Faculdade de Engenharia da Universidade do Porto

Melhoramento do Desempenho do Robot de Serviço de Limpeza

Comparação de desempenho de Filtros de Kalman e de Filtros de Partículas para Auto-Localização

João Manuel Ferreira Martins

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Automação

Orientador: Prof. Dr. Armando Sousa

Junho de 2008

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2007/2008

A Dissertação intitulada

"**Melhoramento do Desempenho do Robô de Serviço de Limpeza**"

foi aprovada em provas realizadas em 16/Julho/2008

o júri

Presidente Professor Doutor Paulo José Lopes Machado Portugal
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Rui Paulo Pinto da Rocha
Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Professor Doutor Armando Jorge Miranda de Sousa
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto

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

Autor - João Manuel Ferreira Martins

Faculdade de Engenharia da Universidade do Porto

Agradecimentos

Agradeço

À minha mãe por me apoiar em tudo o que faço.

Ao meu maior crítico, aquele que sempre me incutiu o desejo de fazer melhor, o meu pai.

À Marta por tudo o que aguenta e por todo o tempo perdido.

Ao professor Armando por acreditar que era possível.

Ao Pedro e ao Gustavo por me confiarem a sua obra.

Muito obrigado

"Para ser grande, sê inteiro,
nada teu exagera ou exclui.
Sê todo em cada coisa.
Põe quanto és no mínimo que fazes.
Assim em cada lago a Lua toda brilha,
porque alta vive."

(Fernando Pessoa)

Resumo

A autonomia é uma necessidade cada mais desejada em sistemas robóticos móveis. Um dos factores de libertação é a mobilidade. Associando autonomia e mobilidade evidencia-se naturalmente o problema da localização. Para o solucionar recorre-se ao uso de sensores que permitam criar uma percepção do ambiente. A melhor forma de o fazer é fundir os dados para para que juntos possam fornecer uma maior e melhor percepção do mundo.

Neste trabalho são implementados dois métodos de fusão de informação sensorial: o filtro de Kalman extendido e o filtro de partículas. No final comparam-se os seus resultados experimentais.

Esta comparação é muito interessante pois permite observar as reacções de cada filtro no mesmo ambiente, usando o mesmo robot e os mesmo sensores, tornando assim legítimo compara-los.

No filtro de Kalman extendido usam-se dois métodos de localização bastante recentes e inovadores: localização por linhas usando sensores de posição ópticos e localização por códigos de barras, que permite obter uma localização completa. Este último método, permite como foi comprovado experimentalmente, resolver o problema da localização global.

No filtro de partículas são verificadas, na prática, algumas das suas características mais interessantes, tais como a capacidade de ultrapassar problemas de posição global.

É também descrita a arquitectura criada, com realce para a sua modularidade e capacidade de armazenar os vários dados usados em cada instante. Esta arquitectura permite o funcionamento em paralelo de algoritmos com objectivos iguais.

Apresenta-se também o conceito de aquisição inteligente de sensores que consiste numa aplicação que através de fusão de sensores permite obter uma melhoria na qualidade das medidas.

São efectuados diversos ensaios experimentais acerca da localização global e do seguimento de posição. É também efectuada a medição da posição real para que se possa calcular o erro da estimativa.

Abstract

The autonomy is a need each time more desired in mobile robotic systems. One of the important factors for autonomy is the mobility. Associating autonomy and mobility naturally appears the problem of localization. To solve this problem are used sensors to create a perception of the environment. The best way to make it, is join all the data so that together they can give a better knowledge of the world.

In this work are used two sensorial information's fusion methods: Extended Kalman filter and particle filter. In the end the experimental results are compared. This comparison is very interesting because allows to observe the reaction of each filter in the same environment, using the same robot and the same sensors.

In the extended Kalman filter are used two localization methods very recent and innovating: localization using lines with position sensitive detectors and localization using barcodes, which allows obtaining a complete knowledge of the localization. This last method helps the EKF solve the problem of global localization .

In the particle filter were verified, in practice, some of its main characteristics, like the capability of solve the problems of global position .

Is also described the architecture made, with attention to the modularity and capability of store the multiple data used in each instant.

The concept of intelligent acquisition of sensors is also presented and consists on application that uses sensorial fusion to get better measures.

Experimental tests were made to obtain the results of the filters in global localization and position tracking.

It also made the measurement of the actual position so we can calculate and present the error of estimate.

Índice

Lista de Figuras

Lista de Tabelas

Abreviaturas

FEUP- Faculdade de Engenharia da Universidade do Porto
DEEC - Departamento de Engenharia Electrotécnica e de Computadores
ISR-P – Instituto de Sistemas e Robótica do pólo do Porto
KF - Filtro de Kalman
EKF – Filtro de Kalman Extendido
PF – Filtro de Partículas
BC – Códigos de Barras
ISA – Aquisição Inteligente de Sensores
UDP – *User Datagram Protocol*
IPC - *Inter-Process Communication*

Capítulo 1

Introdução

1.1 Motivação

Ao longo dos tempos, criaram-se máquinas e ferramentas que libertassem o Homem de trabalhos indesejados ou repetitivos. Tratam-se de automatismos, sistemas capazes de repetir a mesma tarefa inúmeras vezes.

Actualmente, estes sistemas, por si só, não correspondem às necessidades do mercado. Desejam-se conjuntos cada vez mais reconfiguráveis e adaptáveis a novos ambientes e missões. Aumentam as expectativas em torno de máquinas com poder de decisão, capazes de planear e executar múltiplas tarefas de forma independente.

Surgem, como face mais popular da resposta a estes anseios, os sistemas robóticos autónomos.

O conceito de autonomia confunde-se com o de liberdade, consistindo na qualidade de um indivíduo tomar as suas próprias decisões, com base na sua razão[1].

Essencialmente a questão da autonomia em sistemas robóticos, traduz-se na liberdade de tomar algumas decisões, de forma a planear e executar os seus objectivos com eficiência, precisão e segurança.

Idealmente sistemas autónomos serão completamente independentes, sem qualquer necessidade manutenção humana. Múltiplos conjuntos cooperarão em tarefas comuns e pertencerão a uma sociedade própria.

Uma das vertentes interessantes destes sistemas é a área dos robots de serviço. Têm aplicações muito abrangentes, sendo exemplo:

- funções indesejáveis, inacessíveis ou economicamente desvantajosas no uso de trabalhadores humanos;
- acções onde exista perigo significativo para humanos (no espaço, uso militar ou mesmo em ambientes hostis);
- uso humanitário (apoio a pessoas com limitações e tarefas em hospitais ou em salvamentos).

A mobilidade de um robot é uma questão relevante para o seu nível de autonomia. Um robot com a capacidade de se deslocar livremente num dado ambiente, pode realizar de forma eficiente muitas tarefas que de outra forma lhe estariam vedadas. A localização aparece então como essencial para a autonomia dos sistemas robóticos [2].

Auto-localização é assim um assunto crítico na robótica móvel. Se o robot não sabe onde está, não pode eficientemente planear movimentos, encontrar objectos ou completar objectivos [3].

O professor Ingemar Cox refere mesmo que a localização é “A propriedade mais

importante para fornecer a um robot móvel com capacidades autónomas" [4] [5].

Soluções eficazes e o mais genéricas possíveis para a localização, tornam-se assim cada vez mais apetecíveis.

1.2 Contexto

Esta dissertação insere-se na continuação do projecto CleanRob, robot de limpeza, nascido em 2004 e financiado pelo Departamento de Engenharia Electrotécnica e de Computadores (DEEC) e pelo Instituto de Sistemas e Robótica do pólo do Porto (ISR-P).

O CleanRob consiste num robot autónomo, com o objectivo de limpar/varrer os corredores do DEEC. Para tal tem afixado um apetrecho de limpeza semelhante a um aspirador, vendido em qualquer superfície comercial.

Este trabalho tem como objectivo principal, a resolução do problema da auto-localização do robot, para tal encaram-se duas abordagens distintas, uma utilizando filtro de Kalman extendido (EKF) e outra usando filtro de partículas (PF).

São apresentadas técnicas recentes como a localização por linhas e a localização por códigos de barras (BC).

É também apresentada uma arquitectura de aquisição inteligente de sensores utilizada e é feita a prova de conceito desta arquitectura.

1.3 Contribuições

Esta dissertação apresenta as seguintes contribuições de relevo:

- Aplicação de um filtro de Kalman extendido para auto-localização num sistema robótico móvel;
- Aplicação de um filtro de partículas para auto-localização num sistema robótico móvel;
- Criação de um sistema para evitar colisões;
- Comparação entre os resultados do filtro de Kalman e do filtro de partículas;
- Apresentação do conceito de aquisição inteligente de sensores (ISA);
- Exposição de diversos resultados experimentais referentes à localização.

De referir que este projecto foi alvo de acções de divulgação na comunicação social, tendo sido apresentado na revista semanal "Visão", nº 799, de 26 de Junho de 2008, no jornal diário "Jornal de Noticias" do dia 29 de Junho do mesmo ano, na RTPN no dia 10 de Julho e referenciado na página oficial da BBC.

Espera-se que até ao final da dissertação, este trabalho seja ainda mais publicitado ao nível da imprensa escrita e audiovisual.

Refere-se ainda que este projecto será submetido como artigo científico em conferências futuras.

Disponibilizam-se mais informações acerca do desenvolvimento e de objectivos futuros na página do projecto do robot de limpeza[6].

1.4 Estrutura

No 2º capítulo, enquadramento, retratam-se os principais paradigmas dos sistemas robóticos autónomos. Serve este capítulo para aproximar o leitor ao ambiente em que situa este trabalho e às dificuldades envolvidas. A necessidade da fusão de informação sensorial e a problemática da localização são abordadas.

Segue-se o capítulo, métodos estocásticos de fusão de informação, onde se apresentam as ferramentas teóricas de fusão de informação que serviram de base para este projecto. Apresenta-se a formulação teórica do filtro de Kalman e do filtro de partículas, bem como alguns pressupostos essenciais a estes métodos.

No 4º capítulo, descrevem-se os principais constituintes do robot, bem como as secções mais importantes do trabalho desenvolvido. Enunciam-se os diversos componentes já existentes, como os sensores e algumas das suas interfaces e expõem-se os melhoramentos efectuados, tais como o tipo de arquitectura, a aquisição inteligente de sensores e os métodos de localização usados.

Na validação experimental apresentam-se alguns dos dados obtidos relativos à auto-localização. De assinalar os resultados na localização global e o seguimento de posição.

Por último apresentam-se as conclusões gerais acerca do trabalho efectuado, bem como indicações para objectivos futuros deste projecto.

Capítulo 2

Enquadramento

2.1 Introdução

Neste capítulo apresentam-se, de forma ligeira, os principais paradigmas dos sistemas robóticos móveis, com realce para os que foram abordados neste trabalho. É um capítulo de enquadramento, que permite ao leitor situar-se nas linhas gerais deste projecto.

Inicialmente expõem-se algumas características dos sistemas autónomos e caminha-se depois para a descrição dos dois temas mais importantes discutidos neste projecto: a fusão sensorial e a localização.

2.2 Sistemas Robóticos Autónomos

Define-se um sistema robótico autónomo como sendo um sistema com alguma liberdade de actuação e autonomia de decisão. Constituem este sistema: a unidade de decisão, o conjunto de sensores e de actuadores e todos os periféricos necessários.

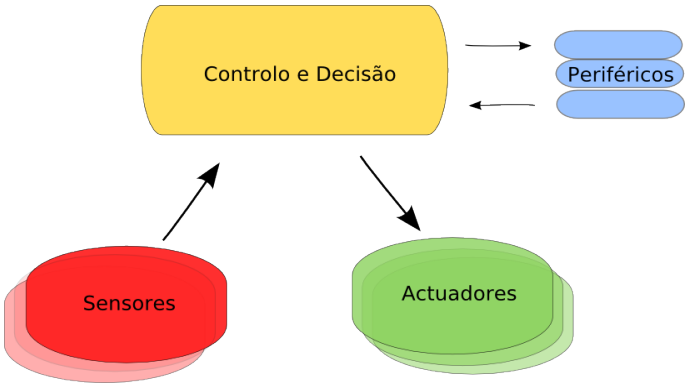


Figura 2.1 Estrutura genérica de um robot

Como é óbvio a construção de um robot envolve diversas áreas como a mecânica,

electrónica e a informática. É necessário conjugar diverso *hardware* e *software* diferente, de forma a garantir a comunicação entre os vários componentes. Uma boa estrutura entre as várias interfaces de comunicação e o sistema de decisão é a base para ter um bom tempo de reacção, e assim uma resposta útil em tempo real às mudanças do ambiente.

Uma das actividades dos sistemas robóticos móveis que mais necessita de respostas em tempo real é localização.

Antes de descrever este problema, é necessário explicar como um robot cria uma percepção do espaço que se encontra à sua volta.

Divide-se a interacção entre o robot e o ambiente que o rodeia, em dois conjuntos de dados:

- o grupo das medidas enviadas pelos sensores, que permitem criar uma percepção do mundo;
- e o grupo das acções de controlo que alteram o estado do robot.

Considera-se o estado como sendo o conjunto de parâmetros que se deseja conhecer e estimar. Neste caso o estado é a pose, que engloba a posição cartesiana e a orientação do robot.

É muito importante esta divisão, pois o primeiro grupo dá informação vital acerca do estado, o que leva a um maior conhecimento do robot. Enquanto que o segundo grupo, por sua vez, leva a uma modificação do estado, que terá ser modelizada, o que acarreta erros e incertezas em relação à nova posição [5].

Realça-se também que a grande maioria sensores que são usados nos robots capta apenas uma pequena informação acerca do estado, sendo portanto necessário juntar os vários conjuntos de medidas recebidos, de forma a conseguir obter um melhor conhecimento do ambiente. Surge assim a necessidade de fusão das várias informações recebidas.

"Fusão de Informação é um quadro formal de trabalho onde se emprega meios e ferramentas para a junção de dados de origens diversas com o objectivo de obter informação de qualidade superior. A definição exacta de qualidade superior depende da aplicação em causa."[2].

Apenas com estas breves definições, é possível concluir que para conhecer a posição de um robot, será necessário juntar os dois conjuntos de dados (acção e percepção). Com o grupo de acções, devidamente modelizado, obtém-se uma previsão da posição do robot no mundo. Com o grupo de sensores actualiza-se e confirma-se essa posição.

Seguem-se algumas definições e explicações acerca da fusão de informação dos sensores.

2.3 Fusão Sensorial

Do ponto de vista económico, passou a ser interessante substituir um sensor de grande

precisão mas de elevado custo, por vários sensores de baixo custo e por sua vez de menor precisão. Esta nova conjectura levou à criação de uma área denominada de fusão multi-sensorial de dados ou sensorização distribuída [7].

Uma possível definição é a seguinte: fusão multi-sensorial de dados é o processo de combinar as observações de vários sensores de forma a fornecer uma descrição robusta e completa do ambiente de interesse [8] [9].

A fusão sensorial é aplicada em várias áreas tais como: reconhecimento de objectos, localização, construção do mapa do ambiente, entre outras.

Motivação para o uso e popularidade da fusão de dados [8]:

- Imperfeição dos sensores (não linearidades, ruído);
- Avaria de sensores;
- Diversas limitações tecnológicas (não permitem que um único sensor capta toda a informação de um ambiente);
- Limitações físicas do modelo do sensor (alcance, fraca precisão, baixa resolução);
- Complexidade do ambiente (necessidade de medições indirectas de características);
- Problemas com sistemas de tempo real.

Existem vários tipos de classificações deste método dependendo, obviamente, do autor, segue-se uma destas divisões.

Segundo Boudjema e Forbes [10]:

- Fusão de sensores: diversos sensores medindo a mesma propriedade. Como exemplo, vários sensores de temperatura medem a temperatura de um objecto.
- Fusão de atributos: nesta situação, um número de sensores medem diferentes propriedades de um sistema, tal como o uso de uma balança e uma filtra métrica permitem determinar o índice de massa muscular de uma pessoa.
- Fusão de domínios: neste caso, vários sensores medem o mesmo atributo com diferentes alcances.
- Fusão ao longo do tempo: as medidas actuais são integradas ao histórico da informação disponível.

Existem três configurações básicas de sensores [11]:

- Sensores Complementares: são combinados de forma a fornecer uma percepção mais completa do fenómeno observado.
- Sensores Competitivos: cada sensor entrega uma medida independente da mesma propriedade. Esta configuração permite obter redundância, que por sua vez reduz os efeitos de incerteza e de medidas erradas.
- Sensores Cooperativos: são misturadas as informações dos vários sensores para medir propriedades impossíveis de atingir com um só sensor.

A grande maioria dos métodos de fusão de informação representa a realidade como sendo um conjunto de sistemas determinísticos somados com um conjunto de processos estocásticos. Desta forma procuram modelar os distúrbios e erros existentes nos modelos dos sistemas através do uso de probabilidades.

Algumas desvantagens do uso de sistemas inteiramente determinísticos são[12]:

- Nenhum modelo matemático é perfeito. Nem a própria realidade o é.
- A dinâmica do sistema não é controlada apenas pelas entradas, mas também por distúrbios e ruídos que nenhum modelo determinístico pode modelar.
- Nenhum sensor transmite a percepção perfeita e completa acerca de um sistema/estado. Por vezes são usados sensores com resoluções e precisões diferentes e até com diferentes métodos de funcionamento, sendo necessário fundir os vários dados para obter uma melhor estimativa. Os sensores não fazem medições completamente exactas, sendo elas também dependentes do ruído e da dinâmica dos próprios sensores.

Torna-se assim evidente o uso de processos estocásticos para modelar os ruídos e distúrbios que podem ocorrer no sistema.

Grande parte dos algoritmos de fusão de informação baseia-se no teorema de Bayes descrito no capítulo seguinte.

2.4 Localização

Para melhor compreensão da problemática da localização é necessário rever algumas definições:

- Seguimento ou rastreio da posição, denominado também por localização "de vizinhança";
- Localização Global;
- Rapto.

O rastreio da posição assume que a pose inicial é conhecida. Desta forma a localização fica confinada a uma área reduzida do ambiente, vizinhança do robot, sendo um problema local. A solução comum para este tipo de localização passa por utilizar a odometria. Adicionam-se à posição actual os deslocamentos medidos pelo sistema de odometria e espera-se assim obter uma pose próxima da posição real do robot. Como será demonstrado experimentalmente, esta solução, por si só, não é suficiente, visto o sistema de odometria acumular erros proporcionais ao deslocamento efectuado, e também porque nem todos os movimentos são possíveis de medir com a odometria. Um exemplo desses movimentos é o acto do robot deslizar.

A localização global parte do principio que a pose inicial do robot é desconhecida. O robot pode encontrar-se em qualquer zona do ambiente e tem pouco conhecimento do seu estado. Resolver a localização global é mais difícil do que o seguimento da posição, aliás engloba este problema. Em termos práticos, a localização global só é usada no arranque do sistema, pois após o robot ter conhecimento da sua pose passa a um seguimento de posição.

O rapto é uma variante da localização global, mais ainda mais difícil de resolver. Consiste na alteração da pose do robot, durante o seu normal funcionamento, sem que este possa

regista-la. A dificuldade acresce, pois enquanto na localização global o robot tem uma baixa confiança no seu estado e por sua vez uma grande incerteza na sua posição, no rapto o robot pode ter uma elevada confiança numa pose que está errada. Em situações reais, raramente este tipo de problema ocorre, mas este problema tem um valor elevado como teste para recuperações de falhas em algoritmos de localização global.

2.5 Conclusões

Neste capítulo apresentaram-se os dois temas abordados neste trabalho, a localização e a fusão sensorial.

Resumiram-se algumas das definições de cada um dos temas, sendo a localização divida em: rastreio de posição, localização global e rapto. A fusão sensorial foi seccionada em: sensores complementares, competitivos e cooperativos.

Explicou-se também a necessidade de fusão de informação em sistemas robóticos e a utilização de processos estocásticos nos métodos de estimação da posição do robot.

Capítulo 3

Métodos Estocásticos de Fusão de Informação

3.1 Introdução

Este capítulo aborda algumas ferramentas teóricas utilizadas neste trabalho.

A teoria das probabilidades permite modelar um dado fenómeno, incerto na sua natureza mas acerca do qual existe algum conhecimento [13]. Desta forma, usam-se processos probabilísticos para representar os ruídos, distúrbios e possíveis erros nos sistemas utilizados.

Apresentam-se os algoritmos utilizados na fusão de informação, sendo eles o filtro de Kalman, na sua versão para sistemas não lineares, e o filtro de partículas. Antes porem, expõem-se algumas das propriedades das variáveis aleatórias usadas nos processos estocásticos.

Descreve-se também o filtro de Bayes que serve de base aos dois algoritmos utilizados e é uma das principais referências na estimação de parâmetros.

3.2 Variáveis aleatórias

3.2.1 Fundamentos acerca de Probabilidades

Considerando U_Y como sendo o universo de todos os valores que uma dada variável *Y* pode assumir (*Y* representa o fenómeno a modelar). *Y* será então uma variável aleatória e se assumir uma variação temporal *Y(t)* designa-se por processo estocástico.

No caso de o processo estocástico assumir apenas valores discretos para a variável t, então é usual indexar o tempo à variável discreta:

$$Y(t) \rightarrow processo\ estocástico$$
$$t \in \mathbb{R}$$
$$Y(k) \rightarrow processo\ estocástico\ em\ tempo\ discreto$$
$$k \in \mathbb{Z}$$

É então necessário estabelecer uma lei de probabilidade que indique quão previsível é que

Y assuma um dado valor, o mesmo é dizer, que o fenómeno sob estudo tenha um dado desfecho. A probabilidade de acontecer um desfecho A é representada por $P(Y=A)$ ou P(A).

Uma dada lei de probabilidade deve ainda obedecer a:

1. $P(A)\geq 0 \; \forall A$
2. $P(U_Y)=1$
3. $P(A_1 \cup A_2 \cup \ldots)=P(A_1)+P(A_2)+\ldots$ para qualquer sequência finita ou não de eventos mutuamente exclusivos.

Definindo uma função densidade de probabilidade (fdp) como sendo a função que calcula para cada valor *x* a sua probabilidade, $f_y(x): \mathbb{R} \rightarrow \mathbb{R}$, esta fdp deve obedecer a:

1. $f_y(x)\geq 0 \; \forall x$
2. $\int_{-\infty}^{+\infty} f_y(x)\, dx \; = \; 1$

O cálculo da probabilidade de um evento A é feito através de:

$$P(A)=\int_A f_y(x)\, dx \tag{3.1}$$

Se a variável aleatória for discreta, então teremos que o universo U_Y é finito e assim deve-se verificar:

$$\sum_i P(Y=i)=1 \tag{3.2}$$

É também possível ter variáveis aleatórias vectoriais $Y=(y_1, \ldots, y_n)$. Teremos $f_y(X)$ com $X \in \mathbb{R}^n$ e será também:

$$\begin{array}{rccc} f_y(x): & \mathbb{R}^n & \rightarrow & \mathbb{R} \\ & (X) & \rightarrow & f_y(X) \end{array} \tag{3.3}$$

satisfazendo

1. $f_y(x)\geq 0 \; \forall X \in \mathbb{R}^n$
2. $\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\cdots\int_{-\infty}^{+\infty} f_y(x)\, dx_1 \cdots dx_n \; = \; 1$

Para obter P(A):

$$P(A)=\int_A \cdots \int f_y(X)\, dX \tag{3.4}$$

3.2.2 Atributos de Variáveis Aleatórias

Para simplificar o tratamento destes conceitos, é interessante utilizar restrições quanto às classes de distribuições em causa. Nesse sentido definem-se alguns atributos básicos das variáveis aleatórias.

A média:

$$\mu_y = E(Y) = \int_{-\infty}^{+\infty} x\, f_y(x)\, dx \tag{3.5}$$

A variância σ^2 e o desvio padrão σ :

$$\sigma_y^2 = VAR(Y) = E\left(\left(Y-\mu_y\right)^2\right) = E(Y^2) - (\mu_y)^2 \tag{3.6}$$

E a covariância entre as variáveis aleatórias Y e Z, com médias μ_y e μ_z :

$$cov(Y,Z) = E\left((Y-\mu_y)(Z-\mu_z)\right) = E(YZ) - \mu_y\mu_z \tag{3.7}$$

No caso multidimensional em que $Y=(y_1,\cdots,y_n)$ teremos então que $\mu_y = \left(E(y_1),\cdots,E(y_n)\right)$ e a matriz de covariância é a seguinte:

$$\sigma^2 = \begin{bmatrix} cov(y_1,y_1) & cov(y_1,y_2) & \cdots & cov(y_1,y_n) \\ cov(y_2,y_1) & cov(y_2,y_2) & \cdots & cov(y_2,y_n) \\ \vdots & \vdots & \ddots & \vdots \\ cov(y_n,y_1) & cov(y_1,y_2) & \cdots & cov(y_n,y_n) \end{bmatrix} \tag{3.8}$$

Note-se que os $VAR(y_i)$ encontram-se na diagonal da matriz.

Para um processo estocástico *Y(t)* os parâmetros média e (co)variância aparecem também em função do tempo.

Podem-se definir ainda as funções seguintes:

A correlação em instantes t_1 e t_2:

$$R_y(t_1,t_2) = E\left(Y(t_1)Y(t_2)^T\right) \tag{3.9}$$

e uma matriz de covariância entre variáveis aleatórias:

$$C_y(t_1,t_2) = cov\left(Y(t_1)Y(t_2)\right) \tag{3.10}$$

Como se verifica que:

$$C_y(t_1,t_2)=R(t_1,t_2)-\mu_y(t_1)\,\mu_y(t_2) \tag{3.11}$$

teremos então para um processo de média nula:

$$R_y(t_1,t_2)=C_y(t_1,t_2) \tag{3.12}$$

Para processos estacionários verifica-se que estas funções só dependem da diferença entre os instantes de tempo $\tau=t_2-t_1$ e assim é possível escrever $R_y(\tau)$ e $C_y(\tau)$.

3.2.3 Propriedades dos Atributos

O valor esperado de variáveis aleatórias *X* e *Y* tem as seguintes propriedades [14]:

$$\begin{aligned} &E(K)=K, \qquad K\in\mathbb{R} \\ &E(X+Y)=E(X)+E(Y) \\ &E(KX)=K\cdot E(X), \qquad K\in\mathbb{R} \end{aligned} \tag{3.13}$$

Se *X* e *Y* são variáveis aleatórias independentes então:

$$E(X\cdot Y)=E(X)\cdot E(Y) \tag{3.14}$$

A variância tem as seguintes propriedades:

$$\begin{aligned} &VAR(K)=0 \qquad K\in\mathbb{R} \\ &VAR(KX)=K^2\cdot VAR(X) \end{aligned} \tag{3.15}$$

Se *X* e *Y* são variáveis aleatórias independentes:

$$VAR(X+Y)=VAR(X)+VAR(Y) \tag{3.16}$$

3.2.4 Distribuições Gaussianas

A função densidade probabilidade de uma distribuição gaussiana tem a forma analítica:

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.17}$$

Onde μ é a sua média e σ é o seu desvio padrão.

Processos estocásticos que sigam uma distribuição de forma gaussiana são completamente

definidos pela sua média e covariância:

$$\mu_Y(t)=E(Y(t)) \quad \text{e} \quad C_y(t_1,t_2)=cov(Y(t_1),Y(t_2))$$

Estes processos gaussianos têm propriedades interessantes:

1. A combinação linear de processos gaussianos é ainda um processo gaussiano;
2. Muitos sinais reais são representáveis por processos gaussianos.

A primeira propriedade garante que se pode representar com exactidão e simplicidade combinações lineares de processos estocásticos.

A segunda propriedade é essencial para garantir a correcta modelização da realidade. Uma justificação da propriedade é dada pelo Teorema do Limite Central que refere o seguinte:

Sendo *Z(n)*, $n\in\mathbb{N}$, variáveis aleatórias independentes com variâncias $V(Z(n))=\sigma^2<\infty$ então a sequência $\frac{S(n)-E(S(n))}{\sqrt{V(S(n))}}$ onde $S(n)=\sum_{i=1}^{n} Z(i)$ converge para a distribuição gaussiana normalizada[1] quando $n\to+\infty$ desde que $0<\epsilon\le V(Z(n))\le C<\infty$, isto é, desde que a variância esteja limitada superior e inferiormente.

Pode-se assim dizer que o efeito de muitas perturbações, independentemente das distribuições individuais, se soma para formar uma distribuição gaussiana. Por estes motivos, a utilização de processos gaussianos é muito relevante para o estudo dos mais diversos fenómenos na actualidade.

3.3 Filtro de Bayes

O filtro de Bayes é o algoritmo mais genérico para estimar a confiança num determinado estado. Este método calcula a confiança, normalmente representada por uma função de densidade da probabilidade, através dos dados do estado, das medidas e das acções de controlo [5].

Este filtro necessita do conhecimento de três distribuições de probabilidade: da confiança inicial $p(x_0)$, da probabilidade das medidas, $p(z_t|x_t)$ e da probabilidade do modelo do sistema de transição do estado, $p(x_t|u_t,\ x_{t-1})$ [15].

Tem dois passos essenciais: a previsão e a actualização. Na primeira fase, usam-se as acções de controlo u_t e o estado x_t para obter a confiança à priori $\overline{bel}(x_t)$. Na segunda fase, usam-se as medidas y, para actualizar a confiança do estado, $bel(x_t)$.

1 Com média nula e variância (e desvio padrão) unitários

Previsão:

$$\overline{bel}(x_t) = \int p(x_t|u_t,\ x_{t-1})\ bel(x_{t-1})\ dx_{t-1} \tag{3.18}$$

Actualização:

$$bel(x_t) = \eta\ p(z_t|x_t)\ \overline{bel}(x_t) \tag{3.19}$$

Facilmente se observa que o produto na fase actualização raramente será uma probabilidade, sendo necessário normalizar através da variável η.

O filtro baseia-se no teorema de Bayes, que é usado em diversas aplicações como por exemplo nos filtros *anti-spam* dos *emails*. Diz o seguinte:

$$p(x|y) = \frac{p(y|x)\ p(x)}{p(y)} \tag{3.20}$$

Permite calcular a probabilidade *à posteriori* sabendo a probabilidade *à priori*.

Traduzindo para este caso específico, se $p(x)$ representar a confiança que se tem num estado, $p(y|x)$ representar a probabilidade *à priori*, sendo y o conjunto de medidas dos sensores, então a probabilidade $p(x|y)$, probabilidade *à posteriori*, representa a confiança no estado com as medidas já incorporadas [5].

O filtro de Bayes tem também outro requisito importante, supõe que o processo a estimar é uma cadeia de Markov.

De forma simples, chama-se cadeia de Markov a um processo temporal, quando o seu estado actual é completo, ou seja, é o melhor previsor do futuro. Por outras palavras, o estado actual engloba todos os dados vindos de estados, medidas e acções de controlo anteriores. Possuindo o conhecimento do estado anterior, não há qualquer necessidade de informação acerca do passado.

Este requisito permite que as probabilidades condicionadas envolvidas sejam referentes apenas ao estado actual, não sendo necessário guardar toda a informação de dados anteriores.

Para mais considerações acerca do filtro de Bayes considera-se a referência [5] como fundamental.

3.4 Filtro de Kalman

O filtro de Kalman (KF) é um dos métodos mais difundidos de fusão de informação. Deve o seu nome ao seu criador R. E. Kalman, que o apresentou num artigo científico em 1960 [16].

Trata-se de um algoritmo recursivo que permite obter a estimativa óptima da variável a seguir, na medida em que sendo o sistema envolvido linear e os ruídos presentes brancos e

gaussianos, então o KF minimiza o quadrado dos erros dos parâmetros estimados.

O termo "filtro" remete do facto de o KF retirar a estimativa óptima do estado do sistema a partir de medidas afectadas por ruído.

A sua popularidade mantém-se devido à sua garantia de convergência para a solução óptima, ao seu baixo peso computacional, à sua recursividade e ao facto de permitir a fusão de diversos tipos de medidas.

Uma das qualidades do KF é a de incorporar toda a informação que tenha sido recebida. Processa todas as medidas disponíveis, independentemente da sua precisão, para estimar o estado actual do sistema [12].

O termo recursivo vem do facto de não precisar de guardar todos os dados anteriores e de não necessitar de ser completamente reprocessado sempre que uma medida nova é recebida. Esta característica é de uma importância vital na praticabilidade da implementação do filtro em tempo real[12].

Em resumo, o KF espera três requisitos chave, para garantir a convergência para uma solução óptima:

- O sistema envolvido seja linear;
- Os ruídos do modelo do sistema e das medidas sejam sequências de ruído branco (ou seja, sejam independentes, não correlacionados e com média nula);
- A distribuição estatística dos ruídos seja gaussiana.

Desta forma, obtemos o melhor estimador linear.

Considerando um sistema genérico com dinâmica linear cujas saídas são combinação linear do estado [2]:

$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t)+Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{3.21}$$

onde

$$\begin{aligned} x &= [x1 \;\; x2 \;\; \ldots \;\; xn]^T \\ u &= [u1 \;\; u2 \;\; \ldots \;\; un]^T \\ y &= [y1 \;\; y2 \;\; \ldots \;\; yn]^T \end{aligned} \tag{3.22}$$

$dim(A)=n\times n$, n é a dimensão do sistema.

$dim(B)=p\times n$, p é o número de entradas.

$dim(C)=q\times n$, q é o número de saídas.

É possível converter este modelo em tempo contínuo para um equivalente em tempo discreto:

$$\begin{aligned} x(k) &= \phi(k-1)+Gu(k-1) \\ y(k) &= Hx(k) \end{aligned} \tag{3.23}$$

sendo k um número natural que representa o k- ésimo instante de amostragem.

Sabendo que $\Delta t_k = t_k - t_{(k-1)}$, é ainda possível demonstrar que :

$$\phi(k)=\mathrm{e}^{A\Delta t_k} \tag{3.24}$$

$$G=\int_{t_{(k-1)}}^{t_k} \mathrm{e}^{A\Delta t_k} B\,dt \tag{3.25}$$

$$H=C \tag{3.26}$$

O modelo descrito é um modelo determinístico, para obter a sua versão estocástica é necessário introduzir a incerteza proveniente da dinâmica do modelo do sistema *w(k)* e das medidas *v(k)*:

$$\begin{aligned} x(k) &= \phi(k-1)+Gu(k-1)+w(k-1) \\ y(k) &= Hx(k)+v(k-1) \end{aligned} \tag{3.27}$$

Para que o filtro tenha garantia de convergência é necessário que:

$$R_w(k,i)=\begin{Bmatrix} Q(k) & se & i=k \\ 0 & se & i\neq k \end{Bmatrix} \tag{3.28}$$

$$R_v(k,i)=\begin{Bmatrix} R(k) & se & i=k \\ 0 & se & i\neq k \end{Bmatrix} \tag{3.29}$$

$$E(w(k)v(i)^T)=0 \quad \vee k,i \ , \tag{3.30}$$

Estas restrições definem $w(k)$ e $v(k)$ como processo estocásticos compostos por sequências independentes de variáveis aleatórias com variância $Q(k)$ e $R(k)$, respectivamente. Mais ainda a equação 3.30, garante que $w(k)$ e $v(k)$ são processos não correlacionados [17].

$$\begin{aligned} P(w) &\sim N(0,Q) \\ P(v) &\sim N(0,R) \end{aligned} \tag{3.31}$$

Para que a estimativa seja óptima é necessário também que $w(k)$ e $v(k)$ sejam processos com distribuição gaussiana. Neste caso específico, o KF torna-se num estimador de estado de variância mínima, e visto ser um estimador não enviesado, minimizar a variância é o mesmo que minimizar o erro quadrático médio [18].

As matrizes $Q(k)$ e $R(k)$ são diagonais, a primeira representa a covariância do ruído de cada parâmetro do modelo do sistema. A segunda representa a covariância do ruído do modelo do sensor. Todos os ruídos são independentes e não correlacionados, entre si, logo $cov(x,y)=0$. Sendo as matrizes $Q(k)$ e $R(k)$ iguais à equação 3.8, , facilmente se compreende o facto de serem diagonais.

3.4.1 Filtro de Kalman Discreto

O Filtro de Kalman Discreto é a versão deste algoritmo para sistemas lineares definidos em tempo discreto.

Tal como todos os principais métodos de estimação de origem baysiana, divide-se em duas fases: previsão e actualização.

A operação de previsão consiste em usar o modelo do sistema e o estado anterior para prever o estado actual. Nesta operação realiza-se também a previsão da covariância do estado, através do conhecimento da covariância anterior, do modelo do sistema e da sua variância do ruído.

Se existirem medidas referentes ao instante actual, segue-se a fase de actualização, em que se altera o estado, para que este reflicta toda a nova informação actual.

Considerando então as condições iniciais:

$$E\{x(0)\}=X_0 \quad cov\{x(0)\}=P_0 \tag{3.32}$$

E o modelo do sistema:

$$x(k) = \Phi x(k-1)+Gu(k-1)+w(k-1) \tag{3.33}$$

Com as observações

$$y(k) = Hx(k)+v(k) \tag{3.34}$$

A propagação do estado será :

$$x(k^-) = \Phi x(k-1)+B u(k-1) \tag{3.35}$$

A sua covariância associada:

$$P(k^-) = \Phi P(k-1)\Phi^T+Q(k) \tag{3.36}$$

Para a fase de actualização, calcula-se o Ganho de Kalman:

$$K(k) = P(k^-)H^T(H\,P(k^-)\,H^T+R(k))^{-1} \tag{3.37}$$

Actualiza-se o estado:

$$x(k) = x(k^-)+K(k)\left(y(k)-Hx(k^-)\right) \tag{3.38}$$

E a covariância:

$$P(k) = \left(I-K(k)H\right)P(k^-) \tag{3.39}$$

Por vezes em lugar da equação 3.38 usa-se:

$$P(k) = \left(I-K(k)H\right)P(k^-)\left(I-K(k)H\right)^T+K(k)R\,K(k)^T \tag{3.40}$$

Apesar de ser mais complexa, é valida para qualquer ganho $K(k)$, mesmo que este não seja óptimo, o que pode ser interessante em algumas aplicações práticas.

Observando a equação 3.38 conclui-se que quando a covariância do erro de medida R tende para 0, o ganho K faz pesar com maior importância a parcela denominada de inovação $\left(y(k)-Hx(k^-)\right)$ da equação 3.38. Em concreto fica:

$$\lim_{R_k\to 0} K_k = H^{-1} \tag{3.41}$$

Quando a covariância da estimativa do erro do sistema P_k^- aproxima-se de 0, então a parcela inovação terá um peso inferior, tal como se pode verificar a partir de:

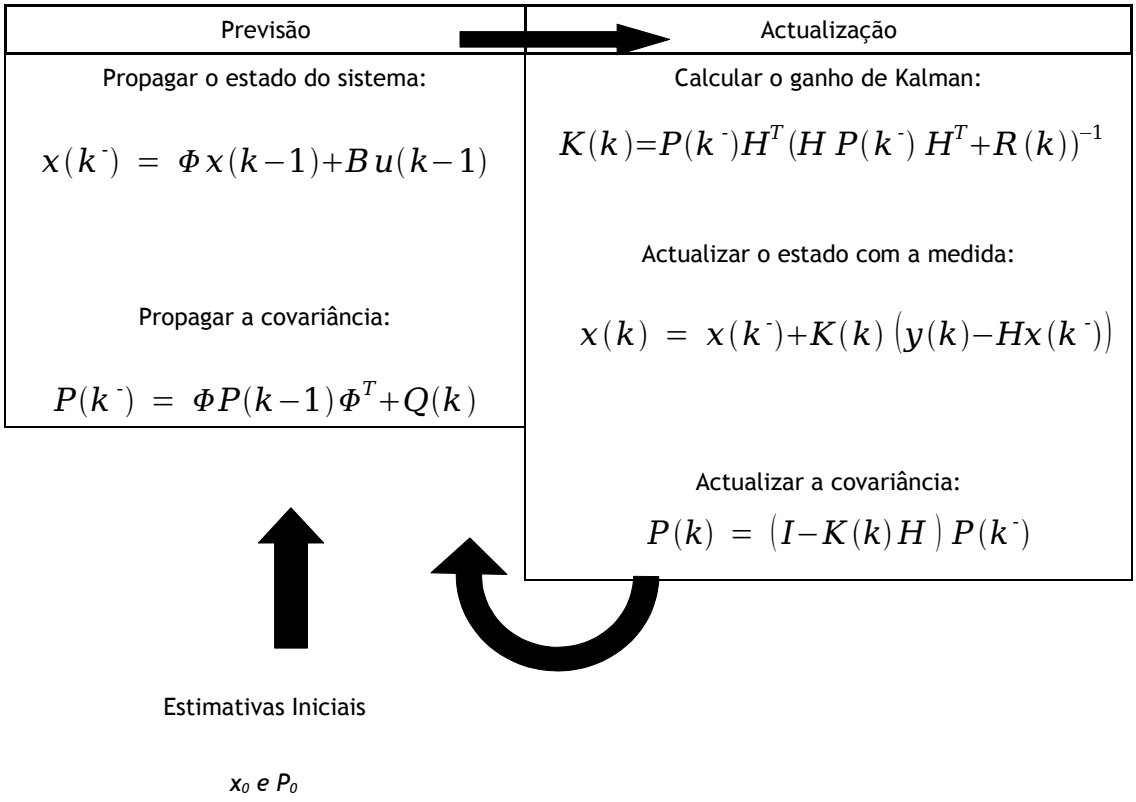
$$\lim_{P_k\to 0} K_k = 0 \tag{3.42}$$

Estas observações fazem sentido, visto que se a matriz R é próxima de 0, significa que existe uma grau elevado de confiança nas medidas recebidas, o que por sua vez, deve afectar, com um peso elevado, a estimativa do estado.

No caso da matriz P_k^- ser próxima de 0, significa que existe uma grande confiança na previsão, ou seja, supõe-se que a estimativa se encontre muito próxima do real estado do sistema. Desta forma, o peso das medidas deve ser reduzido.

Como resumo, observe-se a tabela 3.1 que ilustra o funcionamento do KF.

Tabela 3.1 Resumo das operações do KF



3.4.2 Filtro de Kalman Extendido

O Filtro de Kalman Extendido (EKF) deve ser utilizado em sistemas não lineares. Este filtro, em cada instante, lineariza a função de transferência do sistema, em torno do valor estimado e actualiza as estimativas baseado num KF aplicado a esse sistema linear calculado. Mantém o requisito de que os ruídos presentes sejam brancos e gaussianos.

O EKF tornou-se na ferramenta de estimação do estado mais popular na robótica. A sua força reside na simplicidade e na eficiência computacional [5].

Apesar de não apresentar garantias teóricas de convergência nem de solução óptima, na prática um projecto cuidadoso evita que estes problemas se manifestem [2],

Uma importante limitação do EKF vem do facto de aproximar as transições do estado e das medidas usando expansões de Taylor lineares, desta forma a qualidade da aproximação usada no EKF depende principalmente de dois factores: o grau de incerteza e o grau de não-linearidade das funções a aproximar [5].

O EKF é o melhor filtro linear possível para sistemas não lineares. Filtros não lineares podem ter resultados superiores.

Com as seguintes condições iniciais:

$$E\{x(0)\}=X_0 \quad cov\{x(0)\}=P_0 \tag{3.43}$$

Considere-se o sistema estocástico não linear com as dimensões anteriormente descritas para o caso do KF discreto:

$$\begin{aligned} x(k) &= f\left(x(k-1),u(k),w(k-1)\right) \\ y(k) &= h\left(x(k),v(k)\right) \end{aligned} \tag{3.44}$$

Onde $w(k)$ e $v(k)$ mantém as restrições do KF. Os valores dos ruídos do sistema e da medida, são, obviamente, desconhecidos e têm média nula. É então válida a aproximação:

$$\begin{aligned} \tilde{x}(k) &= f\left(x^{-}(k-1),u(k),0\right) \\ \tilde{y}(k) &= h\left(x^{-}(k),0\right) \end{aligned} \tag{3.45}$$

Onde x^{-} é uma estimativa à *posteriori* do estado (resultante de uma iteração anterior do EKF).

Linearizando as equações do modelo do sistema e das medidas em torno dos resultados anteriores da equação 3.45:

$$\begin{aligned} x(k) &\approx \tilde{x}(k)+A\left(x(k-1)-x^{-}(k)+W\,w(k-1)\right) \\ y(k) &\approx \tilde{y}(k)+h\left(x(k)-\tilde{x}(k)\right)+V\;v(k) \end{aligned} \tag{3.46}$$

É necessário conhecer os jacobianos da descrição do modelo e das observações:

$$A=\frac{\partial f}{\partial x}=\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{3.47}$$

$$H=\frac{\partial h}{\partial x}=\begin{bmatrix}\frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n}\\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial h_n}{\partial x_1} & \frac{\partial h_n}{\partial x_2} & \cdots & \frac{\partial h_n}{\partial x_n}\end{bmatrix} \tag{3.48}$$

Com os jacobianos dos ruídos:

$$W=\frac{\partial f}{\partial w}=\begin{bmatrix}\frac{\partial f_1}{\partial w_1} & \frac{\partial f_1}{\partial w_2} & \cdots & \frac{\partial f_1}{\partial w_n}\\ \frac{\partial f_2}{\partial w_1} & \frac{\partial f_2}{\partial w_2} & \cdots & \frac{\partial f_2}{\partial w_n}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial f_n}{\partial w_1} & \frac{\partial f_n}{\partial w_2} & \cdots & \frac{\partial f_n}{\partial w_n}\end{bmatrix} \tag{3.49}$$

$$V=\frac{\partial f}{\partial v}=\begin{bmatrix}\frac{\partial f_1}{\partial v_1} & \frac{\partial f_1}{\partial v_2} & \cdots & \frac{\partial f_1}{\partial v_n}\\ \frac{\partial f_2}{\partial v_1} & \frac{\partial f_2}{\partial v_2} & \cdots & \frac{\partial f_2}{\partial v_n}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial f_n}{\partial v_1} & \frac{\partial f_n}{\partial v_2} & \cdots & \frac{\partial f_n}{\partial v_n}\end{bmatrix} \tag{3.50}$$

Para simplificar a notação, não se explicitou a dependência de A , H , W e V do instante de amostragem k .

Pode-se agora definir os erros de predição e o resíduo da medida:

$$\begin{aligned}\tilde{e}_x(k)&\equiv x(k)-\tilde{x}(k)\\ \tilde{e}_y(k)&\equiv y(k)-\tilde{y}(k)\end{aligned} \tag{3.51}$$

Pode-se assim descrever os erros:

$$\begin{aligned}\tilde{e}_x(k)&\approx A\left(x(k)-\hat{x}(k-1)\right)+\epsilon(k)\\ \tilde{e}_y(k)&\approx H\tilde{e}_x(k)+\eta(k)\end{aligned} \tag{3.52}$$

onde $\epsilon(k)\ e\ \eta(k)$ são variáveis aleatórias independentes com média nula e covariâncias WQW^T e VRV^T respectivamente, com Q e R a obedecerem às mesmas restrições referidas para o caso linear.

São notórias as semelhanças das equações apresentadas em 3.52 com as equações de um KF discreto. Pode-se então considerar um KF hipotético para estas variáveis em que a estimativa à *posteriori* do processo inicial será:

$$\hat{x}(k)=\tilde{x}(k)+e(k) \tag{3.53}$$

com as seguintes distribuições gaussianas:

$$\begin{aligned} &p\left(\tilde{e}_x(k)\right)\sim N\left(0,E\left[\tilde{e}_x(k)\tilde{e}_x^T(k)\right]\right)\\ &p\left(\epsilon_x(k)\right)\sim N\left(0,E\left[W\,Q(k)\,W^T\right]\right)\\ &p\left(\eta_x(k)\right)\sim N\left(0,E\left[V\,R(k)\,V^T\right]\right) \end{aligned} \tag{3.54}$$

Fazendo que o valor esperado de $\hat{e}(k)$ seja 0 vem:

$$\hat{e}(k)=K_k\tilde{e}_y(k) \tag{3.55}$$

o que resulta em:

$$\hat{x}(k)=\tilde{x}(k)+K(k)\tilde{e}_y(k) \quad = \quad \tilde{x}(k)+K(k)\left(y(k)-\tilde{y}(k)\right) \tag{3.56}$$

que é a equação da actualização do EKF.

A propagação do estado será:

$$\hat{x}(k^-)=f\left(\hat{x}(k-1),u(k),0\right)$$

A propagação da covariância:

$$P(k^-)=A(k)P(k-1)A(k)^T+W(k)Q(k-1)W^T(k) \tag{3.57}$$

O cálculo do ganho de Kalman *K(k)*:

$$\begin{aligned} &L(k)=\left(H(k)P(k^-)H(k)^T+V(k)R(k)V(k)^T\right)^{-1}\\ &K(k)=P(k^-)H(k)^TL(k) \end{aligned} \tag{3.58}$$

Actualização da covariância:

$$P(k)=\left(I-K(k)H(k)\right)P(k^-) \tag{3.59}$$

E finalmente a actualização do estado:

$$x(k)=x(k^-)+K(k)\left(y(k)-h\left(x(k^-),0\right)\right) \tag{3.60}$$

O EKF utiliza uma dinâmica linearizada pelo que mesmo que a estimativa inicial do estado $x(0)$ seja gaussiana, não há garantias que assim se mantenha ao longo do tempo. A propagação da covariância é também feita com recurso à linearização em torno do estado estimado. Estes dois factores de imprecisão são tanto mais importantes quanto menos válida for a aproximação utilizada.

Na discussão acima considera-se que o EKF utiliza a matriz A do processo dinâmico em tempo discreto.

Como se procura uma aproximação da dinâmica do sistema em tempo discreto, parte-se da seguinte descrição dinâmica do sistema no tempo contínuo:

$$\frac{dX(t)}{dt} = f(X(t), u(t_k), t), \qquad t \in]t_{k-1}, t_k] \tag{3.61}$$

Calcula-se de seguida:

$$A_k = \left.\frac{\partial f}{\partial x}\right|_{\substack{x=x(t_k)\\ u=u(t_k)\\ t=t_k}} \tag{3.62}$$

E considerando os valores constantes entre instantes de amostragem obtém-se finalmente:

$$A(k) = \exp\left(A_k (t_k - t_{k-1})\right) \tag{3.63}$$

Uma característica importante dos Filtros de Kalman e do EKF em particular é que o jacobiano H_k da equação 3.48 que calcula o ganho de Kalman propaga correctamente a informação que a medida traz acerca de cada elemento do estado. Desta forma se a medida não afectar todos os elementos do estado as covariâncias do erro do estado são afectadas de acordo com essa situação e a informação que a medida traz.

O KF e o EKF só lidam com um conjunto de medidas sempre disponível. A implementação pode, no entanto, contornar a inexistência de medidas introduzindo covariâncias grandes nas medidas não disponíveis mas tal procedimento poderá trazer problemas numéricos. Outra solução possível é repetir a actualização para as diferentes medidas que existam de facto. O processo descrito é uma aproximação à teoria mas é utilizado na prática [19].

Se o conjunto das medidas não trouxer (nunca) informação acerca de todas as partes do vector de estado, então o filtro torna-se divergente visto o processo não ser observável.

Tabela 3.2 Resumo da operação do EKF

Previsão	Actualização
Propagar o estado do sistema: $\hat{x}(k^-)=f\left(\hat{x}(k-1),u(k),0\right)$ Propagar a Covariância: $P(k^-)=\left(A(k)P(k-1)A(k)^T+ +W(k)Q(k-1)W^T(k)\right)$	Calcular o Ganho de Kalman $K(k)$: $L(k)=\left(H(k)P(k^-)H(k)^T+V(k)R(k)V(k)^T\right)^{-1}$ $K(k)=P(k^-)H(k)^T L(k)$ Actualizar estado com a medida: $x(k)=x(k^-)+K(k)\left(y(k)-h\left(x(k^-),0\right)\right)$ Actualizar Covariância $P(k)=\left(I-K(k)H(k)\right)P(k^-)$

Estimativas Iniciais
x_0 e P_0

3.5 Filtro de Partículas

O Filtro de Partículas (PF) também denominado por localização de Monte Carlo é um algoritmo recente, mas que já se tornou muito popular [5].

O filtro de partículas nasce como resposta aos diversos problemas da localização de Markov baseada em grelhas [5]. Este algoritmo baseia-se numa versão discreta do filtro de Bayes e consiste na divisão do ambiente em células. Cada uma destas células representa uma pose do robot. Seguidamente com as medidas recebidas dos sensores procura-se a célula que melhor se encaixa na percepção do ambiente recebida. Este algoritmo para ter uma boa resolução necessita de um elevado esforço computacional, o que em certas situações, como ambientes muito grandes, impossibilita a sua utilização.

O PF veio resolver este problema pois utiliza o método de Monte Carlo, que é uma forma rápida de resolver problemas morosos, usando números aleatórios.

Para resolver um problema através do método de Monte Carlo é usada uma série de tentativas aleatórias. A precisão do resultado final depende em geral do número de tentativas. Esse equilíbrio entre a precisão do resultado e o tempo de computação é uma característica extremamente útil dos métodos de Monte Carlo. Se se deseja somente uma

solução aproximada, então este método é uma boa opção.

Assim sendo, dependendo do número de tentativas, é possível obter uma boa estimativa do estado do robot.

De salientar que este método existe há séculos, mas só em 1949 foi formalmente apresentado [20]. Com a evolução da capacidade de processamento dos computadores, ganhou ainda mais importância.

Segundo vários artigos científicos, o PF apresenta melhores resultados, sendo mais rápido e preciso, que a localização por grelhas. Outra das grandes vantagens do PF é o facto de resolver o problema da localização global, descrita anteriormente [21].

Segue-se a formulação do método com base num artigo científico de Ioannis Rekleitis, que é visto como um tutorial de aplicação do PF à localização de um robot móvel [22].

O principal objectivo do filtro de partículas, tal como todos os outros referidos, é de seguir a evolução da variável de interesse ao longo do tempo. Uma das vantagens deste método, é o facto de permitir usar distribuições não gaussianas e *inclusive* poder integrar diversas distribuições probabilísticas no mesmo problema [23].

A base deste método consiste em descrever a confiança através de um conjunto de amostras da função de densidade de probabilidade. Essas amostras são denominadas de partículas. Cada partícula é composta por dois parâmetros: um estado possível da variável de interesse e um valor representativo da confiança naquele estado, normalmente denominado de peso, w_j^k .

Neste caso de localização, a variável de interesse é a pose $\boldsymbol{x}^{\boldsymbol{k}} = \left[x^k , y^k , \theta^k \right]$, que compreende a posição cartesiana e a orientação do robot. Assim sendo no instante $t = k$ a pose do robot é representada por um conjunto de M partículas, $S_i^k = \left[\boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{k}} , w_j^k \right] \ : j = 1 \ldots M$, onde j designa cada partícula.

Como todos os filtros de origem baysiana encontra-se divido em duas fases: previsão e actualização. Existe mais uma fase, mas não essencial para a fusão de informação, daí não ser integrada na divisão referida acima. Trata-se de uma secção de manutenção do algoritmo, chamada de re-amostragem.

Na fase de previsão, usa-se um modelo para simular o efeito de uma acção de controlo, no conjunto de partículas que representa a distribuição do estado do robot.

Na fase de actualização, usa-se a informação recebida dos sensores para actualizar o peso de cada partícula, de acordo com a proximidade desta à percepção recebida.

Na fase de re-amostragem, eliminam-se as partículas com peso menor, e criam-se novas em torno das partículas com maior peso. Desta forma aumenta-se a probabilidade de obter uma melhor estimativa.

Existem depois vários métodos, que serão apresentados mais tarde, para decidir qual a partícula ou conjunto de partículas que melhor representa o estado do robot.

Seguidamente descrevem-se as principais fases deste algoritmo, tendo já como objectivo a aplicação na localização de um robot móvel.

3.5.1 Previsão

Nesta fase usa-se o modelo de odometria para prever a alteração da pose do robot no mundo.

Como já foi referido, existem diversos tipos de erros relacionados com os dados recebidos da odometria, torna-se assim necessário modelar o ruído.

Existem muitas abordagens para a minimização do impacto deste tipo de erro, a maior parte usa a adição de um modelo gaussiano do ruído para a movimentação [24] [25] [26].

Neste trabalho decidiu-se seguir o método descrito no tutorial sobre filtro de Partículas de Ioannis Rekleitis , que é baseado em estudos feitos por J.Borenstein.

Assumindo que a pose inicial é:

$$X_{k-1}=[x_{k-1}, y_{k-1}, \theta_{k-1}]^T \tag{3.64}$$

e que os dados obtidos pela odometria (Ox_k , Oy_k) fornecem uma estimação das distâncias absolutas percorridas em cada uma das componentes do sistema de eixos, poderemos então definir:

$$\begin{aligned} \Delta x_k &= Ox_k - Ox_{k-1} \\ \Delta y_k &= Oy_k - Oy_{k-1} \end{aligned} \tag{3.65}$$

$$\Delta \theta_k = arctan2(\Delta y / \Delta x) \tag{3.66}$$

Sendo as componentes da translação parcial 3.67 e rotação parcial 3.68:

$$\delta \rho_k = \sqrt{(\Delta x_k)^2 + (\Delta y_k)^2} \tag{3.67}$$

$$\delta \theta_k = \Delta \theta_k - \theta_{k-1} \tag{3.68}$$

Reunindo as equações 3.67 e 3.68, obtém-se a matriz que define o movimento linear do robot 3.69, que serve para modelar a acção de movimentação.

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + \delta \rho_k * \cos(\theta_k) \\ y_{k-1} + \delta \rho_k * \sin(\theta_k) \\ \theta_{k-1} + \delta \theta_k \end{bmatrix} \tag{3.69}$$

É agora necessário obter o modelo do ruído, para tal divide-se a movimentação em dois modelos independentes: um que retrata a translação parcial definida por $\delta \rho_k$ e outro responsável pela rotação parcial definida por $\delta \theta_k$, ambos nos instante de tempo $t=k$.

Surgem três modelos matemáticos de ruído proveniente da odometria: um para a translação, outro para rotação e o último para os deslizamentos susceptíveis de acontecer durante o movimento [5] [23].

- Incerteza proveniente da rotação

$$randN(\delta\theta_k * M_{rot}, \delta\theta_k * \sigma_{rot}) \tag{3.70}$$

A função 3.70 caracteriza o desvio e respectivos erros existentes aquando da rotação do robot em torno do seu centro. Em que $\delta\theta_k$, é o deslocamento parcial em $t=k$. M_{rot} é um erro sistemático que teoricamente seria possível eliminar e face às dificuldades inerentes à calibração da odometria, só é possível minimizá-lo. E σ_{rot} é o desvio padrão que caracteriza a dispersão de valores prováveis em torno de M_{rot} .

- Incerteza proveniente da translação

$$randN(\delta\rho_k * M_{trs}, \delta\rho_k * \sigma_{trs}) \tag{3.71}$$

A função 3.71 caracteriza o desvio e respectivos erros existentes aquando da translação do robot, dado que a odometria 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

$$randN(\delta\rho_k * M_{drft}, \delta\rho_k * \sigma_{drft}) \tag{3.72}$$

A função 3.72 caracteriza o desvio e respectivos erros existentes a aquando da translação do robot 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.

A função "*randN*" é uma função de geração de números aleatórios com base na distribuição normal. Esta distribuição foi escolhida para a modelação do ruído proveniente da odometria dado que é a mais adequada face aos resultados experimentais obtidos aquando da sua calibração.

Para a modelização de qualquer outra acção é conveniente usar uma distribuição adequada, (não sendo de carácter obrigatório o uso da distribuição normal) dado não existir qualquer restrição por parte do filtro de partículas neste aspecto.

É ainda necessário adicionar o modelo do ruído, pelo que a modelização mais adequada à realidade será dada pela matriz:

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + (\delta\rho_k + randN(\delta\rho_k * M_{trs}, \delta\rho_k * \sigma_{trs})) * \cos(\theta_k) \\ y_{k-1} + (\delta\rho_k + randN(\delta\rho_k * M_{trs}, \delta\rho_k * \sigma_{trs})) * \sin(\theta_k) \\ \theta_{k-1} + \delta\theta_k + randN(\delta\theta_k * M_{rot}, \delta\theta_k * \sigma_{rot}) + randN(\delta\rho_k * M_{drft}, \delta\rho_k * \sigma_{drft}) \end{bmatrix} \tag{3.73}$$

Como é óbvio este modelo é aplicado individualmente a cada uma das partículas.

3.5.2 Actualização

Quando há medidas dos sensores referentes aos instante actual $t=k$, efectua-se a fase de actualização.

De forma simples, nesta etapa procuram-se as partículas que apresentam maior proximidade à percepção construída com os dados dos sensores.

Aproxima-se a probabilidade à *posteriori* descrita no filtro de Bayes (3.19), da seguinte forma:

$$p(x_k|z_k)\approx\sum_{i=1}^{M} w_k^i\,\delta(x_k-x_k^i) \tag{3.74}$$

sendo w_k^i o peso da partícula i , e $\delta(x_k-x_k^i)$ representa a probabilidade do estado da amostra i na distribuição do modelo das medidas recebidas dos sensores [23]:

$$\delta(x_k-x_k^i)\ \propto\ p(x_k|x_k^i;z_k) \tag{3.75}$$

A equação 3.75 é então multiplicada ao peso de cada partícula. Desta forma as amostras com um estado menos provável ficam com um peso menor.

Os pesos das partículas são depois normalizados por forma a que a sua soma seja igual a um.

$$\sum_{i=1}^{M} w_k^i=1 \tag{3.76}$$

Em cada iteração do algoritmo é necessário calcular a melhor estimativa a partir das partículas existentes. Para esse processo existem três métodos descritos no tutorial seguido, sendo estes:

- Melhor Partícula

$$\bar{X}_s=X_s^{max} \tag{3.77}$$

Limita-se a escolher a partícula com peso maior.

- Média Ponderada

$$\bar{X}_s=\sum_{j=1}^{M} X_s^j*w_j \tag{3.78}$$

Faz uma média ponderada usando todas as partículas.

- Média Robusta

$$\bar{X}_s=\sum_{j=1}^{L} X_s^j*w_j:|X_s^j-X_s^{max}|\leq\epsilon \tag{3.79}$$

Faz uma média ponderada usando apenas as partículas com um peso maior que uma determinada constante.

3.5.3 Re-amostragem

Esta fase é uma das mais importantes na manutenção do filtro. Para entender a sua importância, recorre-se a um exemplo: imagine-se a utilização do PF na localização de um robot móvel. Em cada iteração efectua-se com o modelo do movimento a previsão da pose de cada uma das partículas. Quando são recebidas medidas dos sensores, actualizam-se os pesos das partículas de acordo com a probabilidade destas estarem próximas da percepção criada pelos sensores.

Assim sendo ficam com peso menor as partículas que representam uma pose menos provável do robot, de acordo com a informação recebida.

Ao longo das várias iterações, as partículas menos prováveis ficarão com um peso cada vez menor, ou seja para além de não representarem poses úteis, ocupam tempo de cálculo importante.

A solução apresentada em diversas aplicações do PF, [5], para este problema consiste na eliminação das partículas com um peso menor que uma determinada constante. Segue-se depois a criação de novas partículas copiando as poses das partículas com peso mais elevado. Este acto traduz-se num maior conjunto de amostras em torno da pose mais provável do robot, o que leva a uma melhor precisão na estimativa.

É essencial não colocar todas as partículas criadas em torno das poses mais prováveis, pois em caso de um rapto do robot, ou algum acontecimento similar, corre-se o risco de não existirem partículas nessa zona.

Em resumo eliminam-se as partículas com peso menor, criam-se novas em torno das partículas com peso mais elevado e espalham-se aleatoriamente algumas das criadas.

3.6 Conclusões

Foram apresentadas as principais ferramentas teóricas de fusão de informação utilizadas neste trabalho.

O filtro de Bayes é uma das mais importantes bases dos algoritmos estocásticos de fusão. O filtro de Kalman é um algoritmo recursivo de fusão de informação, que para sistemas lineares garante a convergência para a solução óptima. O filtro de partículas apresenta-se como um método recente que permite ultrapassar algumas das limitações do FK, tais como a aceitação de diversas distribuições estatísticas e resolução do problema de localização global.

Capítulo 4

Trabalho Desenvolvido

4.1 Introdução

Neste capítulo apresenta-se todo o trabalho efectuado, bem como algum do material herdado de projectos anteriores.

Inicialmente surge uma descrição breve do robot de limpeza e seguidamente apresentam-se os seus sensores e principais programas de controlo. Expõe-se também a sua arquitectura e o conceito de aquisição inteligente de sensores.

4.2 Robot de limpeza

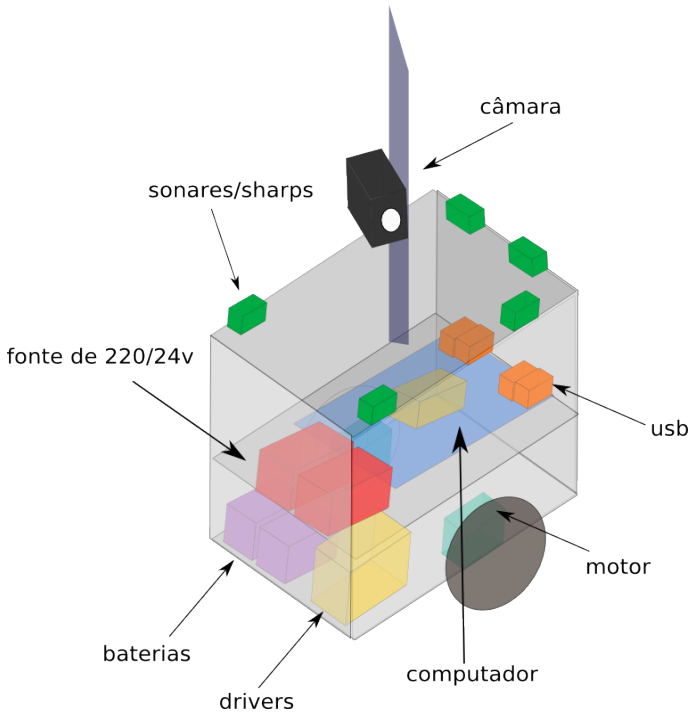


Figura 4.1 Representação dos componentes mais importantes do robot de limpeza

O robot de limpeza é um computador pessoal capacidade de se movimentar graças ao seus motores. Tem uma fonte que permite converter 220v em 24v e desta forma carregar as baterias e funcionar por alimentação da rede eléctrica. Tem diversos conjuntos de *drivers* que fazem o controlo dos sensores e dos motores. Como sensores tem uma câmara e sensores de posição como sonares e *sharps* infravermelhos. O facto de o seu cérebro ser um computador normal, permite vantagens ao nível de interfaces de comunicação e também permite a ligação à *internet* sem fios da FEUP. Outra vantagem de usar um computador é a possibilidade de usar linguagem de alto nível como o pascal no *software Lazarus*.

O robot de limpeza apresenta uma estrutura em alumínio, cúbica, com uma configuração de locomoção do tipo diferencial.

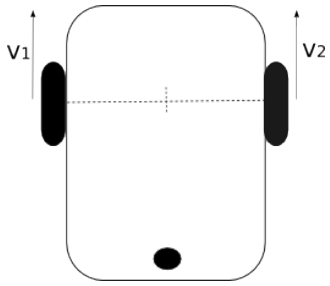


Figura 4.2 Tracção diferencial

Segue-se uma breve explicação dos componentes mais importantes do robot de limpeza, iniciando-se pelo conjunto de sensores utilizados.

4.2.1 Odometria

Os *encoders* incrementais são elementos fundamentais no cálculo da odometria de um robot. Neste caso existe um *encoder* ligado a cada roda motriz. Através dos dados enviados pelos sensores é possível calcular o deslocamento de cada roda. Informações acerca da calibração da odometria podem ser encontradas nas referências [27] e [28].

4.2.2 Sonares

Figura 4.3 Sonar *Devantech* SRF08

O robot está equipado com seis sonares *Devantech* SRF08 cujo interface de comunicação é através do protocolo I2C e tendo como característica física um ângulo de abertura na ordem

dos 30 graus e um alcance máximo de 6 metros [29]. De forma a garantir medidas úteis em cada instante de amostragem, optou-se por diminuir o alcance para 2metros.

As definições do ganho, alcance e sequências, podem ser redefinidas *on-line* e permanecem guardadas dado que são programadas em memória *EEPROM*.

O grande potencial dos sonares é a detecção de obstáculos, devido à sua grande abertura de eco.

Para informações referentes ao *firmwire* implementado consultar [27] e [28] .

Algumas das suas desvantagens estão relacionadas com o ângulo de reflexão do eco do sensor ou com *cross-talk* em que um sonar recebe o eco de outro [30].

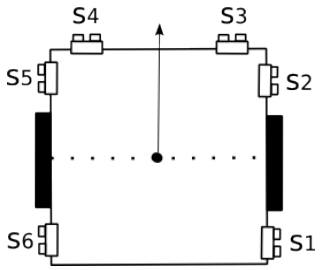


Figura 4.4 Disposição dos sonares no robot

4.2.3 Sharps

Figura 4.5 *Sharp*

O robot tem equipado um conjunto de quatro sensores que medem distâncias por infravermelhos (Sharp GP2Y0A02YK0F), com gama de medição que se situa entre os 15cm e os 150cm, tendo uma disposição no robot similar aos sonares [31].

A sua saída é analógica, necessitando assim de uma fase de tratamento para que possam ser enviados os dados. Esta fase é efectuada numa placa dedicada de conversores A/D que envia por RS-232 os dados para o computador.

Para reduzir a incerteza da medida foi implementada na placa responsável pela conversão uma média ponderada das últimas quatro conversões.

As vantagens destes sensores consistem na sua precisão, no seu baixo custo e na disponibilidade em fornecer medidas. Existe também a possibilidade de usar um grupo de

sensores com um maior alcance (*Sharp* GP2Y0A700K0F).

O método de medição consiste na medição do ângulo através de uma lente que observa o raio infravermelho e graças à trigonometria, sabendo a distância entre a lente e o infravermelho calcula a distância ao objecto, como se pode ver na figura 4.6.

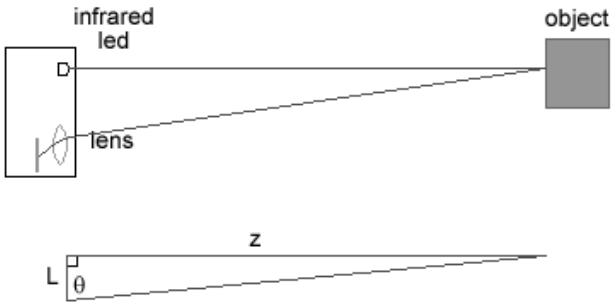


Figura 4.6 Método de medição do s*harp*

A disposição dos *sharps* no robot é a seguinte:

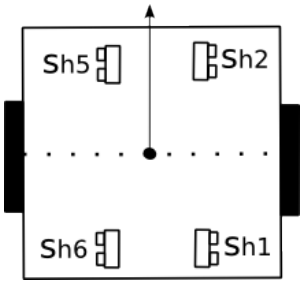


Figura 4.7 Disposição dos *sharps* no robot

4.2.4 Câmara

Figura 4.8 Câmara de leitura dos BC

O processamento de imagem da câmara, foi realizado pelos professores Paulo Costa e Armando Sousa, numa ferramenta de desenvolvimento denominada de *Kylix* utilizada no

futebol robótico. Este processamento de imagem usa a câmara como um sensor que possibilita a detecção e identificação de códigos de barras no formato "*Interleaved* 2 de 5" . Através da inclinação das linhas do código de barras,calcula o ângulo da câmara em relação ao código de barras. E dado que estes marcadores tem um tamanho conhecido, é possível determinar as distâncias entre a câmara e os códigos.

Este processamento é assim uma enorme vantagem, pois um único código de barras permite obter informação acerca de todas as componentes da pose do robot.

Este programa comunica com a aplicação de controlo através do envio de pacotes UDP por sockets.

4.2.5 Software

Neste projecto usada uma distribuição de linux denominada de *Ubuntu*. Esta distribuição tem origem numa distribuição base chamada de *Debian* orientada para servidores devido à sua grande estabilidade.

Como plataforma de desenvolvimento usou-se o *Lazarus*, que é uma ferramenta relativamente nova, que vem no seguimento de programas como o *Delphi*. Tem a grande vantagem de ser *opensource*.

Uma funcionalidade muito interessante é a possibilidade de conectar o robot à rede de *internet* sem fios da FEUP e desta forma ser possível controla-lo em qualquer local com *internet*. Esta capacidade permite também o controlo do robot sem necessidade de usar uma ligação física e mesmo assim implementar as funcionalidades típicas de um cordão umbilical.

4.2.6 Principais aplicações desenvolvidas

- *CleanRob Map Creator*: ferramenta de construção ou reconstrução de ficheiros de mapa. Este tipo de ficheiros são utilizados por algumas das aplicações desenvolvidas, para representação do "mundo" onde o Robot de Limpeza se encontra. Possibilita, de forma simples, a criação de mapas, o que facilita a utilização do robot noutros espaços.
  No ficheiro de mapa são também definidas zonas inválidas, que são zonas fisicamente impossíveis do robot frequentar. Também é condensada informação relativa a marcadores (códigos de barras) no ambiente.

- *CleanRob Log Replay*: esta ferramenta permite ler todos os dados guardados durante um ensaio. Assim é possível verificar as decisões tomadas, bem como as convergências dos vários métodos de localização em *offline*. Tem diversas funcionalidades sendo que umas das mais importantes é a capacidade de mostrar em imagem, a posição do robot estimada pelos vários métodos, as medições efectuadas pelos sensores, as incertezas das estimativas e até é possível observar a posição real medida. Foi com este

programa que grande parte das figuras da validação experimental foram criadas.

- *CleanRob Control Program*: aplicação fulcral para todo o funcionamento do projecto, esta aplicação contempla toda a "inteligência" do robot de limpeza, sendo responsável pelo seu controlo e decisões. Esta aplicação gera um ficheiro com a informação obtida e processada em cada ciclo de controlo, podendo este ficheiro depois ser lido pela aplicação "*CleanRob Log Replay Program*" descrita acima. É neste programa que se faz a localização, navegação e planeamento de tarefas, bem como o tratamento dos dados dos sensores e o envio das ordens aos motores.

4.2.7 Modelos de ruído

Todos os modelos de ruído dos sensores foram aproximados a distribuições gaussianas. Dois motivos essenciais para essa decisão:

- Possibilidade de utilização do filtro de Kalman extendido, visto a distribuição normal ser um requisito deste algoritmo.
- A distribuição gaussiana pode ser completamente descrita através da sua média e do seu desvio-padrão. Este facto traduz-se numa maior facilidade em representar o modelo do ruído.

4.2.7.1 Modelo da Odometria

Tabela 4.1 Modelo do erro de translação

Translação	Média	Desvio-Padrão
X (m)	0,0212	0,0040
Y (m)	0,0139	0,0035
θ (rad)	0,0212	0,0040

Tabela 4.2 Modelo do erro de rotação

Rotação	Média	Desvio- Padrão
θ (rad)	0,0086	0,0068

Tabela 4.3 Modelo do erro de deslizamento

Deslizamento	Média	Desvio-Padrão
θ (rad)	0,0013	0,0027

4.2.7.2 Modelo do Sonar

Tabela 4.4 Modelo do erro do sonar

Modelo	Média	Desvio-Padrão
d (m)	0,0088	0,0579
θ (rad)	0,1715	0,3948

4.2.7.3 Modelo do *Sharp* GP2Y0A02YK0F (1.5 m)

Tabela 4.5 Modelo do erro do *sharp(1m)*

Modelo	Média	Desvio-Padrão
d (m)	0,0032	0,0399
θ (rad)	0,0214	0,0493

4.2.7.4 Modelo do *Sharp* GP2Y0A700K0F (5 m)

Tabela 4.6 Modelo do erro do *sharp(5m)*

Modelo	Média	Desvio-Padrão
d (m)	0,012	0,645
θ (rad)	0,0567	0,0854

4.2.7.5 Modelo da Câmara

Tabela 4.7 Modelo do erro da câmara

Modelo	Média	Desvio-Padrão
X (m)	0,0182	0,0562
Y (m)	0,0192	0,1810
θ (rad)	0,0383	0,1386

Para mais considerações acerca do equipamento do robot e do seu funcionamento, consultar as teses de Fernando Pedro Pinto e Gustavo Pimentel [27] [28].

4.3 Arquitectura

4.3.1 Arquitectura tecnológica

É possível visualizar na figura 4.9 os diferentes tipos de comunicação utilizados, para a integração total de todos os sistemas no robot de limpeza.

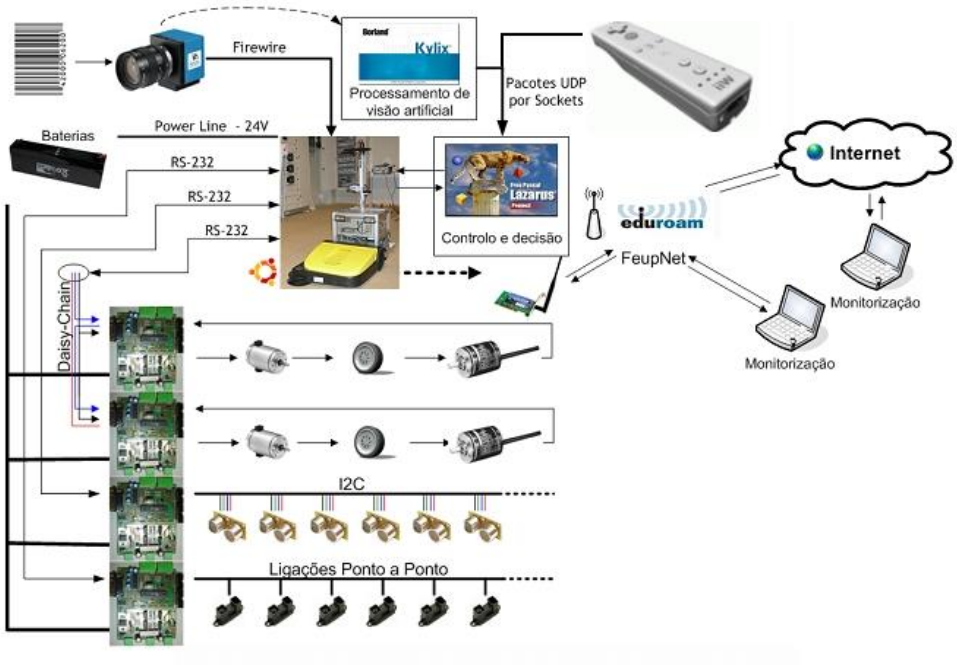


Figura 4.9 Arquitectura Tecnológica – melhorada da Tese de Fernando Pinto

Os vários tipos de comunicação são:

- *Sockets* com pacotes UDP, usados na ligação entre o programa de localização por códigos de barras e o programa de controlo. E são também usados pela aplicação do comando *wiimote;*
- *FireWire*, entre a câmara e o robot;
- Porta-série RS-232:
  - aplicando uma lógica de *daisychain* de modo a interligar mais do que um dispositivo através da mesma conexão física. Estratégia utilizada entre os programas de controlo e drivers dos motores.
  - numa ligação série, mas do tipo *unicast*, estratégia utilizada para as outras duas placas de drivers, de sonares e sharps.

Observa-se também, na figura 4.9, a ligação à rede de *internet* sem fios da FEUP, que permite o controlo do robot a partir de qualquer computador com *internet*.

4.3.1.1 Adaptação ao comando *Wiimote*

Acerca do comunicação com o comando da *wii* é necessário esclarecer que este funciona como um comando local, tomando se desejado o controlo do robot.

O comando transmite por *bluetooth, e* é recebido por uma aplicação de reconhecimento de movimentos criada por outro aluno. A comunicação entre o programa de controlo do robot e aplicação da *wiimote* é feita por *sockets* com pacotes UDP, em que é indicado o tipo de controlo que o utilizador quer efectuar.

Neste momento é possível apenas controlar as velocidades do robot, mas futuramente será implementado um controlo de alto nível, em que se poderá dar ordens de movimentos mais complexos ao robot. Existe como é óbvio, medidas para proteger o robot, tal como limitadores de velocidade, e temporizadores para verificar se a *wiimote* deixou de transmitir.

4.3.2 Aquisição Inteligente de Sensores (ISA)

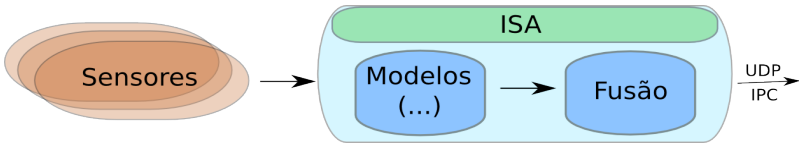


Figura 4.10 Constituição básica do ISA

Durante este projecto tornou-se evidente a necessidade de criar uma ferramenta que recebesse e tratasse os dados dos vários sensores, de forma a libertar o programa principal de controlo destas acções.

Este programa ou agente inteligente dependendo da denominação preferida, teria que reconhecer quais os sensores ligados ao robot através de uma lista predefinida. Enviar os

pedidos de informação aos sensores e receber e converter os dados para valores que possam ser usados pelo programa de controlo. Teria também que efectuar uma primeira verificação da validação da medida, como por exemplo confirmar se a medida tinha ultrapassado o valor de fim de escala definido.

Como requisito mais interessante surge a fusão de informação. Assim sendo o programa deveria usar dados de vários sensores de forma a garantir uma medida melhor. Exemplo disso é a utilização das medidas dos *sharps* para verificar se os dados vindos dos sonares se encontram numa zona não linear. Ou seja na zona não linear dos sonares funcionam como sensores complementares, nas zonas comuns de medição actuam como sensores redundantes. Há autores [30] que chamam a este método de usar sensores que fornecem o mesmo resultado mas medem grandezas diferentes de redundância lógica.

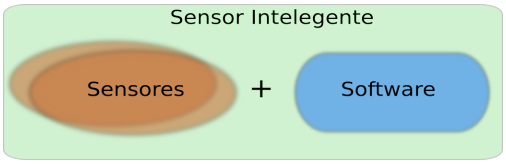


Figura 4.11 Constituição de um sensor inteligente

Outra funcionalidade interessante seria a de ter vários sensores inteligentes ou sensores lógicos com este programa. Um sensor inteligente [9], denominado por outros autores como sensor lógico [30] , é o conjunto do sensor mais o programa que converte os seus dados numa medida mais interessante. Um exemplo disto seria a transformação das duas medidas enviadas pelos sensores na lateral do robot numa distância e num ângulo em relação à parede. Por outras palavras, fisicamente ter-se-ia dois sensores, que poderiam ser um par de *sharps* ou um par de sonares, ou outro tipo de dispositivo semelhante, que devolveriam duas distâncias. E na saída deste sensor lógico ter-se-ia um distância e um ângulo em relação à parede. Assim resultaria num sensor melhor graças à fusão dos dados. Esta solução é usada no EKF na localização por linhas.

Outra opção de fusão também muito aliciante seria a junção de vários sensores com o objectivo de obter um sensor semelhante mas com uma resolução e alcance muito melhores. Neste caso usava -se os dados de um dos sonares laterais, do *sharp* correspondente, e juntar-se-ia um outro *sharp* de alcance de 5metros. Nas distâncias reduzidas usar-se-ia o *sharp* de alcance pequeno, nas maiores distâncias o *sharp* de alcance maior, e nas medidas intermédias juntavam-se os dados dos três sensores, resultando numa medida com um erro muito menor.

A comunicação com o programa de controlo é feita por *sockets* por pacotes UDP ou então por IPC – *Inter Process Comunication*, sistema de comunicações entre processos, disponível no Lazarus.

Resumidamente neste módulo, recebe-se e trata-se os dados dos sensores, efectuam-se os primeiros testes de verosimilhança e faz-se uma fusão dos dados de forma a obter medidas mais interessantes e úteis.

Resta acrescentar que a ferramenta ISA não foi completamente finalizada, devido à falta

de tempo, mas também devido ao enorme número de funcionalidades possíveis de implementar neste bloco.

4.3.3 Arquitectura de Controlo

A arquitectura de controlo é uma arquitectura híbrida, composta por uma parte reactiva e uma parte hierárquica. No nível reactivo, não existe planeamento ou decisões a longo prazo. No entanto nesse nível encontram-se funções importantes como por exemplo evitar colisões. Nesta função apenas se lêem os dados dos sensores e actua-se sobre os motores.

No nível superior, a estrutura de controlo é muito semelhante a uma máquina de estados. Inicialmente tenta-se localizar o robot, depois com a pose estimada o organizador decide o objectivo e o navegador define o caminho a seguir e as velocidades a enviar para os controladores dos motores.

Uma das características interessantes desta estrutura é a capacidade de guardar em registo todos os dados importantes em cada instante. Desta forma cria-se um repositório de todas as variáveis úteis e das decisões tomadas. Assim através deste sistema é possível reproduzir *offline*, todos os acontecimentos passados durante a missão do robot, examinando assim a evolução dos diversos algoritmos usados, bem como as medidas recebidas dos sensores e as ordens enviadas aos motores.

Outra característica relevante é aquisição inteligente de sensores explicada na secção anterior. Salienta-se também a modularidade implementada no nível hierárquico, em poucas palavras, é possível alterar facilmente qualquer bloco da estrutura, tal como acrescentar novos algoritmos em cada secção. Um exemplo: neste trabalho usaram-se dois métodos de localização, numa arquitectura normal, só seria possível usar um método de cada vez. Na arquitectura criada é possível ter vários algoritmos dentro do mesmo bloco, ou seja, é simples colocar os dois métodos de localização a funcionar em simultâneo e se desejado comparar os resultados entre ambos. Várias vezes foi usada esta opção durante os ensaios experimentais, para verificar a estimativa de cada algoritmo.

Trata-se assim de uma arquitectura bastante útil e muito modular, própria para sistemas que actuam em tempo real.

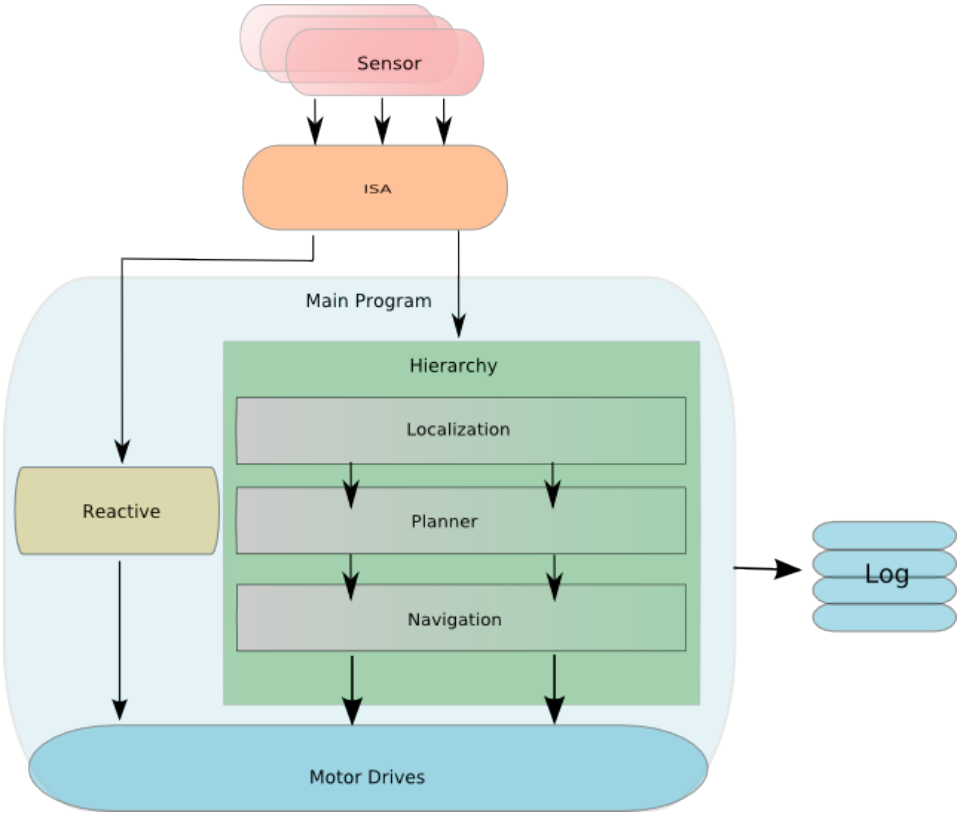


Figura 4.12 Resumo da arquitectura implementada

4.3.4 Ciclo de Controlo e Eventos

Em qualquer projecto cujo objectivo é actuar em tempo real, é necessário avaliar todos tempos de processamento, e dar alguma atenção à forma como o sistema operativo usado trata dos eventos referentes ao programa de interesse [2].

Na figura 4.13 descreve-se o funcionamento do programa de controlo. Apesar do termo "*event*" se encontrar dentro dos programas, este na realidade pertence ao sistema operativo, mas optou-se por esta representação por ser mais compreensível.

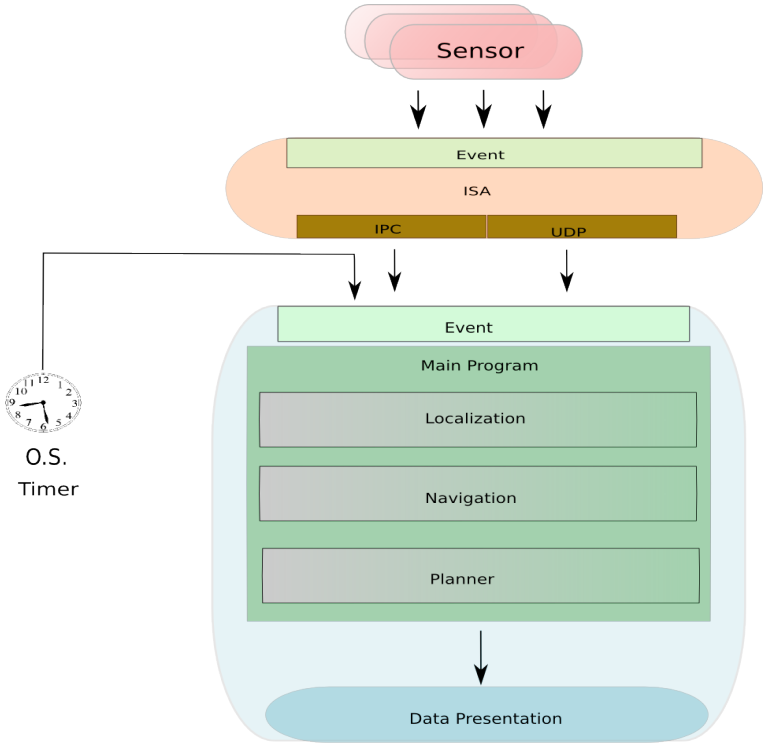


Figura 4.13 Descrição dos eventos do programa de controlo

Neste caso, os eventos são causados por vários factores:

- transmissão de dados dos sensores via porta-série;
- envio de dados do ISA por sockets UDP ou por IPC;
- temporizador do sistema operativo, que controla o intervalo entre ciclos do programa;
- apresentação gráfica de dados.

A aplicação programada em *Lazarus* não tem uma forma de lidar com os eventos de acordo com prioridades e este é um facto muito relevante. Se houver um excesso de chegada de eventos corre-se o risco de não se atenderem todos em tempo útil. Um exemplo prático: se se tentar representar muitos dados no ecrã, tal como gráficos, texto ou vídeos, os dados dos sensores podem não ser atendidos no instante em que são úteis.
Devido ao possível *jitter* que pode acontecer em qualquer sistema, torna-se aconselhável manter uma folga entre o tempo utilizado e o tempo máximo disponível por cada ciclo de controlo. Uma das soluções encontradas para diminuir o tempo gasto, foi alterar o número de vezes em que se apresentam dados no ecrã. Ou seja em vez de se mostrar dados em todos os ciclos de controlo, decidiu-se apresenta-los em intervalos mais alargados. Como o tempo de cada ciclo é de 100ms, se apenas forem mostrados dados de 3 em 3ciclos, significará uma actualização de ecrã de cerca de 300ms, que é mais que suficiente para o ser humano saber o que se passa no robot.

Mais informações acerca dos intervalos de tempos do programa encontram-se na secção denominada de validação experimental.

4.4 Filtro de Kalman Extendido

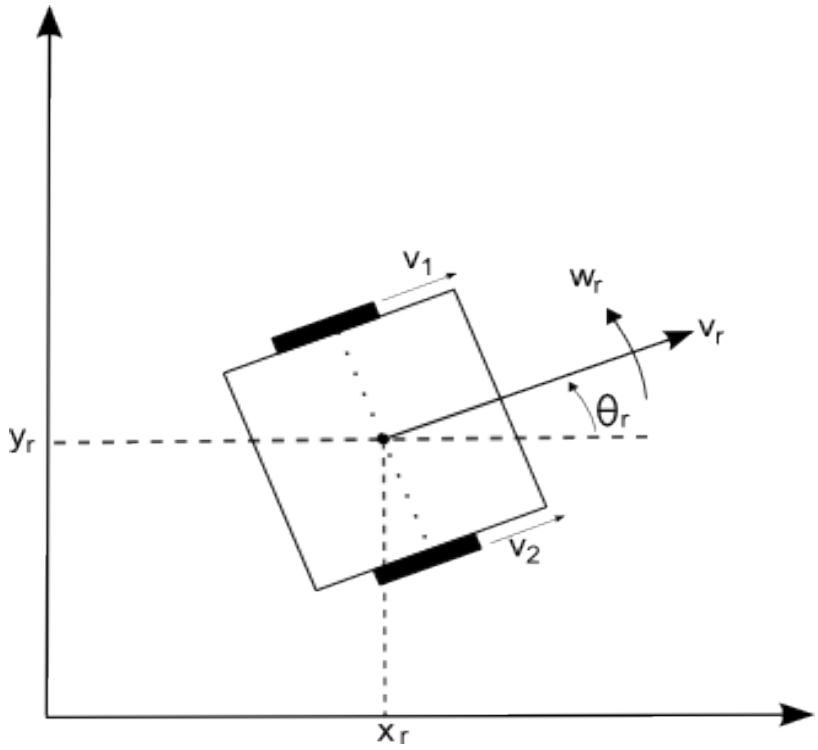


Figura 4.14 Representação do robot no plano *xy*

O vector de estado escolhido para o sistema é:

$$X(t)=\left[x_r(t)\ \ y_r(t)\ \ \theta_r(t)\ \ v_r(t)\ \ \omega_r(t)\right]^T \tag{4.1}$$

onde (x_r, y_r) representa a posição cartesiana do robot, θ_r é a orientação, v_r é a sua velocidade tangencial e ω_r , a velocidade angular. Para a obtenção do modelo, considera-se a tracção diferencial sem qualquer tipo de deslizamento, ou seja a velocidade das rodas no ponto de contacto com o chão é sempre perpendicular ao seu eixo.

O modelo do movimento do robot é caracterizado pela velocidade linear, v_r , e pela velocidade angular, ω_r . Através das velocidades v_1 e v_2 das rodas motrizes do robot conclui-se:

$$\begin{cases} vt = \dfrac{(v1+v2)}{2} \\ w = \dfrac{(v1-v2)}{b} \end{cases} \tag{4.2}$$

sendo b, a distância entre rodas do robot.

O sistema de odometria, composto por *encoders*, fornece valores que são convertidos nas velocidades de cada roda v_1 e v_2 . Como os valores de vr e wr são proporcionais às velocidades das rodas, são por sua vez, facilmente obtidos.

Entre instantes de amostragem consecutivos, estes valores apresentam variações lentas. Assim sendo, em robótica móvel costumam usar-se as velocidades linear e angular obtidas pelo sistema de odometria, como entradas de comando.

Esta suposição não é de todo irreal, pois analisando brevemente o assunto comprova-se a sua legitimidade.

Segundo a teoria, deveriam ser usadas como entradas, as velocidades de referência v e w impostas aos controladores dos motores.

Um das vantagens de usar a odometria como fonte de dados para o conjunto de acções de controlo, reside no facto de captar informação acerca da movimentação do robot, que de outra forma não seria observada. Serve como exemplo deste facto, o acto de empurrar o robot, que apesar das velocidades de referência serem nulas, a velocidade real do robot é diferente de zero.

Outra das desvantagens de usar as velocidades de referência, é que por vezes a velocidade imposta aos controladores não é concretizada, basta por exemplo que algum objecto trave as rodas.

Ou seja nem sempre as velocidades de referência traduzem a situação real do robot.

A opção de usar as velocidades obtidas pela odometria como entradas, resolve os problemas assinalados e retratam de forma mais fiável o movimento do robot. Ficam assim demonstrados os benefícios desta decisão.

A equação da cinemática do robot no tempo continuo é:

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} v(t)*\cos(\theta(t)) \\ v(t)*\sin(\theta(t)) \\ \omega(t) \end{bmatrix} \tag{4.3}$$

No sentido de obter a equação de actualização do sistema em tempo discreto considera-se que os sinais de controlo são constantes entre cada instante de amostragem. De acordo com as equações 3.33 a dinâmica em tempo discreto do sistema é:

$$\begin{aligned} X(t_k) &= A_k X(t_{k-1}) + Bu(t_{k-1}) \\ Y(t_k) &= HX(t_k) \end{aligned} \tag{4.4}$$

E a matriz de estado na sua versão discreta fica:

$$A_k = \begin{bmatrix} 0 & 0 & -v(t_k) * \sin(\theta(t_k)) \\ 0 & 0 & v(t_k) * \cos(\theta(t_k)) \\ 0 & 0 & 0 \end{bmatrix} \tag{4.5}$$

Em termos do modelo, para o erro correspondente ao modelo dinâmico do sistema, este é assumido como, o modelo do erro da odometria, já que o modelo é baseado neste sistema.

$$Q = \begin{bmatrix} \delta^2 x_r & 0 & 0 \\ 0 & \delta^2 y_r & 0 \\ 0 & 0 & \delta^2 \theta_r \end{bmatrix} \tag{4.6}$$

Assim sendo a matriz *Q* será preenchida pelas variâncias do modelo do ruído da odometria.

4.4.1 Auto-Localização por linhas

A grande maioria dos robots usa anéis de sensores ou lasers para conseguir localizarem-se no espaço onde se encontram, nesta tese, seguiu-se um conceito diferente.

Como o ambiente de trabalho escolhido para este projecto é no interior de um edifício, mais propriamente num corredor, optou-se por usar algumas características comuns à generalidade destas áreas. As zonas mais características são as paredes que constituem o corredor, se se observar unicamente o plano *xy*, é legitimo representar as paredes por linhas paralelas ao corredor.

Sabendo a posição destas linhas, obtém-se uma informação muito útil acerca da posição e orientação do robot.

Assim sendo optou-se por usar grupos de dois sensores que medem distâncias, nas laterais do robot. Com as duas medidas recebidas, é possível calcular o ângulo do robot e a sua distância em relação à parede.

Foram usados dois tipos de sensores: sonares e *sharps* infravermelhos, para que fosse possível comparar resultados entre eles e também para que exista redundância lógica da informação [30].

Esta solução pode ser ampliada, colocando grupos à frente e atrás do robot.

Segue-se a formulação teórica deste método:

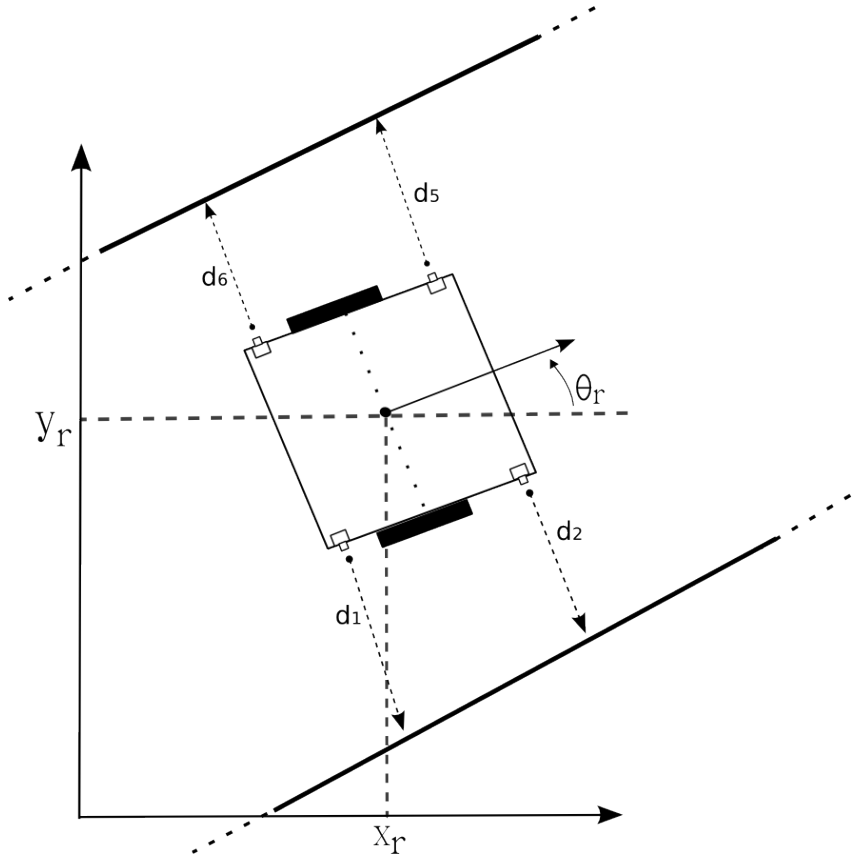


Figura 4.15 EKF Auto-localização por linhas

Com estes grupos de sensores obtém-se 4 medidas relativas ás distâncias entre o robot e as paredes. Os ângulos entre o robot e as paredes são:

$$\theta_{1,2} = \arctan\left(\frac{d_2 - d_1}{dist_{1,2}}\right) \tag{4.7}$$

$$\theta_{5,6} = \arctan\left(\frac{d_5 - d_6}{dist_{5,6}}\right) \tag{4.8}$$

Como distância à parede usa-se a média das duas distâncias em cada lado, desta forma a distância obtida é referente ao eixo central do robot.

Os vectores das medidas a fundir ficam :

$$Y_L^{Sh} = \begin{bmatrix} \frac{(d_1 + d_2)}{2} \\ \theta_{1,2} \end{bmatrix} \tag{4.9}$$

$$Y_R^{Sh} = \begin{bmatrix} \frac{(d_5 + d_6)}{2} \\ \theta_{5,6} \end{bmatrix} \tag{4.10}$$

Usando as equações 4.9 ou 4.10, e sabendo que $d_{sharp} = \frac{(d1+d2)}{2}$ e $\theta_{sharp} = \theta_{1,2}$,então o modelo dos sensores fica:

$$\begin{cases} \theta_w = \theta_{sharp} + \theta_r \\ d_w = d_{sharp} + x_r * \sin(\theta_w) - y_r * \cos(\theta_w) \end{cases} \tag{4.11}$$

sendo θ_w , o ângulo da parede e θ_r o ângulo do robot.

Colocando em ordem ás medidas dos sensores:

$$\begin{cases} \theta_{sharp} = \theta_w - \theta_r \\ d_{sharp} = d_w - x_r * \sin(\theta_w) + y_r * \cos(\theta_w) \end{cases} \tag{4.12}$$

Desta forma, chega-se ao jacobiano da medida:

$$H^{Sh} = \frac{d(d_{Sh}, \theta_{Sh})}{d(x_r, y_r, \theta_r)} = \begin{bmatrix} \sin(\theta_w) & \cos(\theta_w) & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{4.13}$$

O modelo do erro é dado pela variância do ruído dos sensores:

$$R^{Sh} = \begin{bmatrix} \sigma^2_{d_{sh}} & 0 \\ 0 & \sigma^2_{\theta_{sh}} \end{bmatrix} \tag{4.14}$$

Este método, assim como as equações referidas são similares para os sonares e *sharps*, alterando apenas os valores das variâncias do ruído.

Na implementação do EKF, foi aplicado um filtro de verosimilhança para cada um dos conjuntos de medida, onde se testa o valor de cada uma das medidas, afim de se decidir a sua validação. Compara-se também o valor absoluto dos ângulos dados por cada lateral, de forma a garantir que não existem diferenças elevadas entre eles, evitando assim más medidas provocados por objectos ou pessoas.

4.4.2 Localização por Códigos de Barras

A explicação pormenorizada deste método pode ser encontrada na tese do Professor Armando Sousa [2].

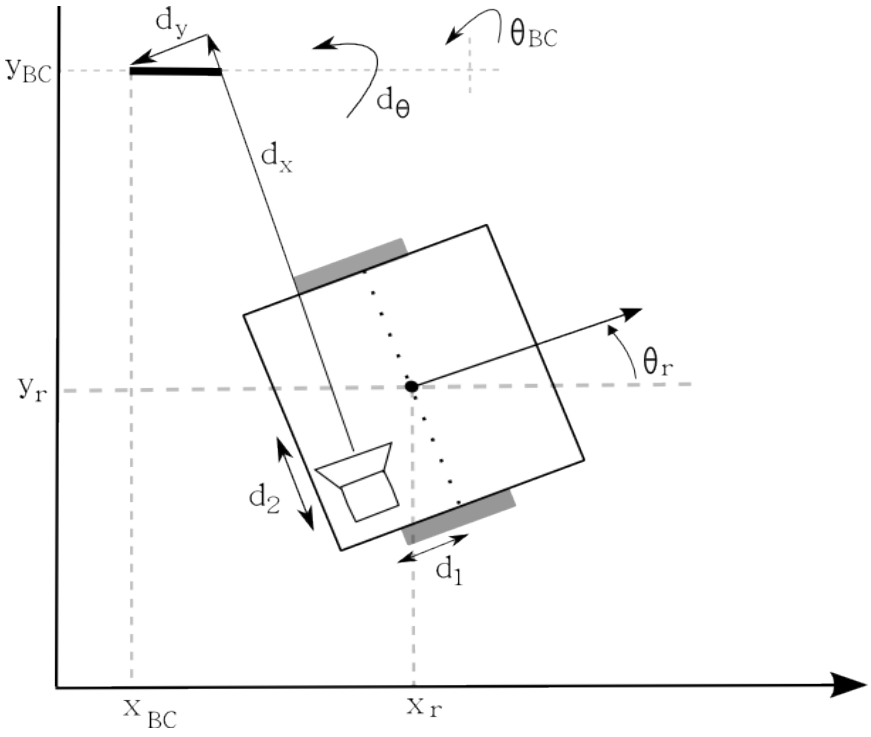


Figura 4.16 EKF Auto-localização por códigos de barra

É possível determinar a posição do robot, baseado nas medidas enviadas pelo programa de analise e localização de código de barras como:

$$Pos_{BC}=[x_{BC} \quad y_{BC} \quad \theta_{BC}] \tag{4.15}$$

$$Pos_{r}=[x_{r} \quad y_{r} \quad \theta_{r}] \tag{4.16}$$

$$Pos_{BC-r}=[dx \quad dy \quad d\theta] \tag{4.17}$$

$$\begin{cases} xr = x_{BC} + dx\cos(\theta_{BC}) + dy\sin(\theta_{BC}) - d_1\sin(\theta_{BC}) - d_2\cos(\theta_{BC}) \\ y_r = y_{BC} + dx\sin(\theta_{BC}) + dy\cos(\theta_{BC}) + d_1\cos(\theta_{BC}) - d_2\sin(\theta_{BC}) \\ \theta_r = \theta_{BC} + 2\pi - d\theta \end{cases} \tag{4.18}$$

Considerando, não as medidas enviadas pelo programa de localização códigos de barras, mas sim as projecções dessas medidas no referencial do mundo e para o centro do robot, fica-se com as seguintes equações:

$$\begin{cases} x_r = x_{BC} + dx' \\ y_r = y_{BC} + dy' \\ \theta_r = \theta_{BC} + d\theta' \end{cases} \Leftrightarrow \begin{cases} dx' = x_r - x_{BC} \\ dy' = y_r - y_{BC} \\ d\theta' = \theta_r - \theta_{BC} \end{cases} \quad (4.19)$$

onde:

$$\begin{cases} dx' = dx\cos(\theta_{BC}) + dy\sin(\theta_{BC}) - d_1\sin(\theta_{BC}) - d_2\cos(\theta_{BC}) \\ dy' = dx\sin(\theta_{BC}) + dy\cos(\theta_{BC}) + d_1\cos(\theta_{BC}) - d_2\sin(\theta_{BC}) \\ d\theta' = 2\pi - d\theta \end{cases} \quad (4.20)$$

Tendo em consideração as equações das medidas apresentadas em 4.19 é necessário agora calcular o jacobiano da medida:

$$H^{BC} = \frac{d(dx', dy', d\theta')}{d(x_r, y_r, \theta_r)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.21)$$

Em termos do modelo do erro da medida, aproximado a uma distribuição gaussiana considerando os erros não correlacionados entre si, é dado pelas variâncias das medidas:

$$R^{BC} = \begin{bmatrix} \sigma^2_{dx'} & 0 & 0 \\ 0 & \sigma^2_{dy'} & 0 \\ 0 & 0 & \sigma^2_{d\theta'} \end{bmatrix} \quad (4.22)$$

Considera-se que os erros da medida, obtidos pela câmara no seu referencial, são os mesmos que quando referenciados ao centro do robot no referencial do mundo, havendo somente a necessidade de realizar a rotação entre os referenciais. Sendo assim:

$$V \,.\, R_{BC} \,.\, V^T = \begin{bmatrix} \sigma^2_{dx'}\cos(\theta_{BC}) & \sigma^2_{dy'}\sin(\theta_{BC}) & 0 \\ \sigma^2_{dx'}\sin(\theta_{BC}) & \sigma^2_{dy'}\cos(\theta_{BC}) & 0 \\ 0 & 0 & \sigma^2_{d\theta'} \end{bmatrix} \quad (4.23)$$

Na implementação do EKF no robot, inclui-se um teste de verosimilhança para as medidas, utilizando para esse facto as informações adicionais obtidas pelo programa de leitura de códigos de barras. Sendo assim, só são consideradas medidas válidas, se o número de linhas processadas for superior a cinco.

4.5 Filtro de Partículas

A implementação do filtro de partículas neste trabalho foi divida em diversas secções para uma maior comodidade de implementação e de compreensão. A descrição do algoritmo encontra-se na figura 4.17.

A população inicial é um procedimento utilizado apenas no arranque do programa de controlo. Esta função tem duas características associadas, dado que o filtro de partículas não requer uma pose inicial definida. Pode-se aleatoriamente espalhar partículas pelo mundo tendo apenas em consideração as restrições do mapa. O problema de localização torna-se global.

Por outro lado pode-se definir uma pose inicial, por exemplo sabendo que o robot parte sempre de um ponto predefinido. Neste caso o problema da localização é apenas de seguimento de posição.

Sabendo à partida a pose inicial, pode-se atribuir a esta, uma certa percentagem de partículas, garantindo assim que a pose estimada se localiza naquele local.

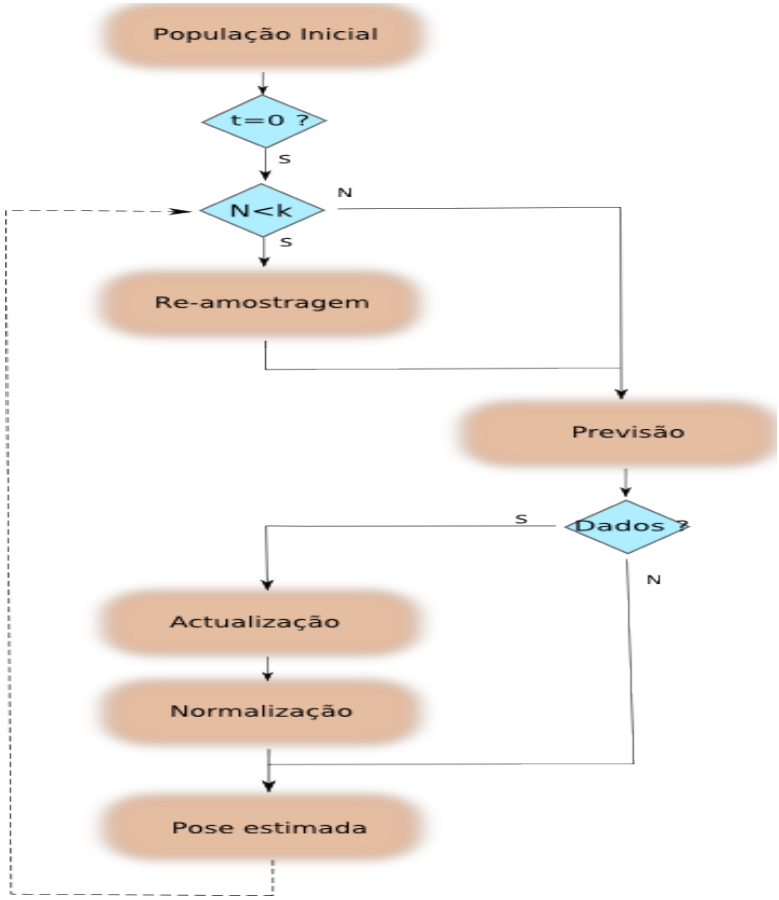


Figura 4.17 Descrição do algoritmo do Filtro de Partículas

A re-amostragem já foi explicada anteriormente e só acontece quando o número de partículas com peso reduzido é muito elevado.

A fase de previsão foi também já referida e explicada. Quando existem dados dos sensores efectua-se actualização. Depois normalizam-se os pesos das partículas como descrito na equação 3.76.

Segue-se a estimação da pose do robot por um dos métodos anunciados na página 30.

Surge a necessidade de explicitar como se efectua a actualização do PF com os sensores disponíveis no robot.

Como se aproximou o modelo dos sensores a uma distribuição gaussiana fica:

$$W(Observação, X_k^i) = p(x_k | x_k^i; z_k) = \frac{1}{\sigma \sqrt{2\pi}} * \exp\left(-\frac{(X_k^i - u)^2}{2\sigma^2}\right) \tag{4.24}$$

Afectam-se os pesos segundo esta probabilidade de a medida calculada se encontrar na distribuição da medida recebida.

$$w_k^i = w_{k-1}^i * W(Observação, X_k^i) \tag{4.25}$$

E normalizam-se os pesos:

$$w_k^i = \frac{w_k^i}{\sum_{i=1}^{M} w_k^i} \tag{4.26}$$

garantindo-se a legitimidade da equação 3.76. Define-se assim o algoritmo de filtro de partículas utilizado.

Acrescenta-se ainda que ao longo do projecto, foram analisados vários resultados que levaram a concluir que só os conjuntos de sensores laterais não são suficientes para obter um resultado interessante com o PF.

Como solução decidiu-se colocar um *sharp* de longo alcance, cerca de 5 metros, na frente do robot, de forma a conseguir obter mais informação que possa dissipar ambiguidades no mapa.

Capítulo 5

Validação Experimental

5.1 Introdução

Nesta secção são apresentados diversos resultados experimentais referentes essencialmente à questão da localização.

Inicia-se como a descrição dos tempos de processamento envolvidos no programa de controlo.

Segue a apresentação de vários ensaios efectuados para validar e testar o Filtro de Kalman Extendido e o Filtro de Partículas. Nestes ensaios comparam-se as estimativas dos filtros com as poses reais medidas. O método de medição da pose real é também descrito neste capítulo.

5.2 Arquitectura

Apresentam-se os tempos de processamento das acções mais importantes do programa de controlo. Os valores descritos são referentes ao pior caso.

Dentro do ciclo de controlo, são chamadas diversas funções, sendo as mais importantes: as de localização, a apresentação de dados ao utilizador, a escrita dos dados em *log* e o processamento das medidas dos sensores.

Tabela 5.1 Tempos referentes a cada acção de controlo

Acção	Tempo (ms)
Ciclo de Controlo	~ 100
Processar sensores	~ 2
Filtro de Kalman	~ 1
Filtro de Partículas	~ 24
Guardar dados em Log	~ 18
Apresentação de dados no ecrã	~ 14
Outras acções menores	~ 2

Como se observa a soma de todas as acções que ocorrem durante o ciclo de controlo, é cerca de 60 *ms*, que é menor que o tempo máximo disponível para cada ciclo, 100*ms*. Este facto é propositado e tem como base a explicação dada na página 45. Ou seja deixar uma folga para que o sistema operativo possa tratar de outros eventos em espera.

Como se verifica o filtro de partículas ocupa muito mais tempo de processamento que o filtro de Kalman. Esse peso computacional durante muito anos fez deste algoritmo inapropriado para sistemas em tempo real. Actualmente é perfeitamente possível usa-lo nesses sistemas, como é exemplo este trabalho.

Segue-se a indicação dos intervalos de tempo entre o pedido de medidas aos sensores e a resposta dos mesmos.

Tabela 5.2 Tempos referentes às respostas dos sensores

Sensor	Tempo (ms)
Sonar	~ 60
Sharp	~ 9

Os *sharps* são extremamente mais rápidos a responder a pedidos, muito devido ao seu tipo de saída e ao seu método de medição. Os sonares demoram bastante mais tempo, pois é necessário sincronizar os pedidos aos vários sonares de forma a evitar erros como o *cross-talk*, em que um sonar recebe o eco de outro.

5.3 Medição da pose real

Criou-se uma função no programa de controlo semelhante a um cronómetro. Sempre que é ultrapassado um determinado intervalo de tempo definido pelo utilizador, o robot pára. É necessário depois activar um botão colocado no robot, para reiniciar o cronómetro e permitir que o robot entre em movimento.

Durante as pausa do robot, mede-se a posição cartesiana (x, y) do ponto de contacto de cada roda motriz.

Com este sistema garante-se que os pontos medidos não são seleccionados pelo utilizador, o que revela uma maior imparcialidade nas poses reais levantadas.

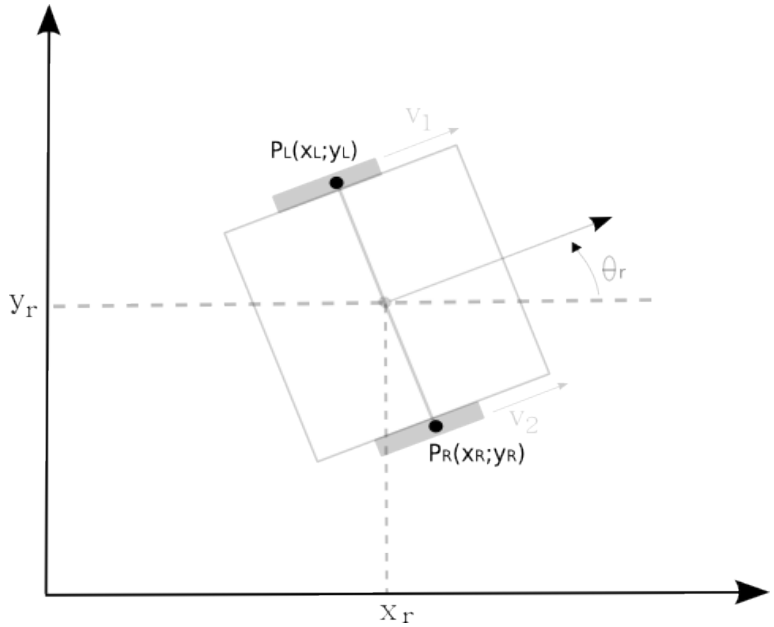


Figura 5.1 Representação da medição da pose real

Como método para prevenir erros humanos nas medições, usam-se apenas os pontos, se a distância calculada entre rodas, for próxima do seu valor real.

Sendo $p_{\text{Left wheel}}=(x_L;y_L)$ e $p_{\text{Right wheel}}=(x_R;y_R)$, as posições cartesianas das rodas esquerda e direita respectivamente. Então:

$$p_r=\left(\frac{(x_L+x_R)}{2};\frac{(y_L+y_R)}{2}\right) \tag{5.1}$$

é a posição cartesiana do robot.

A orientação é calculada da seguinte forma:

$$\theta_r=\arctan\left(\frac{(y_R-y_L)}{(x_R-x_L)}\right) \tag{5.2}$$

Através da repetição da experiência da medida da pose real do robot, é possível caracterizar o pior erro deste método de medição da pose real do robot como sendo na ordem dos 4 cm para x e y e de 7 graus sexagesimais para o ângulo obtido.

5.4 Filtro de Kalman Extendido

5.4.1 Modelo de Previsão

Como já foi referido usou-se como modelo de previsão o modelo da odometria. Segue-se uma figura com a evolução das covariâncias ao longo do tempo.

A imagem apresentada é resultado da indicação dos dados em intervalos de 20 amostras do ciclo de controlo.

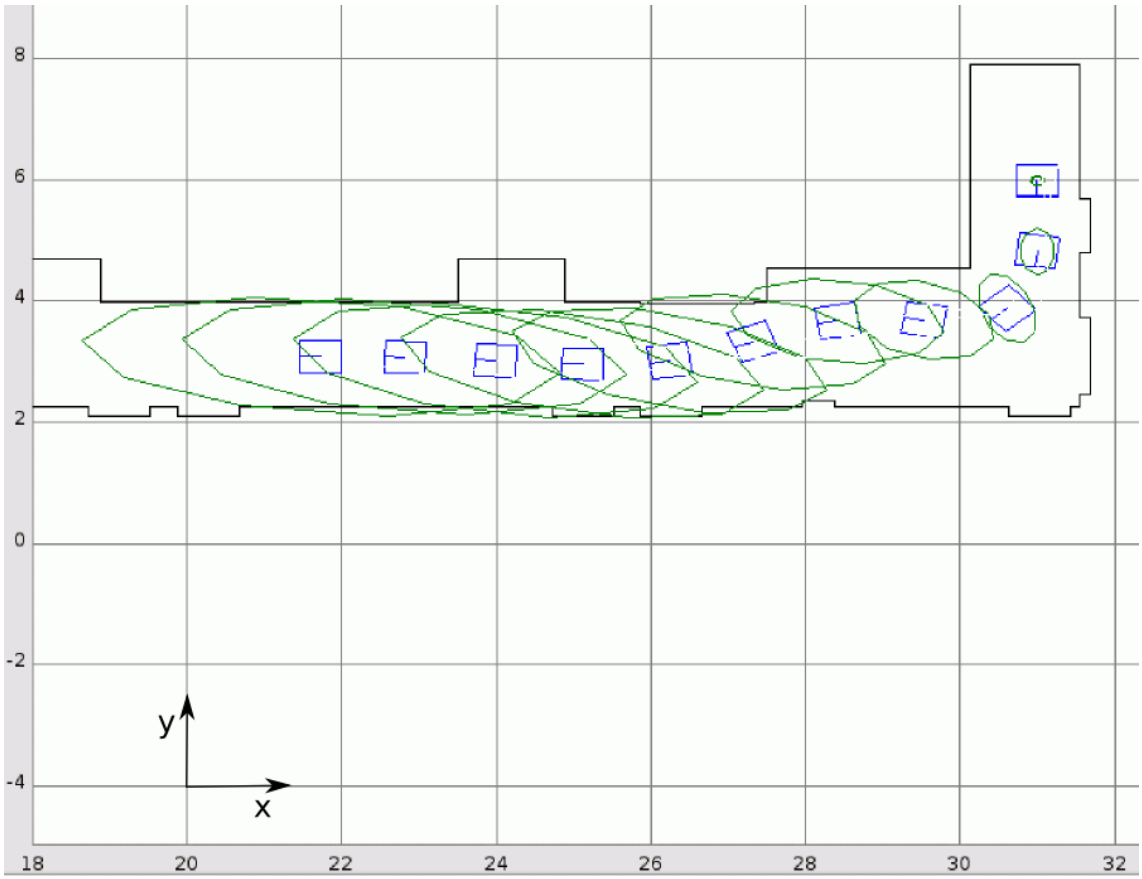


Figura 5.2 EKF modelo de Previsão

Decidiu-se representar apenas a covariância nas componentes *x* e *y*, de forma a manter uma imagem de duas dimensões. Como é óbvio a representação da matriz *P* deveria ser uma elipsóide de 3 dimensões, representando as covariâncias em *x* , *y* e θ .

Aproximou-se as duas covariâncias a um polígono de dez pontos tentando de certa forma representar uma elipse.

Representa-se na imagem o sistema de eixos usados, este sistema manter-se-á ao longo de todas as imagens usadas neste trabalho, desta forma, para evitar sobrecarga de indicações nas figuras, e também diminuir o tratamento efectuado à imagem, mantendo assim a sua

integridade.

Na figura 5.2, observa-se que as covariâncias aumentam ao longo do tempo e de acordo com o movimento do robot. Ou seja quando o robot se movimenta num determinado eixo, a covariância aumenta mais rapidamente sobre a componente referente a esse eixo. Tal facto é compreensível, pois quanto maior for o deslocamento numa determinada componente maior será a incerteza sobre ela.

Nota-se também que durante a rotação a incerteza aumenta aproximadamente igual nas duas componentes.

5.4.2 Localização por linhas

Neste caso foram usados os dois pares de *sharps* nas laterais do robot em fusão com os dados da odometria.

Segue-se uma descrição de toda a experiência que se baseou no método de medição da pose real descrito acima. A azul representa a pose estimada pelo EKF, e a vermelho a pose real nos locais onde foi feita a medição. As distâncias medidas pelos *sharps* são apresentadas a preto e o ângulo calculado por este método é representado a verde-claro.

Na figura 5.3 é possível observar que a pose inicial do EKF tem algum erro em relação à pose inicial real.

Figura 5.3 EKF localização por linhas usando *sharps* (1)

Verifica-se também que o erro em relação às várias poses reais é reduzido. Nota-se ainda que quando o EKF recebe dados úteis dos sharps a sua estimativa é mais próxima da realidade.

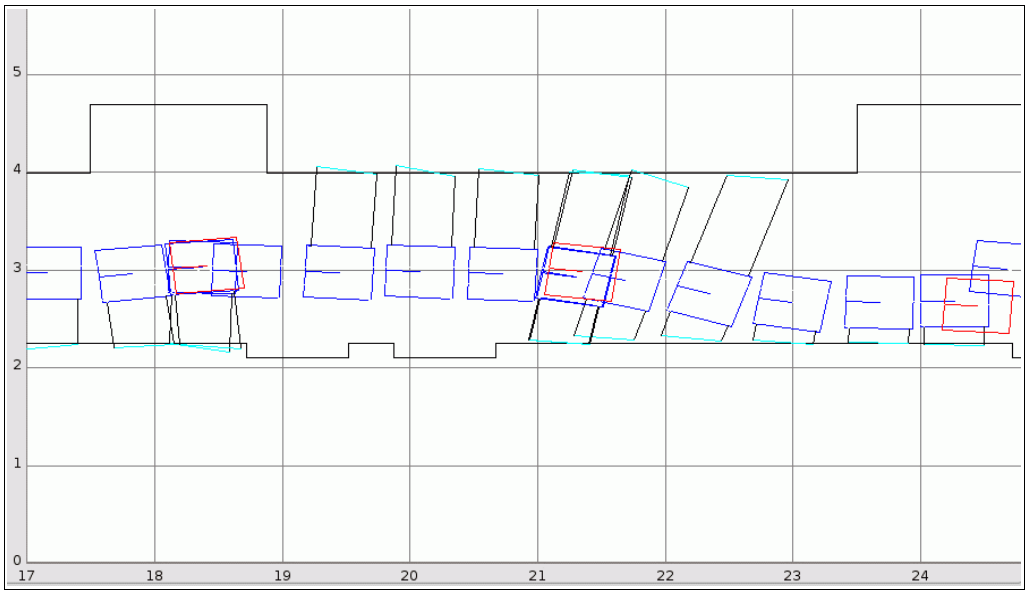


Figura 5.4 EKF localização por linhas usando sharps (2)

Na continuação da observação, nota-se na figura 5.4, que o erro na componente *x* é maior que o erro na componente *y* . Este acontecimento deve-se ao facto de os sharps, nesta zona, apenas fornecerem medidas referentes à componente *y*, que são onde se situam as linhas/paredes de referência. Ou seja neste local a componente *x* é não observável, sendo por isso normal que o erro nesta componente cresça.

É necessário referir que a importância das medidas dos sensores é notória entre as imagens 5.3 e 5.4. Se observar a figura 5.3, a partir de $x=26$, não são utilizados os dados dos sensores. Na figura 5.4, perto de $x=24$, nota-se a grande diferença entre o erro da estimativa quando apenas foi realizada a previsão e seguidamente quando acontece a actualização com os dados dos sensores, o erro diminui substancialmente.

Figura 5.5 EKF localização por linhas usando *sharps* (3)

Na figura 5.5, apresentam-se as ultimas três poses medidas. Mantém-se o erro na componente x , e é possível notar a forma como o EKF actualiza a sua pose de acordo com os dados recebidos pelos sensores. É visível que o ângulo e as distâncias calculadas através dos dados dos sharps são coincidentes com as paredes/linhas do corredor.

Em seguida apresenta-se o erro entre a estimativa e a pose real, em valor absoluto nas componentes cartesianas x e y .

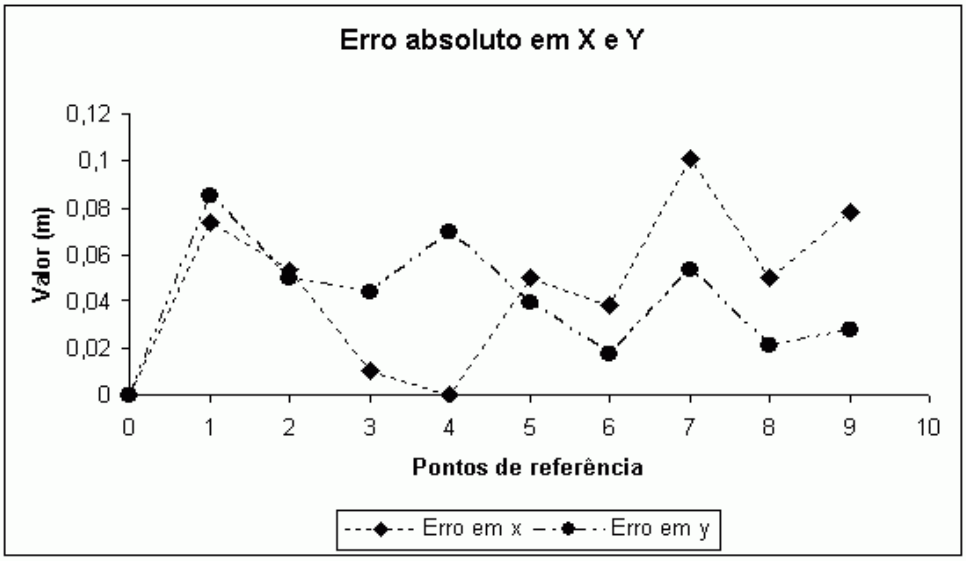


Figura 5.6 EKF erros relativos à localização por linhas.

Neste gráfico observa-se que o erro em *y* diminuiu ao longo do percurso, o que é compreensível, pois nos primeiros pontos não existem paredes de referência nesta componente. A partir do 3 ponto surgem as linhas de referência e por sua vez o erro diminuiu.

Com o erro em *x* acontece o contrário, pois as paredes de referência encontram-se nos pontos iniciais. Depois ao longo do percurso torna-se numa componente não observável pelos sensores, o que leva ao crescimento do seu erro.

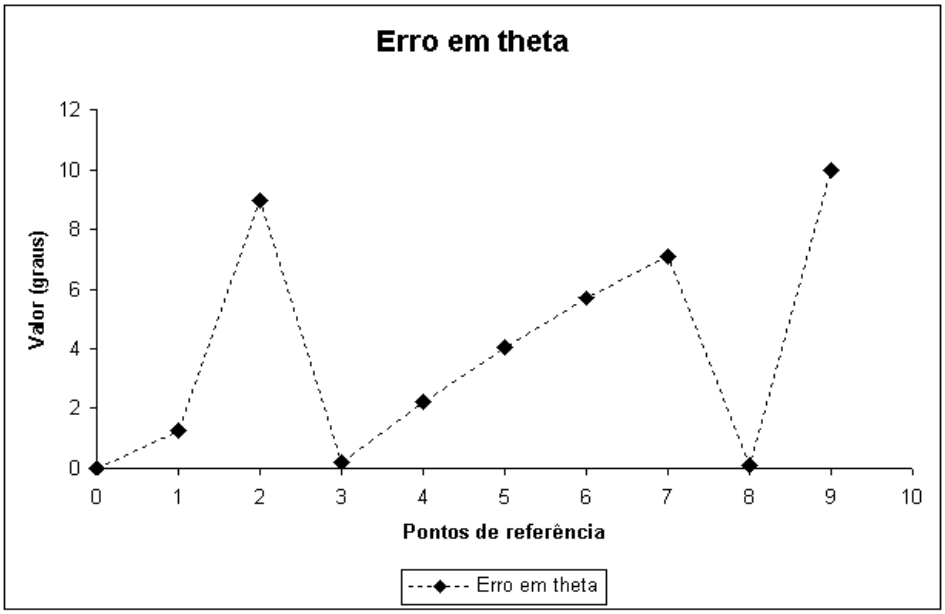


Figura 5.7 EKF erro da orientação relativo à localização por linhas.

É possível verificar que o erro na orientação é sempre menor que 10º, e que existem pontos em que o erro é muito diminuto.

Algumas palavras acerca da confiança na estimativa. Como falado anteriormente a matriz da covariância 3.59, é inversamente proporcional à confiança na pose estimada.

Na figura 5.8, observa-se a evolução da covariância nas componentes *x* e *y*. Como é possível observar a covariância aumenta na componente em que existe movimento. Este facto deve-se ao modelo da odometria.

Quando existem medidas úteis dos sensores, actualiza-se a covariância, e esta diminui nas componentes que foram observadas, de acordo com a confiança das medidas. Nesta figura 5.8, é notória a diminuição da covariância na componente *y*, quando existem medidas. Na componente *x* o mesmo não acontece pois não existem medidas acerca desta componente.

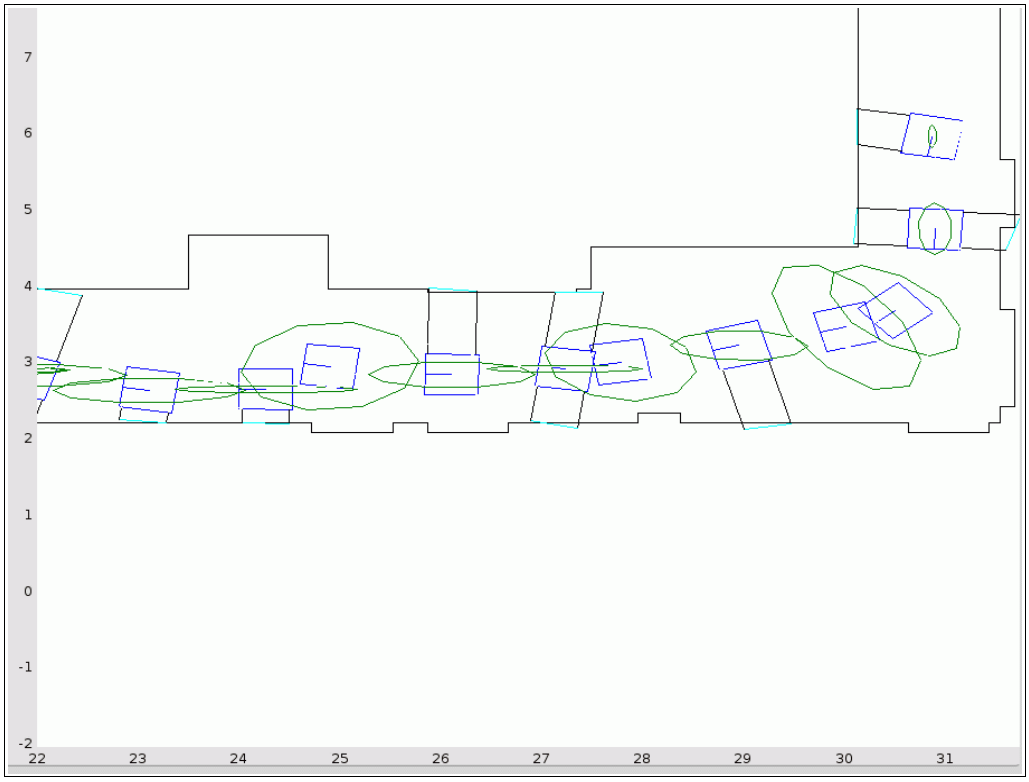


Figura 5.8 EKF localização por linhas, evolução das covariâncias (1)

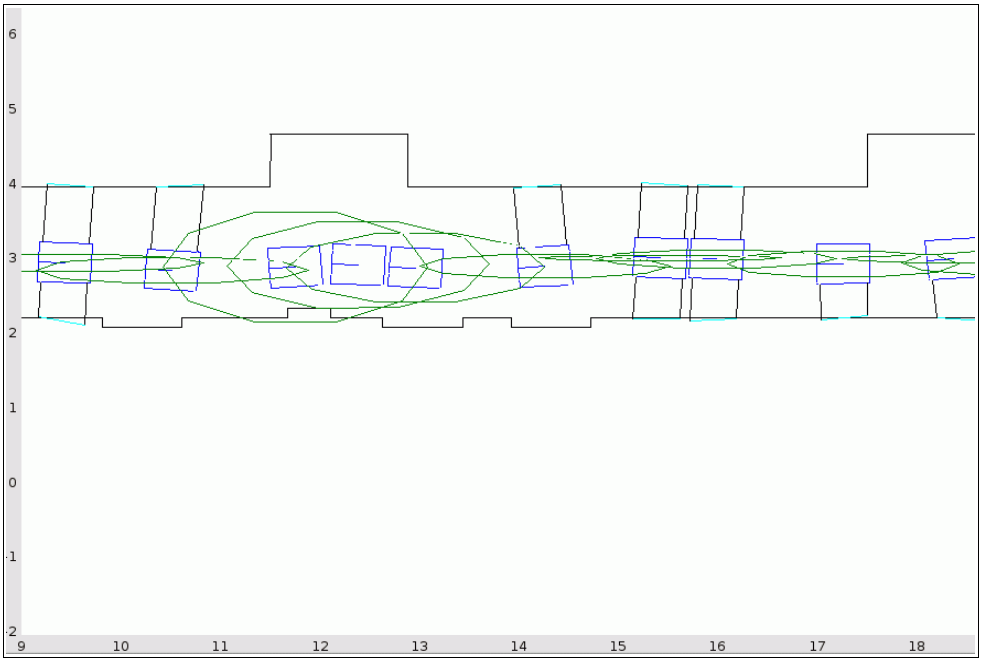


Figura 5.9 EKF localização por linhas, evolução das covariâncias (2)

Na imagem 5.9, nota-se muito bem as zonas onde existe muito confiança na estimativa devido às actualizações com os dados dos sensores e os locais onde a incerteza aumenta, por não haver dados disponíveis.

5.4.3 Localização por Códigos de Barras

Os códigos de barras funcionam como marcadores, apenas fornecem informação quando são avistados pela câmara, o que só acontece quando o robot se encontra nas imediações do código de barras.

Para comprovar experimentalmente a localização por códigos de barras optou-se por usar simultaneamente a localização por linhas. Desta forma testa-se a qualidade da estimativa do EKF usando todos os métodos descritos neste trabalho.

Os códigos de barras são representados nas imagens seguintes como pontos no mapa. Estes pontos encontram-se sobre as paredes pois estão colados a elas no corredor real. Cada ponto representa o canto superior esquerdo do código de barras.

A visualização do código de barras efectuada pela câmara é apresentada como um "L" de cor castanha, em que o segmento maior representa a distância do centro da câmara ao canto superior esquerdo do código de barras e o segmento menor representa a largura do código de barras, sendo este segmento constante. O ângulo entre os dois segmentos é proporcional ao ângulo fornecido pela câmara.

Em resumo, numa situação perfeita, sem ruídos ou erros, o segmento maior do "L" ligaria o robot ao ponto representativo do código de barras. Se o ângulo medido pela câmara estiver correcto, o segmento menor do "L" ficará sobre a parede onde se encontra o código.

Refere-se mais uma vez que as imagens consistem em sobreposições da pose do robot em intervalos de 20ciclos de controlo. Por este facto nem sempre é possível visualizar o "L" representativo da visualização do código de barras, apesar do mesmo ter sucedido.

Mais uma vez a posição real medida é representada a vermelho, a pose estimada a azul.

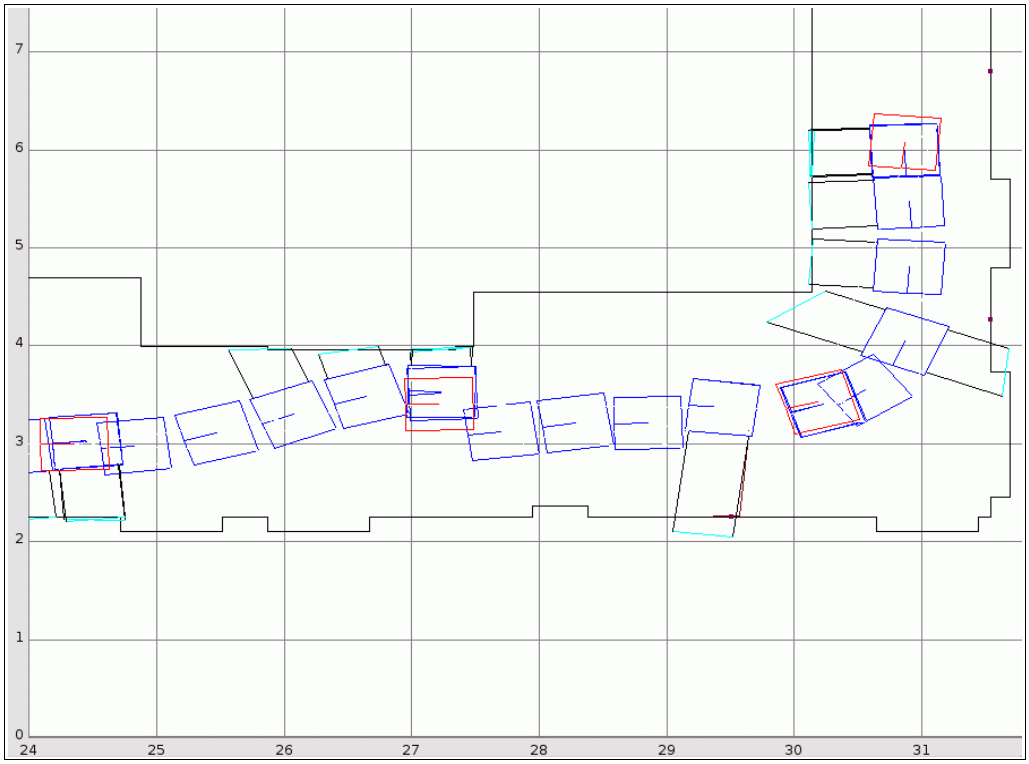


Figura 5.10 EKF localização por CB e por linhas (1)

Na figura 5.10, verifica-se que o algoritmo se inicia com uma pose ligeiramente diferente da real. Apesar desse facto nota-se que o erro em relação às poses reais é menor que usando apenas a localização por linhas.

É de assinalar que entre os 29 e os 30 metros na componente x, observa-se uma representação da visualização do código de barras, que tem um erro bastante baixo. É também notório o efeito do erro do modelo de previsão, pois entre $x=29$ e $x=28$, o algoritmo não recebe nenhuma medida válida que permita efectuar a actualização. Desta forma o erro na estimativa cresce tanto, que quando o algoritmo recebe as primeiras medidas dos *sharps* em $x=27$, o erro entre essa pose e a anterior é de quase 0.5 metros.

Na figura 5.11, observam-se duas visualizações do códigos de barras, perto de $x=17$ e $x=23$, nota-se que existe algum erro entre o ponto que representa o BC e a distância fornecida pela câmara. Por outro lado o ângulo indicado é quase coincidente com a parede.

Ao longo do corredor verifica-se que o robot segue aproximadamente uma linha recta perto de $y=3$, que é a referência dada ao sistema de navegação do robot. A fusão entre o método de localização por linhas e a localização por códigos de barras parece ter diminuindo os erros na orientação.

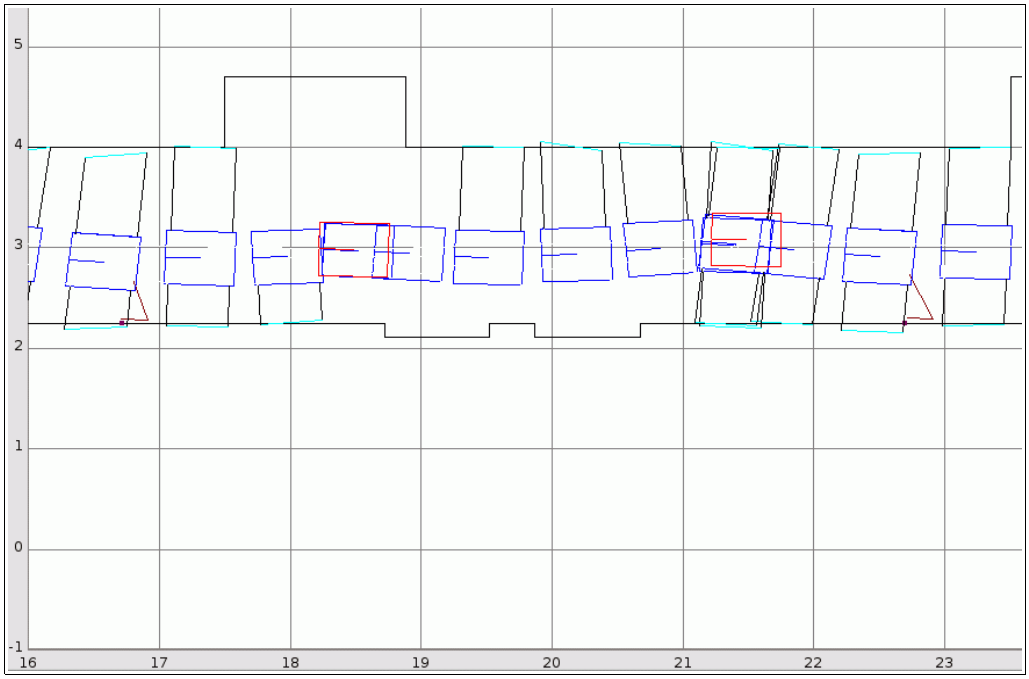


Figura 5.11 EKF localização por CB e por linhas (2)

Na figura 5.12, observa-se a continuação do ensaio e o erro na ultima posição real medida é muito baixo em comparação com a localização por linhas. Este facto deve-se à visualização do código de barras perto de $x=11$, que permite uma medida para as três variáveis que constituem a pose.

Avalia-se também que o erro na posição de referência 5 situada entre $x=12$ e $x=13$ é grande em comparação com os outros, pois não existem medidas válidas nos instantes anteriores, o que leva ao aumento do erro provocado pelo modelo da previsão.

Figura 5.12 EKF localização por CB e por linhas (3)

Em seguida apresentam-se os erros, nas componentes cartesianas *x* e *y*, relativos à diferença entre a pose real medida e a pose estimada.

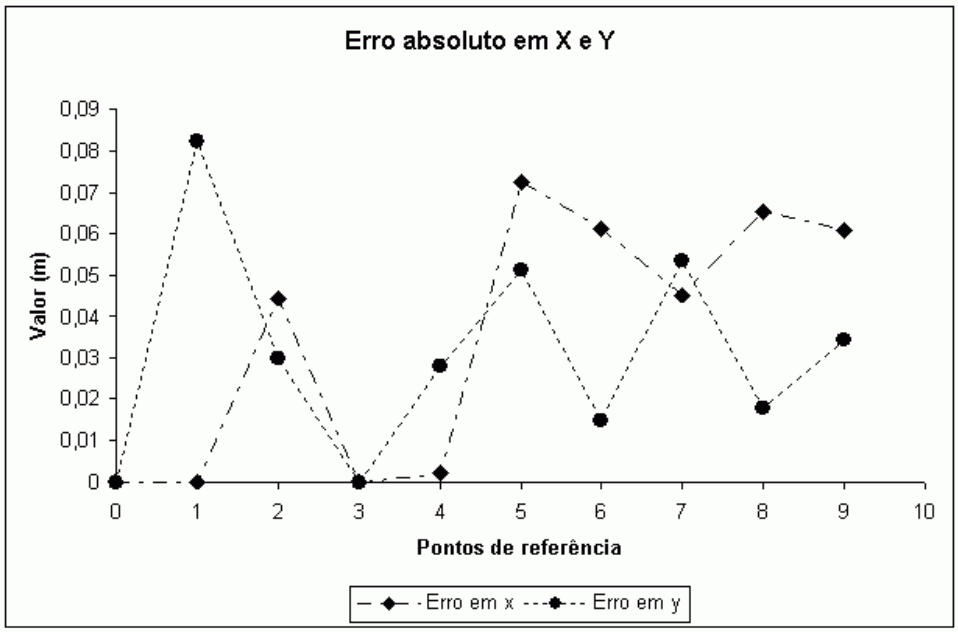


Figura 5.13 EKF erros relativos à localização por CB e por linhas

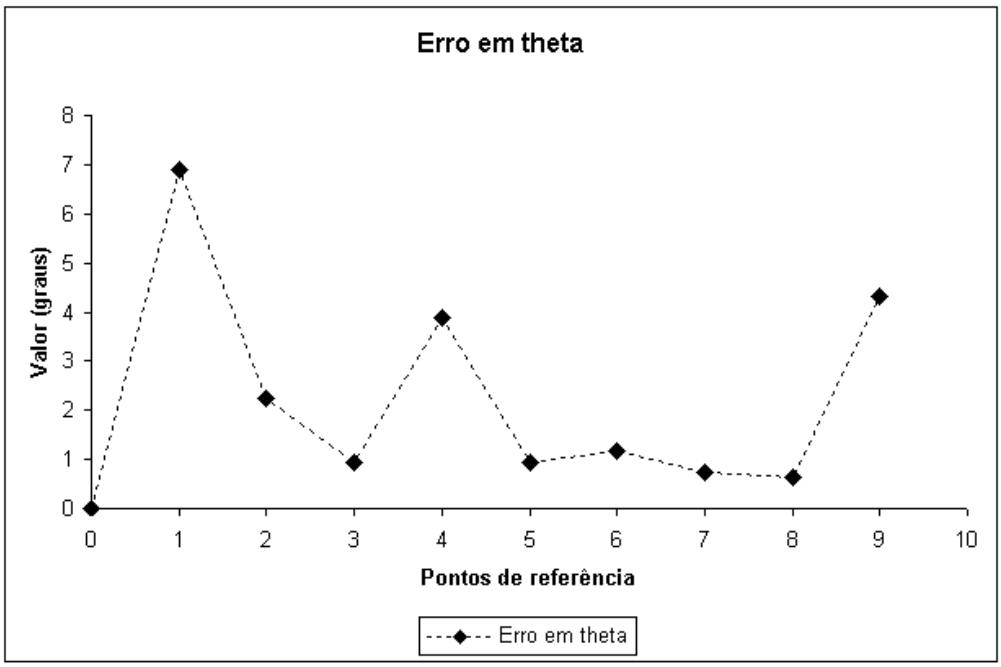


Figura 5.14 EKF erro da orientação relativo à localização por linhas e por BC

Neste gráfico observa-se que os erros são menores que os apresentados na localização por linhas. O erro em *y* diminui ao longo do percurso, o que é compreensível, e já foi explicado anteriormente. A partir do 2º ponto surgem códigos de barras e linhas de referência que levam o erro a diminuir. O valor do erro, com excepção do 1º ponto, situa-se entre 5 e 2 cm, o que é um resultado interessante.

Com o erro em *x* acontece o contrário, pois as paredes de referência encontram-se nos pontos iniciais. Depois ao longo do percurso, com a ajuda dos códigos de barras, o erro nesta componente situa-se abaixo de 7cm.

O erro na orientação também diminuiu quando comparado com o ensaio anterior, como se pode observar na figura 5.14. Este facto demonstra que a fusão entre a localização por BC e com a localização por linhas produz melhores resultados que só com um método.

Segue-se a analise da confiança da estimativa ao longo do ensaio.

Na imagem 5.15, nota-se claramente a diminuição das covariâncias após a visualização do código de barras situado na vizinhança de $x=29$. Isto deve-se ao facto da câmara fornecer medidas relativas às três variáveis *x*, *y* e θ do robot, o que leva ao aumento de confiança nessas três componentes e por sua vez à diminuição de cada covariância, visto a confiança e a covariância serem características inversamente proporcionais.

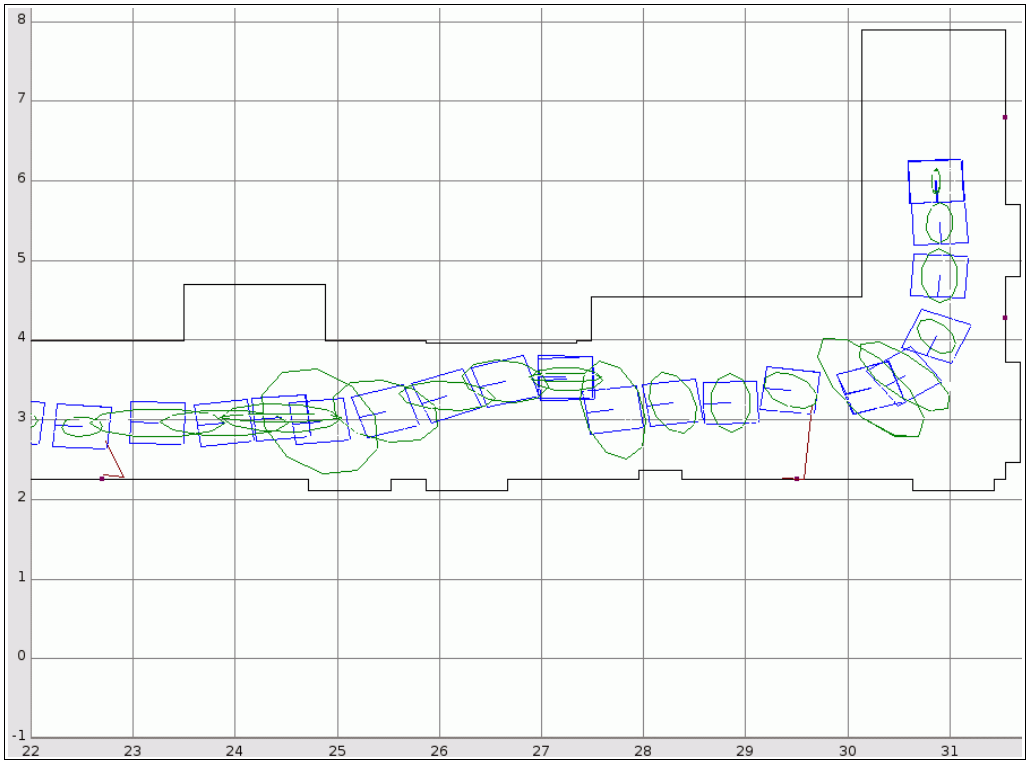


Figura 5.15 EKF localização por CB e por linhas, evolução das covariâncias (1)

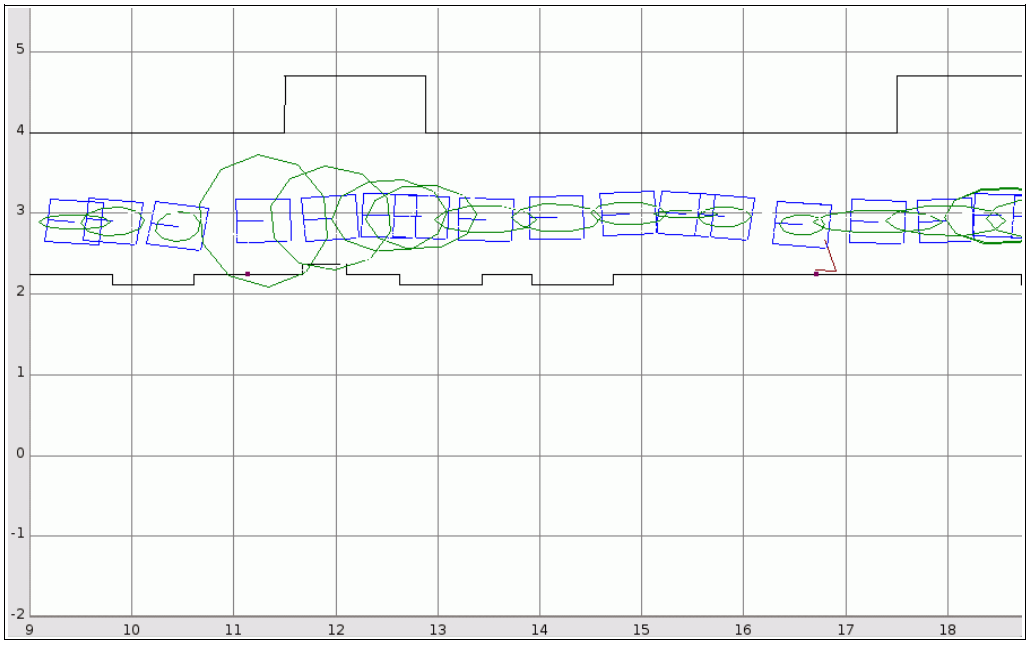


Figura 5.16 EKF localização por CB e por linhas, evolução das covariâncias (2)

Na figura 5.16, é possível verificar a forma diferente como as medidas dos *sharps* e da câmara afectam as covariâncias. Enquanto as medidas dos *sharps* são referentes apenas à

componente perpendicular às paredes/linhas de referência, e assim só aumentam a confiança nessa componente, a câmara, por sua vez, fornece dados sobre todas as componentes.

Esta afirmação é visível da seguinte forma: se se observar a covariância num instante em que o robot visualiza o código de barras, como por exemplo em $x=17$, nota-se que esta tem a forma de um circulo, ou seja diminuiu em todas as variáveis. Nos locais onde apenas se recebem medidas dos *sharps,* a covariância tem a forma de uma elipse, em que o eixo menor se situa sobre a componente observada.

5.4.4 Localização Global

Para testar a localização global, colocou-se o robot parado numa posição do corredor nas imediações de um código de barras. Iniciou-se o EKF com uma pose inicial errada.

Neste ensaio específico, a pose inicial do filtro e a pose real do robot têm mais de 6 metros de diferença, e uma alteração na orientação de cerca de 90º. Um EKF normal não conseguiria recuperar deste erro, mas com a utilização dos códigos de barras como marcadores, é possível enfrentar este problema.

Como vem sendo usual, a vermelho representa-se a pose real do robot, a azul a estimativa do algoritmo. Neste teste apenas será usada a localização por códigos de barras.

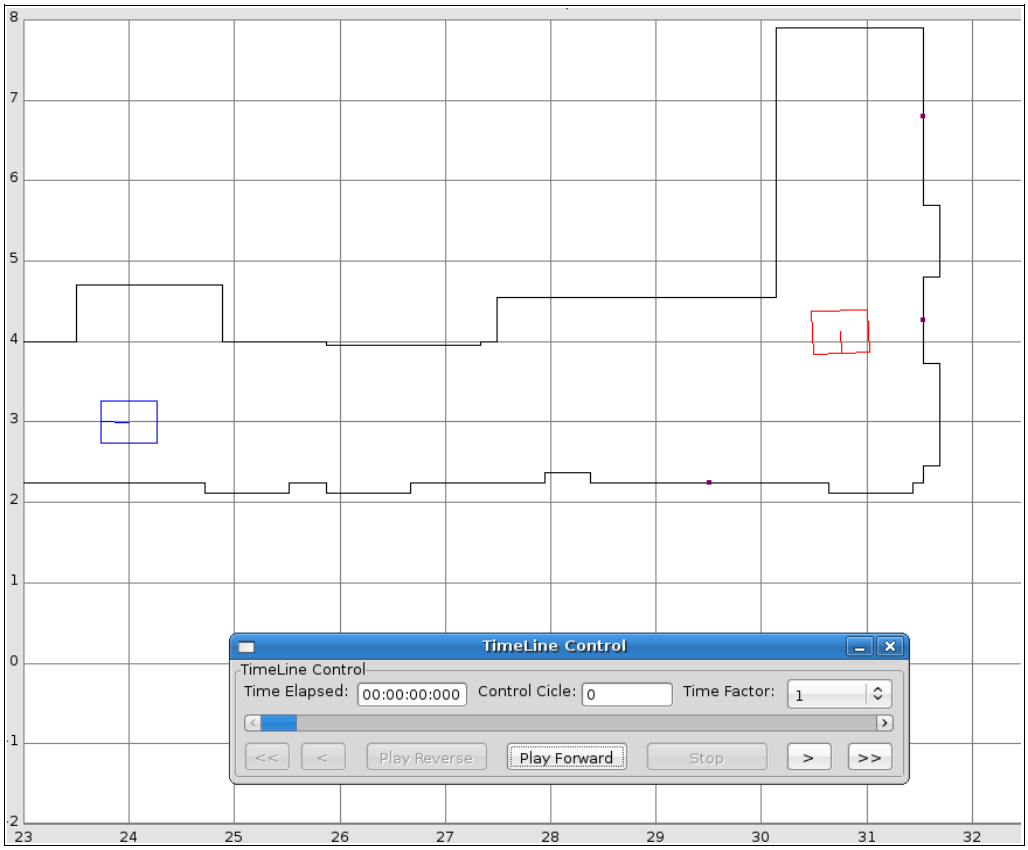


Figura 5.17 EKF localização global usando CB (1).

Na figura 5.17 observa-se a descrição efectuada anteriormente. O EKF inicia-se com uma pose errada, verificam-se claramente as diferenças entre pose real e estimada.

Na figura 5.18 nota-se que o filtro está a convergir para a pose real, esta imagem especifica é referente ao instante $t=10,1$ segundos. A diferença entre as poses é já menor que um metro.

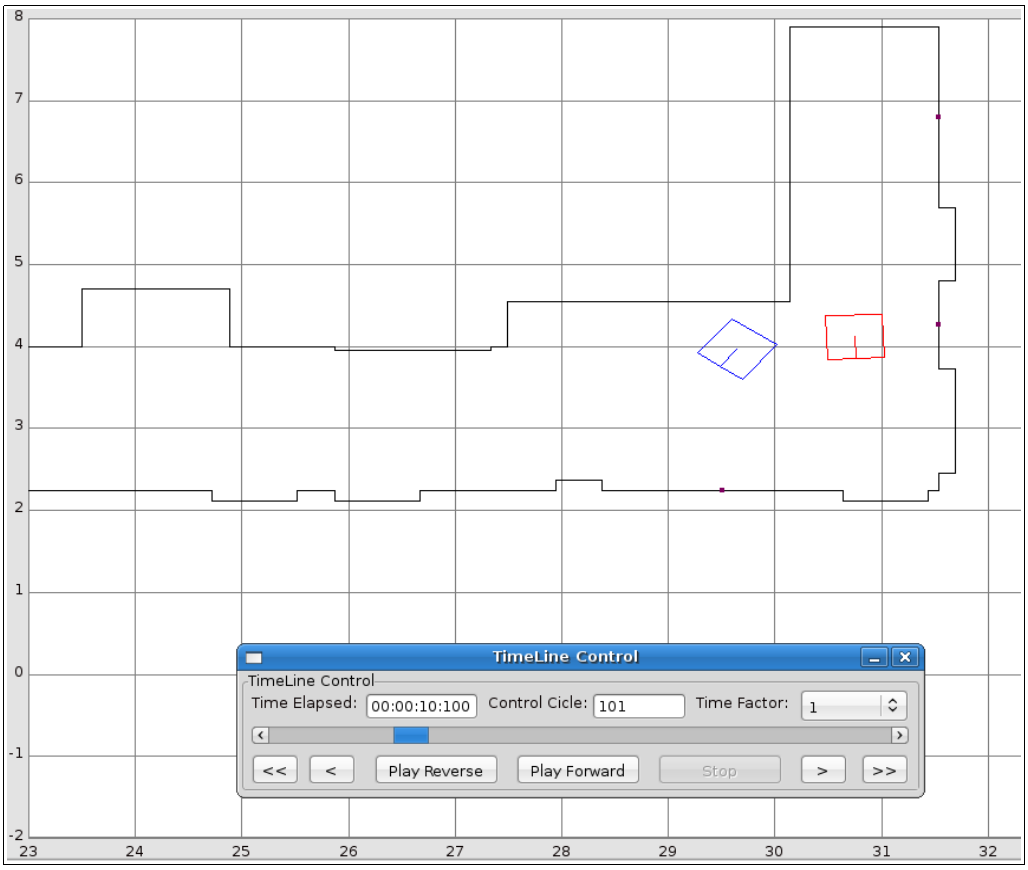


Figura 5.18 EKF localização global usando CB (2).

Por último, na imagem 5.19, verifica-se que a estimativa do filtro se encontra muito perto da pose real, realçando-se apenas algum erro na orientação. Este resultado não é surpreendente, visto que os códigos de barras são marcadores que fornecem medidas, para a posição e orientação, com boa qualidade. Apesar que este tipo de marcadores apresenta por vezes algum erro na medição do ângulo, o que provoca alguma incorrecção na orientação do robot. Resta acrescentar que esta figura é referente ao instante $t=25$ segundos.

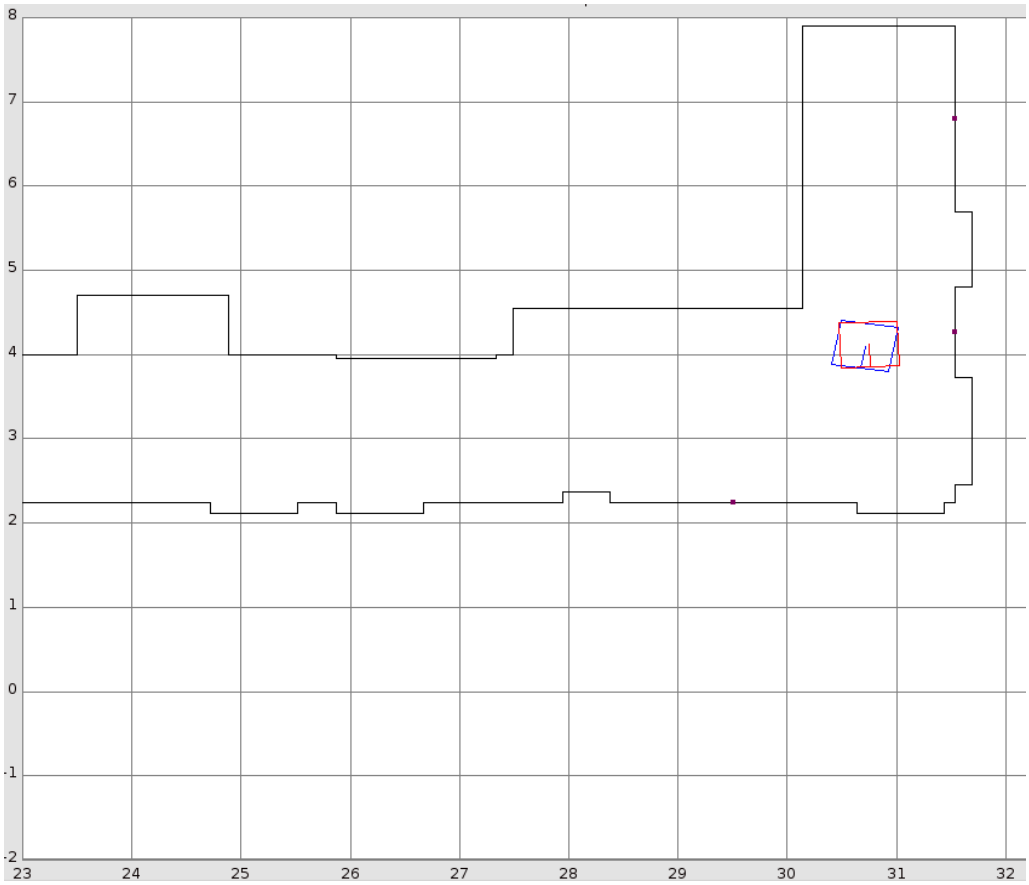


Figura 5.19 EKF localização global usando CB (3).

Na figura 5.20, apresenta-se a evolução das covariâncias neste ensaio. Quando o algoritmo começa, a sua confiança na pose é igual à confiança noutros ensaios, em que a pose real era próxima da inicial. Ou seja o filtro inicia-se com a convicção de que se situa na pose correcta, quando na realidade se encontra muito longe disso. Este facto quase que torna este problema de localização global, num problema de rapto, pois o filtro não sabe que o conhecimento que tem desta posição está errado. Seguidamente com as várias actualizações efectuadas as covariâncias diminuem em todas as componentes da pose, ou seja existe cada vez mais confiança na pose estimada.

Outra característica interessante que se realça na imagem é a evolução das actualizações da pose. Inicialmente a actualização é muito grande, como se verifica entre a posição inicial (24;3) e a pose seguinte situada em $x=26{,}5$, depois as correcções na posição tornam-se cada vez menores, como se observa entre $x=29$ e a pose real. Este facto é muito característico do filtro de Kalman, e é provocado pelo ganho de Kalman 3.58, tal como foi explicado em 3.42 . Em resumo, quando a confiança num estado é muito elevada, o ganho de Kalman tem um peso menor no cálculo da actualização, o que provoca correcções mais ligeiras na pose. Dependendo da confiança nas medidas, quanto maior for a incerteza num estado maior será a correcção no estado estimado.

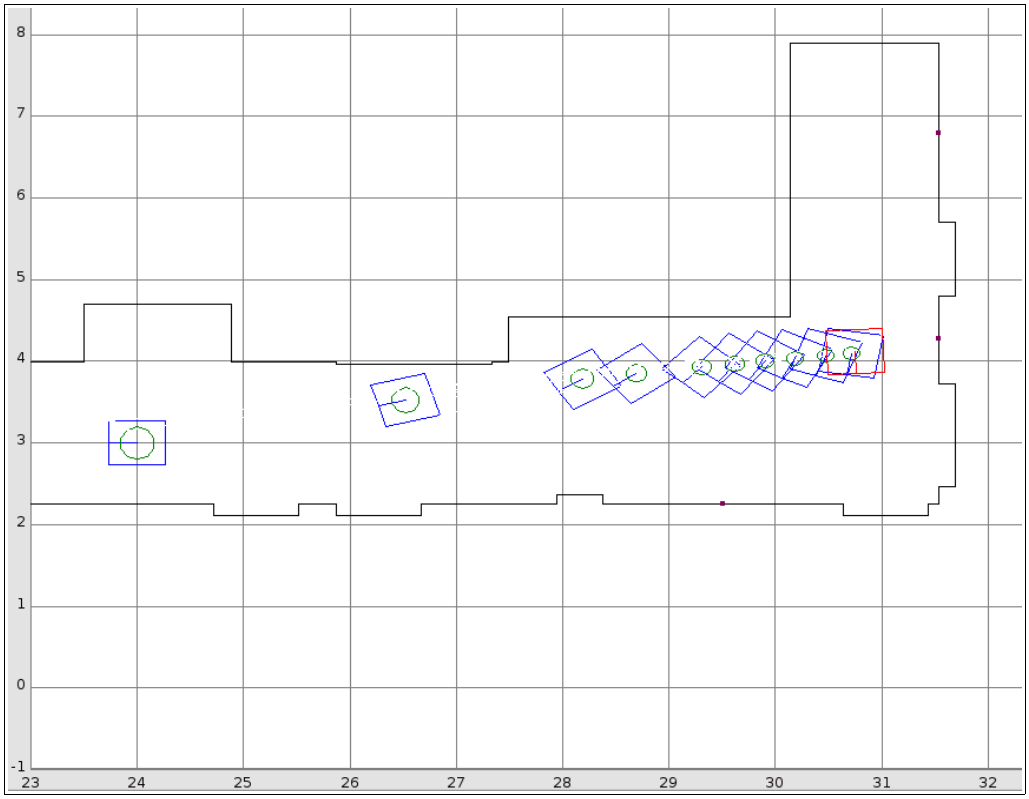


Figura 5.20 EKF localização global usando CB, representação das covariâncias.

Segue-se uma tabela com os valores absolutos dos erros nas várias componentes.

Tabela 5.3 EKF Erro na localização global

Erro em	Valor
X	0,07 (m)
Y	0,11 (m)
θ	7,57(°)

Como se verifica o erro nas componentes *x* e *y* é próximo de 10cm, o que é um valor bastante reduzido, se se tiver em conta que o filtro começou com uma pose completamente errada afastada mais de 6 metros da pose real. Em termos da orientação nota-se um erro um pouco mais elevado, mas compreensível, visto neste ensaio não se fazer uso dos sensores laterais. E também porque a medição do ângulo efectuada pela câmara é mais afectada por ruído.

5.5 Filtro de Partículas

5.5.1 Seguimento da posição do robot

5.5.1.1 Sharps

Neste primeiro ensaio de PF usou-se apenas a odometria e os grupos laterais de sharps, bem como o sharp frontal.

No instante inicial espalharam-se aleatoriamente partículas por todo o mapa útil do robot. Os pontos no mapa assinalam a pose das várias partículas, estão a ser usadas 900. Após várias iterações e com o robot parado, o PF convergiu para uma estimativa próxima da pose real representada a vermelho perto do ponto $(31;6)$. Esta convergência deve-se em parte ao *sharp* frontal que permite dissolver as ambiguidades possíveis se se usasse apenas os conjuntos laterais.

No entanto o erro na estimativa em relação ao 1º ponto de referência é já muito elevado, tal deve-se a dois motivos: o *sharp* frontal usado tem um erro muito elevado, como pode ser observado na tabela 4.6 . Segundo motivo: o filtro não tem qualquer conhecimento da pose inicial.

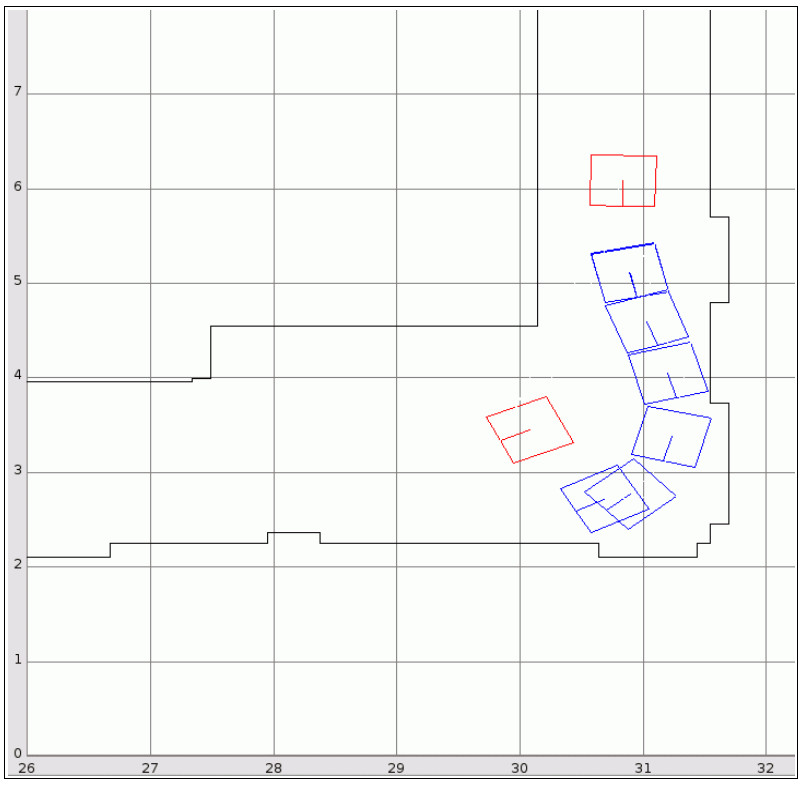


Figura 5.21 PF localização usando *sharps* (1)

Quando se inicia o movimento, a estimativa do PF é afectada pelo seu modelo de previsão

com os dados de odometria, sendo que o erro entre a 2ª pose real e a estimativa dada é notório.

Após o robot voltar a movimentar-se a estimativa do PF ultrapassa a área útil do mapa e elimina um enorme número de partículas, o que leva à criação de uma nova população espalhada aleatoriamente pelo mapa.

Na figura 5.22, observa-se a pose para onde o PF convergiu após a nova população criada.

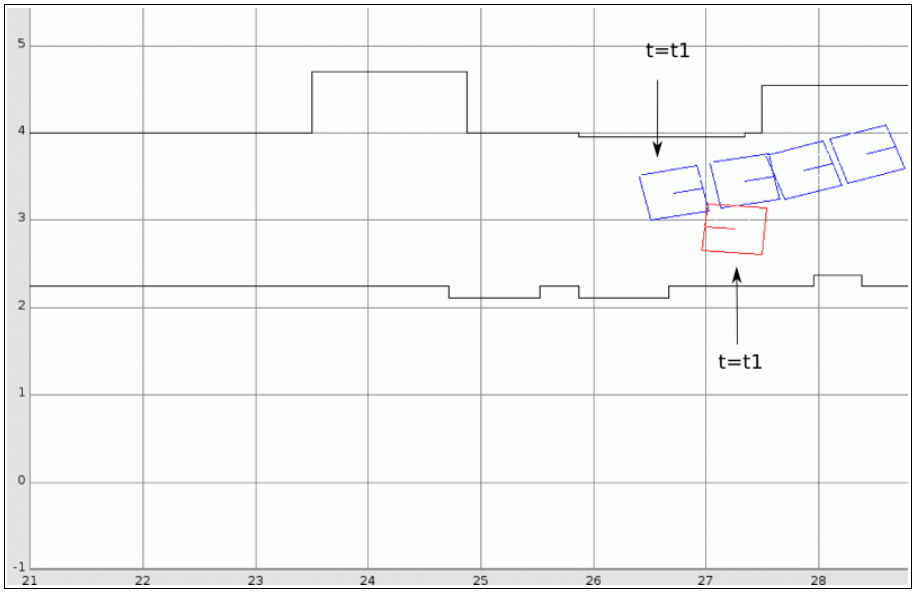


Figura 5.22 PF localização usando *sharps* (2)

É de salientar o facto de que a estimativa se encontra muito próxima da pose real medida, no mesmo instante de tempo, como é descrito na imagem. O grande problema é a orientação da estimativa estar no sentido inverso à pose real. Tal facto deve-se à estrutura do corredor e ao uso de conjuntos laterais de sensores, pois como as paredes são paralelas, é muito difícil descobrir a orientação. O uso de uma bússola poderia resolver este problema.

Obviamente que quando o robot entra em movimento, o PF prevê o movimento no sentido contrário e a sua estimativa tende a divergir da pose correcta. Passados alguns segundos o PF volta a fazer uma re-amostragem e a criar uma nova população pelo mapa.

Quando o PF converge novamente, a estimativa apresentada é a indicada na imagem 5.23, por $t=t1$, enquanto que a pose real nesse instante é a pose a vermelho com o mesmo identificador.

Na figura 5.23, é apresentada uma característica típica deste tipo de filtros multi-modais: o problema com as semelhanças do mapa.

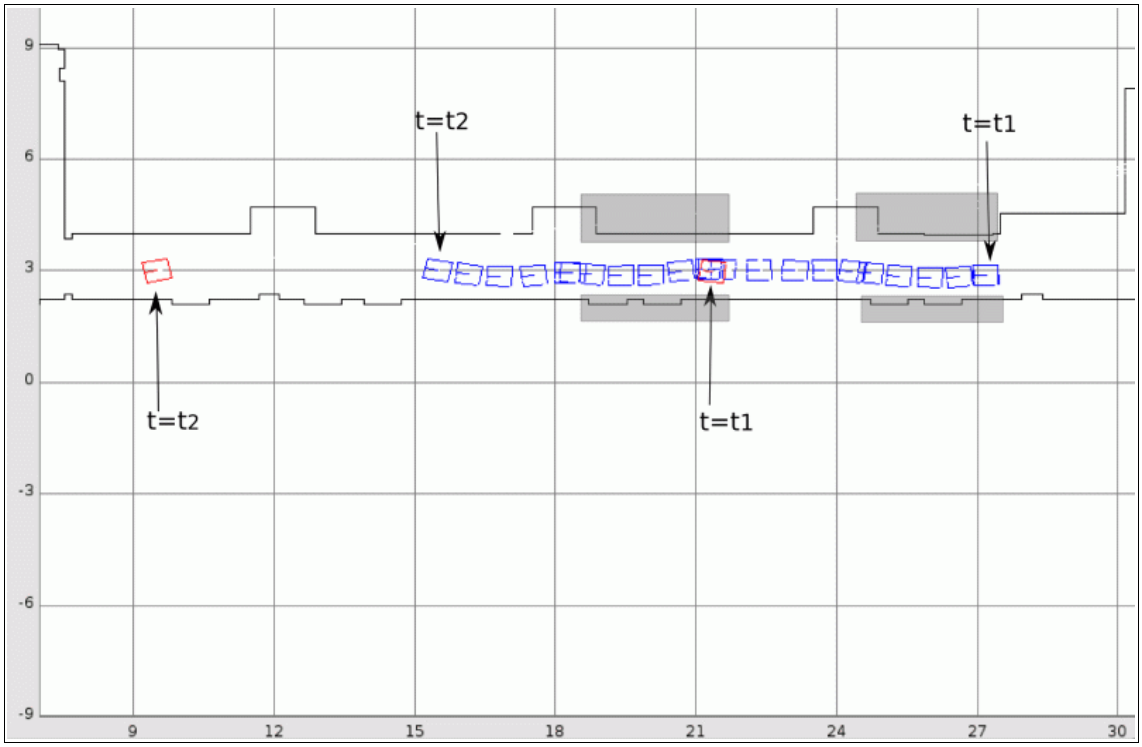


Figura 5.23 PF localização usando *sharps* (3).

Relembra-se que que nesta situação apenas são usados os *sharps* laterais, pois o *sharp* frontal apenas detecta distâncias fora do seu alcance, ou seja não válidas

A diferença entre a estimativa e a realidade é de cerca de 6metros, o que é um erro enorme. Mas analisando bem o mapa, chega-se à conclusão que as zonas marcadas a cinzento são semelhantes, ou seja para o PF, de acordo com os dados recebidos dos sensores, a sua estimativa está correcta.

Nestas situações a única solução é encontrar algo característico do mapa que acabe com a ambiguidade. Neste caso, tal não aconteceu, pois o mapa é muito semelhante nas zonas marcadas a cinzento.

O PF previu o movimento do robot, e actualizou a sua pose de acordo com as medidas recebidas. Quando chegou ao instante $t=t2$ continuava com um erro, na componente *x*, de cerca de 6metros, mas com grande confiança na sua estimativa, pois o mapa permanece semelhante ao longo caminho percorrido.

Com este caso, demonstrou-se um dos maiores problemas deste tipo de filtros: a indefesa contra zonas semelhantes do mapa. Ou seja a sua incapacidade para distinguir esses locais.

Segue-se a apresentação dos erros usando os três métodos de calculo da estimativa da pose referidos na página 30.

5.5.1.2 Melhor partícula

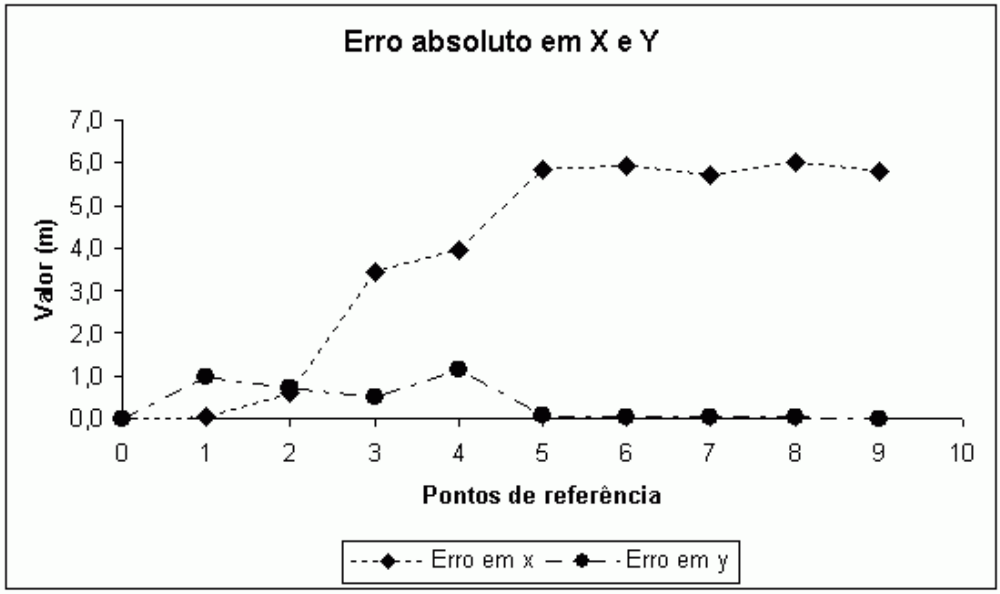


Figura 5.24 PF Erro na localização usando *sharps* (Melhor Partícula)

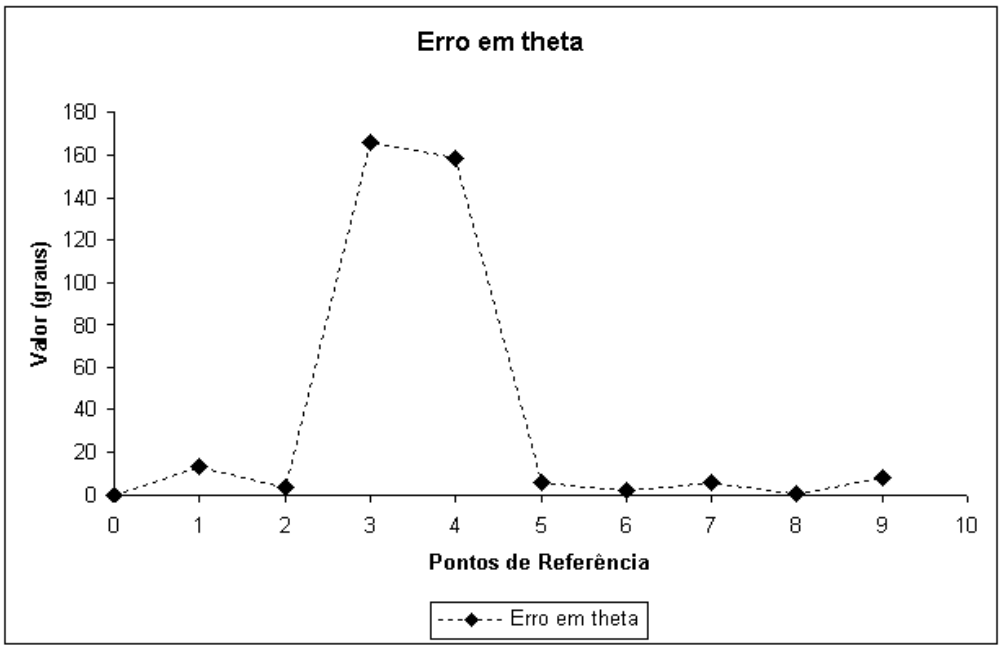


Figura 5.25 PF Erro na orientação usando *sharps* (Melhor Partícula)

5.5.1.3 Média ponderada

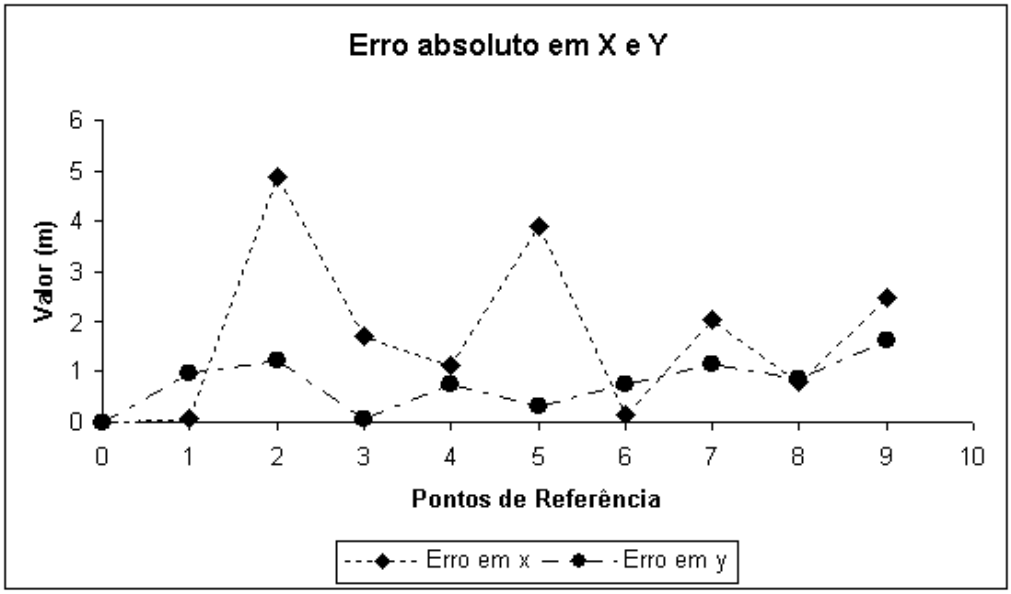


Figura 5.26 PF Erro na localização usando *sharps* (Média Ponderada)

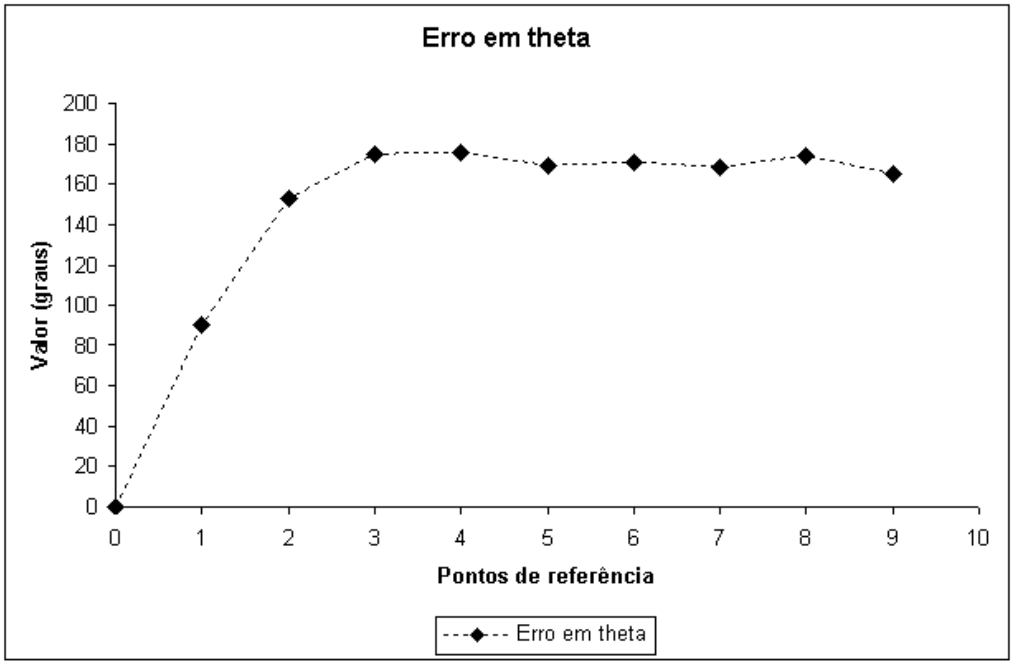


Figura 5.27 PF Erro na orientação usando *sharps* (Média Ponderada)

5.5.1.4 Média Robusta

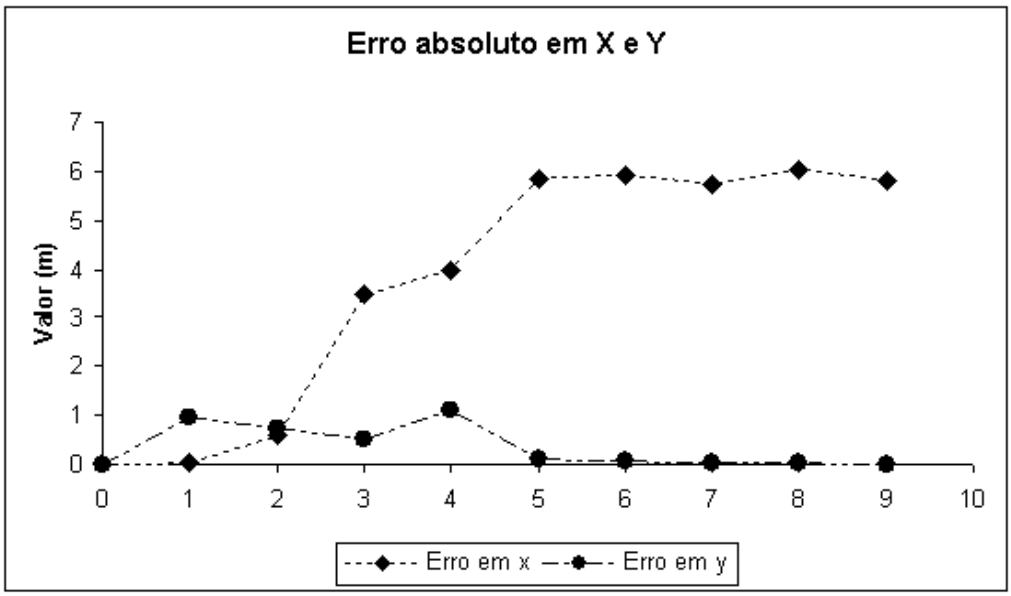


Figura 5.28 PF Erro na localização usando *sharps* (Média Robusta)

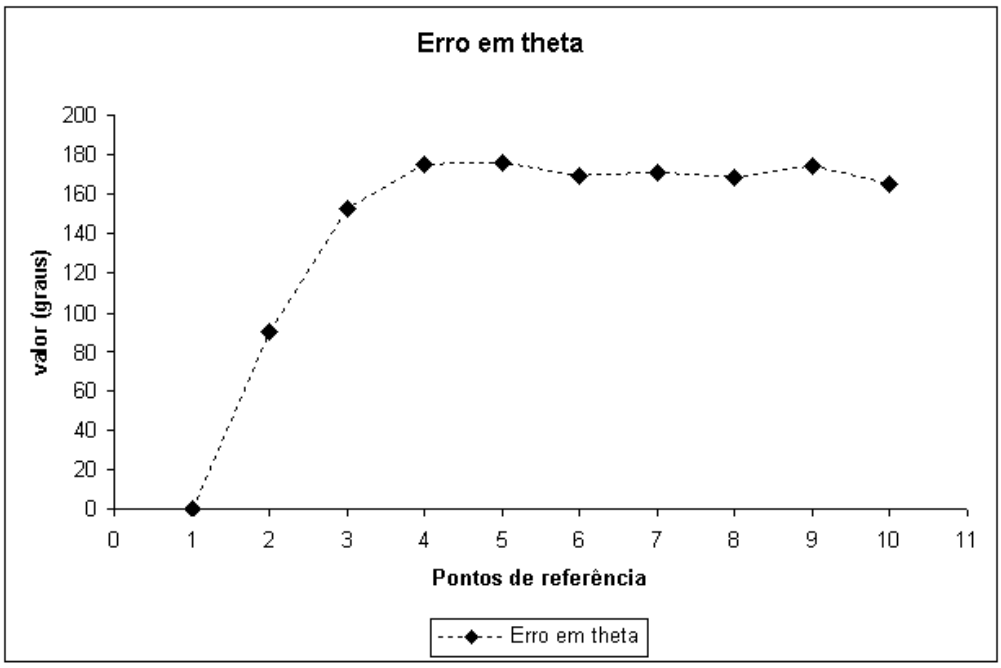


Figura 5.29 PF Erro na orientação usando *sharps* (Média Robusta)

Verifica-se nos gráficos 5.24 e 5.28 aquilo que se referiu acerca deste ensaio. O erro em *y*

tende para zero, o que significa que o PF tem uma grande confiança na sua estimativa, pois a probabilidade das suas distâncias calculadas pertencerem à distribuição das medidas dos sensores é muito elevada, o que se observa pelo erro muito pequeno nesta componente que os sensores mediram.

O erro elevado na componente x, deve-se às semelhanças do mapa, explicadas anteriormente.

Entre os diferentes métodos de estimação da pose do PF, verifica-se que o resultado da "melhor partícula" é muito semelhante ao da "média robusta", mas muito mais eficaz no erro da orientação. A "média ponderada" tem uma evolução diferente mas não melhor, pois o seu erro é quase sempre maior que 2 metros.

5.5.1.5 Sharps e Códigos de Barras

No segundo ensaio de PF optou-se por usar a odometria, os grupos laterais de sharps, o *sharp* frontal e a câmara de leitura dos códigos de barras.

No instante inicial espalharam-se aleatoriamente partículas por todo o mapa útil do robot. Após várias iterações e com o robot parado, o PF convergiu para uma estimativa próxima da pose real representada a vermelho perto do ponto $(31;6)$. Esta convergência já foi explicada anteriormente.

Observando a figura 5.30, o erro na estimativa em relação ao 1º ponto de referência é elevado, tal como no ensaio anterior. Este facto não apresenta surpresa, visto terem sido usadas nas actualizações apenas as medidas dos *sharps*, pois não existiam códigos de barras visíveis perto da pose inicial. Assim sendo é normal que resulte um pose semelhante ao ensaio anterior.

Quando o robot inicia o movimento, a estimativa é revista de forma a traduzir esse movimento, e ao passar perto do 2º código de barras perto de $(31.5;4.2)$, a estimativa é actualizada com os dados da câmara. Nota-se que graças a essa actualização, o erro em relação à 2ª pose real é muito menor que o erro na mesma situação no ensaio anterior.

Após a medição da 2ª pose, a estimativa diverge e acaba por ser efectuada uma re-amostragem, criando uma nova população pelo mapa.

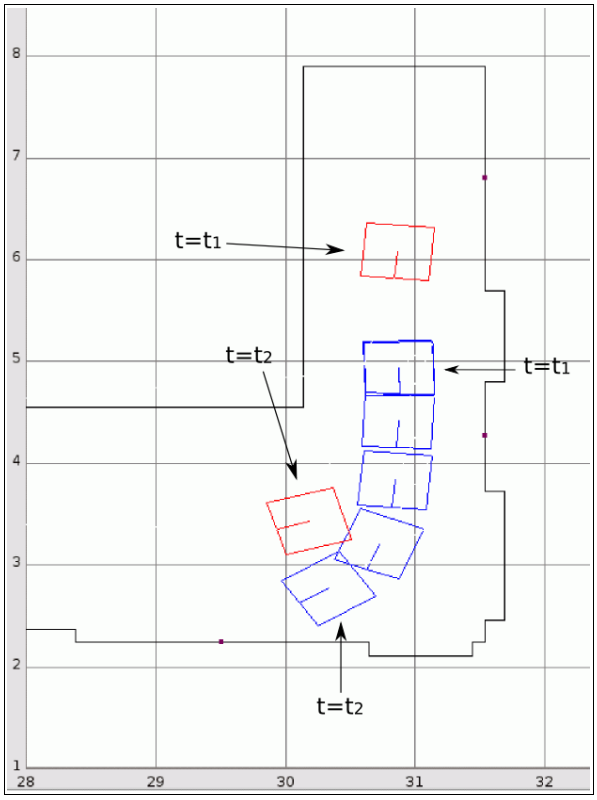


Figura 5.30 PF localização usando *sharps* e CB (1)

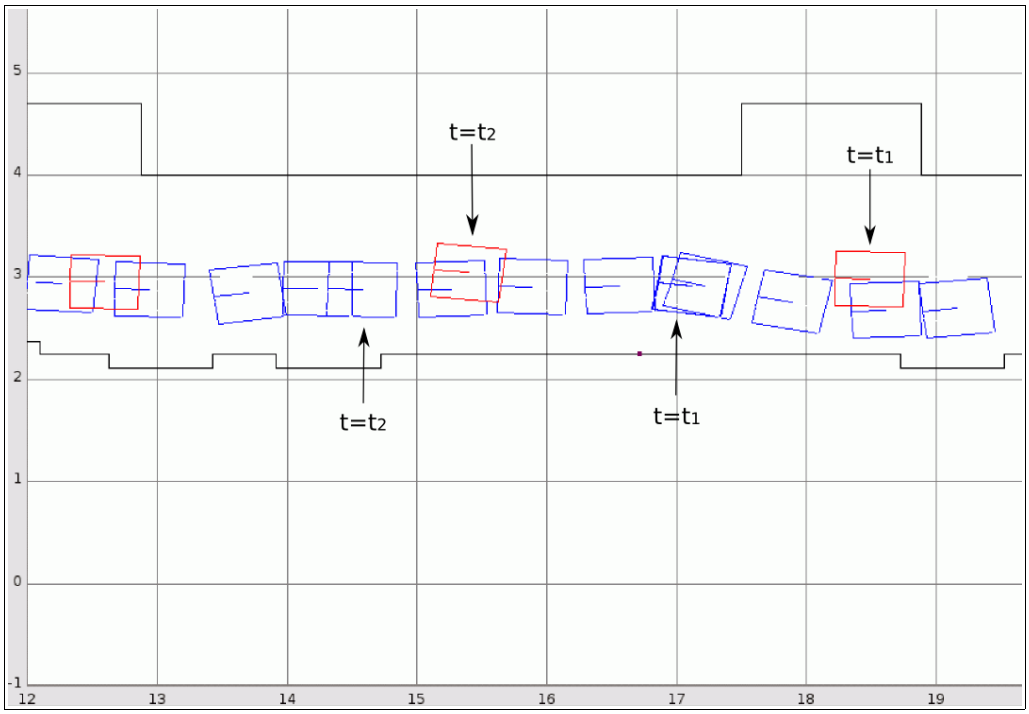


Figura 5.31 PF localização usando *sharps* e CB (2)

Graças aos códigos de barras, o PF rapidamente converge para uma situação próxima da pose real. Esta afirmação verifica-se nas imagens 5.31 e 5.32, em que se nota que as estimativas são próximas das posições reais medidas. Apesar que o erro na componente *x* ainda é elevado, cerca de 1metro, facto devido à escassez de medidas nessa componente.

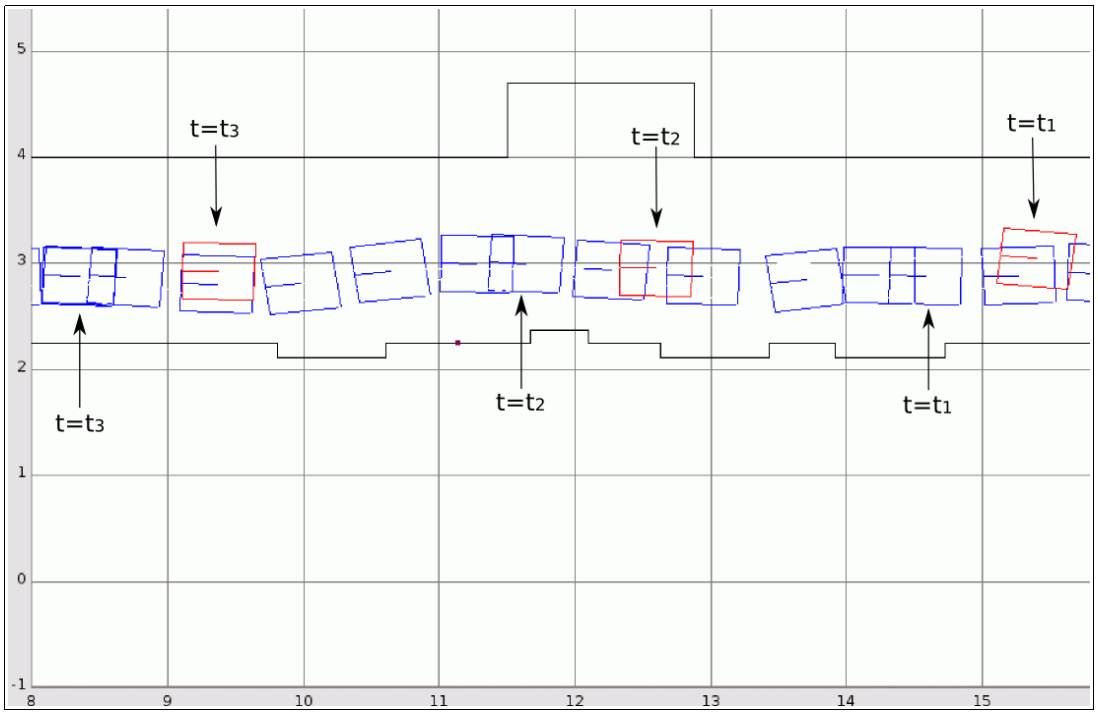


Figura 5.32 PF localização usando *sharps* e CB (3)

Segue-se a apresentação dos erros usando os três métodos de calculo da estimativa da pose.

5.5.1.6 Melhor Partícula

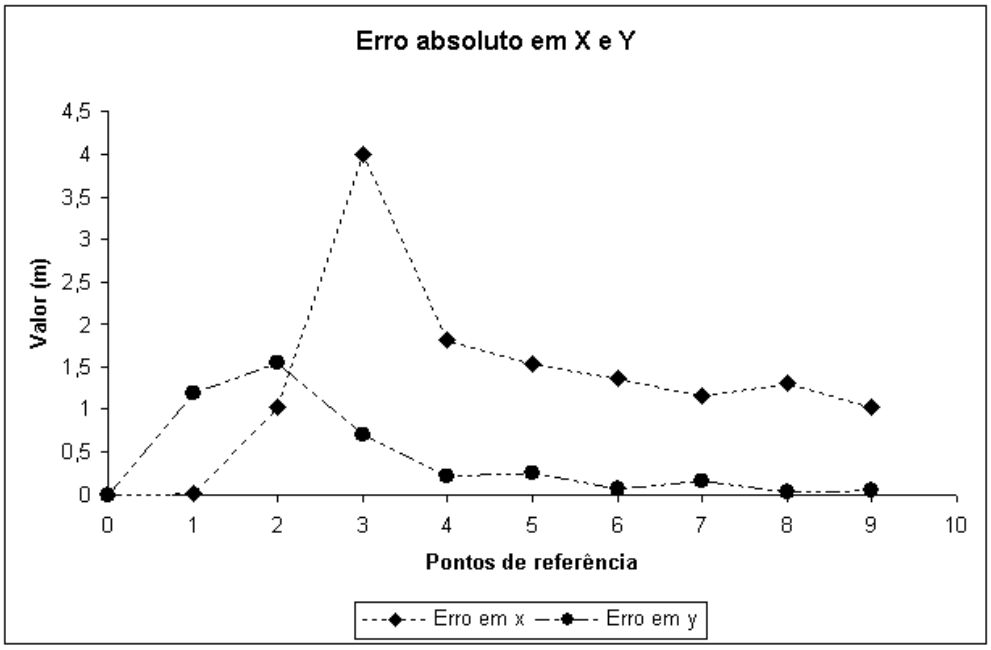


Figura 5.33 PF Erro na localização usando *sharps* e CB (Melhor Partícula)

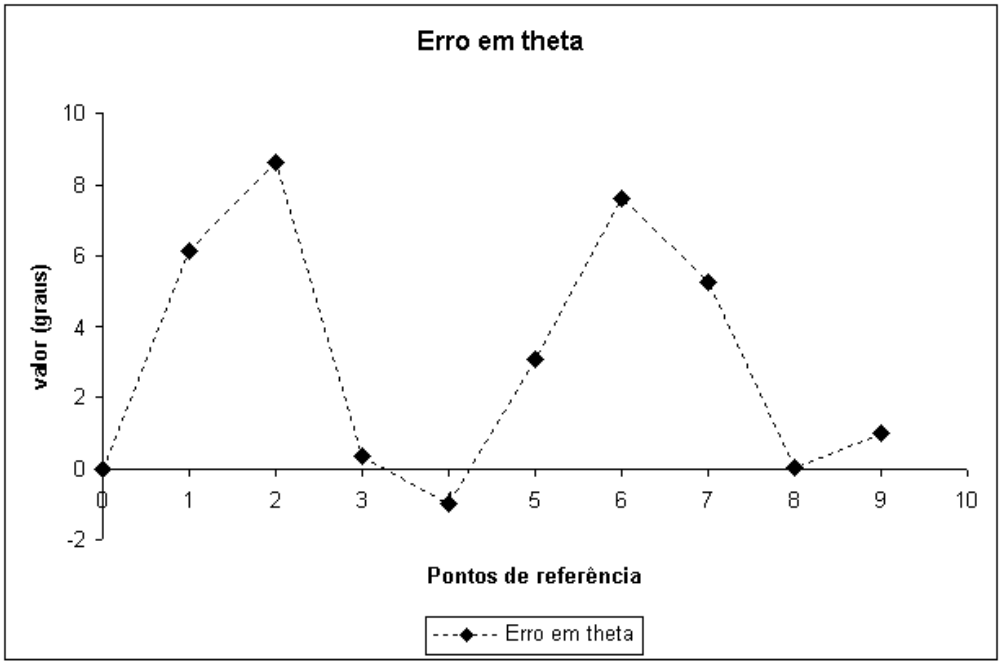


Figura 5.34 PF Erro na orientação usando *sharps* e CB (Melhor Partícula)

5.5.1.7 Média Ponderada

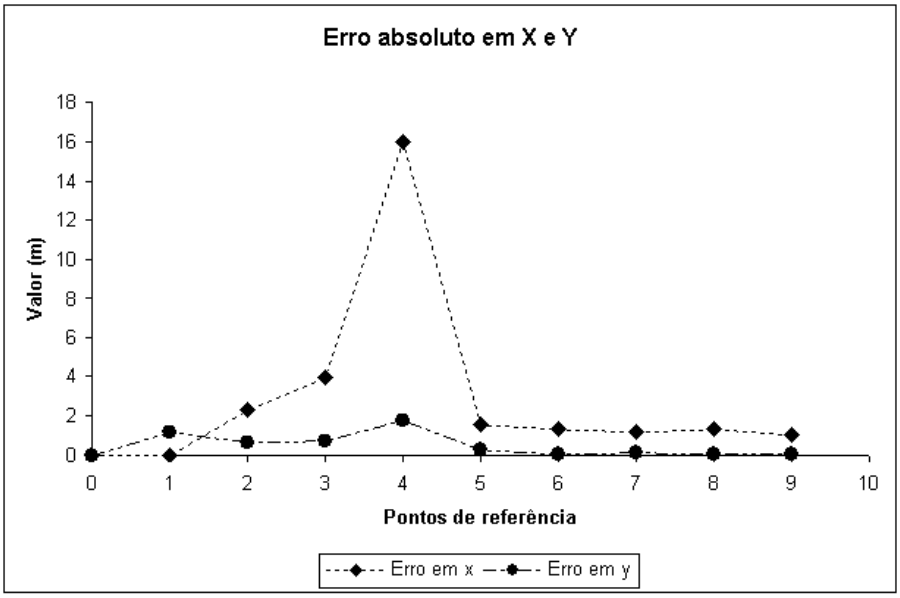


Figura 5.35 PF Erro na localização usando *sharps* e CB (Média Ponderada)

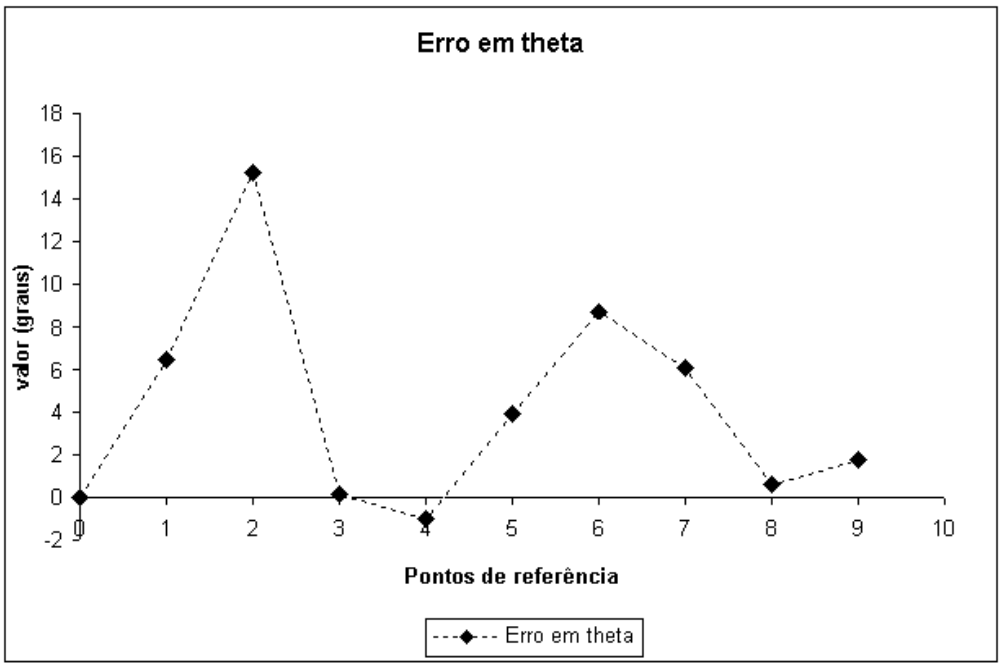


Figura 5.36 PF Erro na orientação usando *sharps* e CB (Média Ponderada)

5.5.1.8 Média Robusta

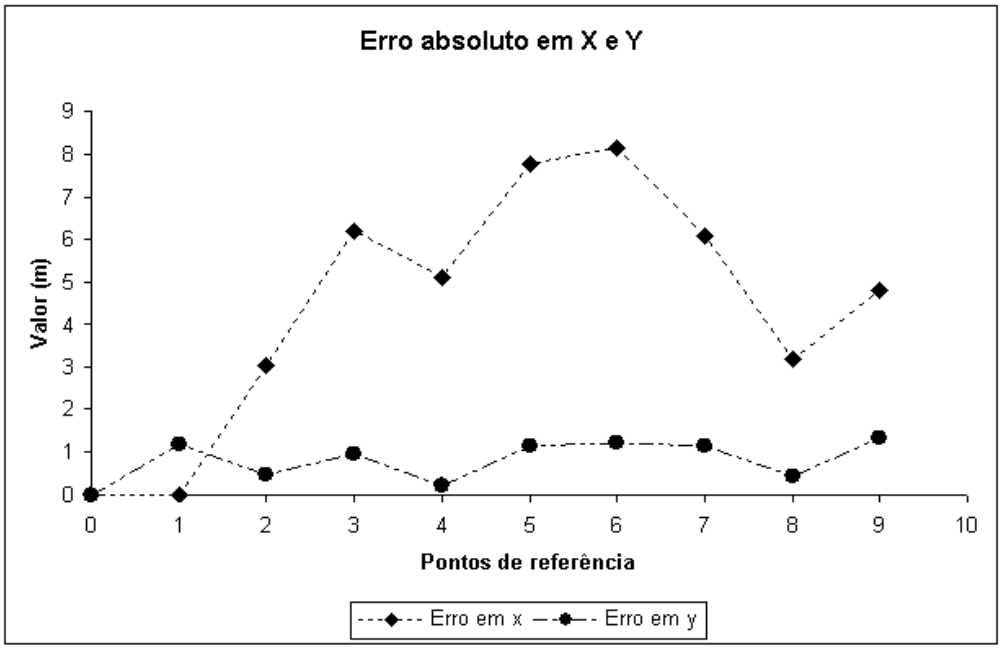


Figura 5.37 PF Erro na localização usando *sharps* e CB (Média Robusta)

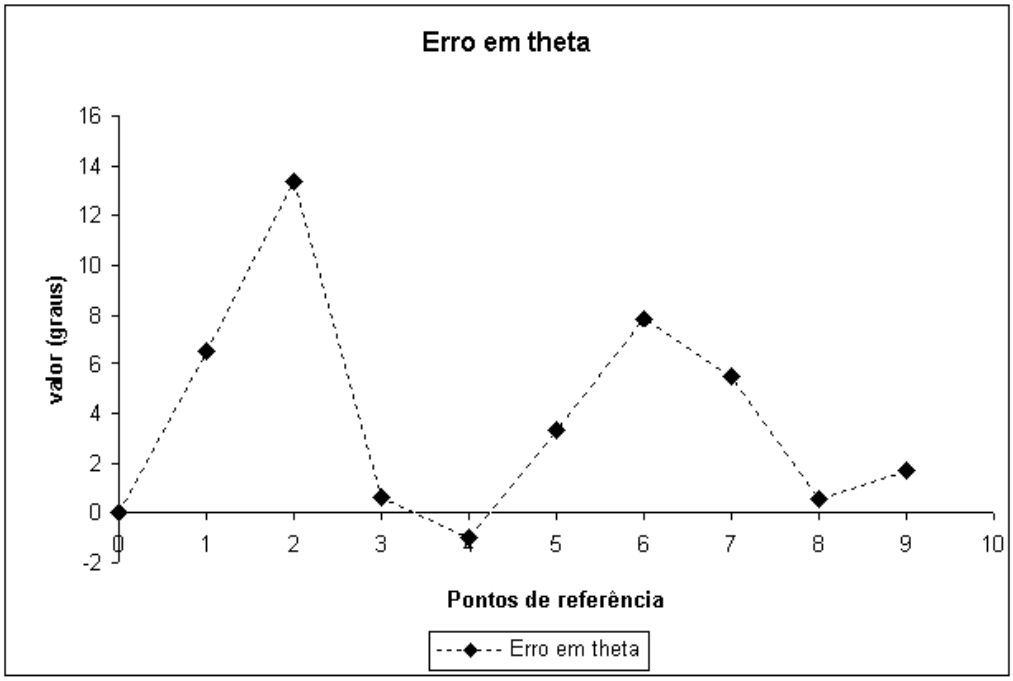


Figura 5.38 PF Erro na orientação usando *sharps* e CB (Média Robusta)

De todos os métodos estudados do cálculo da estimativa do FP, a “melhor partícula” parece ser a que melhores resultados deu neste ensaio. Tanto a “média robusta” como a “média ponderada” divergiram bastante em relação aos resultados reais.

Nos gráficos do erro de orientação, os pontos que foram rejeitados devido ao seu erro demasiado elevado têm o valor de -1. Estes pontos são referentes a momentos em que o filtro divergiu.

Novamente a “melhor partícula” apresenta melhores resultados com um erro na orientação bastante reduzido.

5.5.2 Localização Global

Para testar a localização global, colocou-se o robot parado numa posição do corredor. Iniciou-se o algoritmo do filtro de partículas usando apenas o conjunto de sharps como sensores.

No instante inicial espalham-se aleatoriamente partículas por todo espaço útil do mapa. A vermelho representa-se a pose real do robot. Os pontos no mapa assinalam a pose das várias partículas, estão a ser usadas 900.

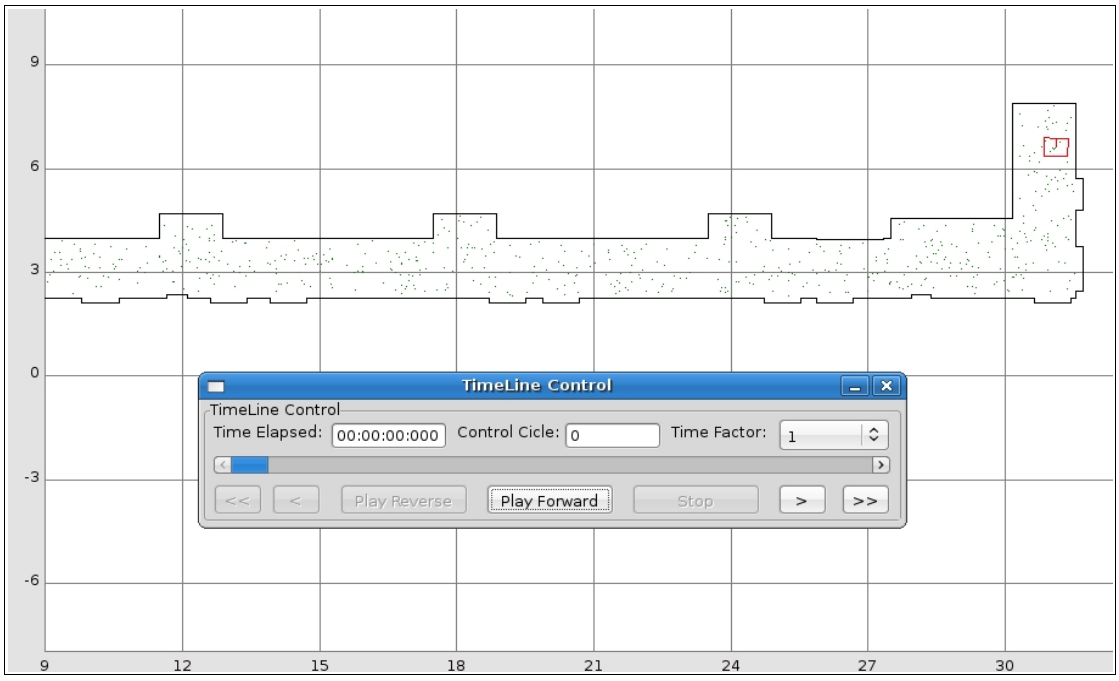


Figura 5.39 PF localização global (1)

Calcula-se a estimativa da pose usando o método da “melhor partícula”, 3.77. A azul é representada a pose estimada do filtro de partículas.

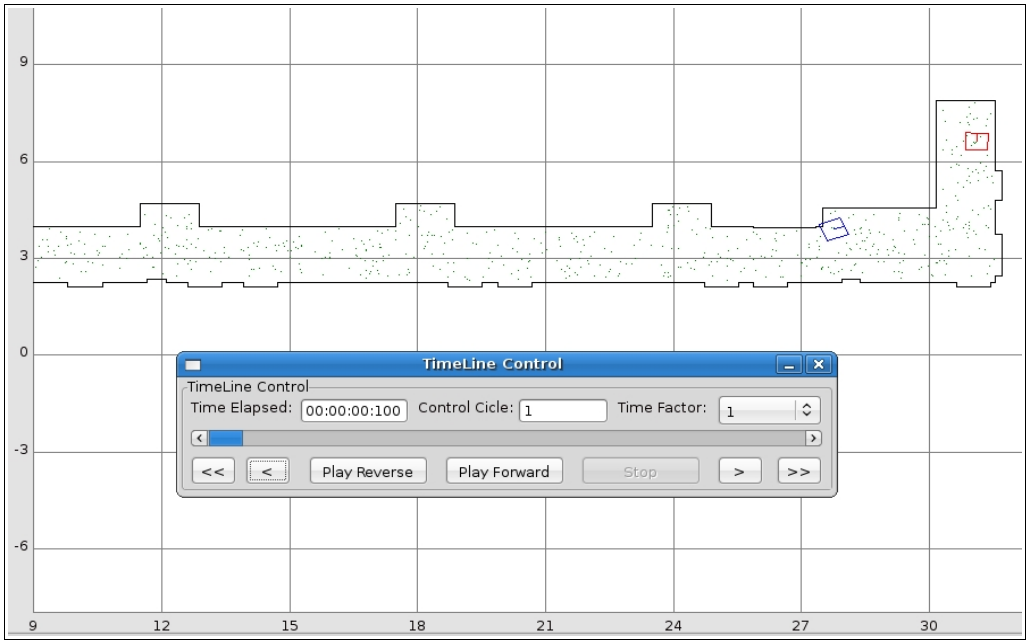


Figura 5.40 PF localização global (2)

Passados alguns instantes, o PF converge para uma pose muito próxima da pose real do robot. Como é possível observar na figura 5.41 ainda há muita incerteza nesta estimativa, pois existem muitas partículas espalhadas pelo mapa.

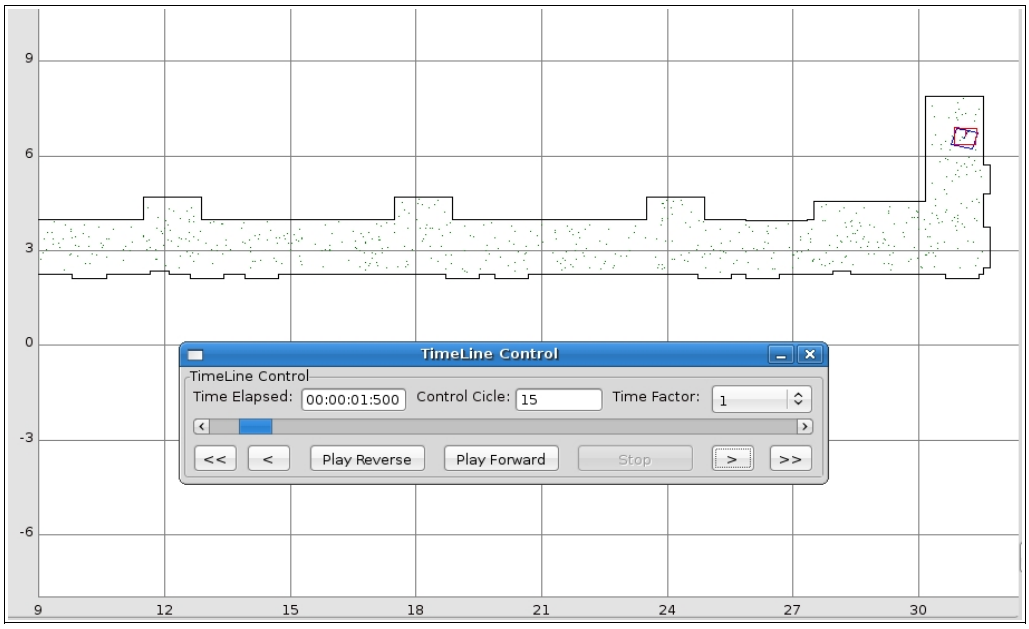


Figura 5.41 PF localização global (3)

Mais alguns instantes passados e a confiança na estimativa aumenta consideravelmente, como é possível notar pelo número reduzido de partículas espalhadas pelo mapa, na figura 5.42. Pois a grande maioria destas encontra-se nas imediações da pose mais provável.

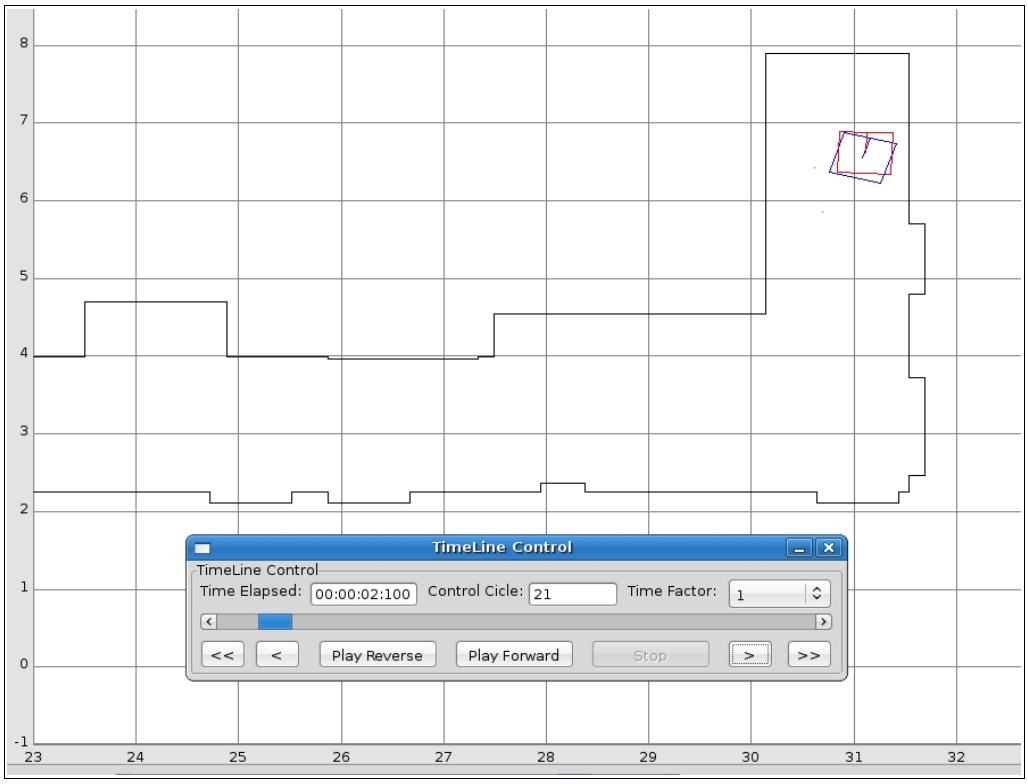


Figura 5.42 PF localização global (4)

Assim tal como é possível verificar na figura 5.42, em 21 ciclos do algoritmo, ou seja em cerca de 2 segundos, o filtro de partículas forneceu uma pose estimada muito próxima da pose real do robot.

Segue-se uma tabela com os valores absolutos dos erros nas várias componentes.

Tabela 5.4 PF Erro na localização global

Erro em	Valor Absoluto
X	0,027 (m)
Y	0,070 (m)
θ	13,6 (°)

Analisa-se que o erro é reduzido, tendo em conta que o algoritmo começou sem qualquer conhecimento da pose real do robot.

5.6 Conclusões

Este capítulo reúne o trabalho experimental realizado na área da auto-localização em ambiente semi-estruturado.

Foram abordados dois algoritmos de fusão de informação: Filtro de Kalman Extendido (EKF) e Filtro Partículas (PF).

Para o EKF usaram-se dois métodos de auto-localização:

- localização por linhas;
- localização por códigos de barras.

A técnica de localização por linhas é similar à adoptada no futebol robótico com uma diferença muito importante: neste trabalho não se usa uma câmara para visualizar as linhas, como no futebol, mas sim sensores de baixo custo, que medem distâncias. Adiciona-se o facto de que este método é usado com as paredes do corredor, não havendo na realidade linhas marcadas no ambiente.

Os resultados obtidos foram muitos interessantes, pois com uso dos sensores *sharp* conseguiram-se estimativas com erros menores que 7 cm. O erro na estimativa da orientação também é reduzido. na ordem dos 10º, no pior caso.

A localização por códigos de barras é um método inovador que permite a localização (x, y, θ) do robot através do processamento de visão.

Comprovou-se experimentalmente as vantagens deste método, que permite obter estimativas com erros muito baixos, sendo assim um marcador muito fiável. Demonstrou-se que com a utilização dos códigos de barras é possível ao EKF enfrentar e resolver o problema da localização global.

Foi também importante o estudo feito à confiança da estimativa usando as covariâncias da pose estimada. Esta análise permitiu verificar certos fundamentos teóricos do filtro de Kalman que de outra forma seriam complicados de observar.

Em resumo regista-se uma boa resposta do EKF ao problema de localização, conseguindo num mapa de cerca de 54 m^2 , ter erros nunca superiores a 10cm. Este facto em muito se deve aos testes de verosimilhança implementados, que permitem uma maior confiança nas medidas recebidas dos sensores.

De assinalar também que o EKF implementado é robusto, e foi testado em ambientes dinâmicos provocados pelas seguintes situações: movimento de pessoas pelo corredor, deslizamentos e choques provocados por terceiros no robot, objectos que não constam no mapa, portas abertas, etc. Em todos os casos o resultado foi sempre muito bom, tendo a estimativa da localização mantido a sua qualidade.

O filtro de partículas, comprovou a sua popularidade nos ensaios de localização global, em que apenas com o conjunto de sensores *sharp*, conseguiu, em poucas iterações, fornecer uma estimativa com erros menores que 10cm.

Nos outros ensaios, verificaram-se algumas das lacunas e problemas que existem neste algoritmo. Entre elas, a que mais se destaca é a dificuldade em trabalhar com zonas semelhantes do mapa. Na realidade este problema poderia ser resolvido de duas formas: ou colocando mais sensores que permitissem ter mais informações sobre o ambiente que rodeia o

robot e assim dissipar as ambiguidades, ou mudando para um tipo de localização activa. Como neste trabalho se considerou necessária uma metodologia de localização passiva, em que o algoritmo de localização, não controla as decisões ou tarefas do robot, simplesmente tenta estimar a melhor pose enquanto o robot cumpre a sua missão. Numa situação de localização activa, em caso de dúvida na sua posição, o robot pára a sua missão e procura pontos característicos no mapa para que se possa encontrar.

Usando estas duas soluções, certamente seriam melhores os resultados obtidos pelo filtro de partículas. De qualquer forma, com este trabalho analisou-se em pormenor o PF, e apresentam-se as suas virtudes e defeitos, bem como algumas das possíveis soluções para a obtenção de melhores resultados.

Acerca dos métodos estudados para o calculo da estimativa do PF, chegou-se à conclusão, por comparação com vários ensaios, que a técnica de "melhor partícula" é a que apresenta resultados mais estáveis, apesar de nem sempre ser a melhor estimativa. Nos ensaios descritos neste capítulo nota-se que a técnica de "melhor partícula" é mais estável e também que na grande maioria dos casos é a que apresenta menor erro.

Comparando os resultados dos dois filtros chega-se à conclusão que o EKF obteve melhores resultados e através do uso de códigos de barras conseguiu até competir na localização global com o PF. Relembrando que este problema específico de localização é dominado pelos filtros multi-modais como o PF.

É necessário referir que são filtros com características diferentes e que cada um tem as suas vantagens, é portanto muito perigoso fazer generalizações.

Capítulo 6

Conclusões

Esta dissertação aborda diversos melhoramentos feitos a um robot autónomo de serviço de limpeza.

Foi feita a prova de conceito de uma arquitectura para a integração fácil de novos sensores e fusão de informação.

Foi ainda alterada a arquitectura do *software* de decisão do robot para passar a aceitar comandos provenientes de um comando *wiimote*, para permitir um comando local do robot. O trabalho de desenvolvimento dessa interface de comando propriamente dita é alvo de outro trabalho de dissertação.

Foram ainda feitos diversos testes de desempenho do conjunto, nomeadamente foram medidos os tempos de execução relevantes para o programa de controlo.

O trabalho central deste projecto foi a auto-localização de sistemas robóticos móveis.

Foi projectado e implementado um filtro de Kalman extendido para a fusão sensorial dos dados disponíveis, de forma a obter uma melhor estimativa da localização do robot. Ficou demonstrada, na prática, a convergência deste método.

Utilizaram-se duas técnicas de localização usando linhas e códigos de barras. Verificaram-se que estas técnicas podem ser adaptadas a situações reais. Na localização por linhas adoptaram-se as paredes como referências e obtiveram-se resultados bastantes interessantes com sensores de posição de baixo custo. Na localização por códigos de barras, comprovou-se que o uso destes marcadores é muito vantajoso, pois fornecem uma pose completa nas suas medidas. O facto de se tratarem de marcadores constituídos por uma folha de papel, acresce ainda mais o seu valor, pois são fáceis de aplicar no local desejado, simples de criar e não requerem manutenção ou energia como outros tipos de marcadores. De referir também que o uso deste método permitiu ao filtro de Kalman resolver a situação de localização global, com erros bastante baixos, o que de outra forma seria muito difícil.

Foi também projectado e implementado um filtro de partículas, com o objectivo de estimar a localização do robot. Verificaram-se algumas das suas características tal como a resolução do problema de localização global, e a sua indefesa a mapas com muitas zonas semelhantes. O uso de códigos de barras foi também implementado e os seus resultados melhoram a estimativa do filtro.

De referir que os testes de verosimilhança implementados são de uma grande importância nos bons resultados da localização. A sua função de filtrar as medidas ruidosas é fundamental na fusão sensorial. Mais testes de verosimilhança devem ser implementados e deverá dar-se uma maior importância aos mesmos, pois neles reside parte do sucesso da fusão. Graças aos vários testes efectuados, o EKF consegue manter a qualidade da sua estimativa em ambientes dinâmicos, compostos por pessoas, objectos estranhos ao mapa e colisões com o robot.

Outro tema abordado foi a arquitectura do sistema de controlo, esta foi revista e alterada de forma a ser constituída por blocos modulares, tais como a localização e a navegação. É também possível colocar vários algoritmos em paralelo no mesmo bloco, como por exemplo, na secção de localização optar-se por usar o filtro de Kalman e o filtro de partículas em simultâneo, para que seja possível comparar as suas estimativas. Esta opção foi várias vezes usada durante o trabalho.

Um conceito interessante que surgiu da remodelação da arquitectura foi a aquisição inteligente de sensores. Este baseia-se na necessidade de ter uma aplicação que reconheça e atenda os dados dos vários sensores e os agrupe por características comuns. Este programa deve para além de garantir a validade da medida recebida, deve fundir dados dos sensores de forma a obter medidas mais robustas. Pode também fundir dados para avaliar características que não podem ser medidas por um único sensor. Um exemplo disso é a localização por linhas, em que se usa um par de sensores que fornecem cada um, uma distância, e através de cálculos são convertidas num ângulo com uma confiança associada.

Comparando os dados obtidos experimentalmente, chega-se à conclusão que o filtro de Kalman obteve melhores resultados que o filtro de partículas, isto deve-se a diversos factores, sendo os mais importantes: os testes de verosimilhança usados no EKF e o uso de códigos de barras que permitiu ao EKF resolver a localização global. A posição e o número de sensores também influenciou o resultado, pois enquanto a disposição actual fornece dados úteis ao EKF como o ângulo em relação à parede, no PF apenas estes dados são por vezes insuficientes para que este possa dissipar as incertezas em relação à sua posição. Tal particularidade deve-se ao facto dos sensores se encontrarem nas laterais, o que permite apenas ter informação na componente perpendicular à orientação do robot, o que num ambiente igual ao usado, um corredor, leva a duvidas entre as várias zonas semelhantes do mapa. Outro motivo para a menor qualidade da estimativa do PF deve-se ao tipo de localização usada. Neste trabalho optou-se por uma localização passiva, ou seja o robot tinha a sua missão definida e o algoritmo de localização não interferia na navegação. A localização activa melhoraria a localização ao dirigir o robot para locais distintos do mapa, encontrando assim a sua pose e acabando com as incertezas mas atrasando a sua missão de limpeza.

Em resumo, este trabalho permitiu comparar diferentes algoritmos de fusão sensorial no âmbito da localização. A estrutura de controlo criada é robusta e permite uma localização com boa qualidade em ambientes dinâmicos.

6.1 Trabalho futuro

Ao nível da localização os objectivos mais lógicos a curto prazo seriam:

- Implementar e testar novos algoritmos tal como a localização por grelhas ou filtro de Kalman multi-hipótese.

- Aumentar o número de sensores e se possível experimentar outros, na tentativa de obter melhores resultados.
- Descobrir novos métodos de localização que possam ser implementados em ambientes não estruturado, uma solução possível seria usar as luzes do tecto como marcadores usando uma câmara.

Ao nível de fusão de informação, torna-se muito atraente a aquisição inteligente de sensores. Deve ser estudada e melhorada, de forma a servir não só para libertar o tratamento dos sensores do programa principal como para criar vários sensores inteligentes capazes de medir características impossíveis de medir com um único sensor.

Em relação ao projecto efectuado, C*leanrob*, é um trabalho que tem bases sólidas para seguir em frente. Os objectivos próximos deverão passar por aumentar a robustez da localização e do método de evitar colisões. A melhor forma de o fazer é implementar mais testes de validação das medidas para que assim se possa aumentar a sua confiança e rejeitar os dados inválidos ou inutilizados. Esta área é de importância vital para o bom funcionamento de todos os algoritmos que usam a informação vinda dos sensores.

É necessário implementar um método de planeamento de rotas e um gestor de tarefas, para que o robot seja cada vez mais independente. Objectivos como ensaiar o robot noutros corredores mais interessantes, como os do edifico B da FEUP ou colocar o robot a descer de elevador, são aliciantes e aumentam a autonomia e o interesse do projecto.

A médio prazo, era importante dar outro tipo de missões ao robot, tal como entregar correio ou material entre gabinetes, guiar pessoas pelos corredores e monitorizar zonas comuns do edifico em acções de vigilância.

Capítulo 7

Referências Bibliográficas

[1] http://pt.wikipedia.org/wiki/Autonomia, acedido a 28 de Maio de 2008

[2] Armando jorge Sousa, "Arquitecturas de sistemas robóticos e localização em tempo real através de visão", 2004, Tese de doutoramento FEUP

[3] Clark F. Olson , " Probabilistic Self Localization for Mobile Robots", IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, VOL. 16, NO. 1, FEBRUARY 2000

[4] Ingemar J. Cox, " Blanche: An Experiment in Guidance and Navigation for an Autonomous Robot Vehicle", 1991

[5] Sebastian Thrun, Wolfram Burgard, Dieter Fox, "Probabilistic robotics", 2005, MIT press

[6] http://paginas.fe.up.pt/~cleanrob/wiki/doku.php, acedida a 30 de Junho de 2008

[7] Jaime Francisco Cruz Fonseca, "Aplicação de técnicas de fusão sensorial de dados no levantamento do relevo de objectos", 1999, Tese de doutoramento

[8] Armando Jorge Sousa, " Data Fusion, Kalman Filtering, Self Localization", 2007

[9] H.B.Mitchell, "Multi-Sensor Data Fusion", 2007, Springer

[10] Boudjemaa R., Forbes A.B, " Parameter estimation methods for data fusion", 2004

[11] Durrant-Whyte H.F, " Sensor models and multisensor integration", 1988

[12] Peter S. Maybeck, " Stochastic models, estimation, and control", 1979

[13] A Gelb, J. Kasper, Raymond Nash, Charles Price, Arthur Sutherland, "Applied Optimal Estimation", 1989, The M. I. T. Press

[14] Manuel Ferreira e Isabel Amaral, "Probabilidades e estatística - Fornulário - 5a Edição", 2001, Edições Sílabo

[15] Dieter Fox, Jeffrey Hightower, Lin Liao, " Bayesian Filters for Location Estimation", 2003

[16] R. E. Kalman, " A New Approach to Linear Filtering and Prediction Problems", 1960

[17] Paulo Costa, "Localização em tempo real de multiplos robots num ambiente dinâmico", 1999, Tese de doutoramento

[18] Paulo Lopes dos Santos, " Apontamentos da disciplina Identificação e Estimação", 2007

[19] Eric Wan e Alex Nelson, "Dual EKF Methods", 2001, Kalman Filtering and Neural Networks, Wiley Publishing, editors Simon Haykin, 2001

[20] Nicholas Metropolis, S. Ulam, " Journal of the American Statistical Association", 1949

[21] Frank Dellaert, Dieter Fox, Wolfram Burgard, Sebastian Thrun , " Monte Carlo Localization for Mobile Robots", 1999

[22] Ioannis M. Rekleitis, " A Particle Filter Tutorial for Mobile Robot Localization", 1998

[23] M. Sanjeev Arulampalam, Siomn Maskell, Neil Gordon, " A Tutorial on Particle Filters for Online NonLinear/Non Gaussian Bayesian tracking", 2002

[24] J.Borenstein, H.R.Everett, L. Feng, ""Where am I?", Sensores and Methods for mobile robot positioning", 1996, University of Michigan

[25] J. Borenstein, H. R. Everett, and L. Feng, " Navigation Mobile Robots: Systems and Techniques", 1996

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

[27] Gustavo Pimentel, "Aplicação do Filtro de Partículas como sistema de fusão de informação", 2008, Tese de mestrado

[28] Fernando Pedro Pinto, "Aplicação do Filtro de Kalman na Auto-Localização de um Sistema Robótico Autonomo", 2008, Tese de mestrado

[29] http://www.acroname.com/robotics/parts/R145-SRF08.html, Acedido a 18 de Março de 2008

[30] Robin R. Murphi, "Introduction to AI Robotics", 2000, MIT Press

[31] http://www.acroname.com/robotics/parts/R144-GP2Y0A02YK.html, acedido em 15 de Março de 2008

Anexo 1

Ficha Técnica

Escrito em Open Office 2.4.0 sob Windows

Fonte: Trebuchet MS, Times New Roman

Muitas figuras desenhadas utilizando Inkscape.

URL da página do Autor: *www.fe.up.pt/~ee03122*

URL da página institucional do projecto CleanRob: *http://www.fe.up.pt/~cleanrob*

João Manuel Ferreira Martins,
30 Junho 2008