-se entre 100cm e os 500cm, de momento não foi possível utilizar este sensor dado que não estão esgotados em stock a algum tempo.

Ambos sensores são analógicos, pelo que foi necessário desenvolver e implementar um "*driver*" que possibilitasse a interligação destes com o robô, para tal usou-se como base um PIC16F882 com 11 ADC que comunica com o robô através de uma interface RS-232. Dada à instabilidade do sinal do sinal proveniente do medidor de distâncias por infra-vermelhos, foi implementada no PIC uma média ponderada das últimas quatro conversões, e sendo esse valor enviado ao robô. Esta implementação possibilitou a estabilização dos valores na conversão do sinal analógico do medidor de distâncias por infra-vermelhos.

5.1.5 Sistema Informático

Sistema Operativo

Nos robôs desenvolvidos pelo futebol robótico, tem sido utilizado a distribuição “*Mandriva”* como sistema operativo, porém neste projecto foi proposto o uso de outra distribuição de “*Linux*” denominada “*Ubuntu”*.

Esta distribuição tem sido alvo de grande adesão por parte dos utilizadores como uma alternativa ao sistema operativo “*Microsoft Windows*” quer pela sua simplicidade de utilização, bem como pelo grande suporte existente na comunidade “*Linux*”.

- A mudança para esta distribuição tem diversos objectivos, sendo

estes os seguintes:

- Possibilidade de interligação à "*FeupNet*" através da tecnologia "*wireless*" do projecto "*eduroam*".

- Possibilidade de documentar todos os passos necessários para futuras realizações de projectos semelhantes.

- Melhor suporte de "*drivers*" para o equipamento utilizado no robô.

- Desafio ao uso de uma distribuição usada normalmente no desenvolvimento de projectos semelhantes (futebol robótico)

- Desenvolvimento de "*software*" compatível ao uso de aplicações robóticas, de forma a futuramente ser incluído noutro projecto chamado "*FeupLIVE*", como incentivo a futuros alunos a ingressarem nesta área e a desenvolverem projectos novos ou até mesmo o prolongamento deste.

- É completamente livre garantido pela licença "*GPL*", ao contrário da distribuição "*Mandriva"* que é necessário ser membro oficial de um clube sem registo gratuito.

- Grande suporte de informação por tutoriais "*online*" e da comunidade "*open source*" o que permite reduzir tempos de aprendizagem.

- Pelo facto de ter como base a distribuição "*Debian*" (distribuição utilizada amplamente em servidores) permite que tenha uma garantia de qualidade elevada.

Plataformas de desenvolvimento

Optou-se por usar duas plataformas de desenvolvimento, o "*Kylix3"* e o "*Lazarus"*.

O "*Kylix3*" é uma ferramenta de desenvolvimento disponível há já bastante tempo, pelo que existem diversas bibliotecas de programação para os mais diversos fins, o mesmo já não se acontece com o "*Lazarus"* que é uma ferramenta relativamente nova, tendo sido desenvolvida para colmatar a extinção do desenvolvimento do "*Kylix3*" por parte da empresa "*Borland"*.

Contudo o objectivo principal é promover a interacção entre ferramentas. As sockets permitem que programas desenvolvidos em diferentes aplicações e até mesmo em diferentes linguagens de programação possam comunicar entre si trocando informações. Dado que as sockets são independentes das linguagens ou ferramentas de

desenvolvimento, este tipo de abordagem permite retirar o melhor de cada tipo de ferramenta de desenvolvimento e até trocar no futuro um determinado programa por outro sem ter que recomeçar do ponto zero. Este foi um dos motivos pelo quais optou-se por esta solução.

Programa de Controlo

Um dos requisitos do projecto, era passar da teoria à prática, aplicando os conhecimentos adquiridos durante a primeira fase do projecto sobre o estudo de métodos de auto-localização, implementando-o num sistema real.

Desde do início do projecto foi necessário remodelar e/ou desenvolver de raíz vários programas com a finalidade de permitir uma análise e testes de diversos componentes associados ao robô. O programa de controlo desenvolvido inicialmente encontrava-se desorganizado, com baixa fiabilidade, fraca/inexistente documentação e de difícil interpretação, fruto dos inúmeros testes, correcções e alterações realizadas no código pelos diferentes intervenientes nas diferentes fases de vida do projecto. Estes factos tornaram inviável e possivelmente inglório o esforço de documentar convenientemente toda a aplicação de controlo.

Sendo assim, tomou-se a decisão e o desafio de desenvolver a aplicação de controlo integralmente em “*Lazarus”*, como foi referido acima, é nos nossos dias uma plataforma de desenvolvimento em constante melhoramento, algo que não acontece com o “*Kylix3*” já que foi descontinuado pela “*Borland”*.

Da versão antiga versão do programa de controlo desenvolvida em “*Kylix3”*, foram unicamente aproveitados os procedimentos de comunicação com os “*drivers”*, responsáveis pelo envio de ordens e/ou pedidos de dados, assim como da análise e interpretação das tramas de resposta enviadas.

Mesmo assim, os novos procedimentos criados em “*Lazarus”*, sendo conceptualmente idênticos aos do "*software*" anterior, foram modificados devido a alteração global da arquitectura e estratégia de controlo.

Todos os sistemas requerem controlo, desde dos mais simples e básicos aos mais complexos. Um robô não é uma excepção, devido à complexidade existente na coordenação dos diferentes aspectos que o compõem no que respeita ao "*hardware*" e à complexidade do controlo de navegação por forma a executar o seu objectivo principal no que respeita ao "*software*".

As arquitecturas mais simples são as reactivas, que possuem uma

inteligência muito reduzida ou até mesmo nula, a informação é passada praticamente dos sensores para os actuadores, sendo afectados de um ganho ou passando por uma função de controlo simples, enquanto as arquitecturas baseadas em comportamento são as mais complexas, dado que são mais orientadas a operarem em ambientes semi-estruturados e não-estruturados, sendo as suas acções deliberadas por várias camadas de "*software*" organizadas em diversos de controlo.

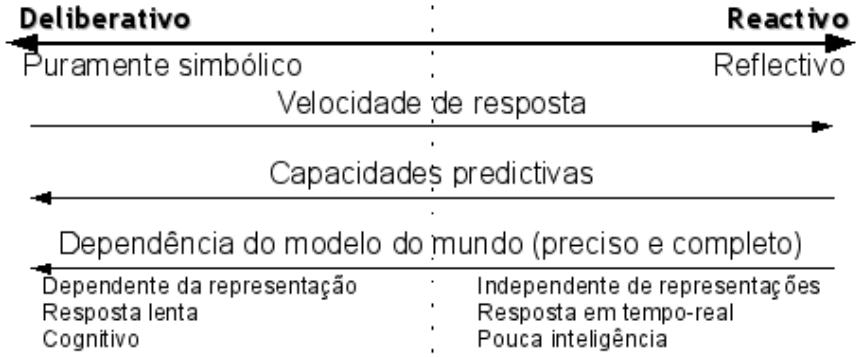


Figura 5.15: Espectro dos Sistemas de Controlo Robóticos

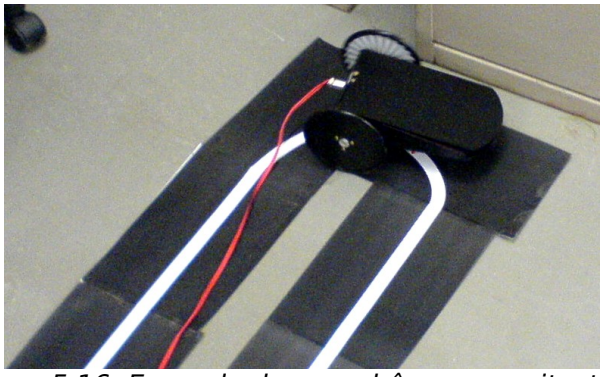


Figura 5.16: Exemplo de um robô com arquitectura reactiva que segue linhas brancas

Neste projecto, à semelhança de muitos outros cujo o objectivo é criar um robô autônomo, foi necessário criar as diversas camadas de "*software*", tendo sido cada uma desenvolvida inteiramente e exclusivamente para este projecto. Como constatamos pela figura 5.17, a base de deste sistema é comportada pela aquisição de informação do mundo obtida pelos sensores e respectivamente a execução de uma ou mais acções no mesmo através dos actuadores. A transição entre o "*hardware*" e o "*software*" é realizada através de "*drivers*", tendo sido estes criados pela equipa de futebol robótico da FEUP, porém de modo a corresponder aos requisitos deste projecto, foi necessário o desenvolvimento de novos "*firmwares*" para os mesmos.

A camada de auto-localização, foi realizada com base em filtro de fusão de informação tais como o Filtro de Partículas e o Filtro de Kalman Extendido, possibilitando assim o um patamar fundamental do qual a camada superior, nomeadamente planeamento de trajectórias, possa

operar em toda a sua potencialidade. A camada de navegação respectiva às funções de navegação foi também desenvolvida e implementada face às necessidades do projecto, introduzindo nestas funções potencialidades requeridas à função que o robô irá executar, que possibilitam o controlo maior por parte da camada de planeamento de trajectórias.

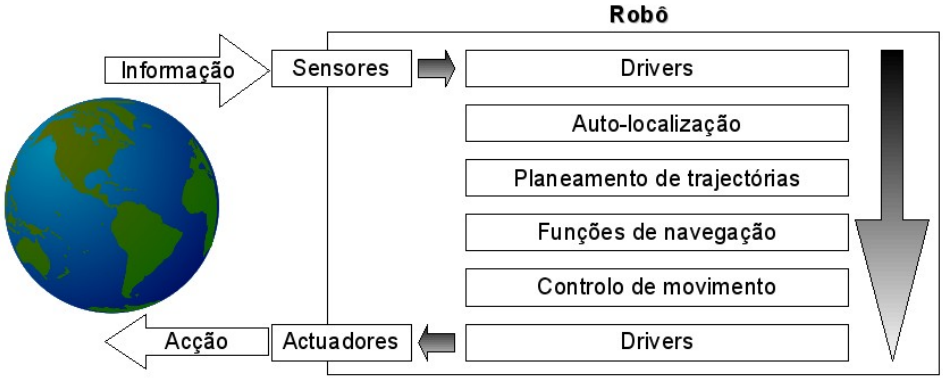


Figura 5.17: Camadas do sistema de controlo

A camada de controlo de movimento foi implementada, sendo esta a mais simples, devido ao desenvolvimento do robô e de todas as camadas inerentes ao controlo deste, com base na física que regula o movimento do robô, tornando-se assim numa mera questão de relação entre a camada de funções de navegação e os comandos enviados a cada "*driver*" que regula o respectivo actuador.

Uma das potencialidades do programa da controlo é a criação de "*logs*" em ficheiro, devidamente estruturado, de acordo com as camadas existentes, para uma análise posterior da missão e consecutivo despiste de anomalias.

5.1.6 Mapa do Edifício I - Piso 2 - Poente

Dado que o robô terá como função aspirar as áreas comuns interiores dos edifícios que constituem a faculdade, é necessário recriar o ambiente em que este actuará na sob forma de mapas.

Devido à dimensão das instalações da faculdade de engenharia, é imperativo que a construção destes mapas seja realizada de uma forma fácil e eficiente permitindo a alteração, pelo que dividiu-se o mapa em diversos mapas de menor dimensão.

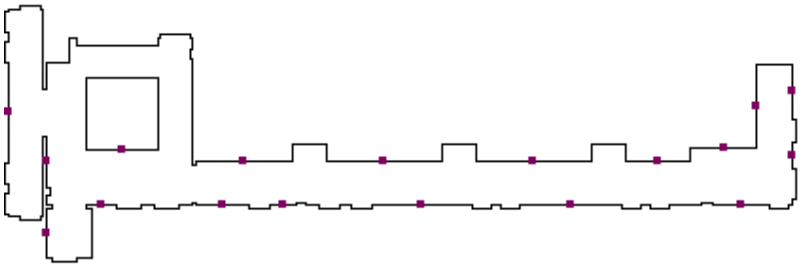


Figura 5.18: Mapa do Edifício I - Piso 2 - Poente

O ficheiro de mapa é criado com base numa aplicação denominada "*Map Creator*", desenvolvido com o único objectivo de permitir a maior rapidez e facilidade, na criação/alteração de mapas sem que com isso houvesse a necessidade de alterar o programa de controlo.

O mapa poderá ser definido numa primeira fase como um conjunto de pontos onde há intercepção de paredes físicas e cujo o ponto inicial do mapa é o ponto final deste conjunto. Porém como observamos na figura 5.18, existe uma abertura no piso junto aos elevadores. Esta abertura é delimitada por outros pontos diferentes relativamente às paredes principais.

Para haver separação destes conjuntos de pontos é criado no mapa, vários conjuntos, neste caso especifico dois objectos, podendo noutro mapa serem criados inúmeros mais reflectindo os diferentes conjuntos existentes.

No ficheiro de mapa é também definido zonas inválidas, estas zonas são zonas que é fisicamente impossível o robô estar, por exemplo no interior de paredes ou na abertura do piso. Este tipo de informação é utilizada no programa de controlo a quando da fusão de informação. Também é condensada informação relativa a marcadores (códigos de barras), esta informação reside na localização destes e na sua orientação.

5.1.7 Sistema de Auto-Localização

A implementação do filtro de partículas no projecto "*CleanRob*", foi divida em diversas secções para uma maior comodidade de implementação e de compreensão. Pelo que foram seccionadas na seguinte forma:

1. População inicial
2. Reamostragem
3. Previsão

4. Sensorização

5. Actualização

6. Normalização

7. Estimação da "*pose*"

A população inicial é um procedimento utilizado apenas no arranque do programa de controlo. Esta função tem duas características associadas, dado que o filtro da partículas não requer uma "*pose*" inicial definida pode-se popular o mundo com partículas de uma forma aleatória, tendo apenas em consideração as restrições do mapa que se fornece, evitando assim popular o mundo com partículas em zonas inválidas e permitindo assim que a convergência do método encarrega-se de estimar uma "*pose*".

Por outro lado quando definida uma "*pose*" inicial, por exemplo sabendo que o robô parte sempre de uma estação de abastecimento sabemos à partida a sua "*pose*" inicial, pelo que poderá atribuir-se esta logo ao filtro de partículas que definiria uma certa percentagem de partículas associadas à "*pose*" atribuída, com uma variação gaussiana em cada uma das componentes das diversas partículas para que o filtro de partículas tenda a convergir mais rapidamente implicando que o robô conheça a sua posição actual mais rapidamente e com isso tome decisões acertadas.

A reamostragem ou "*resampling*" é uma função que tem como objectivo analisar todas as partículas, copiando as que tem uma probabilidade elevada e sorteando aquelas cuja probabilidade é reduzida decidindo assim a sua sobrevivência ou a sua extinção. Isto garante no final que surjam novas partículas de forma a existir sempre um número mínimo definido inicialmente. Podemos seguir melhor este procedimento pelo algoritmo em pseudo-código definido na tabela 2 (adaptado de [4]).

```
for ( j=1 to M ) do                    /* Cria pesos auxiliares */
    a[j]=√(W[j]);
end
acum=0;
for ( j=1 to M ) do                    /* Calcula o somatório dos pesos auxiliares */
    acum=acum+a[j];
end
for ( j=1 to M ) do                    /* Normaliza os pesos auxiliares, T é o número total */
    a[j]=T*a[j]/acum;                  /* de partículas desejadas*/
end
i=1;
for ( j=1 to M ) do                    /* Para cada partícula */
```

```
   if ( a[j]≥1 ) then              /* Aceita as partículas com pesos auxiliares > ou = 1 */
      for ( l=1  to  a[j] ) do     /* Cria a[j] cópias da partícula */
          S'[i]=S[j];
          i=i+1;
      end
   else
          R=rand(1);
          if ( a[j]≥R ) then       /* Partícula sobreviveu ao sorteio */
              S'[i]=S[j];
              i=i+1;
          end
    end
end
for ( i  to  T ) do                /* Cria partículas até atingir o máximo de T */
    S'[i]=S[j];
    i=i+1;
end
S=S';
```

Tabela 2: Algoritmo em pseudo-código do procedimento de reamostragem ou "resampling"

A previsão é o procedimento que se ocupa de propagar a todas as partículas todas as acções desempenhadas pelo robô. No "*CleanRob*", esta acção é caracterizada pela movimentação deste pelo meio que o envolve. A acção de movimentação executada pelo robô pode ser descrita em três diferentes aspectos os quais foram analisados e implementados na modelização da movimentação, sendo estes divididos em translação, deslizamento e rotação.

Estes movimentos são considerados em separado para uma maior comodidade em termos matemáticos e de análise.

Para a modelo da acção de translação e de deslizamento foi implementado um algoritmo sugerido por Ioannis Rekleitis [4] que tem como fundamento transformar uma translação em K menores translações, conseguindo assim modelizar melhor a influência do deslizamento (tabela 3). O valor de K foi escolhido numa relação de compromisso entre uma boa modelização e o peso computacional envolvido neste processo.

Nos desvios-padrões para este tipo de modelização são requeridos o uso de desvios-padrões diferentes dos modelos de translação (σ_{trs}) e de deslizamento (σ_{drft}) como podemos observar pelas equações definidas em 5.8 e 5.9.

$$\sigma'_{trs} = \sigma_{trs} * \sqrt{(K)} \qquad (5.8)$$

$$\sigma'_{drft} = \sigma_{drft} * \sqrt{(K/2)} \qquad (5.9)$$

Entrada: Definição de M Partículas::S; Translação de uma distância:: ρ

$\delta\rho = \frac{\rho}{K};$
for ($j=1$ to M) do /* Para cada partícula */
  for ($j=1$ to K) do /* A cada passo K */
    $E_{trs} = randN(\delta\rho * M_{trs}, \delta\rho * \sigma'_{trs});$
    $E_{drft} = randN(\delta\rho * M_{drft}, \delta\rho * \sigma'_{drft});$
    $\theta[j] = \theta[j] + E_{drft};$
    $x[j] = x[j] + (\delta\rho + E_{trs} * \cos(\theta[j]));$
    $y[j] = y[j] + (\delta\rho + E_{trs} * \sin(\theta[j]));$
    $E_{drft} = randN(\delta\rho * M_{drft}, \delta\rho * \sigma'_{drft});$
    $\theta[j] = \theta[j] + E_{drft};$
  end
  $S'[j] = [x[j], y[j], \theta[j]]^T$
end

Tabela 3: Algoritmo em pseudo-código da modelização da acção de translação e de deslizamento

Para a modelização da rotação o algoritmo aplicado é bastante mais simples (tabela 4), onde simplesmente é adicionado ao ângulo anterior o deslocamento parcial efectuado juntamente com ruído gaussiano.

Entrada: Definição de M Partículas::S; Rotação :: $\delta\theta$
for ($j=1$ to M) do /* Para cada partícula */
  $E_{rot} = randN(\delta\theta * M_{rot}, \delta\theta * \sigma'_{rot});$
  $\theta[j] = \theta[j] + \delta\theta + E_{rot};$
  $S'[j] = [x[j], y[j], \theta[j]]^T$
end

Tabela 4: Algoritmo em pseudo-código da modelização da acção de rotação

Estes algoritmos definidos nas tabelas 3 e 4, constituem a acção ao qual se associou o termo α. Esta acção irá como referido anteriormente propagar a mesma acção executada pelo robô afectada de ruído gaussiano às diversas partículas, desta forma as partículas irão mover-se pelo mundo em diferentes direcções conforme a sua orientação definida pela sua "*pose*". Esta é a fase de previsão definida na tabela 5).

for ($j=1$ to M) do /* Fase de previsão devido à acção α */
  $X_j^k = \hat{f}(X_j^{k-1}, \alpha);$
end

Tabela 5: Algoritmo em pseudo-código do procedimento de previsão

Devido a todos os sensores estarem associados a "*drivers*" de controlo externos ou a programas de comunicação que controlam as diversas informações obtidas, a fase de sensorização não foi implementada directamente no algoritmo de fusão de informação. Isto deve-se no caso dos "*drivers*" associados aos "*encoders*", sonares, medidores de distâncias por infra-vermelhos terem um tempo de

aquisição de informação não desprezável ou no caso da câmara, esta disponibiliza informação sempre que tenha dados relativos à detecção de um código de barras.

Assim optou-se por definir o pedido de informação aos respectivos sensores, com a excepção da câmara no início do ciclo de controlo, para que a informação esteja já disponível aquando do processo de fusão de informação ou no caso específico da câmara, a informação irá ser acumulada até ao momento em que será processada por este processo.

Dado que o programa de controlo foi realizado, assim como o Filtro de Partículas numa linguagem orientada por eventos, tira-se desde modo vantagem da potencialidade de processamento de informação à medida que certos e determinados eventos são encadeados, permitindo o processamento de informação ou execução de outras funções enquanto não se recebe informações relativas aos "*drivers*".

Dado que o ciclo de controlo encontra-se definido nos 100ms e a velocidade bastante reduzida do robô, movimentando-se cerca de 0,5 m/s, o mesmo significa que durante um ciclo de controlo o robô terá andado apenas 5cm na sua velocidade máxima definida para aspirar, é possível considerar que as informações obtidas através dos sensores correspondem ao mesmo instante, tirando uma espécie de fotografia do estado do mundo naquele instante.

Desta forma aquando da fusão de informação toda a informação sensorial estará disponível e pronta a ser integrada na observação.

A fase seguinte define-se como actualização das probabilidades associadas com base nas observações. De facto, a actualização poderá ser realizada várias vezes ou em contrapartida nem uma única vez, conforme a informação obtida no momento. Dado ao conjunto de sensores disponíveis no robô, 6 sonares, 6 medidores de distâncias por infra-vermelhos e 1 câmara que poderá transmitir um conjunto de códigos de barras visíveis no momento.

No caso de modelização de sensores com base na distribuição gaussiana deverá ser aplicada a equação 5.10, cuja média é definida por μ e o respectivo desvio-padrão por σ.

$$W(Observação_i, X_j^k) = \frac{1}{\sqrt{(2*\pi*\sigma^2)}} * e^{\left(\frac{-(Xm_{ij}^k-\mu)^2}{2*\sigma^2}\right)} \qquad (5.10)$$

No caso de modelização de sensores com base na distribuição poisson deverá ser aplicada a equação 5.11, cuja média é definida por μ e o respectivo desvio-padrão por $\sqrt{(\mu)}$.

$$W(Observação_i, X_j^k) = \frac{e^{-\mu} * \mu^{Xm_{ij}^k}}{Xm_{ij}^k!} \quad (5.11)$$

No caso de modelização de sensores com base na distribuição binomial deverá ser aplicada a equação 5.12, cuja média é definida por np e o respectivo desvio-padrão por $\sqrt{(n*p*(1-p))}$.

$$W(Observação_i, X_j^k) = \frac{n! * p^{Xm_{ij}^k} * (1-p)^{(n-Xm_{ij}^k)}}{Xm_{ij}^k! * (n-Xm_{ij}^k)!} \quad (5.12)$$

Em que $Observação_i$ é a informação fornecida pelo sensor que permitirá obter os valores de média e desvio-padrão devidamente ajustados à informação obtida e Xm_{ij}^k é a medida de um sensor virtual i semelhante ao sensor que forneceu a informação, pertencente à partícula j no instante de tempo k.

Caso o sensor em causa forneça mais do que uma informação, por exemplo o caso da câmara, é possível simplificar o problema transformando um sensor que fornece três informações em três sensores que fornecem apenas uma única informação, tendo atenção a alguma relação entre as medidas.

Todo o processo de actualização está definido em pseudo-código na tabela 6, com base numa função genérica de cálculo de nos pesos.

```
for ( j=1 to M ) do
   for ( i=1 to Z ) do          /* Z é o número total de observações a serem aplicadas */
      w_j^k = w_j^(k-1) * W(Observação_i, X_j^k);
   end
end
```

Tabela 6: Algoritmo em pseudo-código do procedimento de actualização

Após a actualização das observações, o somatório do universo de todas as probabilidades referentes às partículas dificilmente será igual a 1. Este facto vai contra uma das propriedades básicas das probabilidades, equação 4.38. No seguimento deste dado, é necessário normalizar a probabilidade das partículas, para que a função densidade de probabilidade seja uma fiel representação do estado do mundo, usando para tal o algoritmo estabelecido na tabela 7.

```
for ( j=1 to M ) do          /* Normaliza os pesos */
   w_j^(k+1) = w_j^(k+1) / (sum_{j=1}^{M} w_j^(k+1));
end
```

Tabela 7: Algoritmo em pseudo-código do procedimento de normalização

A estimação é realizada através de três métodos possíveis,

especificados respectivamente através das equações 4.22, 4.23 e 4.24. Na implementação do "*CleanRob*", foram utilizados os três métodos na executados de uma forma sequencial, para permitir uma comparação e respectiva análise, tendo como base a mesma informação que permite aos três métodos estimar, de outra forma a não seria válido para este tipo de análise.

Na figura 5.8, está representado a forma fluem os diversos estágios do filtro de partículas, sendo um resumo abstracto do que foi descrito acima. A recursividade do filtro de partículas na imagem é apresentada na figura 5.8 pela ligação directa entre o estágio de estimação da "*pose*" e a reamostragem. Esta ligação não é inteiramente verdadeira, dado que o filtro de partículas é chamado pelo programa de controlo a cada ciclo de controlo, estando esta recursividade bem delimitada em intervalos de tempo bem definidos.

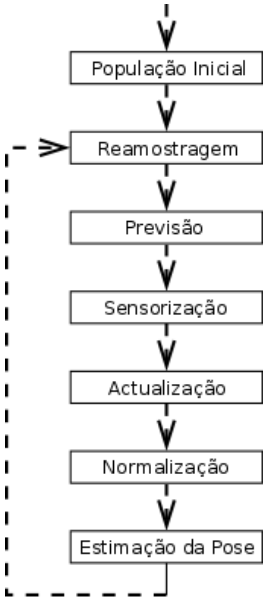


Figura 5.19: Recursividade do Filtro de Partículas

5.2 Calibração dos PID

5.2.1 Abordagem Implementada

A calibração dos PID, foi realizada por um colega que trabalha no mesmo projecto porém numa outra vertente de fusão de informação, pelo que transcrevo aqui um excerto da sua tese.[8]

De modo a controlar verdadeiramente os motores e especialmente a sua resposta ao degrau, foi necessário calibrar os controladores PID desenvolvidos em "*software*" e integrados nos microcontroladores

presentes nas placas de circuito impresso dos "*drivers*" de potência.

Na realidade, o controlador implementado no microcontrolador não é um simples controlador PID, mas contém também um filtro de medida, um filtro de referência e ainda uma malha de "*feed-forward*".

Devido a existir uma camada de "*software*", mais uma camada de controlo, necessitando ambas de ser calibradas, optou-se por utilizar uma somente o controlador de PID, uma vez que, para a aplicação pretendida as outras afinação não trazem mais valias significativas.

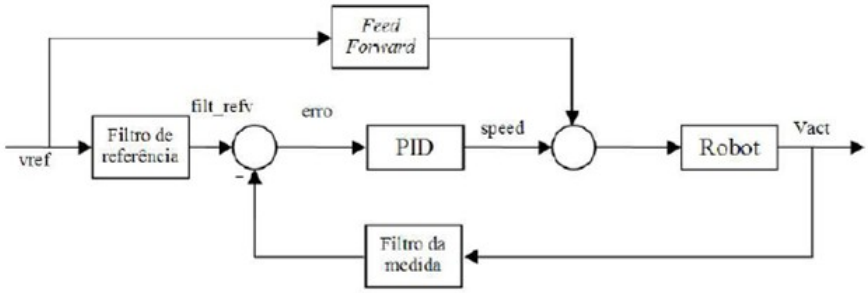


Figura 5.20: Esquema do controlador PID implementado nos "drivers" de controle dos motores

Para a determinação dos parâmetros do PID foi utilizada uma abordagem do tipo controlo por modelo interno.

Esta decisão foi realizada devido a planta do sistema não ser completamente conhecida, devido a facilidade de determinação dos valores dos parâmetros para o PID e apresentar um único factor da ajuste que se encontra directamente relacionado com o tempo de resposta do sistema.

Foi também escolhido o modelo de primeira ordem com atraso.

A opção de sistema com atraso é justificada pelo facto de existir uma diferença entre o instante em que a informação do robô é obtida e o instante em que a informação é processada pelo sistema de controlo, já que o programa realiza um ciclo de controlo cada 100ms.

Em termos de o robô se tratar de um sistema de primeira ordem, este pressuposto é considerado já que é assumido que o robô é capaz de realizar acelerações constantes, e como o sistema dinâmico considerado é em termos de velocidade, então estamos perante um sistema de primeira ordem.

Modelo do processo	**Parâmetros do controlador**		
	$k_c \cdot k$	τ_i	τ_D

$\frac{k_p \cdot e^{-Ls}}{1+\tau s}$	$\frac{\tau}{\tau_{CL}+L}$	τ	

Tabela 8: Modelo de primeira ordem com atraso, definido para a calibração dos parâmetros do PID pelo método de Chien

O método de Chien define, para este tipo de modelo, que o melhor controlador será do tipo PI, podendo os parâmetros determinados como apresentados na tabela 8.

O factor τ_{CL} corresponde a medida que controla a relação entre o tempo de resposta em malha aberta e a pretendida em malha fechada.

Foi assumido que, se pretendia um tempo de resposta que seria poderia ser até ao dobro do conseguido em malha aberta, sendo assim, definiu-se:

$$\tau_{CL}=2 \tag{5.13}$$

Com a calibração do PID pretende-se diminuir o erro em regime permanente, assim como, definir um melhor tempo de resposta do sistema.

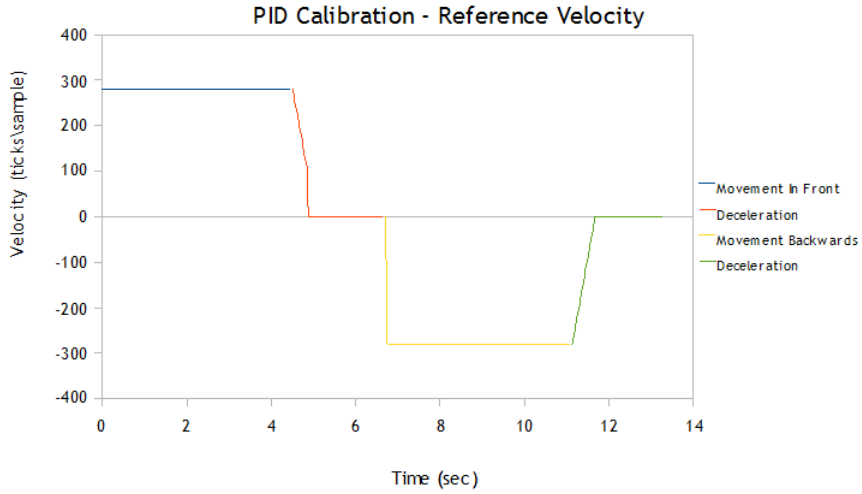


Figura 5.21: Referências de velocidade aplicadas para a determinação dos parâmetros do PID

Para a determinação dos dados sobre os motores, de modo a realizar a calibração, foi utilizada a seguinte abordagem: colocou-se os motores em funcionamento em malha aberta, aplicando quatro degraus de velocidade:

1. velocidade no sentido positivo, durante aproximadamente 4 segundos, com um valor de 280 ticks/amostragem;

2. rampa de desaceleração até uma velocidade de referência de 0

ticks/amostragem, permanecendo nesta fase durante 3 segundo;

3. velocidade no sentido negativo, durante aproximadamente 4 segundos, com um valor de -280 ticks/amostragem;

4. rampa de desaceleração até uma velocidade de referência de 0 ticks/amostragem, permanecendo nesta fase durante 3 segundo.

5.2.2 Dados Experimentais

De seguida, apresentarei os dados já tratados para cada um dos motores, através de representações gráficas dos "*runs*" realizados e ainda dos valores calculados para cada um dos parâmetros do controlador, sendo desta vez apresentado em forma tabular.

O atraso do sistema é de 100ms, correspondendo este valor ao período de entre chamadas do ciclo de controlo no programa que supervisiona e controla todo o processamento do robô.

O tempo de estabelecimento corresponde ao tempo decorrente desde do instante em que o sistema começa a responder a solicitação de movimento e o instante em que o valor de regime permanente é obtido.

As referências enviadas para os "*drivers*", provenientes do programa de controlo, são em termos de velocidade, sendo assim, é sobre os dados presentes nos gráficos apresentados nas figuras 5.22 e 5.23 que se calculam os parâmetros do PID para os motores.

Motor da Direita

Analisemos agora os dados referente ao motor da direita.

Inicialmente este irá se movimentar no sentido positivo considerado para a movimentação do robô, sendo então a velocidade medida positiva, passando depois num segundo momento a velocidade a ser negativa.

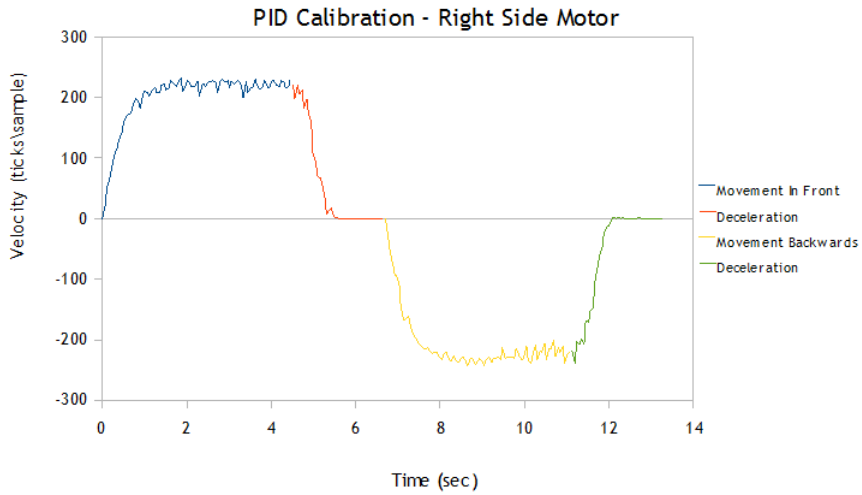


Figura 5.22: Gráfico da resposta em malha aberta do motor da direita, em termos de velocidade

É possível resumir os dados experimentais, obtidos para o motor da direita, e necessários para a calibração do controlador, sendo estes apresentados na tabela 9.

Características de dinâmica para o motor da direita	***Movimentação no sentido positivo***	***Movimentação no sentido negativo***
Atraso (seg)	0,1000	0,1000
Valor médio em regime permanente (ticks/amostragem)	216,3683	-227,0039
Tempo de estabelecimento (seg)	0,9496	0,7542

Tabela 9: Resumo dos principais dados obtidos sobre a dinâmica do motor da direita

Tendo em consideração os dados apresentados na tabela 9 e fórmulas de cálculo dos parâmetros de configuração do PID, segundo o Método Chien, presente na tabela 8, podemos determinar os parâmetros a utilizar para o motor da direita, que são apresentados na Tabela 9.

Método de Chien	***Movimentação no Sentido Positivo***	***Movimentação no Sentido Negativo***
Coeficiente de Chien	*2,0000*	*2,0000*
Ganho da Planta	*0,0009*	*-0,0007*
Ganho do Controlador	*0,1882*	*-0,2379*
Tempo Integral	*0,0095*	*0,0075*
Tempo Derivativo	*0,0000*	*0,0000*

Tabela 10: Resumo dos dados obtidos para a calibração do controlador PID, segundo o método de Chien, para o motor da direita

Uma característica, que é necessário também controlar, ou pelo

menos ter em consideração é a corrente. A resposta em termos de corrente para o motor da direita e com a aplicação das referência de velocidade utilizadas encontra-se graficamente representada na figura 5.23.

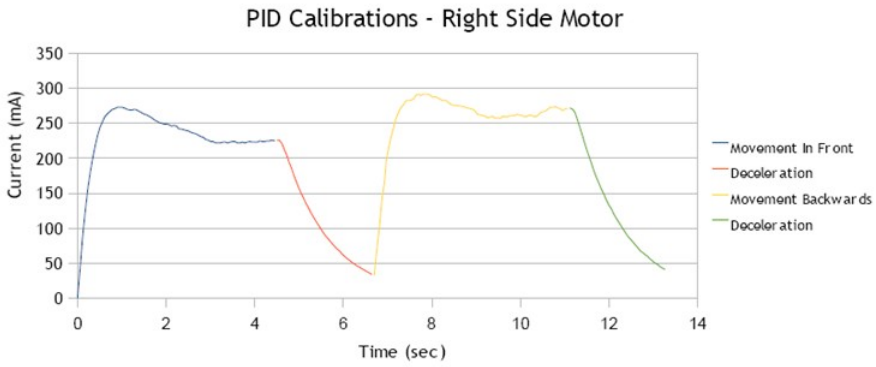


Figura 5.23: Gráfico da resposta em malha aberta do motor da direita, em termos de corrente

Podemos verificar que, em termos de corrente, o tempo de estabelecimento é muito superior ao que acontece em termos do gráfico da velocidade, sendo neste caso, de aproximadamente 3 segundos. É possível verificar a existência de "*overshoot*" no valor da corrente, que neste caso é de aproximadamente 22%.

Como o robô normalmente irá realizar deslocamentos no sentido positivo, iremos considerar os parâmetros para colocar no PID como os valores apresentados para o sentido positivo.

Ganho do Controlador	*0,1882*
Tempo Integral	*0,0095*
Tempo Derivativo	*0,0000*

Tabela 11: Parâmetros do PID utilizados para o motor da direita

Com estes parâmetros, o motor da direita passou a ter as respostas em termos de velocidade e corrente apresentadas respectivamente nas figuras 5.24 e 5.25.

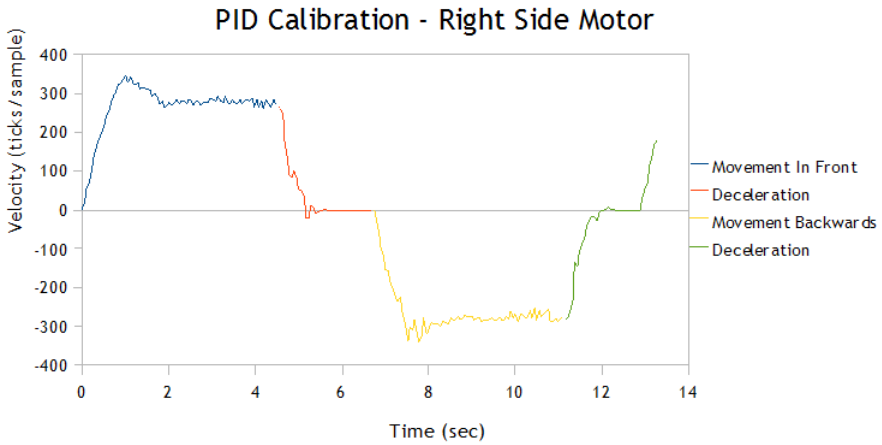


Figura 5.24: Gráfico da resposta em malha fechada e controlada do motor da direita, em termos de velocidade

Em termos de velocidade verificamos que o erro em regime permanente foi bastante reduzido mas, passou a existir um "*overshoot*" de velocidade aproximadamente de 24%.

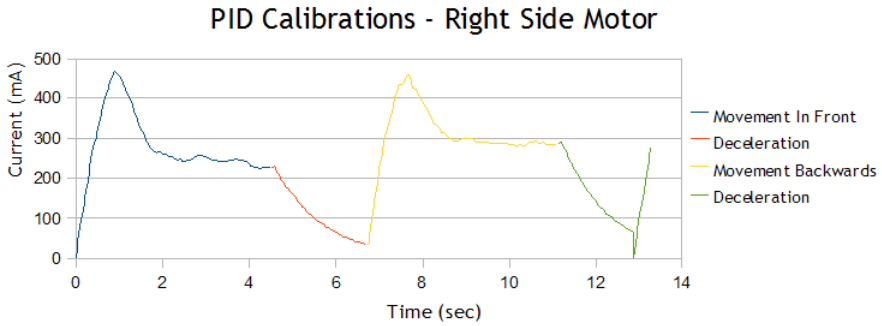


Figura 5.25: Gráfico da resposta em malha fechada e controlada do motor da direita, em termos de corrente

Em termos de corrente verificamos que me termos do valor médio da corrente este não é muito alterado, mas em termos de "*overshoot*" é aumentado consideravelmente passando agora para perto dos 85%, mesmo assim abaixo do limite de corrente aplicado.

Motor da Esquerda

Agora é a vez de analisar os dados referente ao motor da esquerda.

Inicialmente, este irá movimentar-se no sentido positivo considerado para a movimentação do robô, sendo então a velocidade medida negativa, passando depois num segundo momento a velocidade a ser positiva.

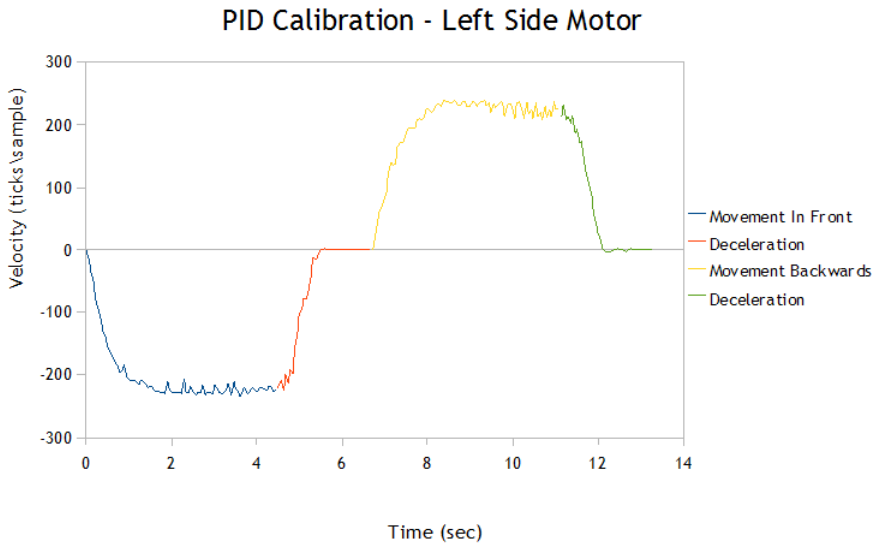


Figura 5.26: Gráfico da resposta em malha aberta do motor da esquerda, em termos de velocidade

É possível resumir os dados experimentais, obtidos para o motor da esquerda, e necessários para a calibração do controlador, sendo estes apresentados na tabela 12.

Características de dinâmica para o motor da esquerda	***Movimentação no sentido positivo***	***Movimentação no sentido negativo***
Atraso (seg)	*0,1000*	*0,1000*
Valor médio em regime permanente (ticks/amostragem)	*-219,8629*	*219,4519*
Tempo de estabelecimento (seg)	*0,9629*	*0,8386*

Tabela 12: Resumo dos principais dados obtidos sobre a dinâmica do motor da esquerda

Tendo em consideração os dados apresentados na tabela 12 e fórmulas de cálculo dos parâmetros de configuração do PID, segundo o Método Chien, presente na tabela 8, podemos determinar os parâmetros a utilizar para o motor da esquerda, que são apresentados na tabela 13.

Método de Chien	***Movimentação no Sentido Positivo***	***Movimentação no Sentido Negativo***
Coeficiente de Chien	*2,0000*	*2,0000*
Ganho da Planta	*-0,0009*	*0,0008*
Ganho do Controlador	*-0,1891*	*0,2113*
Tempo Integral	*0,0096*	*0,0084*
Tempo Derivativo	*0,0000*	*0,0000*

Tabela 13: Resumo dos dados obtidos para a calibração do controlador PID, segundo o método de Chien, para o motor da direita.

Uma característica, que é necessário também controlar, ou pelo menos ter em consideração é a corrente. A resposta em termos de corrente para o motor da esquerda e com a aplicação das referência de velocidade utilizadas encontra-se graficamente representada na figura 5.26.

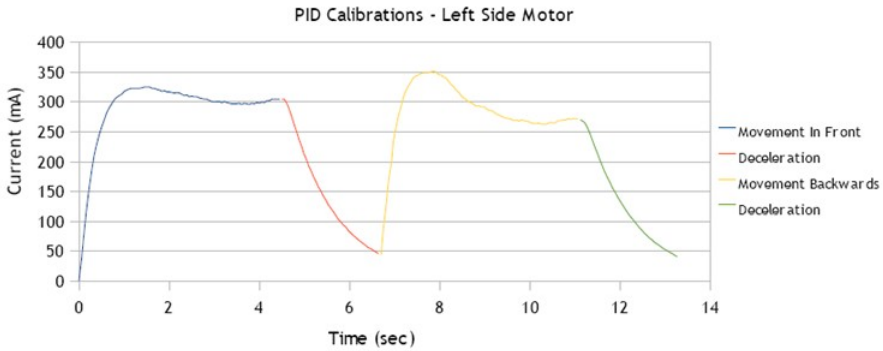


Figura 5.27: Gráfico da resposta em malha aberta do motor da esquerda, em termos de corrente

Podemos verificar que, em termos de corrente, o tempo de estabelecimento é muito superior ao que acontece em termos do gráfico da velocidade, sendo neste caso, de aproximadamente 3 segundos. É possível verificar a existência de "*overshoot*" no valor da corrente, sendo neste caso de 6% no caso da movimentação no sentido positivo e de 30% no sentido negativo.

Como o robô normalmente irá realizar deslocamentos no sentido positivo, iremos considerar os parâmetros para colocar no PID como os valores apresentados para o sentido positivo.

Ganho do controlador	*0,1891*
Tempo integral	*0,0096*
Tempo derivativo	*0,0000*

Tabela 14: Parâmetros do PID utilizados para o motor da esquerda

Com estes parâmetros, o motor da direita passou a ter as respostas em termos de velocidade e corrente apresentadas respectivamente nas figuras

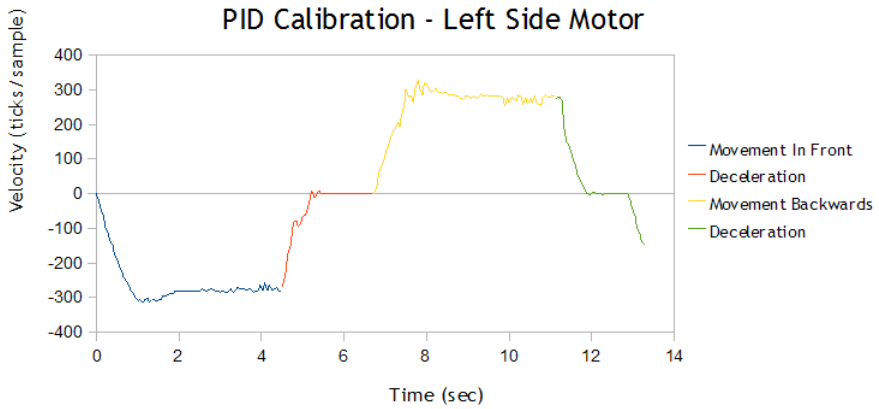


Figura 5.28: Gráfico da resposta em malha fechada e controlada do motor da esquerda, em termos de velocidade

Em termos de velocidade verificamos que o erro em regime permanente foi bastante reduzido mas, passou a existir um "*overshoot*" de velocidade aproximadamente de 12%.

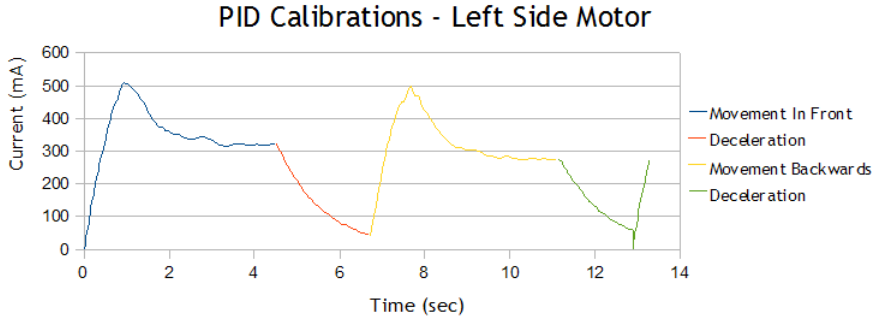


Figura 5.29: Gráfico da resposta em malha fechada e controlada do motor da esquerda, em termos de corrente

Em termos de corrente verificamos que me termos do valor médio da corrente este não é muito alterado, mas em termos de "*overshoot*" é aumentado consideravelmente passando agora para perto dos 50%, mesmo assim abaixo do limite de corrente aplicado.

5.2.3 Conclusões

Em suma, a calibração dos controladores PID veio melhorar a performace dos motores e acima de tudo minimizar o erro entre a velocidade pretendida e a velocidades reais obtidas.

É necessário ter em consideração o valor da corrente máxima, especialmente porque os "*drivers*" de potência apresentam uma limitação no valor de corrente máxima a fornecer aos "*drivers*", sendo este valor de 1A.

Considerando que a velocidade normal de funcionamento do robô

será a velocidade testada, foi possível comprovar que os limites máximos de corrente não são atingidos, mesmo após a calibração que provocou uma maior intensidade na corrente pedida e em consequência um maior valor de “*overshoot*”.

5.3 Calibração e Modelo da Hodometria

5.3.1 Abordagem Implementada

A calibração da hodometria é um procedimento no mínimo moroso e complexo, não existindo nenhum guia de como o fazer correctamente nem equipamento adequado para tal nas instalações da faculdade.

Dado que a hodometria (como referido anteriormente na secção de sensorização acerca dos "*encoders*"), baseia-se sobretudo em duas constantes físicas do robô, ou seja na distância entre rodas e no perímetro das mesmas.

O procedimento mais exacto para a calibração da hodometria seria executar a mesma com o recurso a estruturas que possibilitassem a realização desta com a menor interferência de factores externos e com isto maximizar a repetição de resultados, para tal o conveniente seria a implementação de uma estrutura de duas vias paralelas, como por exemplo (caminho de ferro) no qual o robô se deslocaria, obtendo apenas assim a componente tangencial da velocidade no robô e provocando que a velocidade angular seja nula devido a não haver rotação do robô em relação ao seu ponto central. Tal facto pode ser constatado facilmente pela equação 5.5 que define a velocidade tangencial e a velocidade angular do robô através das velocidades tangenciais de cada roda. O segundo passo seria efectuado através de uma estrutura circular com um raio adaptável de forma a ser facilmente ajustado à distância entre rodas do robô, assim seria possível realizar a rotação do mesmo sobre o seu ponto central, calibrando assim o factor utilizado no cálculo da velocidade angular.

Dado não existir este tipo de equipamento/estrutura, recorreu-se a outro método de calibração, embora criativo, mas que fosse fiável e que minimizasse a possibilidade de erros. Para tal, utilizou-se uma intercepção de paredes rectas como base de referência de um sistema de coordenadas ortogonais. Delimitou-se na parede associada ao eixo das abcissas (XX's) dois pontos distanciados de 1 metro.

Utilizou-se duas almofadas de teflon colocadas no pára-choques do robô de forma a que este deslizasse facilmente usando a mesma parede

como guia, como ilustrado na figura 5.30. Foi posicionado o centro do robô no primeiro ponto e deslocou-se o mesmo manualmente até que o centro do robô se encontrasse no segundo ponto. Desta forma foi adquirido o número total de "*ticks*" fornecidos por cada "*encoder*" associado à respectiva roda e dada a distância percorrida obteve-se a relação entre o perímetro de cada roda com o número de "*ticks*". Esta relação indica-nos a distância equivalente a 1 "*tick*". O motivo de existir duas constantes, cada uma destas associada a uma roda prende-se ao facto das dimensões das rodas, polias serem ligeiramente diferentes e mesmo o posicionamento dos "*encoders*" serem ligeiramente diferentes.

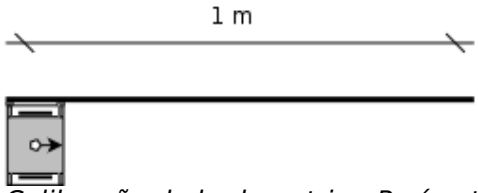


Figura 5.30: Calibração da hodometria - Perímetro das rodas

Resta-nos então a calibração do segundo factor, a distância entre rodas. Para a calibração deste factor, após a calibração do perímetro das rodas, é realizada posicionando o robô a uma distância fixa da parede e com o recurso a um apontador laser posicionado no ponto central do robô, em que a projecção deste incide na origem de um quadro alvo fixado na parede, como se pode ver pela ilustração da figura 5.31.

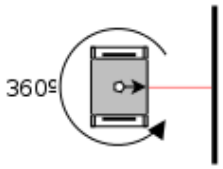


Figura 5.31: Calibração da hodometria - Distância entre rodas

O robô foi rodado 360º em torno do seu eixo com o auxilio de uma calha circular e posicionado com a ajuda do laser na origem do quadro de alvos, medindo-se assim os "*ticks*" fornecidos pelos "*encoders*". À semelhança da calibração dos perímetros das rodas e dado que pela calibração anterior obtemos a relação da distância percorrida com o número de "*ticks*" de cada roda. Assumindo que o ponto central do robô, não se deslocou, a trajectória formada na rotação corresponde a uma circunferência com o diâmetro igual à da distância entre rodas. Então usando a distância percorrida por estas (ambas distâncias idênticas), obtemos assim a distância entre rodas com o recurso à formula de cálculo do perímetro de uma circunferência.

$$P_{circunferência} = 2 * \pi * r \Leftrightarrow \pi * Diâmetro \qquad (5.14)$$

5.3.2 Dados Experimentais

Após as calibrações acima descritas, por mais tentativas que se realize é praticamente impossível conseguir que a hodometria seja perfeita, devido a erros relativos à resolução do "*encoder*", tempos de amostragem, variação do ponto de contacto com o piso, variação da distância entre rodas devido à presença de uma placa metálica de suporte aos motores que provoca um leve amortecimento, deslizamento das rodas, pelo que é lógico considerar a abordagem de um modelo da hodometria para que se possa enquadrar o seu comportamento num conjunto de valores cuja as medidas obtidas pela hodometria estejam contidas. Para tal, à semelhança da calibração dos perímetros das rodas e da distância entre estas, utilizou-se a intercepção das paredes como referência de um eixo de coordenadas ortogonal.

O robô foi posicionado a uma distância X_i e Y_i em das respectivas paredes consideradas referenciais e o seu ângulo θ_i definido como nulo, como demonstrado na figura 5.32. Então através do programa de controlo do robô, foi ordenado que esta percorresse uma distância de 1m a uma velocidade baixa e no final deste obtido os respectivos valores, tendo-se repito este processo várias vezes.

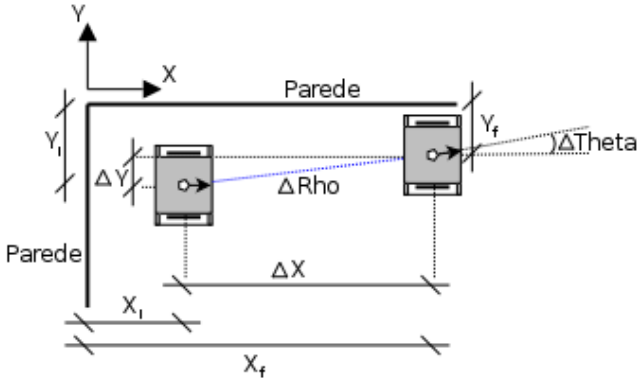


Figura 5.32: Aquisição do modelo de translação e deslizamento

Os dados obtidos por esta experiência encontram-se apresentados na tabela 23 no anexo 1 e permitem formulação dos modelos associados ao movimento de translação e do deslizamento do robô.

O deslizamento do robô é caracterizado por um desvio no ângulo do robô, provocado por uma assimetria dos pesos dos diversos componentes que o compõem, bem como diferentes respostas por parte dos motores originam a respectiva locomoção, causando um comportamento diferente na translação do mesmo.

Para o modelo da hodometria no que respeita às componentes de translação e deslizamento a distribuição que mais de adequa é a distribuição normal pela caracterização dos erros como se poderá observar pela figura 5.33, em que consta-se uma tendência da propagação dos erros dentro de um determinado intervalo. Dado que

nesta experiência foi definido antecipadamente um deslocamento de 1m, os valores obtidos de erro já estão normalizados nas unidades do sistema internacional.

Modelo da hodometria - Translação			
[m / m]	**X**	**Y**	**Rho**
Média	0,0212	-0,0139	0,0212
Desvio padrão	0,0040	0,0035	0,0040

Tabela 15: Modelo da hodometria – Translação

Modelo da hodometria - Deslizamento	
[Radianos / m]	**Theta**
Média	-0,0013
Desvio padrão	0,0027

Tabela 16: Modelo da hodometria - Deslizamento

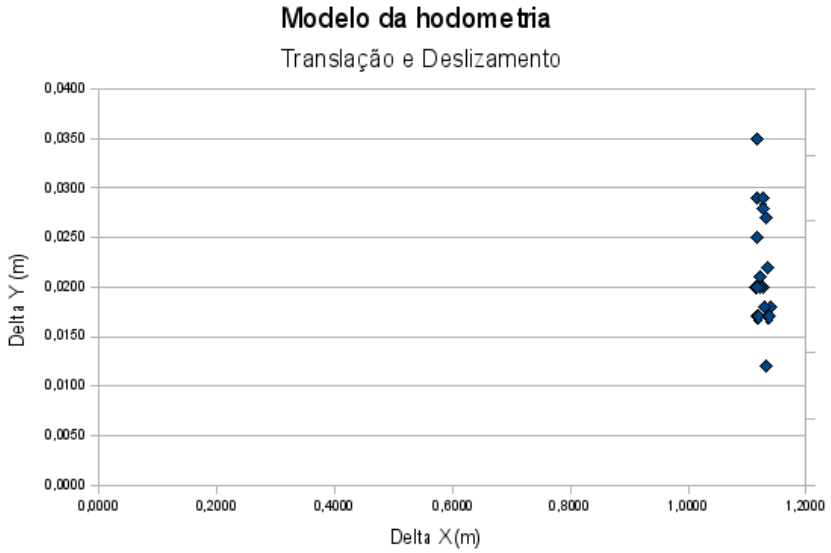


Figura 5.33: Representação gráfica dos dados experimentais na obtenção do modelo de translação e deslizamento

Para o modelização da rotação, posicionou-se o robô à frente de uma parede e com recurso a um ponteiro laser, acertou-se a sua orientação de forma a ficar coincidente com a origem de um quadro de alvos. Através do programa de controlo ordenou-se ao robô perfazer uma volta completa em torno do seu ponto central, como ilustrado pela figura 5.34.

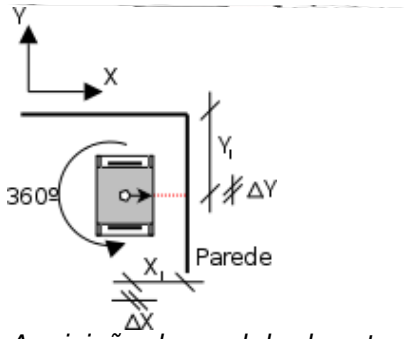


Figura 5.34: Aquisição do modelo de rotação

Dado que o robô não conseguiu rodar em torno do seu eixo (tendo existido sempre um deslocamento deste) optou-se por compensar este deslocamento recalculando o theta final com base nestes deslocamentos, aproximando-se bastante do theta obtido pelo robô. Os dados retirados estão compilados na tabela 26 no anexo 1.

Tal como no modelo da translação e do deslizamento, o modelo da rotação pode ser descrito como uma distribuição normal facilmente visualizado no gráfico da figura 5.35.

Modelo da hodometria - Rotação	
[Radianos / 2π]	**Theta**
Média	-0,0086
Desvio padrão	0,0068

Tabela 17: Modelo da hodometria – Deslizamento

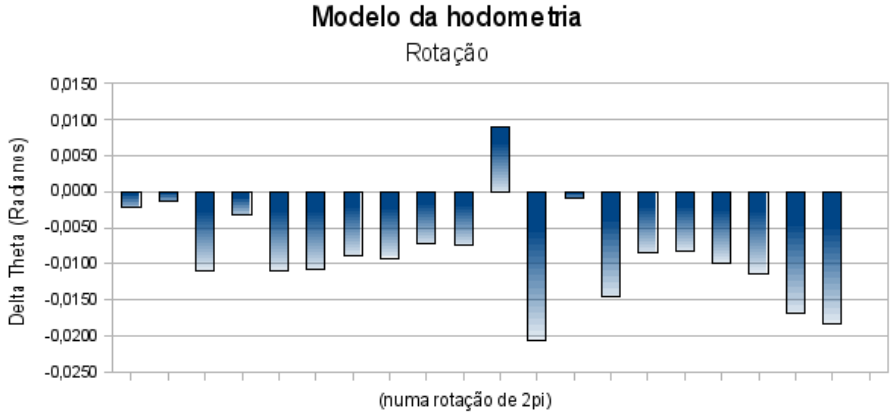


Figura 5.35: Representação gráfica dos dados experimentais na obtenção do modelo de rotação

5.3.3 Conclusões

A hodometria como fonte de informação possui uma importância que não pode ser ignorada. A sua calibração não é realizada de uma forma simples, nem poderá ser exacta, possuindo inúmeros erros que não são possíveis de ser eliminados, quanto muito no melhor esforço, poderão ser minimizados.

Os erros de hodometria possuem a particularidade de serem acumulativos, por outras palavras, os erros são aproximadamente lineares conforme a distância percorrida. Isto significa que recorrendo a um modelo matemático, poderemos prever (dentro de uma certa gama de valores), os erros que a hodometria poderá assumir. Desta forma podemos obter um certo grau de confiança sobre a fiabilidade dos valores desta, reutilizando-os numa fase posterior em algoritmos de fusão de informação de forma a validar ou rejeitar a informação transmitida por esta.

5.4 Modelo da câmara

5.4.1 Abordagem Implementada

Para a integração da informação da câmara na fusão de informação é necessário realizar caracterizar o modelo do erro que acompanha as medidas por esta fornecida. Dado que a câmara é um sensor que possibilita uma grande aquisição de informação, será necessário o uso de algoritmo de processamento de imagem, para retirar a informação útil ao nosso propósito. Neste caso em concreto o uso de um algoritmo detecta um código de barras e que fornece valores respectivos à distância e ângulo do mesmo em relação à câmara.

Estando a análise deste algoritmo fora do âmbito desta tese, não é analisado os erros relativos a todo este processo, contudo assumindo que os dados fornecidos pela câmara são de certa forma equivalentes a qualquer outro sensor, poderemos simplificar o problema da modelização da câmara como sensor.

Assim, aproveitando que o professor Doutor Armando Sousa encontra-se a desenvolver um artigo sobre esta abordagem relativa ao uso da câmara como um detector de código de barras, facultou a este projecto, os valores experimentais obtidos na construção de modelos para o respectivo artigo.

Dados esses que foram analisados por mim e que me permitiram a construção de um modelo para este tipo de sensor.

5.4.2 Dados Experimentais

Dada à extensão dos dados facultados pelo professor Doutor Armando Sousa para a modelização dos erros relativos a este tipo de sensor, não poderei anexa-los a este documento, pelo que poderá ser consultado na página de internet relacionada a este projecto.

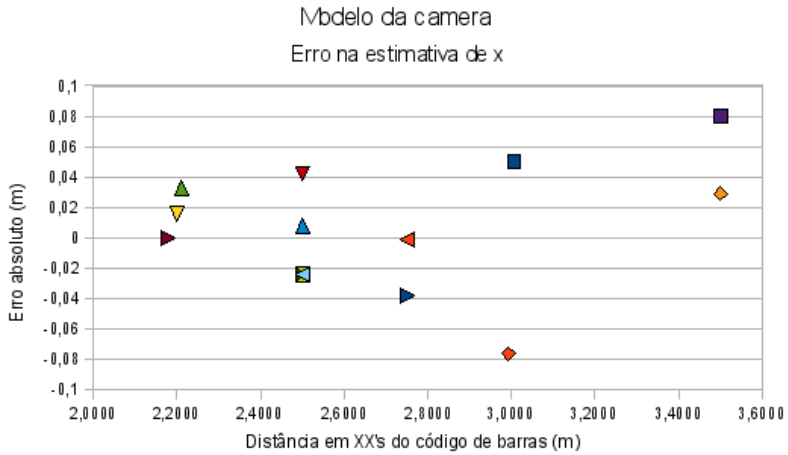


Figura 5.36: Representação gráfica dos erros absolutos da estimação em X do sensor em relação à distância no mesmo eixo

Modelo da câmara	
[m / m]	**X**
Média	0,0182
Desvio padrão	0,0562

Tabela 18: Modelo do erro da estimação em X da câmara

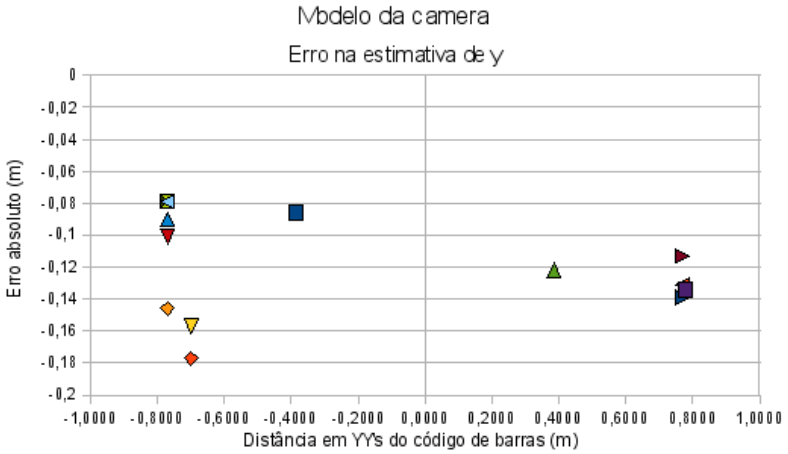


Figura 5.37: Representação gráfica dos erros absolutos da estimação em Y do sensor em relação à distância no mesmo eixo

Modelo da câmara	
[m / m]	**Y**
Média	0,0192
Desvio padrão	0,1810

Tabela 19: Modelo do erro da estimação em Y da câmara

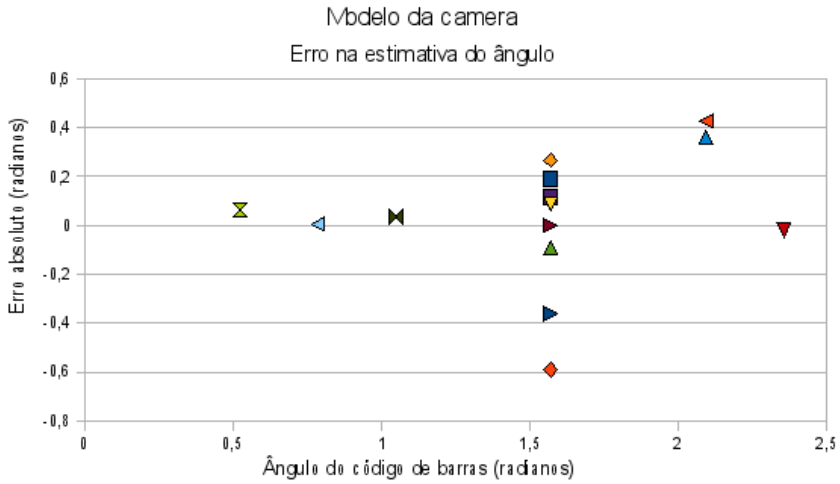


Figura 5.38: Representação gráfica dos erros absolutos da estimação em do ângulo do sensor em relação ao ângulo do código de barras

Modelo da câmara	
[rad / rad]	**Ângulo**
Média	0,0383
Desvio padrão	0,1396

Tabela 20: Modelo do erro da estimação do ângulo da câmara em relação com o ângulo do código de barras

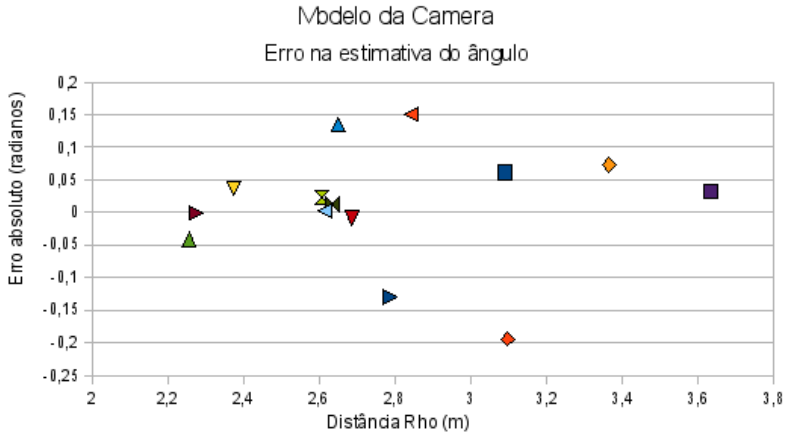


Figura 5.39: Representação gráfica dos erros absolutos da estimação do ângulo do sensor em relação à distância que o código de barras está posicionado

Modelo da câmara	
[rad / m]	**Ângulo**
Média	0,0204
Desvio padrão	0,0812

Tabela 21: Modelo do erro da estimação do ângulo da câmara com a distância que o código de barras está posicionado

5.4.3 Conclusões

Como podemos constatar pelos modelos apresentados nas tabelas

18, 19, 20 e 21, que a câmara como sensor que detecta códigos de barras e fornece distâncias e ângulo relativas ao mesmo, será de grande importância na estimativa da posição de um robô. Contudo estes mesmos modelos, indicam que a estimativa em Y e do ângulo, possuem um erro não muito elevado, porém não desprezável.

No entanto não existem sensores perfeitos e face à modelização do erro a fusão de informação terá que gerir as informação obtida e com esta actuar de forma apropriada.

5.5 Sistema de Controlo

O sistema de controlo foi em grande parte desenvolvido por um colega de projecto, no qual se focou mais nesta área, enquanto procurei focalizar-me nos aspectos mecânicos e eléctricos do robô. Face ao à perspectiva do programa de controlo ser comum a ambos, passo a transcrever esta secção, adaptando certos assuntos ao objectivo desta tese, nomeadamente à fusão de informação usando o Filtro de Partículas. [8]

Para a realização de um controlo e decisão no robô e Limpeza, foi definida uma arquitectura de "*software*" robusta, que utiliza as potencialidades de uma linguagem "*event-driven*" (orientada a eventos), tal como o "*Lazarus*". A arquitectura definida proporcionou a melhor interligação possível com todos os elementos que compõem o projecto.

Tendo por base o programa de controlo e decisão desenvolvido para o robô de Limpeza, é possível definir que os principais eventos, encontrando-se estes representados na figura 5.40.

Cada evento externo é processado de forma independente e assíncrona, ou seja, após a recepção do evento, e logo que o programa termine todas as acções que se encontrava a fazer, é que é propriamente processado. O instante de inicio do seu processamento assim como o tempo de processamento são calculados e guardados.

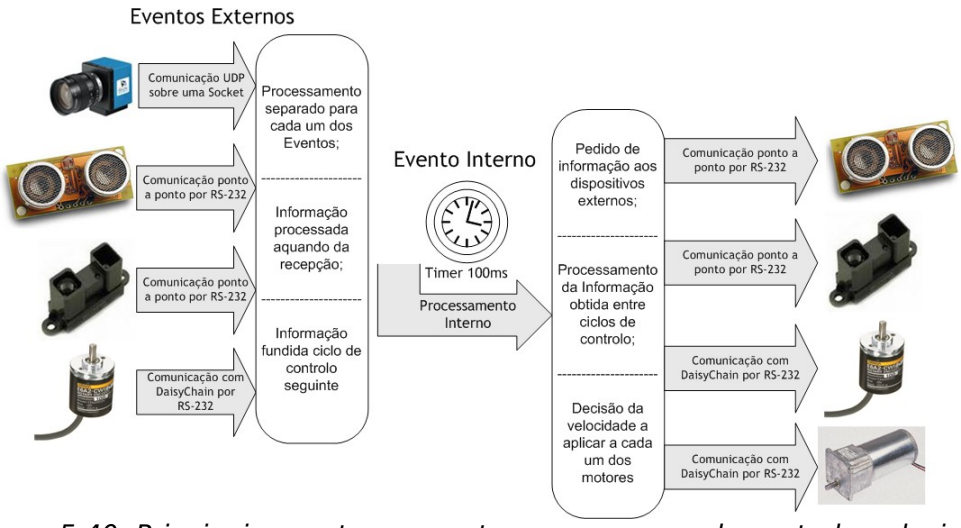


Figura 5.40: Principais eventos presentes no programa de controlo e decisão do "CleanRob"

O evento interno, ou seja, o "*timer*" que controla o tempo entre ciclos de controlo, é lançado periodicamente de 100ms em 100ms. É durante o ciclo de controlo que todas as decisões e processamentos importantes são realizados. Os principais blocos de acções presentes no ciclo de controlo, encontram-se apresentados na figura 5.41.

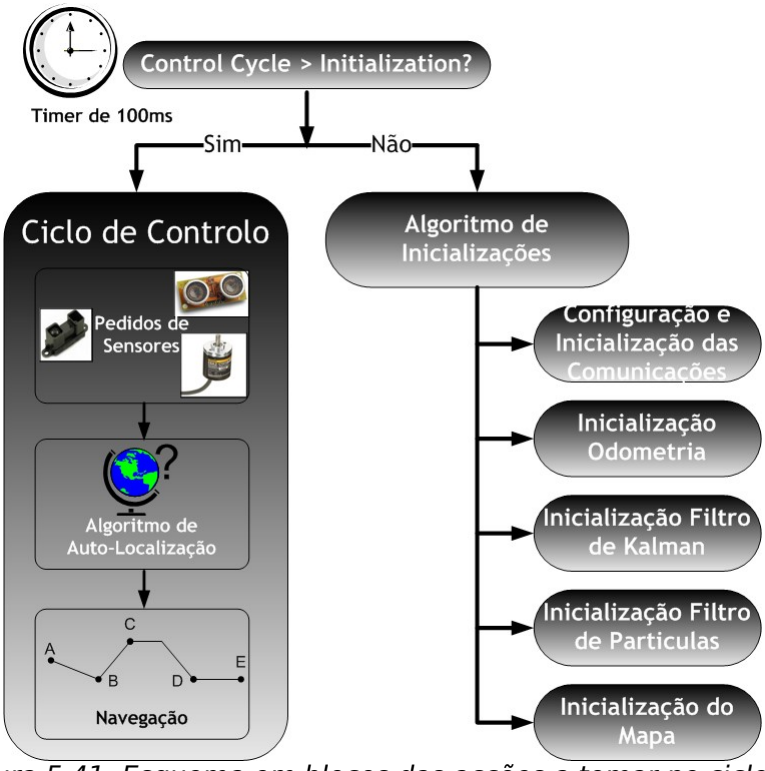


Figura 5.41: Esquema em blocos das acções a tomar no ciclo de controlo

Nas primeiras entradas no ciclo de controlo, é necessário realizar algumas inicializações, tanto ao nível das comunicações com os dispositivos externos, com reconfiguração de alguns parâmetros, como

dos métodos de auto-localização presentes no robô de limpeza (Filtro de Kalman e Filtro de Partículas), ou ainda, a leitura do ficheiro de mapa da área que o robô de limpeza irá limpar, não só para o conhecimento dos códigos de barras da área, paredes, mas também para se poder utilizar como ferramenta de "*debug*" ou de controlo visual da posição actual do "*CleanRob*".

Passado todas estas inicializações, o "*CleanRob*" está pronto para se começar a movimentar-se, realizando sempre os blocos apresentados no lado esquerdo da figura 5.41.

Primeiro de tudo, são realizados os pedidos de medidas dos diversos dispositivos externos, de modo a apresentar no próximo ciclo de controlo informações novas sobre o estado actual do robô de limpeza. Feito isto, agora é necessário estimar a posição actual e qual a acção a desempenhar, acções estas que são responsáveis os dois outros blocos principais dentro do ciclo de controlo.

Em relação ao bloco de auto-localização que inclui o Filtro de Kalman e o Filtro de Partículas, a sua implementação encontra-se esquematizada na figura 5.42.

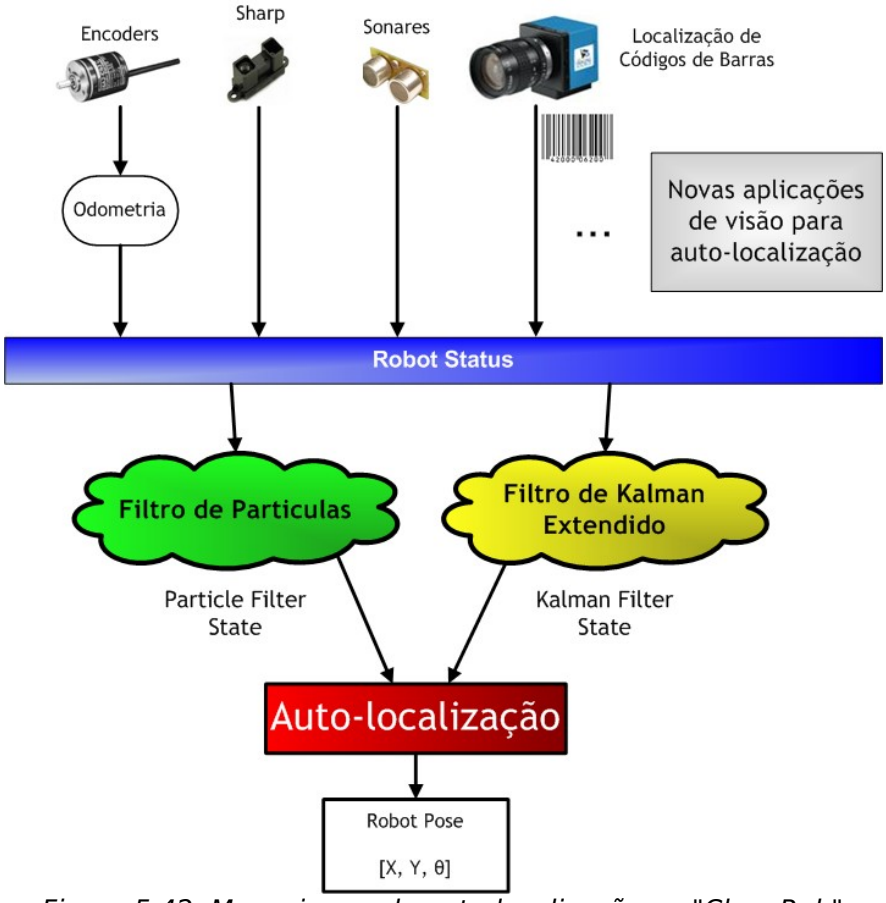


Figura 5.42: Mecanismos de auto-localização no "CleanRob"

A informação sobre o estado dos diversos sensores encontra-se acessível na estrutura robô "*status*" (tabela 22). Os métodos de auto-localização utilizam essa informação, validam-na e integram-na nos seus processamentos afim de chegar a uma estimativa para a posição actual do "*CleanRob*".

Os processos realizados pelo bloco do Filtro de Partículas são apresentados de uma forma mais exaustiva ao longo deste documento.

Em relação ao Filtro de Kalman a sua aplicação no "*CleanRob*" pode ser investigada recorrendo a dissertação do meu colega de projecto responsável por esse método. [8]

Por fim, e voltando ao esquema da figura 5.41 temos a fase de Navegação. Aqui, foi implementado um sistema de "*path planning*" baseado em "*checkpoints*" de posições globais que o robô de Limpeza terá que se passar durante a sua missão.

A relação entre a posição estimada do robô de limpeza e a posição do "*checkpoint*" é então processada e recorrendo a funções de movimentação são definidas as velocidades tangencial e angular, a aplicar ao "*CleanRob*", de modo a que este consiga chegar ao seu objectivo.

Todo o funcionamento do programa de controlo e decisão encontra-se sustentado por uma definição global de uma estrutura que representa uma verdadeira compilação de toda a informação presente no "*CleanRob*".

```
type TStatus=record
  TimeStamp: Dword;
  case StatusData: integer of
    0: (OdometryData: TOdometry);
    1: (SonarData: TSonar);
    2: (BarCodeData: TBarCode);
    3: (SharpData: TSharp);
    4: (KalmanFilterData: TKalmanFilter);
    5: (ParticlesFilterData: TParticlesFilter);
    6: (RobotInfo: TRobotInfo);
end;
```

Tabela 22: Estrutura da variável TStatus

A escolha da utilização de uma estrutura tipo "*union*" para o robô "*status*", foi decidida devido ao assicronismo dos eventos externos. Assim num mesmo ciclo de controlo não é garantido que todas as sub-estruturas fossem preenchidas, evitando assim a alocação de memória desnecessária.

Devido a este facto, e como para um mesmo ciclo de controlo, mais que uma sub-estrutura é escrita, ou seja, informação diferente é avaliada. Este facto levou a duas novas decisões, primeiro a introdução da variável "*timeStamp*", que define o instante em que a estrutura foi escrita e ainda uma forma em "*array*" circular para a variável robô "*status*", o que promove o controlo do tamanho máximo e possíveis usos desnecessários ou descontrolados de memória.

A definição da estrutura do robô "*status*", proporciona, pela sua forma e divisão, a criação de um "*log*", ou seja, um histórico sobre o estado de todas as variáveis do sistema. Esta função de "*log*", encontra-se no robô de limpeza implementada de duas formas: uma de modo "*online*", devido a forma em "*array*" circular da estrutura e do uso de variáveis globais que guardam a posição no "*array*" do ultimo estado processado do sistema, podendo-se assim analisar o estado anterior de qualquer uma das variáveis de qualquer uma das sub-estruturas, e também para um de um modo "*offline*" já que no fim de cada ciclo de controlo é escrito num ficheiro e de forma estruturada toda a informação continua nas últimas posições de cada umas das sub-estruturas.

5.6 Auto-localização por Filtro de Partículas

5.6.1 Simulação

Um dos pontos de interesse é analisar como se comporta o filtro de partículas numa simulação antes da implementação deste no robô, semelhante ao que irá ser realizado nos corredores do Departamento de Electrotécnica e de Computadores (figura 5.18).

Supondo que o robô de cor azul encontra-se inicialmente com a "*pose*" real definida por $X=[1,9,0]^T$. Este robô irá deslocar-se ao longo do corredor, tendo apenas como informação sensorial seis sonares que ladeiam a estrutura do robô (quatro laterais e dois frontais) que fornecem a distância entre o sonar e o obstáculo mais próximo em linha recta e a informação da hodometria.

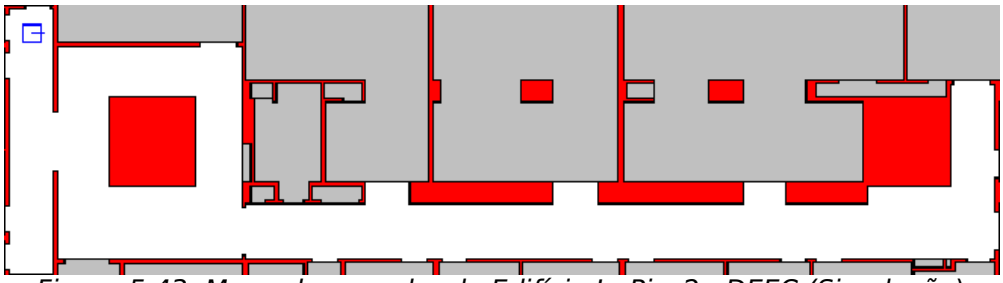
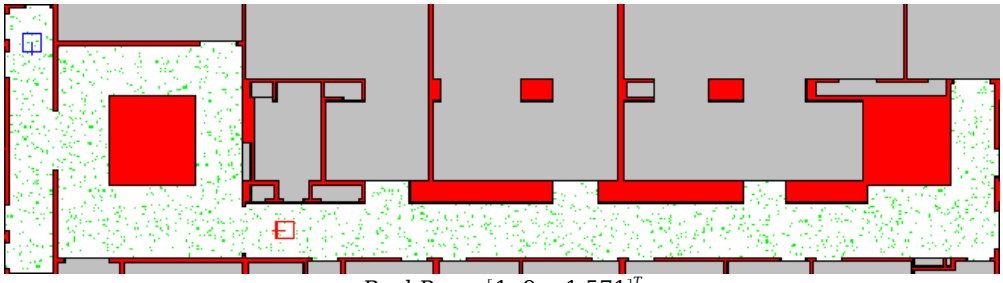


Figura 5.43: Mapa do corredor do Edifício I - Piso2 - DEEC (Simulação)

Nesta simulação o filtro de partículas irá criar cerca de 1200 partículas que irão ser espalhadas pelo mundo, neste caso o corredor. Em todas as iterações desta simulação o tempo de processamento computacional tem sido aproximadamente de 5 segundos, justificando-se pela plataforma utilizada nesta simulação, quer pelo número de partículas introduzidas nesta simulação.

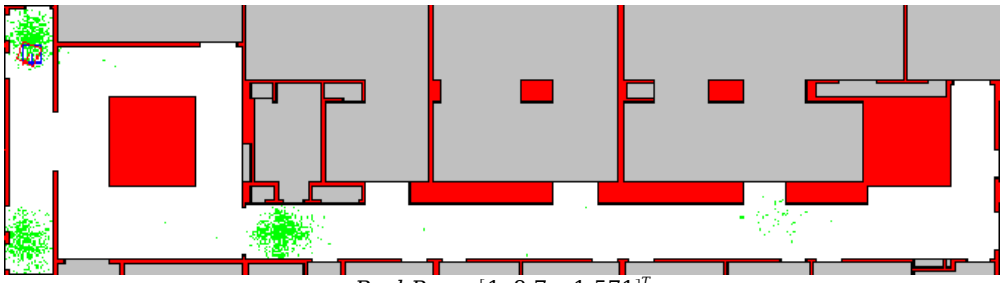
O método utilizado para a estimação da posição do robô, será efectuado pelo método da melhor partícula, devido a ser um método simples e como pouco peso computacional, sendo este um dos factores primordiais na escolha deste método. As "*poses*" são definidas em sempre definidas em unidades do sistema internacional.



$Real\ Pose = [1,\ 9,\ -1.571]^T$
$Estimation\ Pose = [9.3,\ 3.2,\ 3.137]^T$

Figura 5.44: Simulação do filtro de partículas - Iteração 1

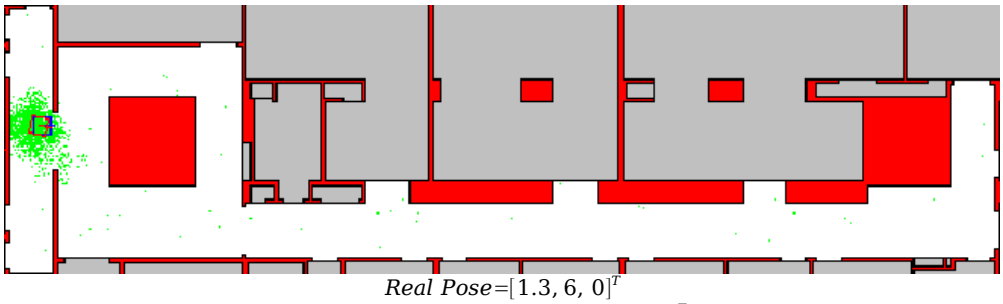
Na figura 5.44, é mostrada a primeira iteração do filtro de partículas, com a respectiva população de partículas assinaladas a cor verde. Devido à limitação inicial de 1200 partículas espalhadas aleatoriamente pelo mundo e à informação sensorial do robô (sonares), a probabilidade condicionada entre cada partícula com a informação sensorial originou uma função densidade de probabilidade mais acentuada em relação ao resto do mundo na posição assinalada pelo robô a cor vermelha ("*pose*" estimada), dado que estamos a usar um método de estimação simples que estima a posição baseando-se na função probabilidade mais elevada, por outras palavras, a partícula com a probabilidade maior da população existente.



$Real\ Pose = [1, 8.7, -1.571]^T$
$Estimation\ Pose = [1.1, 8.7, -1.598]^T$

Figura 5.45: Simulação do filtro de partículas - Iteração 2

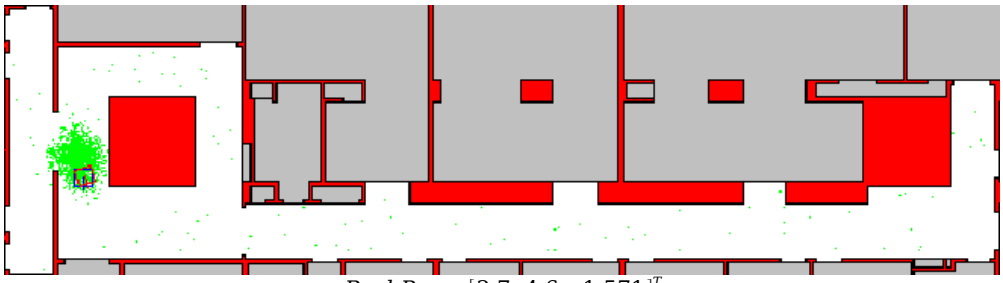
Observamos agora o efeito das observações ao colaboração com a reamostragem, que possibilitou a eliminação das partículas que não correspondiam aos dados adquiridos pelas observações e o aparecimento de novas partículas nos pontos da função densidade de probabilidade com uma probabilidade elevada, conseguindo assim a criação de aglomerados de partículas em áreas com maior probabilidade, sendo que uma dessas áreas existe uma partícula que possui um grau de confiança maior que as restantes, através do método de estimação aplicada esta será considerada a posição estimada, representada pelo robô de cor vermelha, enquanto o robô a cor azul simula o robô real. No entanto existe outras dois focos onde a probabilidade é elevada, isto deve-se às simetrias arquitectónicas do edifício, que possibilitam a hipótese de o robô se localizar nesses pontos, face às mesmas condições de observação obtidas.



$Real\ Pose = [1.3, 6, 0]^T$
$Estimation\ Pose = [1.2, 6, -0.137]^T$

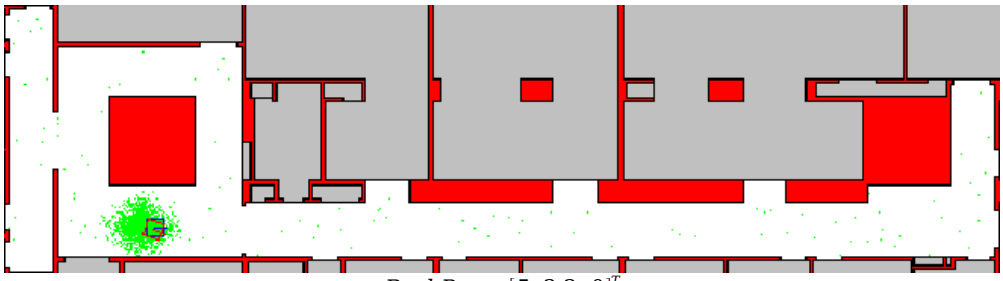
Figura 5.46: Simulação do filtro de partículas - Iteração 10

À medida que o robô vai percorrendo as instalações e o algoritmo do filtro de partículas vai sendo recursivamente aplicado as partículas serão sujeitas consecutivamente restringidas a uma área devido às observações adquiridas. No entanto esse facto não implica que no caso de situações em que as características físicas do mundo sejam semelhantes, não se crie a possibilidade do robô estar noutra "*pose*", o que já nos leva a pensar no caso da possibilidade de “rapto” do robô, esta possibilidade deve-se à possibilidade de deslizamento por exemplo de uma roda afectando assim a hodometria, que fornecerá ao algoritmo uma falsa previsão, porém como o algoritmo do filtro de partículas é um algoritmo recursivo, facilmente adquire confiança na partícula que corresponde com maior grau de confiança às observações adquiridas.



$Real\ Pose = [2.7,\ 4.6, -1.571]^T$
$Estimation\ Pose = [2.7,\ 4.6, -1.500]^T$

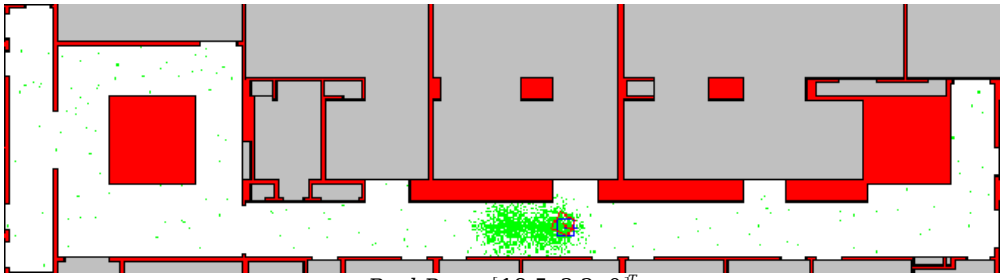
Figura 5.47: Simulação do filtro de partículas - Iteração 20



$Real\ Pose = [5,\ 3.2,\ 0]^T$
$Estimation\ Pose = [5,\ 3.2, -0.365]^T$

Figura 5.48: Simulação do filtro de partículas - Iteração 30

Como poderemos constatar ao longo destas iterações, a posição estimada, mesmo que por um algoritmo simples, como o método da melhor partícula é bastante próxima da real. Não obstante, existe partículas válidas e espalhadas pelo mapa, embora com pesos menores. Permitindo sempre no caso de "rapto" do robô a recuperação com facilidade da nova posição deste.



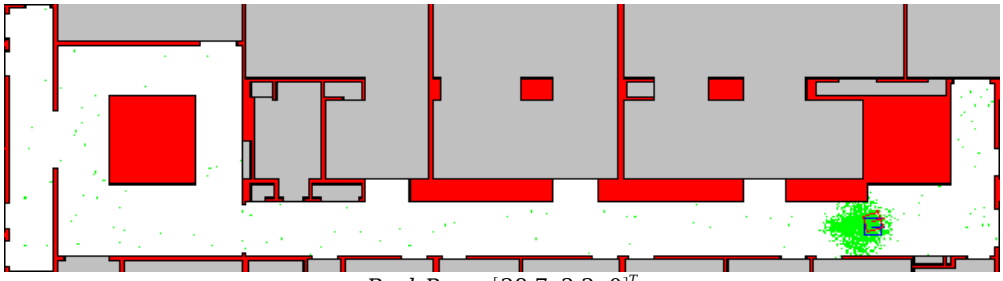
$Real\ Pose = [18.5,\ 3.2,\ 0]^T$
$Estimation\ Pose = [18.5,\ 3.2, -3.431]^T$

Figura 5.49: Simulação do filtro de partículas - Iteração 70

No caso de um corredor comprido com poucas características físicas de relevo que permitam ajudar na localização, o algoritmo, continua a executar a sua função porém com um grau de confiança menor, notando-se pelo arrasto das partículas atrás da posição estimada (figura 5.49).

Na sequência do raciocino anterior, a figura 5.50 indica-nos que a informação sensorial permitiu através das características do mundo melhorar a estimação da posição do robô, reflectindo-se no menor

arrasto das partículas e numa estimação mais próxima da real.



$Real\ Pose = [28.7, 3.2, 0]^T$
$Estimation\ Pose = [28.7, 3.4.7, .137]^T$

Figura 5.50: Simulação do filtro de partículas - Iteração 90

Conclusão

O Filtro de Partículas é um método de auto-localização, que permite desconhecer o ponto inicial do robô, no entanto é necessário ter um conhecimento prévio do mundo onde o robô irá deslocar.

O uso de sensores como sonares e medidores de distâncias por infra-vermelhos torna-se num ponto fraco em mundos com poucas características físicas de relevo, como o ambiente em que o "*CleanRob*" irá operar, sendo portanto necessário o uso de outras características que permitam uma melhor estimação, nomeadamente marcadores distribuídos pelo ambiente.

Permite a realizar a fusão de diversos sensores de uma forma simples, evitando reestruturações no algoritmo.

5.6.2 Implementação

Embora o conceito seja relativamente simples, a sua implementação não o é. Dado que o Filtro de Partículas como sistema de fusão de informação necessita de conhecer o meio onde o robô irá operar é necessário criar uma representação do mundo.

A representação do mundo poderá ser realizada de diversas formas, contudo de modo a manter um nível de processamento considerado baixo leva a que se modelize o mapa num formato vectorial. Este tipo de formato de mapa torna-se rapidamente em algo complexo e com um custo de desenvolvimento extremamente elevado. Face a extrema complexidade, desenvolvi uma ferramenta que possibilitasse o desenvolvimento e/ou modificação do mapa de uma forma simples e rápida, sendo à frente explicada em pormenor.

A parte mais complexa do filtro de partículas é nomeadamente a observação, pelo que é necessário criar funções que possibilitem simular para cada partícula uma representação virtual do sensor que se pretende integrar na fusão de informação. No caso de um medidor de

distâncias por infra-vermelhos, é necessário conhecer previamente a sua posição e orientação em relação ao centro do robô, para quando da integração de uma informação do medidor de distâncias por infra-vermelhos real, consiga-se simular o comportamento num medidor de distâncias por infra-vermelhos homologo virtual associado a cada uma das partículas. Por outras palavras o medidor de distâncias por infra-vermelhos virtual terá que medir a distância da localização virtual do mesmo a uma parede que esteja mais próxima e que tenha a mesma orientação do medidor de distâncias por infra-vermelhos (foi implementada a técnica de atribuição de dimensões à partícula, dado que se possui as dimensões do robô e com essa informação consegue-se localizar a posição do sensor em relação à "*pose*" da partícula). Facilmente se compreende o facto pelo qual o peso computacional do filtro de partículas é elevado, sendo este um dos factores a ter em conta.

O cálculo da distância dada pelo medidor de distâncias por infra-vermelhos virtual é realizado através de inúmeros cálculos geométricos e entre as várias equações de rectas que definem o mapa e a equação da recta que define o medidor de distâncias por infra-vermelhos. Dado ao peso computacional envolvido, organizei as equações universais da recta numa árvore possibilitando um aumento de performance na pesquisa. Este método é realizado para os sensores medidores de distâncias por infra-vermelhos e para os sonares, contudo sendo estes últimos mais complexos, devido a fornecerem a informação do obstáculo mais próximo dentro numa abertura de ângulo aproximadamente de 110º. Face a este grau de complexidade optei por não o modelizar da forma mais realista, embora fosse relativamente fácil de o fazer, mas implicaria um custo computacional ainda mais elevado.

A modelização da câmara foi um processo simples, bastando para tal simular a posição da câmara na partícula, à semelhança dos medidores de distâncias por infra-vermelhos e dos sonares e através das equações de rotação de eixos e da posição conhecida dos códigos de barras permitiu a modelização bastante realista deste sensor.

5.6.3 Dados Experimentais

Fusão de Informação da câmara

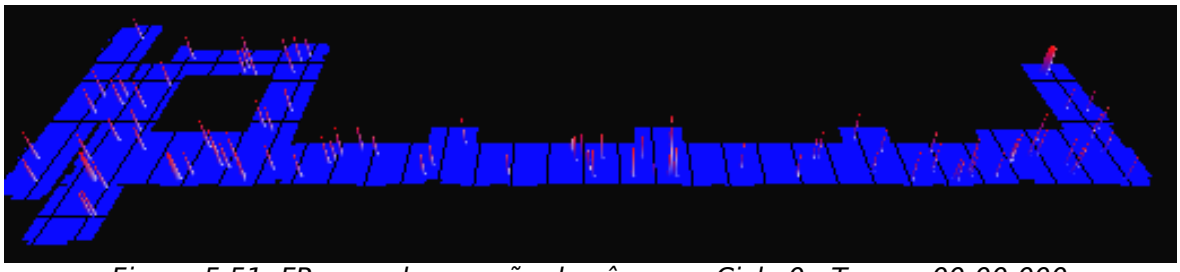


Figura 5.51: FP com observação da câmara - Ciclo 0 - Tempo 00:00:000

Inicialmente foi povoado o mundo com cerca de 500 partículas de uma forma aleatória, concentrando as outras 500 numa "*pose*" inicial de onde o robô irá iniciará (figura 5.51).

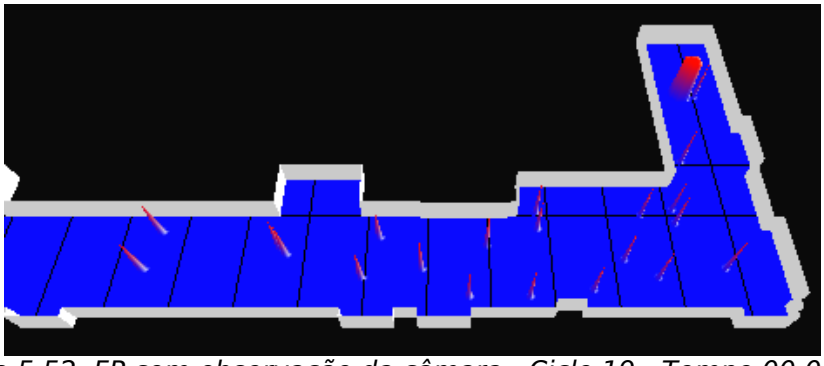


Figura 5.52: FP com observação da câmara - Ciclo 19 - Tempo 00:01:900

Através da hodometria é realizada a previsão, deslocando assim as partículas pelo mundo (figura 5.52). Dado que o robô não obtém num dado da sensorização, o Filtro de Partículas irá continuamente a realizar apenas o "*resampling*", a previsão, normalização e consequentemente a estimação.

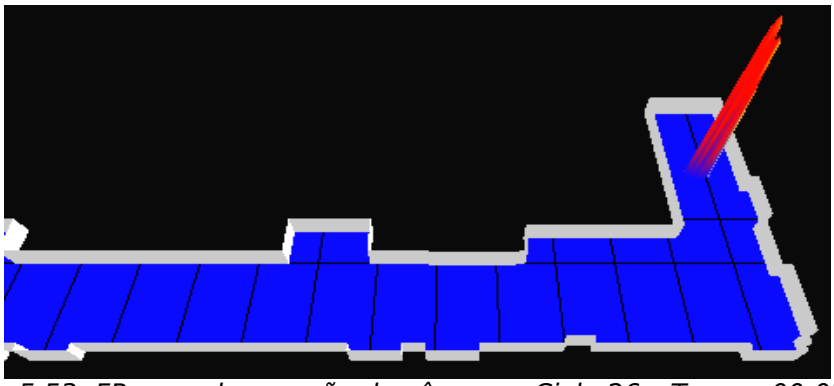


Figura 5.53: FP com observação da câmara – Ciclo 26 – Tempo 00:02:600

Na figura 5.53, o robô adquiriu informação proveniente da sensorização, ou seja da câmara, que detectou um marcador (código de barras) permitindo assim que fosse assim realizada a actualização. Dado que é conhecida a posição do marcador no mundo, a função gaussiana que é utilizada para o cálculo dos novos pesos com base na observação, encarrega-se de automaticamente de eliminar todas as partículas que não tenham informação coincidente com a obtida. ("*Frame*" do vídeo

realizado figura 7.1 do anexo 2).

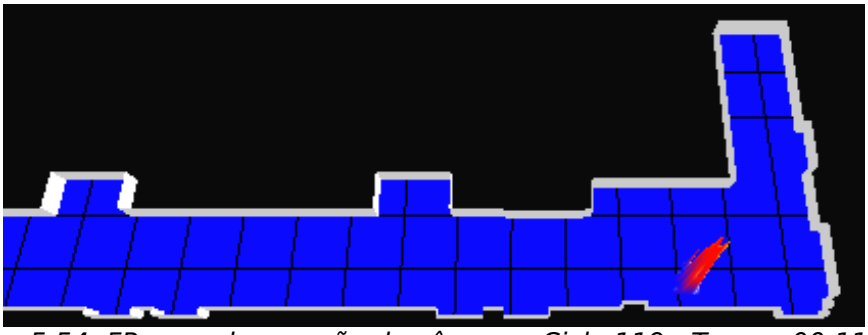


Figura 5.54: FP com observação da câmara - Ciclo 119 - Tempo 00:11:900

Podemos ver que quer na figura 5.54, quer na figura 5.55, o robô não recebeu nenhum informação por parte da câmara detectando marcadores, pelo que o filtro de partículas executa apenas a previsão, dado que a hodometria possui erros acumulativos, o filtro de partículas reage de modo a tentar prever todas as possíveis posições que o robô poderá encontrar-se. Este efeito pode-se ver nas figuras 5.54 e 5.55, constatando-se uma dispersão das partículas.

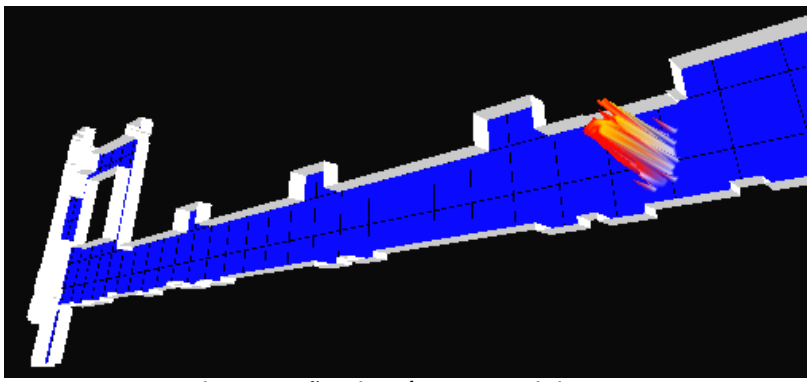


Figura 5.55: FP com observação da câmara - Ciclo 159 - Tempo 00:15:900

Na figura 5.56, observa-se novamente o efeito da observação provocada pela detecção de um código de barras, havendo a eliminação de todas as partículas que não possuíam informação coincidente com a fornecida pela câmara. Pode-se comparar o pico visto na figura 5.56, com o "*frame*" do vídeo existente na figura 7.2 do anexo 2.

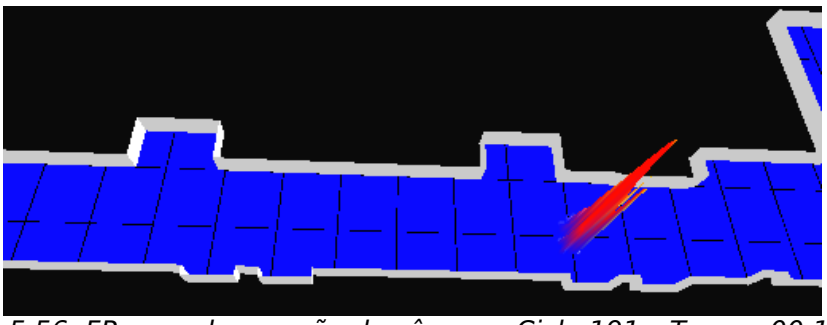


Figura 5.56: FP com observação da câmara– Ciclo 191 – Tempo 00:19:100

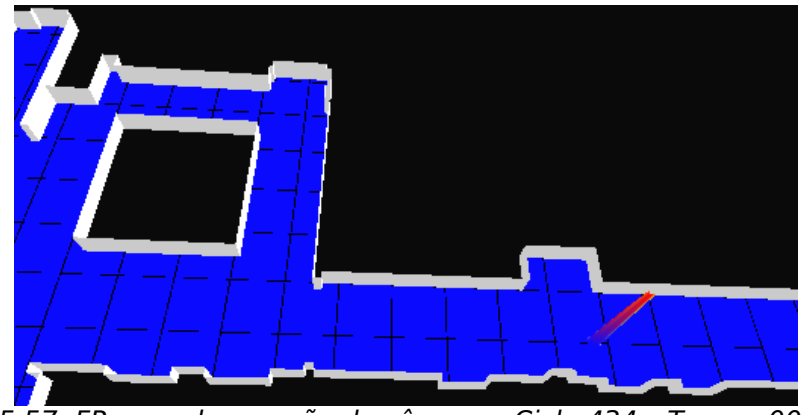


Figura 5.57: FP com observação da câmara – Ciclo 434 – Tempo 00:43:400

A missão de testes termina no ponto no ciclo de controlo 434, visualizado na figura 5.57, novamente é possível confirmar a posição do robô através do "*frame*" do vídeo existente na figura 7.3 do anexo 2. E recorrendo a outra ferramenta do "*Log Replay*" poderemos constatar a orientação do robô comparando a figura 5.58 com o mesmo "*frame*" do vídeo existente na figura 7.3 do anexo 2.

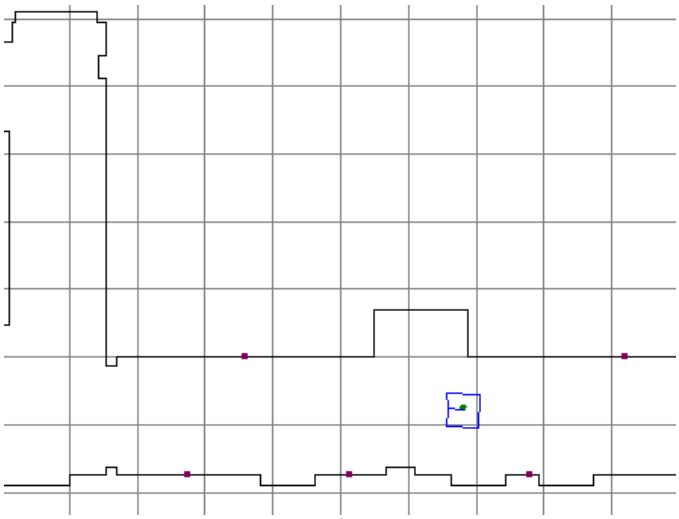


Figura 5.58: FP com observação da câmara – Ciclo 434 – Tempo 00:43:400

Fusão de Informação dos sonares e medidores de distâncias por infra-vermelhos

Infelizmente até ao momento da escrita desta tese não foi possível a realização de testes relativos à integração dos sonares e medidores de distâncias por infra-vermelhos na fusão de informação, contudo o algoritmo do Filtro de Partículas implementado no "*CleanRob*" já contempla o uso destes sensores, sendo que a integração dos mesmos são idênticas. Esta integração foi utilizada na simulação acima descrita, tendo-se observado pelas figuras que acompanham o subcapitulo o resultado da integração deste tipo de sensores.

Conjugação das diversas informações

Dado que não foi possível a realização de testes com sonares e medidores de distâncias por infra-vermelhos, não é possível neste

momento mostrar os efeitos da conjugação destes.

Contudo podemos ter a noção que quer os sonares, quer os medidores de distâncias por infra-vermelhos, auxiliariam o robô a manter concentrada a nuvem de partículas, dado que estes sensores teriam como efeito a correcção do ângulo do robô, num caso extremo em que não houvesse observações da câmara, o robô ao movimentar-se pelo corredor, dado ao "*path planning*" ter sido definido para o robô andar no meio do corredor, existiria uma nuvem de partículas à volta da recta que define o meio do corredor.

Havendo um número grande de partículas à volta do robô, contudo não deixando de aparecer partículas noutros pontos da recta.

5.6.4 Conclusões

Podemos concluir que a câmara é um elemento importante na convergência do método, mas não fundamental. O uso da câmara como sensor permitiu a convergência mais rápida e eficiente do método, contudo é possível realizar-se a auto-localização através de sensores de custo baixo, como por exemplo sonares e medidores de distâncias por infra-vermelhos.

O filtro de partículas mostrou ser um método de auto-localização fiável e robusto, permitindo a integração de diversos sensores de baixo custo. Apesar do seu elevado custo computacional, foi possível a sua implementação com recurso a mapas de formato vectorial e de algoritmos de pesquisa, reduzindo assim o tempo de execução.

Considero o objectivo da implementação do Filtro de Partículas no "*CleanRob*" concluída com sucesso, no entanto será vantajoso a inclusão dos restantes sensores e a adição de outros na fusão de informação analisando o comportamento deste.

5.7 "Map Creator"

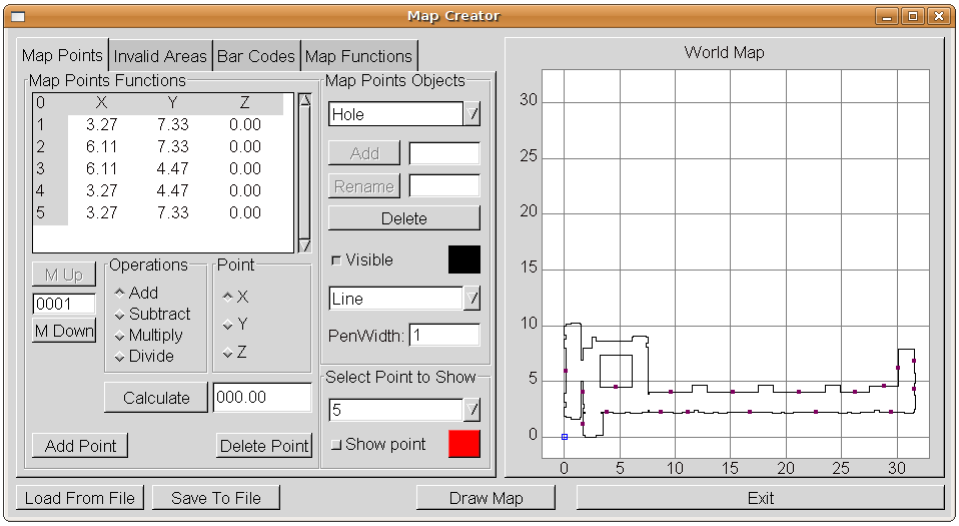


Figura 5.59: Interface da aplicação "Map Creator" – Janela Principal

5.7.1 Introdução

Face à complexidade de manuseamento para a criação/modificação de mapas provenientes de ficheiros "*CAD*"[9] das instalações da FEUP, desenvolveu-se uma ferramenta de edição de mapas que permitisse intuitivamente e facilidade na criação/modificação dos mesmos. Possibilitando assim uma maior versatilidade e redução de tempo no desenvolvimento de mapas relativos aos diferentes cenários de operações da plataforma robótica, com recurso a definição de diferentes conjuntos de objectos (paredes) e à definição de zonas inválidas (zonas onde é fisicamente impossível o robô estar posicionado).

Permite também a adição/modificação de marcadores (códigos de barras) como complemento à localização do robô face ao meio.

5.7.2 Implementação

O "*Map Creator*" foi uma aplicação desenvolvida com a potencialidade de criar/alterar mapas sem que haja com isso necessidade de alterar os programas principais do robô para tal, com um interface gráfico simples e eficiente. Foi pensado e desenvolvido para ser o máximo genérico permitindo assim vários conjuntos de pontos independentes, à semelhança de camadas, podendo haver por exemplo um conjunto destinado às paredes principais que definem o mapa, um outro conjunto para as paredes que definem portas, obstáculos, ou paredes interiores.

Possui também a definição de zonas inválidas, zonas essas que

9 Computer-Aided Design

caracterizam áreas onde o robô ou qualquer outro equipamento que irá usar este tipo de mapa seja fisicamente impossível de estar, como por exemplo o robô localizar-se dentro de paredes ou numa abertura no piso

Possuiu a potencialidade da definição de localização de marcadores (códigos de barras), para isso definindo-se a sua localização e orientação bem como o seu identificador.

5.7.3 Conclusões

Este tipo de programa é bastante útil, devido às suas potencialidades quer a nível de rapidez de desenvolvimento/alteração de paredes, zonas inválidas ou até no posicionamento de marcadores, como sob o ponto de vista gráfico, ou seja, à medida que se insere os pontos relativos à intercepção de paredes, consegue-se ter uma representação no momento do seu efeito na globalidade dos pontos já definidos, permitindo assim ao utilizador realizar no momento acções correctivas.

Dada à estrutura simples e organizada dos pontos definidos cria um ficheiro de mapa, maximizando a integração deste não só neste projecto, como em outros semelhantes que requeiram um mapa com base na representação física do ambiente onde o equipamento actuará.

5.8 "Log Replay"

5.8.1 Introdução

Face ao elevado volume de informação adquirida e à dificuldade de interpretação da mesma simultâneamente ao longo de uma missão, torna-se indispensável uma ferramenta que permita o "*debug*" posterior à mesma.

Para tal foi concebida e desenvolvida uma aplicação que permite o uso da informação adquirida e armazenada num "*Log*" no decorrer de uma missão do robô. Esta informação poderá ser analisada em "*offline*", podendo facilmente manuseada, possibilitando verificação de erros ou falhas por parte do "*software*" ou "*hardware*".

Permite a visualização da missão em mapas a 2D e 3D aliada a uma análise dos dados do mesmo ciclo de controlo. Podendo ser analisada passo a passo ou numa sequência contínua de tempo variável, permitindo assim uma análise mais cuidada ou simplesmente uma representação generalista da missão.

5.8.2 Implementação

O "Log Replay" é uma aplicação desenvolvida para permitir a análise de dados após uma missão. Dados esses que são escritos em ficheiro simultâneamente que é executada a missão. Este tipo de análise permitirá investigar todos os aspectos intervenientes no desempenho do robô e abrange aspectos tais como factores de calibração, correntes instantâneas dos motores, informação sensorial, fusão de informação até às tarefas desempenhadas.

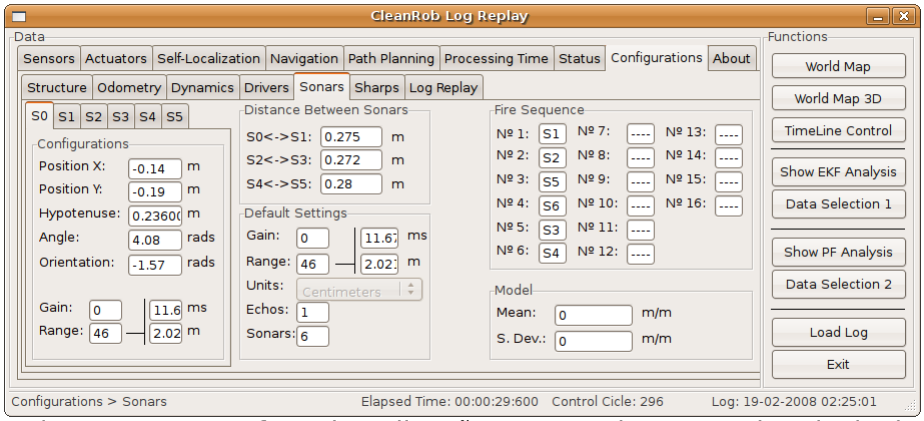


Figura 5.60: Interface da aplicação "Log Replay" – Janela Principal

Outra potencialidade deste tipo de aplicação é conseguir realizar uma análise pormenorizada passo a passo de cada ciclo de controlo e visualizar as transições/variações entre estas. Com recurso a gráficos e mapas é possível utilizar reutilizar esta aplicação com o intuito produzir relatórios pormenorizados sobre factores vitais na missão (figura 5.61).

Figura 5.61: Interface da aplicação "Log Replay" - Time Line Control

Foi implementado com base na aplicação do programa de controlo do robô, isto com o intuito de maximizar a compatibilidade, embora tenha sido criada uma camada superior completamente nova que permitisse providenciar ao utilizador a potencialidade de se deslocar no tempo em relação à duração da missão, bem como interfaces gráficas melhoradas incluindo visualização 3D e gráficos adequados a uma análise mais cuidada (figura 5.62 e figura 5.63).

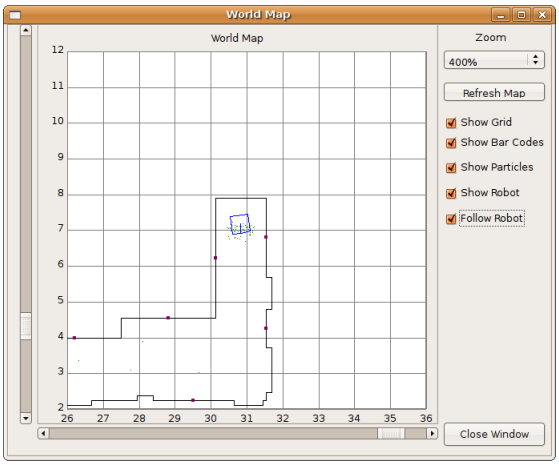


Figura 5.62: Interface da aplicação "Log Replay" - "World Map"

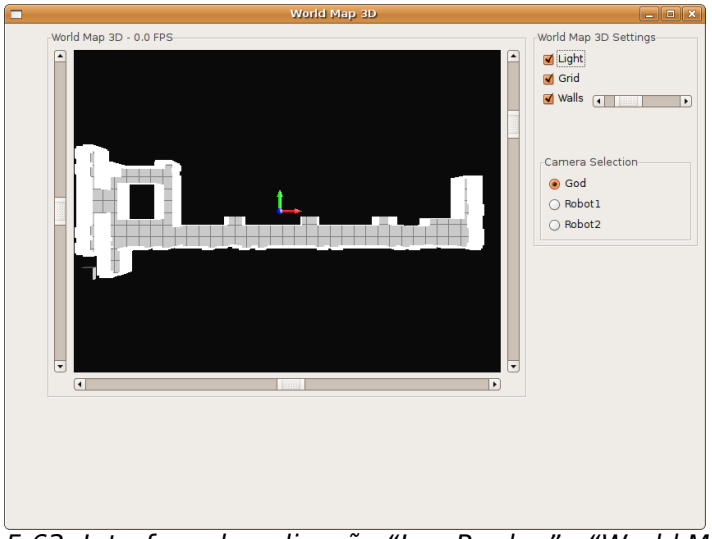


Figura 5.63: Interface da aplicação "Log Replay" - "World Map3D"

5.8.3 Conclusões

Esta aplicação é inovadora, na medida em que no grupo actual de robótica onde este projecto inclui-se não existe actualmente nenhuma ferramenta que possibilite este tipo de análise.

Este tipo de ferramenta tem demonstrado ser de elevada fiabilidade e versatilidade tornando-se indispensável no "*debug*" de erros e falhas da plataforma robótica, reduzindo de forma considerável a detecção de erros, bem como produzindo análises de dados de uma forma rápida e eficiente que possibilitaram o desenvolvimento deste documento.

Devido à sua estrutura simples e organizada permite expansibilidade o que torna este programa numa ferramenta com elevada relevância no desenvolvimento de futuras modificações do "*software*" e "*hardware*" desta e de outras plataformas robóticas.

Capítulo 6

Conclusões

6.1 Conclusão e Trabalho Futuro

Face aos testes realizados, conclui-se que a plataforma robótica encontra-se agora estável, podendo executar a tarefa para a qual foi desenhado. Estando diversas camadas implementadas de momento, o próximo passo a ser desenvolvido deverá ser a implementação de algoritmos de "*path planning*", suficientemente robustos para que esta plataforma robótica autónoma consiga tomar planear rotas num meio dinâmico como é o caso.

É também vantajoso a inclusão de camadas de inteligência artificial que permitam a cooperação com outros robôs autônomos, com o intuito de realizarem melhor o objectivo de limpeza, podendo haver cooperação com outros robôs que não dedicados a este tipo específico de tarefas, tal como robôs vigilantes ou simplesmente uma interacção com o elevador, possibilitando que o "*CleanRob*" expandisse a sua área de acção.

Capítulo 7

Referências Bibliográficas

[1] Vijay Kumar, George Bekey, Yuan Zheng, "*International Study of Robotics Research*", 05-IndustrialPersonal-Kumar-eBook.pdf (*2005*) http://www.wtec.org/robotics/workshop/PDF/

[2] Armando Sousa, Paulo Costa, António Moreira, "*Sistema de localizacao de robôs móveis baseado em EKF*", (*2003*) http://paginas.fe.up.pt/~asousa/wiki/

[3] Samuel Janela, Ing. J. Wendel, "*Self Localization of Mobile Robot using range measurement*", (*2006*)

[4] Ioannis M. Rekleitis, "*A Particle Filter Tutorial for Mobile Robot Localization*", (*2005*) http://www.cim.mcgill.ca/~yiannis/publications/

[5] J. Borenstein, H. R. Everett, and L. Feng, "*Navigation Mobile Robots: Systems and Techniques*", (*1996*) 1-56881-058-X

[6] J. Borenstein and Liquiang Feng, "*Measurement and correction of systematic odometry errors in mobile robots*", (*1996*)

[7] J. Neyman, "*Statistics*", (*1954*)

[8] Fernando Pinto, "*Aplicação do Filtro de Kalman na Auto-Localização de um Robô Autónomo*", (*2007*) www.fe.up.pt/~ee01189

Anexo 1

Dados experimentais do modelo da Hodometria

Dados obtidos pelo robô				Dados reais			
ΔX	**ΔY**	**ΔRho**	**ΔTheta**	**ΔX**	**ΔY**	**ΔRho**	**ΔTheta**
1,1015	0,0322	1,1020	0,0292	1,1170	0,0350	1,1177	0,0358
1,0970	0,0206	1,0972	0,0188	1,1140	0,0200	1,1142	0,0180
1,1096	0,0217	1,1098	0,0196	1,1260	0,0200	1,1262	0,0178
1,0974	0,0268	1,0977	0,0244	1,1160	0,0290	1,1164	0,0260
1,0992	0,0229	1,0994	0,0208	1,1210	0,0200	1,1212	0,0178
1,0995	0,0227	1,0997	0,0206	1,1270	0,0250	1,1173	0,0224
1,0956	0,0226	1,0958	0,0206	1,1160	0,0170	1,1161	0,0152
1,1142	0,0228	1,1144	0,0205	1,1340	0,0220	1,1341	0,0106
1,0968	0,0224	1,0970	0,0204	1,1190	0,0170	1,1191	0,0152
1,0998	0,0219	1,1000	0,0199	1,1210	0,0210	1,1212	0,0187
1,1100	0,0170	1,1101	0,0153	1,1320	0,0120	1,1321	0,0106
1,1055	0,0242	1,1058	0,0219	1,1270	0,0280	1,1273	0,0248
1,1140	0,0160	1,1141	0,0144	1,1400	0,0180	1,1401	0,0158
1,1106	0,0216	1,1108	0,0194	1,1330	0,0270	1,1333	0,0238
1,0997	0,0226	1,0999	0,0205	1,1260	0,0290	1,1264	0,0257
1,1036	0,0188	1,1038	0,0170	1,1180	0,0200	1,1182	0,0179
1,1160	0,0196	1,1162	0,0176	1,1360	0,0170	1,1361	0,0150
1,1030	0,0191	1,1032	0,0173	1,1300	0,0180	1,1301	0,0159
1,1109	0,0204	1,1111	0,0184	1,1370	0,0170	1,1371	0,0150
1,1073	0,0187	1,1075	0,0169	1,1360	0,0170	1,1361	0,0150

Tabela 23: Dados experimentais para a formulação do modelo da hodometria relativos à translação e deslizamento

Erros de Translação		
Erro de X	**Erro de Y**	**Erro de Rho**
0,0155	0,0078	0,0157
0,0170	-0,0006	0,0170
0,0164	-0,0017	0,0164
0,0186	0,0022	0,0186

Erros de Translação		
Erro de X	**Erro de Y**	**Erro de Rho**
0,0218	-0,0029	0,0217
0,0175	0,0023	0,0175
0,0204	-0,0056	0,0203
0,0184	-0,0108	0,0196
0,0222	-0,0054	0,0221
0,0212	-0,0009	0,0212
0,0220	-0,0050	0,0219
0,0215	0,0038	0,0216
0,0260	0,0020	0,0260
0,0224	0,0054	0,0225
0,0263	0,0064	0,0264
0,0144	0,0012	0,0144
0,0200	-0,0026	0,0200
0,0270	-0,0011	0,0270
0,0261	-0,0034	0,0260
0,0287	-0,0017	0,0287

Tabela 24: Erros de Translação

Erros de Deslizamento
Erro de Theta
0,0066
-0,0008
-0,0018
0,0016
-0,0030
0,0017
-0,0054
-0,0099
-0,0052
-0,0012
-0,0047
0,0030
0,0014
0,0044
0,0052
0,0009
-0,0026
-0,0014
-0,0034
-0,0019

Tabela 25: Erros de Deslizamento

Deslocamento do centro do robô		Dados obtidos pelo robô	Dados reais
▲X	**▲Y**	**▲Theta**	**▲Theta**
-0,0081	0,0139	6,6350	6,6328
-0,0106	0,0121	6,6359	6,6347

Deslocamento do centro do robô		Dados obtidos pelo robô	Dados reais
ΔX	**ΔY**	**ΔTheta**	**ΔTheta**
-0,0080	0,0110	6,6395	6,6286
-0,0122	0,0127	6,6148	6,6117
-0,0138	0,0162	6,6753	6,6642
-0,0106	0,0157	6,6174	6,6065
-0,0085	0,0130	6,6433	6,6344
-0,0087	0,0130	6,6228	6,6135
-0,0083	0,0142	6,6433	6,6362
-0,0076	0,0103	6,6481	6,6406
-0,0014	-0,0138	6,6934	6,7024
-0,0042	-0,0117	6,6887	6,6680
-0,0021	-0,0095	6,6565	6,6556
-0,0025	-0,0112	6,7129	6,6983
0,0017	-0,0075	6,6348	6,6264
-0,0017	-0,0109	6,7065	6,6983
-0,0010	-0,0118	6,6364	6,6265
-0,0008	-0,0122	6,6587	6,6474
-0,0008	-0,0102	6,6221	6,6052
-0,0001	-0,0109	6,6469	6,6285

Tabela 26: Dados experimentais para a formulação do modelo da hodometria relativos à rotação

Anexo 2

Fotografias do teste do Filtro de Partículas com a observação da câmara

Figura 7.1: "Frame" do teste do PF com observação da câmara

Figura 7.2: "Frame" do teste do PF com observação da câmara

Figura 7.3: "Frame" do teste do PF com observação da câmara