Análise e implementação de algoritmos para localização de robôs móveis

Leandro Carlos Fernandes

Orientadora: *Profa. Dra. Roseli Aparecida Francelin Romero*

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP – São Carlos
Abril/2003

A Comissão Julgadora:

Profa. Dra. Roseli Aparecida Francelin Romero _____

Prof. Dr. Eduardo Marques _____

Prof. Dr. José Pacheco de Almeida Prado _____

"If knowledge can create problems,

it is not through ignorance that we can solve them."

(Isaac Asimov)

Agradecimentos

Sinto ser apenas umas, as páginas dedicadas aos agradecimentos, pois seria necessário muito mais páginas do que as utilizadas no trabalho, para que pudesse relacionar os nomes de todos a quem eu presto meus agradecimentos. Embora, dedico as linhas restantes, àqueles que certamente tem seus nomes encabeçando esta lista.

Por uma questão de primazia na minha vida, meu primeiro agradecimento é a Deus que sempre me amparou em suas mãos e cobriu-me com suas asas, não deixando que meus pés vacilassem. Em seguida, agradeço a um de seus anjos que habitam a terra, a minha mãe, que mesmo sem conhecer do profundo da ciência dos homens, fielmente esteve comigo mesmo quando distante e criou um homem a quem muitos respeitam e admiram.

Ao meu Pai, que me ensinou a não transgredir as coisas em que acredito ou abrir mão dos meus valores por qualquer que seja o preço a ser pago, mas ser capaz de andar pela vida de cabeça erguida e encarar os outros olhos nos olhos.

A Profa. Dra. Roseli, uma pessoa que indubitavelmente me auxiliou nos momentos mais críticos do meu trabalho, e que durante todo este tempo sempre teve para comigo compreensão e dedicação.

Ao Prof. Dr. Eduardo Marques, que apesar de não ter sido meu orientador, ou ainda, co-orientador, sempre desempenhou esse papel, além de oferecer seus préstimos e recursos do LCR para que eu pudesse desenvolver e concluir meu trabalho.

Imediatamente após, vem todos da minha família, o vô Li, a vó Cida, que cada um em uma ramo, foram os que deram início a isso tudo (família), as tias Cida, Dulce, Dirce, Edna, Joana, Márcia e Rosa, os tios, Ismael, João, Nenê e Serginho, todas as minhas primas, Carol, Cíntia, Mary, Érika, Marina Ciça, os meus primos Jorge, Diego e Vitor, a minha madrasta Conceição, ao seu pai Paulo e sua mãe Maria, ao irmão Paulinho e sua sobrinha Daniela, que, parte em São Paulo e parte em Ribeirão, sempre fizeram a maior torcida pelo meu sucesso.

Ao Jean, ao Henrique, a Vivi e ao César (SP), pessoas que realmente fazem jus à classificação de amigos, e também, aos meus coordenadores e aos colegas de trabalho na UNIP, como a Fabrícia, a Neli, as Danis, Betão e o Fernando.

A amigos de graduação Alexandre e Rovilson, que me acompanharam durante no mínimo quatro anos. Ao pessoal do LCR, Marcão, Denis, Carlão, Menotti, Vanderlei e Murilo, aos

colegas do LABIC, Claudia, Gustavo, Rodrigo, Débora, Gedson e os outros.

Um agradecimento em especial a Profa. Dra. Maria das Graças Volpe Nunes, alguém a quem nunca me reportei diretamente, mas indiretamente resolveu grandes problemas para mim, como por exemplo, a prorrogação do prazo, que já havia sido estendido, em mais cinco dias para que este trabalho pudesse ser concluído.

As meninas que trabalham na secretaria de pós-graduação, Beth, Laura e Ana, e também a uma ex-funcionária desse departamento, a Marília, que sempre foram muito prestativas e atenciosas conosco (alunos).

Não poderia deixar de agradecer ao guardas Alex, Marco, Vagner e o Nunes, que por muitas vezes foram às únicas companhias que eu tinha na USP e que, nas longas noites, preparavam café e nunca deixavam de ligavam pra avisar.

Sumário

Lista de Figuras iv

Índice de Siglas vii

Resumo viii

Abstract ix

Capítulo 1 - Introdução 1

1.1. O que são Robôs? 1

1.1.1. Robôs Móveis *4*

1.2. Incerteza na Robótica 5

1.3. Representando o mundo 7

1.3.1. Diagramas de estado *8*

1.3.2. Mapeamento *10*

1.4. A interação entre o robô e o ambiente 11

1.5. Navegação 13

1.6. Localização 14

Capítulo 2 - Localização de Robôs Móveis 16

2.1. Posicionamento Relativo 16

2.2. Posicionamento Absoluto 17

2.3. Ajuste de Posição usando Odometria 18

2.4. Posicionamento Baseado em Mapas 22

2.5. Landmarks 23

2.6. Trabalhos Relacionados 27

2.6.1. Utilizando Odometria *30*

2.6.2. Utilizando Landmarks *30*

2.6.3. Utilizando Landmarks e Odometria *31*

Capítulo 3 - Embasamento teórico 33

3.1. Projeto Flakey & o sistema Saphira 33

3.1.1. O projeto Flakey *34*

3.2. O sistema de controle robótico – Saphira 35

3.2.1. Arquitetura do Sistema *37*

3.2.1.1. Micro-tarefas do SO *37*

3.2.1.2. Rotinas do Usuário *38*

3.2.1.3. Comunicação de Pacotes *39*

3.2.1.4. Refletor de Estados *39*

3.2.2. A arquitetura de controle do robô *40*

3.2.2.1. Representação do Espaço *40*

3.2.2.2. Controle de Direção e Movimento *42*

3.2.2.3. Rotinas para Interpretação dos Sensores *42*

3.3. O robô Pioneer 1 43

3.4. O sonar Polaroid 6500 44

3.5. Filtro de Kalman 49

3.5.1. Funcionamento do Filtro de Kalman *50*

3.5.2. Filtro de Kalman em Modelos Estocásticos *56*

3.6. Redes Neurais Artificiais 58

3.6.1. Mapas Auto-Organizáveis (Self-Organizing Maps) *59*

Capítulo 4 - Um sistema para Localização 63

4.1. Visão geral do Sistema 63

4.2. Sistema de Navegação 64

4.3. Filtragem dos dados dos sonares 67

4.3.1. Utilizando o Valor Médio *68*

4.3.2. *Utilizando Filtro Kalman* 69
4.3.3. *Ajustando os parâmetros do Filtro* 71
4.4. Localização 72
4.4.1. *Sistema de Orientação* 72
4.4.2. *Odometria* 75
4.4.3. *A Rede de Kohonen* 76
Capítulo 5 - Discussão dos resultados 80
5.1. Utilizando Odometria 80
5.2. Utilizando Landmarks 84
5.3. Utilizando juntos Odometria e Landmarks 86
5.4. Trabalhos futuros 87
Referências bibliográficas 88
Apêndice A 91
Apêndice B 92

Lista de Figuras

Figura 1.1 - Robôs Industriais 3

Figura 1.2 - Robôs Móveis 4

Figura 1.3 - Diagrama de Estados.....9

Figura 1.4 - Interação Robô/Ambiente.....11

Figura 1.5 - A coerência do Modelo de Mundo.....12

Figura 2.1 - Dispositivos utilizados para navegação inercial. 17

Figura 2.2 - Esquema de um codificador óptico relativo 19

Figura 2.3 - Esquema de um codificador óptico absoluto. 19

Figura 2.4 - Ambiente : Sala 4x4.....21

Figura 2.5 - Robô navegando na sala 21

Figura 2.6 - Estabelecendo a correspondência entre o Mapa Local e o Global para um sistema que emprega o paradigma métrico. 22

Figura 2.7 - Exemplo de Landmarks Artificiais para sistemas baseados em visão. 25

Figura 2.8 - Exemplo de Landmarks Naturais para o sistema FINALE baseado em visão. 25

Figura 2.9 - Esquema geral para reconhecimento de Landmarks.....26

Figura 3.1 - Flakey: robô desenvolvido pelo SRI International 34

Figura 3.2 - Arquitetura do sistema Saphira 37

Figura 3.3 - Arquitetura de controle.....40

Figura 3.4 - Representação do espaço ao redor do robô no sistema Saphira. 41

Figura 3.5 - O robô Pioneer 1, equipado com uma câmera de vídeo.....43

Figura 3.6 - Polaroid Série 6500.....45

Figura 3.7 - Desenho esquemático de um transdutor eletrostático. 45

Figura 3.8 - Sistema de medição por ultra-som da Polaroid série 6500. 46

Figura 3.9 - Gráfico da amplitude máxima à 50kHz. 47

Figura 3.10 - Cone de propagação do sinal de ultra-som. 49

Figura 3.11 - Densidade condicional da posição baseada no valor z_1 51

Figura 3.12 - Densidade condicional da posição baseada apenas no valor z_2 51

Figura 3.13 - Densidade condicional da posição baseada nos valores z_1 e z_2 52

Figura 3.14 - Propagação da densidade condicional. 54

Figura 3.15 - Algoritmo do Filtro de Kalman 58

Figura 3.16 - Modelo estrutural de um Mapa Auto-Organizável. 60

Figura 3.17 - Determinação da vizinhança do neurônio vencedor. 61

Figura 3.18 - Atualização de pesos durante a apresentação de duas entradas consecutivas 61

Figura 4.1 - Ambiente do ALDER 65

Figura 4.2 - A Janela Virtual 66

Figura 4.3 - Corredor do Bloco 1 do ICMC 67

Figura 4.4 - Implementação do Filtro de Kalman 69

Figura 4.5 - Gráfico: Leituras do sonar x Saída do filtro 70

Figura 4.7 - Conversão de coordenadas entre dois sistemas de orientação. 73

Figura 4.8 - Sistemas Cartesianos de Orientação Local e Global. 74

Figura 4.9 - Relação entre a detecção sensorial e LPS. 75

Figura 4.10 - Odometria. 76

Figura 4.11 - Definição da classe que representa os neurônios. 77

Figura 4.12 - Definição da classe que modela a Rede de Kohonen.. 78

Figura 4.13 - Atualização da vizinhança, segundo cada um dos algoritmos. 78

Figura 5.1 - Deslocamento real vs Deslocamento medido. 81

Figura 5.2 - Saída gerada pelo sistema odométrico (4 tomadas consecutivas). 82

Figura 5.3 - Falha do sistema odométrico durante colisão. 83

Figura 5.4 - Estimativa odométrica durante um manobra de ré. 84

Figura 5.5 - Landmarks selecionados. 85

Índice de Siglas

ART2: Adaptative Resonace Teory

FRC: Field Robotics Center

GMS: Global Map Space

GPS: Global Position System

LABIC: Laboratório de Inteligência Computacional

LPS: Local Perceptual Space

MEF: Máquina de estados finitos

NREC: National Robotics Engeneering Consortium

RCE: Reduced Coulomb Energy

RIA: Robotics Industries Association

RNA : Rede Neural Artificial

RVL: Robot Vision Laboratory at Purdue University

SO: Sistema Operacional

SOFM: Self-Organize Maps

SONAR: Sound Navigation and Ranging

TCP/IP: Transfer Control Protocol / Internet Protocol

TOF: Time of flight

TTL: Transistor - Transistor Logic

Resumo

A capacidade mais importante de um sistema de navegação para um robô móvel, além de manter-se operacional e evitar colisões, é a que estabelece a sua própria posição em relação ao ambiente. A esta capacidade dá-se o nome de ***localização***, que consiste em atualizar a posição do robô no ambiente, tendo como base, as leituras provenientes dos sensores.

Talvez o fato mais importante obtido a partir do vasto conjunto de pesquisas sobre o tema localização de robôs móveis, seja que, até hoje, apesar de existirem muitas aplicações bem sucedidas, ainda busca-se uma solução elegante e robusta para o problema de localização.

As técnicas existentes para localização de robôs móveis podem ser classificadas nos seguintes grupos: técnicas baseadas em posicionamento relativo e as que se baseiam em posicionamento absoluto. Dentre as pertencentes ao primeiro grupo, destacam-se os métodos de localização odométricos e aos pertencentes ao segundo, os baseados em *landmarks*.

Este trabalho investiga o potencial de alguns algoritmos que implementam métodos odométricos e métodos baseados em *landmarks*, visando uma implementação híbrida dos mesmos. O desempenho dos métodos implementados foi avaliado no simulador Saphira e no robô móvel Pioneer 1, existente no Laboratório de Inteligência Computacional (LABIC), através de aplicações práticas.

Abstract

The most important attribute in a navigation robot system, beyond to keep the robot alive and operating, is the element that establishes the self location in relation to environment. This feature is called *localization*, which means to establish the robot's position (location) into the environment just based on the sensor readings.

It may be, the most important results to found with a lot of research about the theme mobile robots localization is that, until now, even thought we had very much successful applications, any solution was skill and robust enough to solve this problem.

The mobile robot localization algorithms can be classified into two groups: *Absolute Location* and *Relative Localization*. In this work, the potential of some odometric (relative localization) and landmark based (absolute localization) algorithms looking for a hybrid system, is investigated. The system performance was tested at Saphira Simulator and Pioneer 1 belonging to LABIC - USP, through convenient and practical experiments.

Capítulo 1 - Introdução

A literatura proporciona um espaço onde a mente tem total liberdade de criação, mesmo que as propostas apresentadas sejam, em sua época, fisicamente impossíveis ou impraticáveis. Obras classificadas como ficção científica constantemente mostram máquinas antropomórficas e auto-suficientes, dotadas de características inteligentes e capazes de apresentar uma desenvoltura parecida à dos seres humanos nas áreas mais características da natureza humana.

Por muito tempo uma das visões mais exploradas pela literatura e pelos filmes, foi aquela que projetava os robôs como substitutos do ser humano, máquinas superiores a nós em praticamente todos os aspectos, e muitas vezes, até por serem desprovidas de sentimentos, tornavam-se insurretas aos seus criadores, proporcionando futuros incertos para toda a raça humana.

Apesar disto tudo, nossa realidade se mostra bem diferente do que tais descrições. Na verdade, atualmente há inúmeras vertentes que a cada descoberta tecnológica ou biológica, evidencia mais e mais um caminho exatamente oposto a essa visão apocalíptica, principalmente por questionar os limites entre a criação de uma máquina com habilidades humanas e as características que nos tornam propriamente humanos.

1.1. O que são Robôs?

A motivação humana para a construção de robôs está longe do desejo de criar máquinas que visam substituir a existência humana, ou ainda, o de concretizar o desejo dos homens de ser um Deus, através da criação de uma nova existência.

O termo "Robô" foi introduzido em 1921 por uma das pessoas mais proeminentes da literatura tcheca, o romancista Karel Capek, através de seu livro "R.U.R. - Rossum's Universal Robots" [CAPEK,21].

Em sua obra, Capek descreveu uma máquina inteligente e de aparência similar aos seres

humanos, cujo papel era tornar a vida das pessoas mais agradável, pois esta máquina realizava todo o tipo de trabalho que nós humanos não gostamos de fazer.

Tal idéia teria um impacto extraordinário na comunidade humana. Com um mínimo de conhecimento a respeito de si próprio, qualquer pessoa é capaz de compreender quão oportuna seria a concretização de tal sonho. Em primeira instância, alguns poderiam dizer que isso tudo seria, no mínimo, uma situação cômoda e digna de um rei. Mas e se por ventura ousarmos permitir que nossa mente divague livremente? Certamente descobriríamos inúmeras aplicações, nas mais diversificadas áreas e subsidiadas pelos mais diferentes motivos.

Podemos imaginar esses dispositivos inteligentes operando em lugares extremamente inóspitos ao seres humanos, como na remoção de minas terrestres, regaste em ambientes sujeitos a desmoronamento, na exploração de planetas desconhecidos ou combatendo vazamentos em usinas nucleares, ao invés de, simplesmente, arriscarmos pessoas para a realização destes trabalhos.

Entretanto outras aplicações poderiam surgir que não sejam, necessariamente, determinadas pela valorização da vida, mas pela melhoria de sua qualidade. Temas que visam o conforto e bem estar do ser humano, como por exemplo, o de realizar tarefas tediosas como lavar louças e aspirar carpete, bem como, ser capazes de guiar um carro de forma segura, permitindo que o passageiro descanse ou tenha tempo para realizar tarefas mais importantes.

Motivações como essas, que por anos impeliram os escritores a criar máquinas com tais características em suas obras, agora impulsionava os estudiosos a um novo campo de pesquisa dentro da ciência da computação e da engenharia, hoje conhecida como *Robótica*, cujo objeto de estudo são dispositivos que percebem e manipulam o mundo físico.

A palavra e o conceito tornaram-se mais populares cerca de trinta anos depois, com os livros do escritor Isaac Asimov, que na década de cinqüenta, enunciou as leis fundamentais da robótica em sua obra "I, Robot" [ASIMOV,94]. As três leis eram:

> ***1ª Lei:*** *Um robô não pode machucar ou ferir um ser humano, ou, por falta de atitude, permitir que um ser humano venha a se ferir.*
>
> ***2ª Lei:*** *Um robô deve obedecer às ordens dadas por um ser humano, exceto quando essas ordens entrarem em conflito com a primeira lei.*
>
> ***3ª Lei:*** *Um robô deve proteger sua própria existência enquanto a garantia de sua proteção não entre em conflito com a primeira ou segunda lei.*

Tamanha era a genialidade de Asimov que, mesmo ele tendo enunciado estas três leis em um período muito antes da sedimentação da robótica como ciência, praticamente nada se alterou nelas, exceto pela inserção da lei zero, que estende a abrangência das leis, mudando o foco do indivíduo para a toda a humanidade.

Desde então, a definição para o termo robô sofreu inúmeras alterações acompanhando as inovações e descobertas proporcionadas pela Robótica. De acordo com a Robotic Industries Association (RIA) apud Arkin [ARKIN,99], um robô é: um manipulador multifuncional, reprogramável, projetado para mover: materiais, partes, ferramentas ou dispositivos especializados, através de diferentes movimentos programados de forma a realizar várias tarefas.

Os primeiros robôs a ganharem espaço em meio à humanidade certamente foram os industriais (Figura 1.1). Motivados pela desvalorização do ser humano em suas características mais elementares, bem como, dado o caráter repetitivo e de pouco ou nenhum esforço intelectual do trabalho nas linhas de produção industriais, visualizava-se ali um ambiente que desqualificava o ser humano para exercício da tarefa e, ao mesmo tempo, apresentava características que iam de encontro com as expectativas que fundamentaram a robótica.

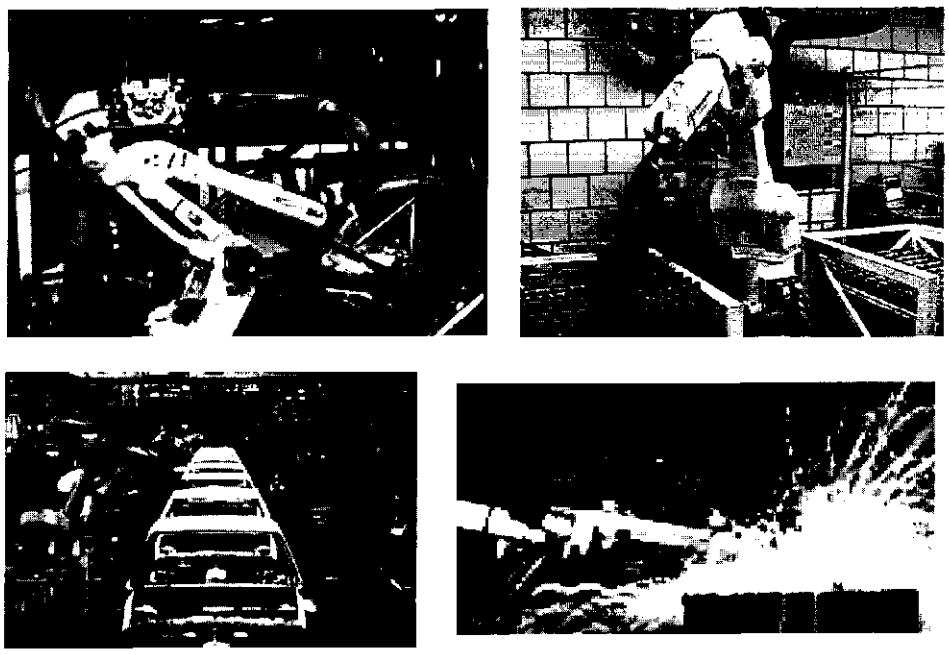


Figura 1.1 – Robôs Industriais

Apesar da infinidade de aplicações possíveis para os robôs, até esse momento da história, a utilização de robôs tinha um caráter de pura automação, como os manipuladores em uma linha

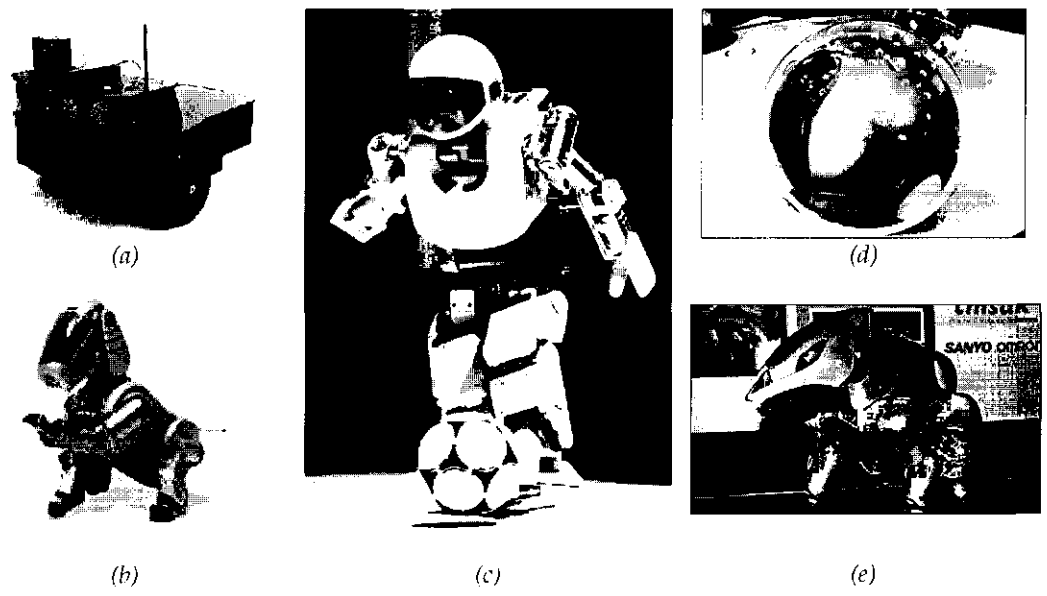
de montagem, desempenhando uma tarefa idêntica todos os dias.

Após este período de uma nova "revolução industrial", as aplicações dos sistemas robóticos têm evoluído para domínios diferentes e mais complexos. Conseqüentemente, a fim de se viabilizar qualquer estudo, é pertinente que se realize algum tipo de agrupamento baseando-se em determinados critérios, ou melhor, criar diferenciação segundo uma classificação.

Deste modo, um robô pode ser classificado partindo-se de diferentes pontos de vista, como: quanto à sua forma física, característica de ação, tipo de deslocamento, materiais do quais é constituído, tipo dos sensores, tipo dos atuadores, sistema de locomoção, sistemas computacionais, ou ainda, não se atendo a estrutura física podemos classificá-los quanto ao seu comportamento e sistemas de controle [ARKIN,99].

1.1.1. Robôs Móveis

Uma destas possíveis classes de robôs são os chamados *Robôs Móveis*, cuja característica fundamental é a capacidade de deslocarem a si mesmos.



(a) o robô Pioneer1, comercializado pela iRobot, utilizado neste projeto. (b) Primeira versão do AIBO, desenvolvida e comercializada pela Sony. (c) Humanóide ASIMO, desenvolvido pela japonesa Honda. Na coluna da direita, os robôs apresentados na Robodex 2002, onde (d) é um modelo fabricado pela Sony que recebe comando de voz e tem forma esférica para evitar obstáculos, e (e) é um robô de vigilância com formato de cão-de-guarda, que vem equipado com câmera de vídeo fabricado pela Sanyo.

Figura 1.2 – Robôs Móveis

Muitas vezes implementados sob a forma de plataformas com rodas, ou com dispositivos

parecidos com pernas, estes robôs são capazes de locomover-se em um ambiente evitando colisões com outros elementos inseridos no mesmo ambiente (Figura 1.2).

Dentro de toda esta gama que compõem os robôs móveis, ainda estamos interessados em uma subclasse formada pelos robôs móveis autônomos e com algum comportamento inteligente.

Existem inúmeras definições para robôs móveis inteligentes. De acordo com Arkin [ARKIN,99], um robô móvel inteligente é uma máquina capaz de extrair informações a partir do ambiente e usar este seu conhecimento sobre o mundo para mover-se de forma segura e significante.

Uma outra definição foi proposta por Ribeiro et al [RIBEIRO,01], onde descreve os robôs móveis como sendo agentes artificiais ativos, com capacidade de locomoção, imersos em um mundo físico real. *Artificiais*, por não fazerem parte da natureza; *ativos*, por atuarem no ambiente de forma racional e não passiva; e *com capacidade de locomoção* por poderem se locomover pelo domínio. Ao longo do texto que se segue, sempre que mencionada a palavra robô móvel, ela deverá ter o significado segundo as definições de robôs móveis inteligentes dadas acima.

1.2. Incerteza na Robótica

Em mundos reais, peculiares por suas constantes mudanças, como pessoas e carros transitando, objetos ocupando diferentes lugares, e toda a sorte de dinâmica característica deste ambiente em que vivemos, os atributos necessários para que um robô seja capaz de operar mudam drasticamente se comparados aos domínios apresentados anteriormente, como era o que se poderia dizer em relação aos estruturados e organizados ambientes das linhas de produção, onde uma determinada peça necessária à soldagem podia ser encontrada sempre no mesmo lugar e na mesma posição.

Deste modo, talvez o atributo mais importante que um robô móvel deva ter é a capacidade de operar em domínios que são, em sua natureza, imprevisíveis e de crescente desestruturação. A esta imprecisão e dúvida causadas por mudanças inesperadas e incontroláveis que acontecem no ambiente damos o nome de *incerteza* [THRUN,98].

O fator incerteza pode ocorrer sob cinco formas diferentes:

- *Nativas do ambiente* - o mundo físico é por sua própria natureza imprevisível, desde que são inúmeras as variáveis incontroláveis que estão presentes e ativas no domínio. Para ambientes bem estruturados o grau de incerteza é pequeno, entretanto para ambientes como casas, escolas e estradas, que são extremamente dinâmicos, o grau de incerteza cresce vertiginosamente.

- *Nas leituras dos sensores* - os dispositivos empregados para aferir grandezas do domínio da aplicação possuem suas próprias limitações, ou seja, os seus alcances e resoluções obedecem às leis da física, por exemplo, câmeras não podem captar imagens através de paredes, e ainda, a resolução da imagem tem seus limitantes. Outro aspecto importante reside no fato de que os sensores estão sujeitos a ruídos (perturbações externas) imprevisíveis, responsáveis por perturbações nos valores de suas medidas, fato este, bem costumeiro aos que utilizam sonares como sensores, uma vez que este tipo de sensor é extremamente suscetível a ruídos.

- *Inerentes aos dispositivos do robô* - dispositivos que compõem o robô podem ter seu comportamento alterado com o decorrer do tempo, isto é, mesmo que construídos com extrema precisão, mecanismos mecânicos como engrenagens e motores podem acumular sujeira ou sofrer desgastes durante o período de seu uso, provocando diferenças na sua velocidade de rotação ou posicionamento.

- *Nos modelos de mundo* – Modelos são imprecisos pelo próprio fato de serem abstrações do mundo real. É extremamente difícil obter um modelo completo o suficiente para representar todas as relações presentes em um ambiente, e, ainda se isto fosse possível, outro determinante seria a complexidade de sua construção, armazenamento e manutenção. Logo, todos os modelos são em sua essência simplificações do mundo real capazes de preservar características que sejam relevantes à aplicação.

- *Característica ao processo computacional* – os sistemas computacionais possuem limitações e, muitas vezes, os processos físicos devem ser adaptados de maneira que sua implementação computacional obedeça a essas limitações, isto é, tais adaptações muitas vezes são obtidas através da utilização de métodos de aproximação, o que penaliza todo o sistema com um erro intrínseco a própria aproximação.

Anteriormente a questão da incerteza era praticamente ignorada pela robótica, mas com a crescente expansão da área de aplicação de robôs através da penetração em ambientes cada vez

mais desestruturados e dinâmicos, essa questão tem se tornado um elemento decisivo para o sucesso de qualquer aplicação neste tipo de domínio.

1.3. Representando o mundo

Típicos na literatura de robôs móveis, os termos *mundo, ambiente* e *meio* denotam o conjunto compreendido por: [WEBSTER]

i. *A área em que alguma coisa vive ou existe; ou*

ii. *A totalidade das coisas que cercam ou estão ao redor; ou ainda*

iii. *As condições, influências e forças que interferem ou modificam os elementos contidos nele.*

Algumas propriedades do ambiente como condições, influências e forças especificam grandezas e podem ser quantificadas, servindo-nos, em muitos casos, como sinalizadores de que o meio sofreu alterações. O *estado de um ambiente* pode ser definido como um conjunto de todas as grandezas mensuráveis quantificadas do ambiente em um instante de tempo determinado [THRUN,00].

Obviamente, quando estes conceitos são aplicados em sistemas computacionais, algumas dessas grandezas são irrelevantes em relação à aplicação e, portanto, podem ser desprezadas sem penalidades, de maneira a adequar o foco da análise dos estados do ambiente tão somente àquelas características que expressem alguma contribuição para norteamento da execução da aplicação.

Por motivos de simplificação, nos referimos a todas essas diferentes entidades que compõem um ambiente como sendo, cada uma delas, uma *propriedade* ou *característica* distinta deste ambiente, independentemente do seu tipo ou grau de contribuição.

O estado de um ambiente é unicamente caracterizado, ou seja, não existem dois estados diferentes que sejam igualmente quantificados – com configurações iguais – para um mesmo ambiente.

Similar a uma fotografia, um estado *representa* as entidades, disposições, condições, e todos os outros elementos pertencentes ao meio (propriedades ou características do ambiente), segundo as circunstâncias em que se encontram para um determinado instante de tempo.

Assim sendo, qualquer alteração nas características do ambiente, mesmo que isso ocorra em tão somente uma delas, implicaria em uma mudança para um novo e diferente estado.

De acordo com esses fatores de alteração, ou melhor, segundo os elementos que incitam as transições entre os diversos estados, podemos diferenciar os ambientes em duas categorias: os *ambientes estáticos* e os *ambientes dinâmicos*.

Ambientes estáticos são todos aqueles que permanecerem inalterados até que seja realizada uma ação por parte do robô, ou seja, nenhum outro fator altera os relacionamentos e objetos do mundo (estado), a não ser o próprio robô durante sua interação com o meio.

Entretanto, quando um ambiente sofre alterações motivadas por outros fatores que não apenas as ações do robô, ou seja, mesmo no caso em que o robô não desempenhe nenhuma ação o ambiente ainda poderia sofrer uma mudança de estado, então a esses denomina-se *ambientes dinâmicos* [BORENSTEIN et al,96].

Em se desejando que o robô interaja com o mundo de forma inteligente e hábil torna-se necessário a ele compreender, da melhor maneira possível, o meio onde está inserido. A modelagem do mundo é uma fase importante, pois esta etapa consiste na criação de um protótipo que reflita ao máximo a realidade do ambiente, representando o conhecimento sob uma forma que capacite o robô à tarefa de inferência e seja, ainda, computacionalmente viável. É, portanto, neste modelo que o robô armazena todo o conhecimento que possui acerca do ambiente ao seu redor.

A interferência humana nesta etapa pode comprometer consideravelmente o desempenho global, visto que os nossos sistemas de percepção – visão, audição, tato, olfato e paladar – ainda que por muitas vezes sejam parecidos com os sensores utilizados na robótica, não tem seus "sinais" processados da mesma maneira.

A situação inversa também ocorre, não obstante, robôs equipados com sensores de alta sensibilidade são capazes de captarem pequenas nuances do ambiente que são imperceptíveis às pessoas. Conseqüentemente, um modelo estabelecido por qualquer entidade, que não o próprio robô, inicialmente tende a não ser ótimo.

1.3.1. Diagramas de estado

Uma das abordagens utilizadas para a modelagem de mundos está fortemente apoiada no conceito de estados, conforme definido no tópico anterior, uma vez que estes quantificam

grandezas do mundo em instantes de tempo.

Independentemente do tipo de ambiente que está sendo modelado (estático ou dinâmico), podemos descrevê-lo através de um conjunto de instâncias assumidas por cada uma de suas propriedades ao longo do tempo (estado), garantindo sempre que um estado seja unicamente definido.

Assim sendo, toda alteração sofrida por um ambiente será representada como uma transição que parte de um estado e leva para outro. É importante lembrar que em se tratando de um modelo criado para representar um ambiente dinâmico, as transições entre os estados não serão determinadas tão somente pelas ações realizadas pelo robô, mas também por fatores externos. Logo, tais influências precisarão ser consideradas pelo modelo.

Suponha um ambiente fechado qualquer que contenha, dentre outros objetos, uma fonte de luz e um interruptor capaz de ligá-la e desligá-la. Em nossa modelagem, a única característica relevante deste ambiente será a fonte de luz, podendo então, serem descartadas todas as outras características do ambiente.

Em um segundo momento, define-se as possíveis configurações (estados) que este ambiente pode assumir em relação à característica escolhida: estar com a luz acesa (estado A) ou estar com a luz apagada (estado B).

A atividade (ação) de apagar e acender a fonte de luz é um dos fatores causadores de alterações no ambiente, determinando assim, as transições entre os estados A e B, conforme mostrados na Figura 1.3.

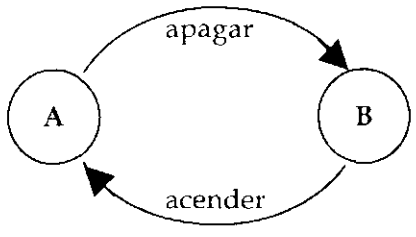


Figura 1.3 – Diagrama de Estados

Essa mesma abordagem pode ser expandida para modelar mundos mais complexos, o que implica no envolvimento de vários estados possíveis, bem como, inúmeras transições para cada um destes estados.

Atualmente, essa técnica é bastante utilizada em robôs reativos, cuja característica é ter suas ações ou respostas determinadas por estímulos, sem antes sofrerem um pré-planejamento

[ARKIN,99].

1.3.2. Mapeamento

Uma outra abordagem para o problema da criação de modelos de mundos é a construção de mapas. Esta solução parece ser muito mais próxima e confortável, uma vez que esta é uma das maneiras como nós compreendemos o mundo que nos cerca.

A complexidade de um mapa é diretamente proporcional à quantidade de características relevantes do ambiente que estão sendo modeladas, assim como os fatores limitantes para o desempenho do sistema ficam a cargo dos métodos empregados para aquisição, armazenamento e recuperação das informações.

Segundo Owen [OWEN,96], a atividade de construir um mapa é, na realidade, a criação de um modelo de mundo através de dados coletados pelos sensores, podendo ser classificada em um destes dois grupos:

- *Mapas Métricos:* consiste em confeccionar um modelo de mundo através da criação de um mapa que contém informações métricas a respeito dos elementos que compõem o ambiente. Esta abordagem conta com a vantagem de ser uma representação simples e facilmente inteligível para os seres humanos. Entretanto, devido à quantidade de detalhes que estes mapas possuem, demandam um consumo de tempo muito grande para serem construídos e, muitas vezes, alguns destes detalhamentos são irrelevantes para a tarefa de navegação geral. O emprego desta técnica permite que o mapa de qualquer meio em que o robô esteja inserido possa ser criado pelo próprio robô ou fornecido, inicialmente, por um projetista.

- *Mapas Topológicos:* nesta outra abordagem, o meio é modelado sob a forma de nós que representam as diferentes localidades que compõem o ambiente, e arcos interligando esses nós, que por sua vez, correspondem aos caminhos que conectam uns locais aos outros. Na realidade, este tipo de mapeamento consiste na criação de uma estrutura do tipo grafo que correspondente ao ambiente. Esta representação é bem compacta uma vez que codifica somente lugares distintos dentro do meio. Adicionalmente, este tipo de mapa é apropriado para o uso de vários algoritmos que tem sido desenvolvido atualmente para o planejamento de rotas. Talvez, o principal problema com este método seja a ocorrência do fenômeno denominado: ambigüidade

perceptual (*perceptual aliasing*), onde localidades distintas dentro do ambiente são entendidas pelos sensores do robô de maneira idêntica. Uma técnica proposta como solução a deste problema é o emprego da chamada fusão de sensores, que consiste em agregar as leituras de diversos dispositivos, muitas vezes até de diferentes modalidades, a fim de se obter um diferencial dos dados nos casos de ambigüidade e, também, uma maior precisão. Apesar de tudo, na prática essa técnica tem sido mais utilizada para reduzir a ocorrência da ambigüidade perceptual do que para eliminá-la realmente. De qualquer forma, outra maneira para solucionar o problema da ambigüidade perceptual é a adoção de diferentes técnicas para se estabelecer o posicionamento do robô em relação ao ambiente.

1.4. A interação entre o robô e o ambiente

Mesmo que da maneira mais simples, como ocupar uma posição no espaço, até a forma mais intensa e frenética de relacionamento entre os elementos que formam um meio, toda e qualquer entidade, nele imerso, influencia e sofre influência dos outros integrantes. Chamamos de *interação*, esta influência mútua que ocorre entre duas entidades.

Conforme definido no item 1.1.1, um robô móvel inteligente é uma entidade com capacidade de locomoção ativa dentro do meio em que está inserido. Diante deste fato, Thrun [THRUN,00] define que a interação entre um robô móvel e seu ambiente pode ocorrer em duas direções, conforme ilustrado na Figura 1.4.

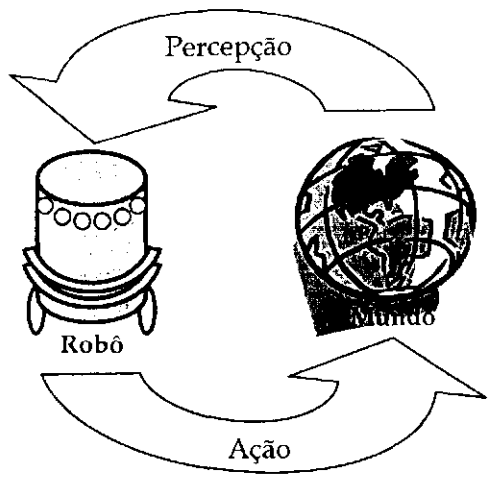


Figura 1.4 – Interação Robô/Ambiente

- *Ambiente → Robô:* ocorre sempre durante a fase de percepção do mundo. Também chamada de *observação, medição* ou simplesmente *percepção*, a forma de interação perceptual compreende toda e qualquer tarefa de coleta de informações a respeito do ambiente realizada através das leituras dos sensores do robô. É, portanto, nesta modalidade de interação que ocorre a modelagem do mundo, a interpretação e a identificação do seu estado atual.

- *Robô → Ambiente:* esta forma de interação ocorre sempre que o robô realiza um ato qualquer que implique em uma alteração de uma ou mais características do ambiente, ou seja, sempre que uma ação proveniente do robô originar uma transição entre os possíveis estados. São exemplos deste tipo de interação: o ato de locomover-se dentro do ambiente e o de manipular um objeto.

Independentemente da direção em que ocorre o processo interativo, as informações resultantes devem ser coletadas para que se possa aumentar o conhecimento que o robô possui do mundo em está inserido. Seja este conhecimento obtido através de dados fornecidos diretamente pelos sensores ou através de suposições derivadas das ações realizadas pelo robô, para que o modelo mantido pelo robô seja fiel ao estado corrente do ambiente, todas as interações entre o robô e o ambiente devem ter uma correspondência no modelo interno (Figura 1.5).

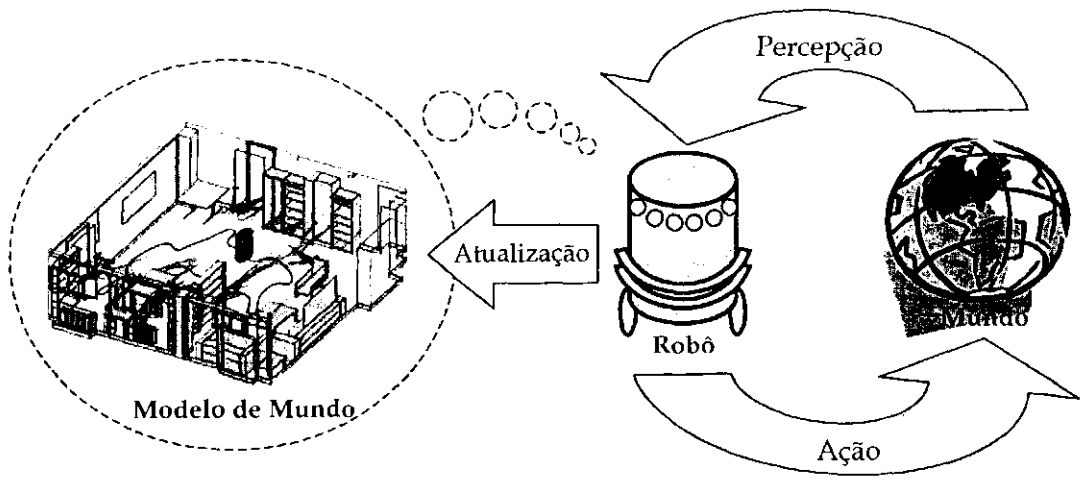


Figura 1.5 - A coerência do Modelo de Mundo

Diferentemente do que se pode pressupor inicialmente, não é somente durante a fase de percepção que as informações referentes ao estado do ambiente são adquiridas. Ora, se cada ação determina uma transição entre dois estados (estado E1 → ação → estado E2), pode-se conjecturar a nova configuração que o ambiente assumirá apenas levando em conta a transição

determinada pela execução de uma ação.

Para ilustrar melhor essa idéia, suponha um corredor com uma porta em uma de suas extremidades e que um robô se encontre em frente à porta, distando quatro metros dela. Sabendo que o robô se deslocou três metros em direção a porta, pode-se supor que, após o término da execução do movimento, o robô estará exatamente a um metro da porta.

Observe que essa informação é obtida simplesmente por meio da análise das conseqüências de uma ação executada, sem que haja a necessidade de se realizar nenhuma leitura dos sensores. É importante ressaltar que neste processo foram consideradas apenas as características que compunham o movimento e a confirmação de sua execução. Então, se por ventura houver uma derrapagem durante o deslocamento, pode ser que o robô diste da porta mais do que o esperado.

1.5. Navegação

> ***"Navigare necesse; vivere non est necesse" - latim, frase de Pompeu, general romano, 106-48 aC, dita aos marinheiros amedrontados, que recusavam viajar durante a guerra, cf. Plutarco, in Vida de Pompeu.***

A memorável frase de Fernando Pessoa "navegar é preciso, viver não é preciso", denota que a tarefa da navegação é, antes de tudo, um ato que demanda exatidão, perfeição ou nas palavras do próprio autor, é uma questão de precisão e não somente de necessidade.

Historicamente, a necessidade sempre foi um fator motivador para que homens e animais se lançassem em viagens exploratórias em seu meio ambiente. Muitas vezes impelidos pela falta de recursos locais, como água e comida, viam-se obrigados a sair em busca destes recursos em outras regiões.

Para realizar tal tarefa, era necessário que esses indivíduos fossem aptos a encontrar um caminho que os levassem até regiões fartas, optar por rotas alternativas quando diante de obstáculos naturais e ainda serem capazes de voltar ao seu local de origem para o trato de sua prole.

De acordo com a Enciclopédia Britânica [BRITÂNICA], *navegação* é a ciência que estuda o direcionamento através da determinação da posição, da rota e da distância a ser percorrida,

tendo como objetivos alcançar quatro pontos básicos: *selecionar uma rota ou curso, evitar colisões com objetos fixos ou em movimento, minimizar o consumo de energia e obedecer às restrições de tempo impostas.*

Agora, enfrentando essas mesmas dificuldades, o uso de robôs móveis tem a finalidade de poder realizar uma variedade de tarefas mais complexas que seus antecessores, os robôs industriais. Certamente, uma parte extremamente relevante nesta interação é a realizada pelo *Sistema de Navegação*, que engloba um conjunto de métodos e procedimentos dos quais o robô faz uso para se locomover e encontrar seu caminho dentro do ambiente em que se encontra.

Assim sendo, para que um robô possa mover-se com desenvoltura, deverá estar equipado com um sistema de navegação robusto, que seja capaz de traçar uma rota de um ponto até um outro ponto qualquer de destino, guiando-o de forma segura, para si e para os outros elementos do meio.

1.6. Localização

De nada adianta ter um sistema de navegação com alto poder de roteamento e apto a desviar de obstáculos que se movem, se o robô não for capaz de estabelecer posições ou localidades dentro do ambiente.

Localização é um item chave para qualquer sistema de navegação, visto que para desempenhar as tarefas típicas à navegação, um robô móvel deve ter uma noção precisa da sua própria posição espacial em relação ao ambiente que o cerca.

Atualmente, apesar da literatura acerca do tema localização de robôs móveis ser bastante vasta, toda técnica empregada para solucionar este problema, pode, de algum modo, ser classificada em uma destas duas categorias:

- *Localização Relativa:* consiste em determinar a posição corrente expressando-a em relação a um ponto pré-estabelecido tomado como referência. Isto é, a localização é obtida a partir de uma estimativa baseada nos deslocamentos do robô em relação a um ponto fixo determinado. Em muitas implementações deste método, os projetistas optam por referenciar o chamado ponto de início, que é a localização do robô no momento da ativação de seus sistemas, a ter que escolher qualquer outro ponto do ambiente.

- *Localização Absoluta:* esta, por sua vez, define qual é a posição atual sem a necessidade de informações a respeito de um ponto referencial único, ou seja, a posição corrente não é expressa sob a forma de uma relação entre a localização atual e a inicial. Na verdade estes métodos empregam técnicas como: a observação do ambiente em busca de características conhecidas ou sinalizadores, comparações entre o modelo de mundo e o estado atual, ou ainda, um sistema de posicionamento global. Segundo Burgard et al [BURGARD,96], essa técnica pode ser utilizada para determinar a posição inicial de um robô que emprega o método de localização relativa.

Existem inúmeras abordagens para o problema de localização, cada qual com seus próprios problemas e soluções. Do ponto de vista estatístico, por exemplo, a tarefa de localizar um robô em um ambiente é considerada um problema de predição [THRUN,00]. Por sua vez, algumas abordagens diferem tão somente pela maneira de representarem o mundo.

Neste trabalho foi realizado um estudo dos principais algoritmos para localização de robôs móveis e alguns destes foram escolhidos para implementação na linguagem C/C++, visando à utilização dos mesmos no robô Pioneer 1, existente no LABIC.

Por ser este um robô simples e também, possuindo algumas limitações, buscou-se entre os algoritmos estudados, aqueles que viabilizavam a sua implementação e teste neste robô. O objetivo a ser atingido é que o robô consiga se localizar em um ambiente proposto, de modo satisfatório e baseado nos algoritmos implementados.

O presente trabalho está estruturado da seguinte forma: neste capítulo, foram apresentados alguns dos principais problemas enfrentados na área de robótica móvel. No capítulo 2, é abordado, com mais detalhes, a questão da determinação da localização do robô em relação ao ambiente em que está inserido. Os trabalhos relacionados com o tema, incluindo algumas das pesquisas mais atuais, são brevemente relatados neste capítulo. O capítulo 3 é inteiramente dedicado à explanação de diversas técnicas que são empregadas neste trabalho. No capítulo 4 são apresentadas as propostas e soluções para resolver os problemas citados. Finalmente, no capítulo 5, são apresentados e discutidos os experimentos realizados e trabalhos futuros.

Capítulo 2 - Localização de Robôs Móveis

Internamente, um robô deve possuir uma descrição ou estrutura que represente, através de um modelo inteligível, o estado em que se encontra o ambiente, de maneira que o robô possa compreender os objetos que constituem este ambiente, bem como suas disposições.

Determinar a localização de um robô no ambiente em que ele está inserido é um requisito fundamental para qualquer sistema de navegação, pois ao interagir com o meio, o robô tornar-se-á parte da descrição do estado, o que implica, em certos casos, ser a entidade motivadora das transições entre as possíveis configurações do ambiente.

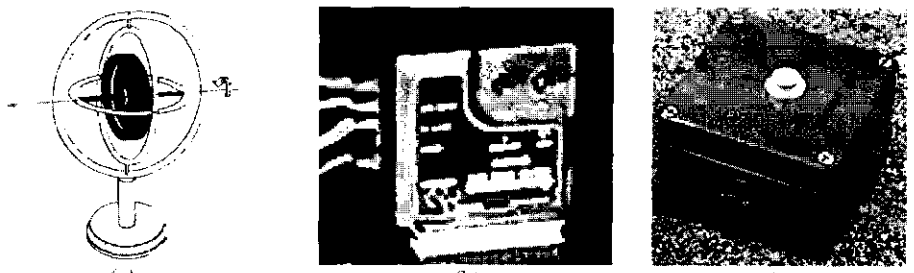
Como já mencionado anteriormente, todas as diferentes abordagens para se solucionar o problema de localização, até hoje, de uma forma ou de outra estão classificadas em um dos dois grupos seguintes:

2.1. Posicionamento Relativo

Também conhecido como método de ajuste de posição, onde a nova localização do robô é dada em relação a uma posição anteriormente conhecida, através de cálculos realizados sobre os valores dos deslocamentos produzidos pelos movimentos do robô. Uma característica peculiar a essas técnicas, é que elas são consideradas auto-suficientes [BORENSTEIN et al,96]. Atualmente existem dois ramos de pesquisas mais difundidos que adotam este paradigma:

- *Odometria:* é um método onde se produz uma estimativa da localização do robô a partir de medições provenientes de decodificadores capazes de medir a rotação e/ou a direção das rodas [NEHMZOW,92], conforme poderá ser visto em maiores detalhes na seção 2.1.
- *Navegação Inercial:* não tão popular quanto o método odométrico provavelmente devido aos altos custos dos sensores necessários, esta técnica fundamenta-se nos conceitos de movimento descritos pela física clássica. Através da utilização de

equipamentos habitualmente empregados na indústria aeronáutica, como giroscópios e acelerômetros (Figura 2.1), que são responsáveis por medir a taxa de rotação e a de aceleração de um corpo, respectivamente, a estimativa da localização é obtida a partir de cálculos das equações que descrevem os movimentos para os valores aferidos por esses dispositivos [BORENSTEIN et al,96].



(a) (b) (c)

(a) Desenho esquemático representando o princípio de construção de um giroscópio e na imagem(b) um desses dispositivos como é vendido no mercado. (c) Um dos diversos modelos de acelerômetros comercializados atualmente.

Figura 2.1 – Dispositivos utilizados para navegação inercial.

2.2. Posicionamento Absoluto

Característicos pela não necessidade de se pré-estabelecer uma posição inicial, tal como ocorre com o posicionamento relativo. A localização do robô é determinada a partir da observação e/ou captação de referências conhecidas que estejam presentes no ambiente ao redor do robô. Atualmente, a maior parte dos trabalhos em que se aplicam estes conceitos podem, certamente, ser agrupados em uma das vertentes abaixo:

- *Balizamento Ativo:* esse método computa a posição absoluta do robô através das medidas de sinais direcionais provenientes de três ou mais fontes de emissão diferentes, em posições conhecidas do ambiente. Consiste na utilização de dispositivos que fornecem um sinal de referência como baliza, muito similar à idéia dos faróis litorâneos, que orientam os navios próximos à costa. Aplicações desta natureza, muitas vezes empregam rádio freqüência, sinais luminosos pontuais ou Global Position System (GPS), como é o caso em [BULUSU,00].
- *Posicionamento baseado em mapas:* neste método a informação adquirida a partir dos sensores é comparada com um mapa ou um modelo do ambiente. Se as características extraídas dos sensores combinarem com o modelo interno, então a localização

absoluta do robô pode ser estimada em relação ao modelo global. Exemplos de utilização desta técnica podem ser encontrados em [SIMMON et al, 95] e [BORENSTEIN et al,96].

- *Reconhecimento de Landmark:* baseando-se na percepção de marcas peculiares e distintas do meio, denominadas landmarks, cujos posicionamentos são conhecidos no ambiente, esta técnica fornece uma estimativa da posição do robô através da observação de um ou mais landmarks e do posterior estabelecimento de uma correlação entre a posição deste landmark e os prováveis locais que permitem que ele seja observado. Em seu trabalho Owen [WIJK,98b] estima a posição do robô a partir da detecção de três landmarks, estabelecendo assim uma triangulação que lhe fornece uma precisão à estimativa do posicionamento.

A seguir métodos que aplicam alguns dos paradigmas definidos acima são apresentados.

2.3. Ajuste de Posição usando Odometria

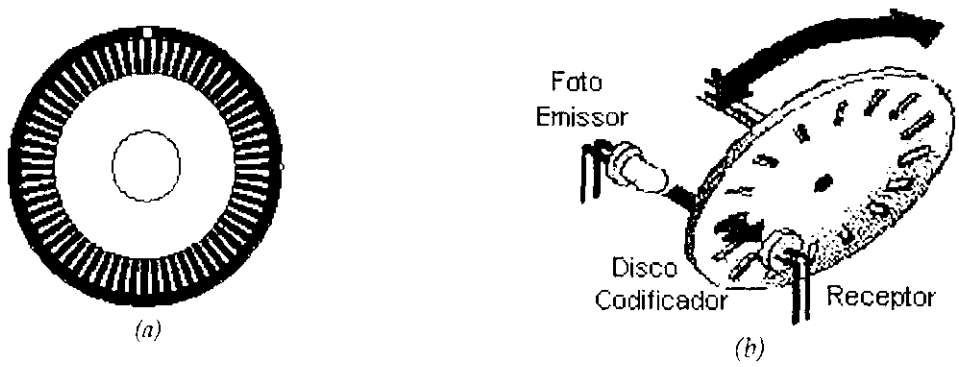
A odometria consiste em um procedimento matemático simples capaz de determinar a localização atual de um corpo, fundamentando-se apenas em informações a respeito de uma posição previamente conhecida e nos dados fornecidos por um dispositivo a cerca dos deslocamentos realizados ao longo de um percurso.

Os dispositivos odométricos, ou no original em inglês, *encoders*, mais comumente utilizados são foto codificadores acoplados diretamente aos eixos dos motores ou engrenagens, capazes de medir variações angulares e, conseqüentemente, medir o grau de rotação do eixo a que estão acoplados.

Essa técnica é a mesma empregada em periféricos como o mouse. O sistema é formado por um foto emissor e um foto receptor, ambos devidamente alinhados. Entre eles, fixo ao elemento que se deseja monitorar, se encontra um disco codificado com um padrão opaco/transparente.

O dispositivo foto emissor envia um feixe de luz ininterrupto na direção do sensor foto sensível. Entretanto, devido ao padrão codificado no disco, este feixe poderá ou não ser bloqueado. Com a execução de um movimento, o eixo ou a haste gira e, juntamente com ela, o disco que está preso. Um pequeno circuito de apoio interpreta os sinais captados pelo foto receptor e gera a resposta do dispositivo. Dois tipos diferentes de codificadores ópticos são

implementados: o *incremental* (Figura 2.2) e o *absoluto* (Figura 2.3).

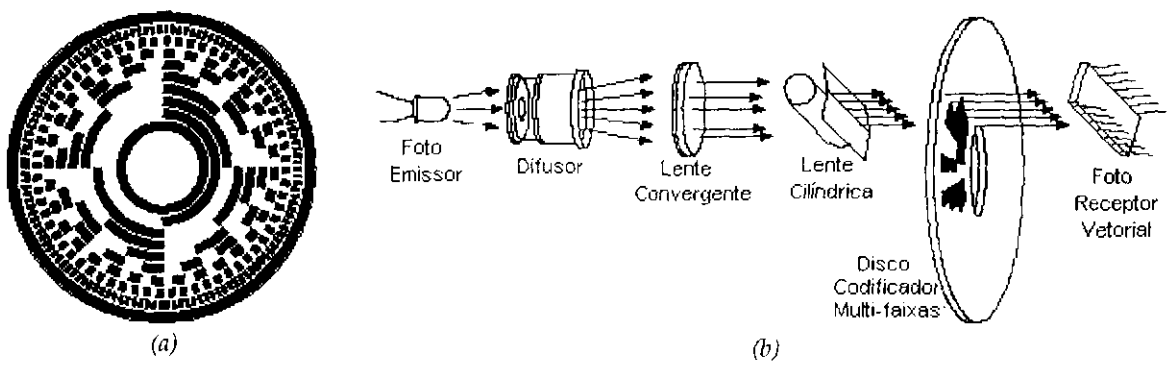


(a) *(b)*

(a) Disco óptico incremental empregado em um sistema óptico relativo.
(b) Esquema simplificado onde se pode observar os diversos elementos que compõe o sistema.

Figura 2.2 – Esquema de um codificador óptico relativo

Um encoder óptico incremental possui um disco com uma única faixa codificada através de ranhuras radiais igualmente espaçadas umas das outras (Figura 2.2a). Ao girar, a luz é interrompida com uma freqüência proporcional a velocidade de rotação (velocidade angular). Assim sendo, se observarmos a taxa de intermitência da luz é possível calcular o deslocamento linear da roda ou da engrenagem a qual o encoder está acoplado (Figura 2.2b). O valor resultante é incrementado ou decrementado de acordo com o sentido do movimento do disco, ou seja, sentido horário ou anti-horário.



(a) *(b)*

(a) Um disco codificando em suas oito faixas, os valores de 0-255 em binário.
(b) Desenho esquemático de um sistema óptico absoluto, com destaque para o foto receptor vetorial.

Figura 2.3 – Esquema de um codificador óptico absoluto.

Já o encoder óptico absoluto possui um disco com múltiplas faixas codificadas (Figura 2.3a), cada qual responsável por uma fração do valor de saída. A resposta do dispositivo é uma composição dos valores obtidos a partir dos diversos sinais captados por um foto sensor vetorial. A luz parte do foto emissor e passa por diversos elementos até que sejam formados vários feixes de luz paralelos e co-planares, como se fosse uma folha luminosa, incidindo cada feixe com uma das faixas do disco codificado. O foto receptor, então, capta as várias faixas de

luz que atravessam o disco compondo um vetor binário, cujo valor é convertido diretamente, gerando a resposta do dispositivo (Figura 2.3b).

Este tipo de encoder fornece sempre o mesmo valor absoluto para cada um dos diversos ângulos que o disco assume em relação aos fotos sensores. Observe que diferentemente dos codificadores incrementais, após uma volta completa do disco, os valores fornecidos pelo dispositivo começam a se repetir, forçando o usuário a realizar por conta própria o controle do número de voltas do disco. Sendo assim, este tipo de enconder é mais utilizado para o controle rotacional de câmeras e outros elementos que não realizam voltas completas em torno de seus eixos.

Uma característica geral e peculiar ao método odométrico é o seu baixo custo de implementação, uma vez que encoders são dispositivos muito baratos quando comparados com câmeras de vídeo, lasers ou sonares.

Como a idéia fundamental da odometria é a integração contínua e incremental da informação que descreve o movimento ao longo do tempo, é inevitável o acúmulo progressivo de erros locais a cada iteração. Além de erros sutis, o método odométrico também está sujeito à ocorrência de erros potencialmente críticos, como por exemplo, uma derrapagem.

Segundo Borestein et al [BORENSTEIN et al,96], existem basicamente duas categorias em que podemos classificar qualquer um dos erros odométricos:

- *Erros sistemáticos:* são assim classificados por serem erros cuja quantificação é proporcional ao longo do tempo e sua ocorrência segue uma freqüência, ou seja, as diferenças entre os valores reais e os valores esperados são proporcionais a cada amostragem de dados, sendo possível então, calcular de antemão o erro para a próxima amostragem. São exemplos deste tipo de erros: medidas informadas diferentes das medidas reais (diâmetro das rodas e distância entre eixos), rodas desalinhadas, bem como, codificadores com pequena precisão ou com taxa de amostragem menor que a velocidade do eixo a que está acoplado.
- *Erros não sistemáticos:* em contraposição aos erros sistemáticos, estes ocorrem em instantes de tempo aleatórios e imprevisíveis. São classificados como erros não sistemáticos: um giro em falso das rodas durante as acelerações ou por conta da falta de aderência com um piso escorregadio; derrapagens em curvas; falta de contato do encoder com o solo; irregularidades na pista, dentre outros.

Erros sistemáticos são particularmente graves devido a sua natureza acumulativa ao longo das iterações, entretanto, dado que sua ocorrência segue a uma cadência pré-estabelecida, estes também podem ser mais facilmente corrigidos, uma vez que tem valores constantes. Talvez o maior problema dos métodos odométricos sejam os erros não sistemáticos, visto que ocorrem inesperadamente e podem gerar grandes erros de posicionamento, bastando para isso uma única ocorrência.

Ilustrativamente, suponha que um robô móvel esteja em uma sala e deseje se movimentar por ela descrevendo uma trajetória em U que passe pelos pontos A, B e C, conforme exibido na Figura 2.4.

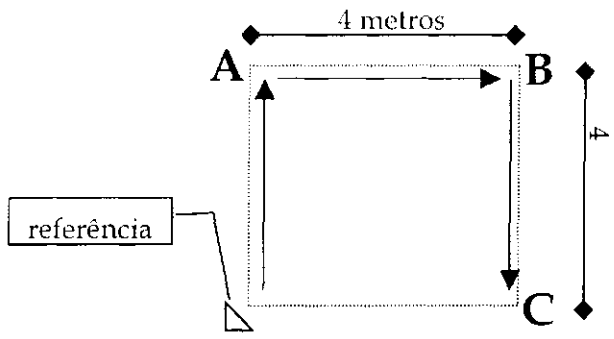


Figura 2.4 - Ambiente : Sala 4x4

Sabendo-se que sua posição atual é um dos cantos da sala (Figura 2.5a), podemos decompor o movimento em três etapas iguais que consistem, cada uma delas, em um deslocamento de 4 metros para frente, seguido imediatamente por um giro de 90º. Supostamente, ao realizar esses movimentos o robô alcançará o ponto C, conforme desejado.

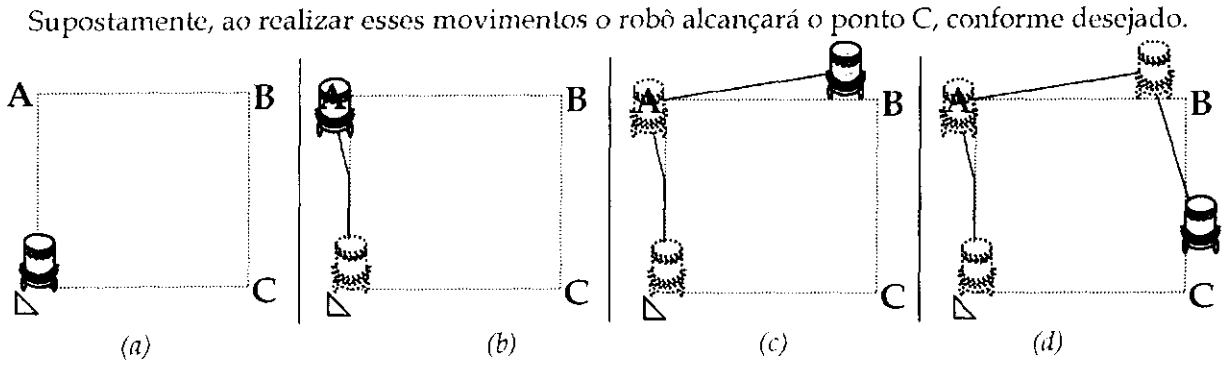


Figura 2.5 - Robô navegando na sala

Devido ao fato da faxineira encerar o piso da sala constantemente, ele é extremamente liso, o que acaba por provocar uma derrapagem da roda esquerda durante a execução do primeiro deslocamento de 4 metros, que corresponde à primeira etapa do movimento (Figura 2.5b). Observe que, nesta situação, o *encoder* acoplado a roda esquerda registra o seu giro, o que será entendido pelo sistema como um deslocamento linear de alguns centímetros, entretanto, devido

ao deslizamento, a medida real é diferente da medida que foi registrada.

Ao final da execução das três etapas que compõem o movimento proposto (Figura 2.5d), o robô "acredita" estar na posição C, apesar de seu posicionamento real ser outro. Ao longo do tempo, após inúmeros movimentos, o erro acumulado pode ser muito grande dada à ocorrência de fatos desta natureza.

Mesmo que aparentemente ineficiente se apresentado como um único método para a estimativa de posicionamento, os modelos odométricos ainda são largamente utilizados para localização de robôs móveis, provavelmente por suas características de baixo custo e pouca demanda de recursos computacionais.

Não obstante, ainda são empregados por muitos pesquisadores em sistemas híbridos, mostrando-se viáveis principalmente quando associados a outras técnicas de localização. Sua utilização pode ter uma contribuição significativa no desempenho global do sistema, pois permite um espaçamento maior entre estimativas de posicionamento absoluto, que são computacionalmente custosas.

2.4. Posicionamento Baseado em Mapas

Também chamada de *map matching* [BORENSTEIN et al,96], esta é uma técnica na qual o robô utiliza, conjuntamente, mapas locais e mapas globais para estimar a localização absoluta.

Através dos dados coletados por seus sensores, o robô constrói um mapa local do ambiente. Este mapa é, então, comparado com um mapa global que deve ter sido armazenado previamente na memória. Quando uma correspondência ocorre (*matching*), então o robô está apto para computar sua localização e orientação em relação ao ambiente.

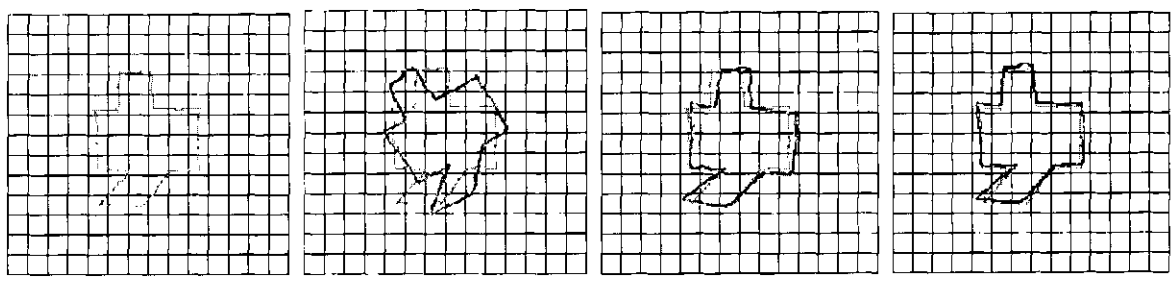


Figura 2.6 - Estabelecendo a correspondência entre o Mapa Local e o Global para um sistema que emprega o paradigma métrico.

No conjunto de imagens que formam a Figura 2.6, pode-se observar uma seqüência de quatro imagens representando os diversos instantes que compreendem o processo para se estabelecer às correlações entre o mapa local (traço em azul) e o mapa global (traço em vermelho).

O exemplo que ilustra o processo de matching descrito anteriormente (Figura 2.6) opera utilizando mapas baseados em grids [BURGARD,96], mas observe que aplicação deste método independe da técnica de mapeamento empregada, uma vez que a busca por correlações entre o mapa local e o mapa global pode ser realizada tanto para os mapas métricos quanto para os mapas topológicos.

Assim sendo, a posição do robô é computada de forma que sejam minimizadas as diferenças das distâncias entre os pontos do mapa local e suas características correspondentes no mapa global. Baseado na sua nova posição, as correspondências são recalculadas e o processo se repete até que o valor do erro agregado à distância entre pontos e segmentos de reta seja mínimo.

2.5. Landmarks

Certamente, o que seria mais pertinente a este tópico antes de qualquer explanação sobre o método de localização propriamente dito é um claro entendimento do significado da palavra *landmark*. De acordo com o dicionário Webster [WEBSTER], as seguintes definições para o termo podem ser encontradas:

- um objeto importante ou ressaltado no ambiente que serve como um guia;
- uma característica diferenciada da paisagem que sinaliza um lugar ou localização;
- alguma coisa usada para delimitar um território.

Em robótica, o termo *landmark* pode ser definido como uma característica ou marca distinta do meio em que um robô está inserido, cujo principal predicado é poder ser reconhecido a partir de dados coletados pelos sensores. Existem inúmeros trabalhos na literatura que abordam este tema, tais como, [OWEN,98], [DUCKETT,99], [NEHMZOW,91] e [GREINER,94].

Os *landmarks* de um ambiente podem ser dos mais diferentes tipos, como, por exemplo:

figuras geométricas (linhas, quadrados, círculos), texturas, imagens, códigos de barra, objetos tridimensionais ou pontos luminosos, dentre uma infinidade de outros tipos [BORENSTEIN et al,96].

A tarefa de eleger uma característica específica do ambiente para desempenhar o papel de landmark implica em se considerar dois itens fundamentais a toda e qualquer escolha:

- *Ser facilmente identificável* - que significa procurar escolher as marcas que possuam um alto contraste em relação ao ambiente, por exemplo, em robôs equipados com sensores fotossensíveis, criar fontes de luz com intensidades variadas e discrepantes, ou ainda, em sistemas baseados em visão, adotar cores diferenciadas às que compõem o cenário de fundo do ambiente.
- *Serem distintos entre si* - escolher características tais que sejam sempre unicamente identificáveis dentre o conjunto formado por todos os landmarks escolhidos. Por exemplo, quando da adoção de símbolos coloridos, procurar utilizar cores diferentes para cada um ao invés de apenas variações de tons da cor. Observe que o caso de *landmarks* em forma de códigos de barra, a utilização de diferentes numerações (códigos) já proporcionaria este efeito.

Quanto mais complexo for o ambiente em que o robô está imerso, maior será a probabilidade de encontrarmos landmarks. Em contrapartida, também é proporcional o aumento no grau de dificuldade em se reconhecê-lo.

Mesmo que contando com uma grande diversidade de tipos potencialmente qualificados como *landmarks*, ainda assim, poder-se-ia classificá-los basicamente em dois grupos:

- *landmarks artificiais* - são marcas ou objetos projetados (construídos e/ou determinados pelo homem) com o propósito exclusivo de fornecer os subsídios necessários que possibilitem a atuação do sistema de navegação. Na Figura 2.7 encontram-se alguns exemplos de landmarks artificiais empregados em sistemas de localização baseados em imagens. Observe que o segundo conjunto (Figura 2.7 (b)) permite a implementação de conceitos como: determinado departamento é representado por quadrados enquanto um outro departamento é representado por círculos, do mesmo modo, as cores podem servir para diferenciar as diversas salas de cada departamento.

(a)

(b)

Alguns exemplos de landmarks artificiais. O primeiro conjunto, indicado em (a), composto por três landmarks que se diferenciam única e exclusivamente pelo padrão do desenho. Em (b), landmarks codificados por cores e formas.

Figura 2.7 – Exemplo de Landmarks Artificiais para sistemas baseados em visão.

- *landmarks naturais* – são marcas ou objetos nativos do ambiente, que desempenham outras funções que não seja necessariamente o suporte para o sistema de navegação. São exemplos de landmarks naturais para o ambiente de um laboratório marcas como portas, janelas, luminárias, cestos de lixo, dentre outros objetos característicos deste tipo de ambiente. Na Figura 2.8 podemos observar alguns landmarks naturais, extraídos e detectados pelo sistema FINALE desenvolvido pelo Robot Vision Laboratory da Purdue University [RVL].

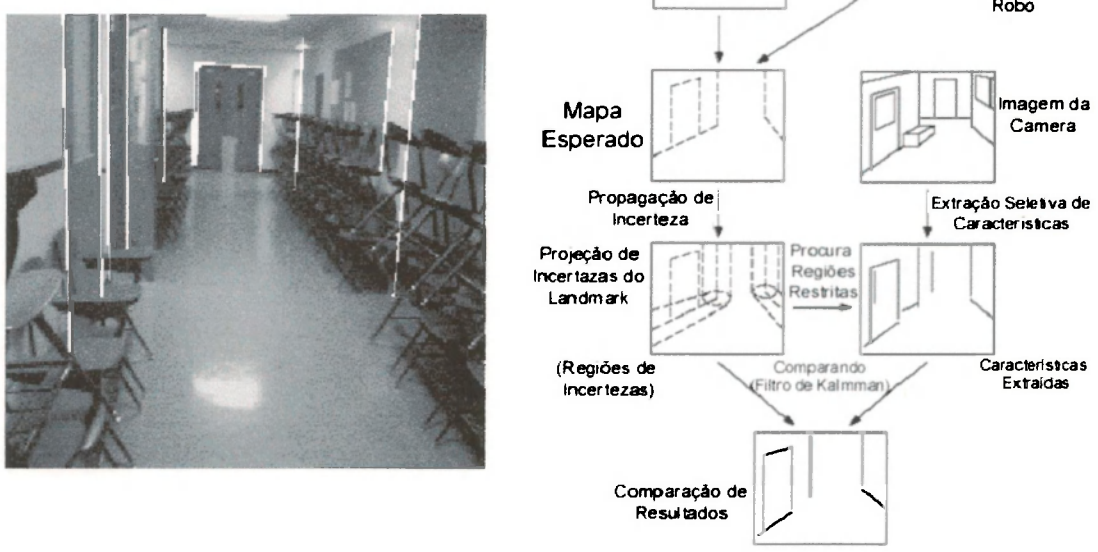


Figura 2.8 – Exemplo de Landmarks Naturais para o sistema FINALE baseado em visão.

A aplicação dos conceitos sobre landmarks em uma técnica para localização consiste em determinar a posição do robô através da observação de um ou mais landmarks e, a partir disto,

determinar os possíveis pontos do ambiente que permitam aquela observação.

Apesar de não se tratar de um método de localização relativa, isso não indica necessariamente que não exista nenhum tipo de relação entre o robô e os landmarks. Ao contrário, lembrando que um landmark deve ter sempre sua posição fixa e conhecida em relação ao ambiente, tão logo este tenha sido reconhecido pelo robô, pode-se estimar a localização atual baseando-se na distância do robô em relação ao landmark observado.

Poderia se dizer que, exceto em ambientes dinâmicos, certamente todos os landmarks seriam estáticos. Até por questão do próprio princípio que motiva a técnica, não se recomendaria estabelecer uma característica dinâmica como um landmak, desde que ela implica ria em uma dificuldade maior para o reconhecimento, ou ainda, seria difícil se estabelecer com precisão qual a sua posição.

Em certos casos ainda, os landmarks podem representar informações adicionais as que já fornecem, como os que se apresentam em forma de código de barras, que, além de ser um marco por si só dentro do ambiente, ainda pode expressar algum tipo de informação decodificada sob a forma do código propriamente dito (vide comentário da Figura 2.7b).

A principal tarefa em um sistema de localização baseado em landmarks é reconhecer de forma convicta um landmarks e, de posse desta informação, calcular a posição do robô. Na Figura 2.9, representam-se todas as fases de um procedimento genérico para a determinação do posicionamento de um robô baseado na observação de *landmarks*.

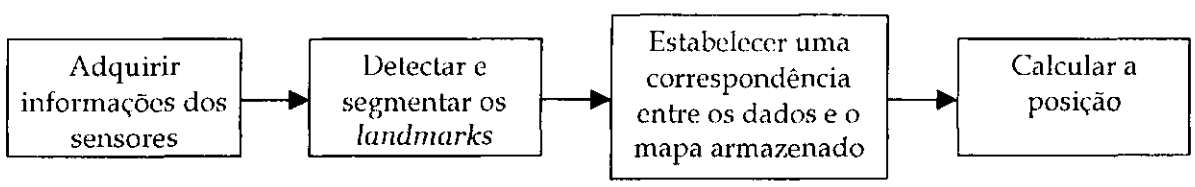


Figura 2.9 – Esquema geral para reconhecimento de Landmarks

Vale ressaltar que um único ciclo de localização compreende a execução de todos os passos, sendo cada etapa responsável por:

- *Aquisição de informações:* Periodicamente o robô deve consultar seus sensores para adquirir informações provenientes do ambiente. A cada leitura dos sensores, análises e filtragens devem ser realizadas a fim de se melhorar a qualidade dos dados a serem processados nas etapas seguintes.
- *Detecção e segmentação:* Consiste em analisar os dados provenientes dos sensores em

busca dos padrões que foram selecionados como landmarks para aquele meio. Cabe aqui ainda, a tarefa de determinar a(s) distância(s) entre o robô e o(s) landmark(s) observado(s).

Nesta etapa, deve-se evitar ao máximo possível que a detecção de um *landmarks* seja uma busca dentro do conjunto de todos os possíveis *landmarks* contidos naquele ambiente, ou seja, o processo de detecção deve, sempre que possível, descartar a procura de *landmarks* que fisicamente não estejam dentro da área de alcance dos sensores do robô.

- *Estabelecendo uma correspondência:* Nesta etapa cada um dos landmarks observados na etapa anterior, deve ser correlacionado com o conjunto de landmarks presentes no modelo de mundo interno mantido pelo robô.
- *Calcular a posição:* é nesta fase que efetivamente é determinada a posição do robô em função das distâncias aferidas entre o robô e cada um dos landmarks observados.

Um sistema de localização que emprega este tipo de técnica tem a vantagem de que erros de posicionamento são limitados, uma vez que, a cada ciclo de localização, as informações a respeito do posicionamento anterior não é relevante. Porém o processo de detecção dos landmarks e a determinação do posicionamento robô, em uma aplicação de tempo real, pode não ser sempre possível.

A detecção e estabelecimento de uma correlação entre os dados captados pelos sensores e as informações armazenadas no modelo de mundo, ainda consistem nas principais dificuldades encontradas pelas implementações de sistemas de localização baseado em landmarks.

Na próxima seção são descritos alguns dos principais trabalhos, referentes ao tema abordado neste trabalho.

2.6. Trabalhos Relacionados

Dado o volume de pesquisas que abordam os tópicos relatados neste trabalho e que, cada vez mais, emanam de diferentes localidades em todo o mundo, pode-se perceber um grande esforço para a melhoria das técnicas aplicáveis a robôs móveis, o que, inevitavelmente, incluem também as técnicas de navegação e de localização.

Por questões históricas, políticas e financeiras, que não cabem serem discutidas aqui, algumas instituições estão na vanguarda tecnológica no que diz respeito à pesquisa robótica.

Uma delas é o Robotics Institute da Carnegie Mellow University .Dividido em diversos centros, cada qual, responsável por uma frente de pesquisa dentro da área da robótica, atualmente a CMU tem se enveredado por uma linha de pesquisas que explora sistemas probabilísticos e baseados em landmarks. Uma vez que no texto são reportados alguns de seus projetos, cabe aqui uma breve citação a respeito de três desses centros:

- *Field Robotics Center – FRC*, que endereça suas pesquisas para os problemas que envolvem o uso de robôs móveis em ambientes típicos do trabalho humano e, também, para ambientes de campo aberto. Este centro atualmente se encontra sob a supervisão de Red Whittaker, que também ocupa o cargo de cientista chefe na RedZone uma empresa fundada após o incidente de Chernobyl em 1986, caracterizada por soluções robóticas para ambientes perigosos.
- *Space Robotics Initiative – SRI* que é o centro de pesquisas cujos projetos tem buscado o desenvolvimento de robôs, bem como, toda a tecnologia de suporte necessária (sistemas de comunicação, manipuladores e a coordenação e cooperação de múltiplos robôs) para a exploração interplanetária, a construção de estações espaciais baseadas em energia solar, dentre outros desafios enfrentados pelos projetos espaciais.
- *National Robotics Engeneering Consortium – NREC* que tem por objetivo incorporar as mais novas técnicas e tecnologias sobre robôs móveis no desenvolvimento de produtos para o mercado comercial americano e fomentar o crescimento da indústria no país.

Uma contribuição importante resultante do trabalho destes centros, juntamente com laboratórios da NASA e o apoio do Departamento de Energia dos Estados Unidos, foi o projeto e a construção de um robô para operar na área de um reator na Usina Nuclear de Chernobyl que apresentou problemas de vazamento. Apesar de se tratar de um robô móvel para a exploração estrutural e das questões radioativas da área, este não se tratava de um robô autônomo, pois era tele-operado por um especialista que poderia estar até 300 pés de distância do robô.

Atualmente, existem diversas linhas de pesquisas na área de robôs móveis que estão sendo exploradas por esta universidade, dentre elas: a criação de veículos autônomos para aplicações

cotidianas, robôs de pequeno porte para trabalho em grupos e para campeonatos de futebol, modelagem tridimensional, sistemas únicos para mapeamento e localização, construção de mapas para ambientes abertos, robôs exploradores para missões a Marte e para a Antártida, dentre outros temas.

Entretanto, o maior destaque para os projetos que envolvem a CMU em relação à localização e mapeamento, fica a cargo da técnica que a instituição dá ao tema. Liderando um segmento cada vez maior, os centros de estudo da CMU aplicam uma abordagem probabilística aos problemas na estimativa do posicionamento de um robô ou mesmo na localização de outros objetos que compõem a ambiente.

Em uma abordagem, completamente diferente, temos a Universidade de Rochester, em Nova York, que explora algoritmos de campos potenciais para a navegação e a construção de mapas deixando a estimativa da localização a cargo de sistemas odométricos.

Não menos importante, temos a Escola de Engenharia da University od South California (USC), que recentemente foi premiada como a segunda maior em pesquisas robóticas dentro dos Estados Unidos. Explorando uma outra abordagem, em seus laboratórios tem-se desenvolvidos projetos na área de modelagem de sistemas robóticos multi-agentes, robôs cooperativos e sistemas de controle na forma de sistemas embarcados em hardware.

Ligado a NASA, temos o Centro de Pesquisas AMES, onde continuam os desenvolvimentos da técnica e da tecnologia utilizada no sistema de visão do robô exploratório enviado a Marte, o Soujorner. Abordagens baseadas em processamento de imagens e visão computacional estão presentes nas principais pesquisas desenvolvidas, deixando em segundo plano a exploração de outras técnicas, haja vista que outras instituições americanas contribuem para o avanço coletivo das pesquisas.

Já a Universidade de Stanford lidera um outro grupo de pesquisas onde se desenvolvem sistemas de navegação e de localização baseados em informações provenientes da triangulação de sinais de satélites, ou seja, sistemas baseados em Global Position System (GPS). Segundo ela, essas e outras tecnologias deverão ser combinadas para desenvolver um sistema superior de navegação para futuras missões com robôs exploradores, os chamados rovers [STANFORD].

Algumas instituições de fora dos Estados Unidos também têm produzido trabalhos significativos na pesquisa de robôs móveis. Na Universidade de Zurique, na Suíça, o principal propósito dos estudos sobre robótica é o entendimento detalhado dos processos envolvidos no

comportamento dos animais durante a execução da tarefa de navegação, de modo a implementar estratégias comportamentais, que sendo executadas por um robô, tenha um desempenho realístico.

A Universidade de Manchester, na Inglaterra, enfatiza uma abordagem semi-simbólica e baseada em comportamento, para tarefas de navegação e mapeamento, e baseada em landmarks naturais para o processo de localização.

2.6.1. Utilizando Odometria

Borestein [BORESTEIN,98] apresenta um algoritmo chamado IPEC – Internal Position Error Correction executado em uma plataforma ominidirecional, ou seja, um robô móvel capaz de executar movimentos em qualquer direção. Esta plataforma é composta por duas bases com rodas, cada uma interligada a outra por intermédio de uma haste localizada na parte superior, diretamente conectada a um sensor angular. Quando uma das bases executa qualquer movimento, a orientação da haste em relação à outra base sofre uma variação angular. Os erros de odometria são detectados e avaliados através das diferenças percebidas por um dos veículos em relação ao outro.

Lindsay Kleeman [KLEEMAN,92] apresenta uma solução híbrida para o problema de localização que se baseia na utilização de balizamento ativo empregando ultra-som juntamente com técnicas odométricas. A estimativa da posição é dada através da utilização de Filtro de Kalman Estendido, que fornece a melhor estimativa do posicionamento do robô uma vez que o filtro modela as suas características de movimento.

2.6.2. Utilizando Landmarks

Owen e Nehmzow [OWEN,98] propõem um sistema de navegação para ambientes dinâmicos através de um mapeamento baseado em *landmarks*, onde a fase de aprendizado é desempenhada por um processo de auto-organização dos dados coletados pelos sensores utilizando um classificador Reduced Coulomb Energy (RCE), enquanto o robo é guiado por uma pessoa. Os experimentos mostram que, durante a fase de reconhecimento, o sistema pode falhar se todos os caminhos conhecidos que levam ao ponto meta estiverem bloqueados.

Preocupando-se com a escalabilidade dos ambientes das aplicações, Duckett juntamente com Nehmzow, mostram que para operar com sucesso em ambientes de média escala, é

fundamental um sistema de orientação global [DUCKETT,99].

Ducket e Nehmzow, confrontando a qualidade do processo de localização com o seu custo computacional para sistemas que empregam as atuais técnicas para reconhecimento de *landmarks*, concluíram que a técnica de *map matching* é uma forte candidata para reconhecimento de *landmarks*. Foram avaliados sistemas que utilizam técnica de *map matching* com mapas métricos, redes de Kohonen, classificadores RCE e Adaptative Resonace Teory (ART2). O experimento utilizava um conjunto de dados gravados em duas etapas. Os dados da primeira etapa eram utilizados para aprendizado dos *landmarks* e os dados da segunda etapa eram utilizados para o reconhecimento [DUCKETT,00].

Implementações baseadas em reconhecimento de *Landmarks* por imagens podem sofrer basicamente dois tipos de erros, (1) alguns ângulos são medidos com pequenos erros e (2) dificuldade de identificar alguns *landmarks* semelhantes entre si [BETKE,97].

Ulrich apresentou em [ULRICH,00] um sistema de reconhecimento de imagens para mapas topológicos, que determina a posição do robô baseando-se em uma imagem colorida representada em seis histogramas unidimensionais. Seus resultados mostraram que o sistema classificava as imagens corretamente a uma taxa de 87.5% a 97.7%.

Infelizmente, alguns *landmarks* não podem ser vistos, ou ainda pior, podem ser confundidos com outros *landmarks*, resultando em desperdício de tempo na busca de *landmarks* invisíveis e também futuros erros na estimativa de posição do robô.

Para lidar com esse problema, R. Greiner e Isukapalli propuseram em [GREINER,94] um método que usa experiências prévias para que o sistema possa aprender uma função de seleção de landmarks. Dados um conjunto de landmarks, esta função retorna um subconjunto dos landmarks que, certamente, podem ser visíveis e encontrados a partir da posição corrente. A eficiência do algoritmo é provada tanto teoricamente como empiricamente, baseada em informação real obtida usando-se um robô implementado.

2.6.3. Utilizando Landmarks e Odometria

Nehmzow [NEHMZOW,91] propôs uma abordagem que utiliza uma rede de Kohonen para localização de um robô num ambiente pré-estabelecido, baseando-se apenas as leituras fornecidas pelos encoders acoplados às rodas. Neste artigo, a rede é representada por um arranjo bidimensional contendo 10x10 nós (neurônios artificiais). O treinamento da rede de

Kohonen [KOHONEN,84] foi realizado com os valores produzidos por dois enconders que registravam os movimentos das duas rodas de tração do robô.

O vetor de entrada para a rede é uma composição dos deslocamentos individuais de cada roda. Em casos de trajetórias na lineares a roda externa ao traçado da curva girava mais que a roda interna produzindo então vetores resultantes diferentes para cada movimento descrito pelo robô. A diagramação da rede, ou seja, a distribuição de agrupamentos no mapa de Kohonen, é então avaliada por um sistema que fornece a localização baseando-se no neurônio vencedor.

Capítulo 3 - Embasamento teórico

Dedicamos este capítulo para a exposição de todos os conceitos necessários ao entendimento da solução proposta pelo trabalho. Inicialmente, uma visão geral do sistema que controla o robô, bem como, um breve relato histórico de seu surgimento, são apresentados nas seções 3.1 e 3.2, enquanto o detalhamento técnico sobre o hardware do robô Pioneer 1 é fornecido na seção 3.3. Tendo em vista que este robô realiza a percepção do ambiente através de vários dispositivos de medição por ultra-som (sonares), descreve-se na seção 3.4, informações sobre o modo como esse tipo de dispositivo opera, questões físicas pertinentes ao seu funcionamento, assim como, algumas considerações a respeito de pontos positivos e negativos do emprego e uso de sonares. Ainda neste contexto, na seção 3.5, poder-se-á encontrar uma explicação didática a respeito de ferramenta probabilística chamada Filtro de Kalman (comumente empregada na filtragem de ruídos em sinais) servindo, inclusive, como uma proposta para a solução ao problema de ruído (perturbações) que ocorrerem durante as leituras realizadas pelos sonares. Finalmente, na seção 3.6 é apresentada a teoria sobre os mapas auto-organizáveis, em particular da rede de Kohonen, que neste trabalho é utilizada para a detecção de landmarks em ambientes simples.

3.1. Projeto Flakey & o sistema Saphira

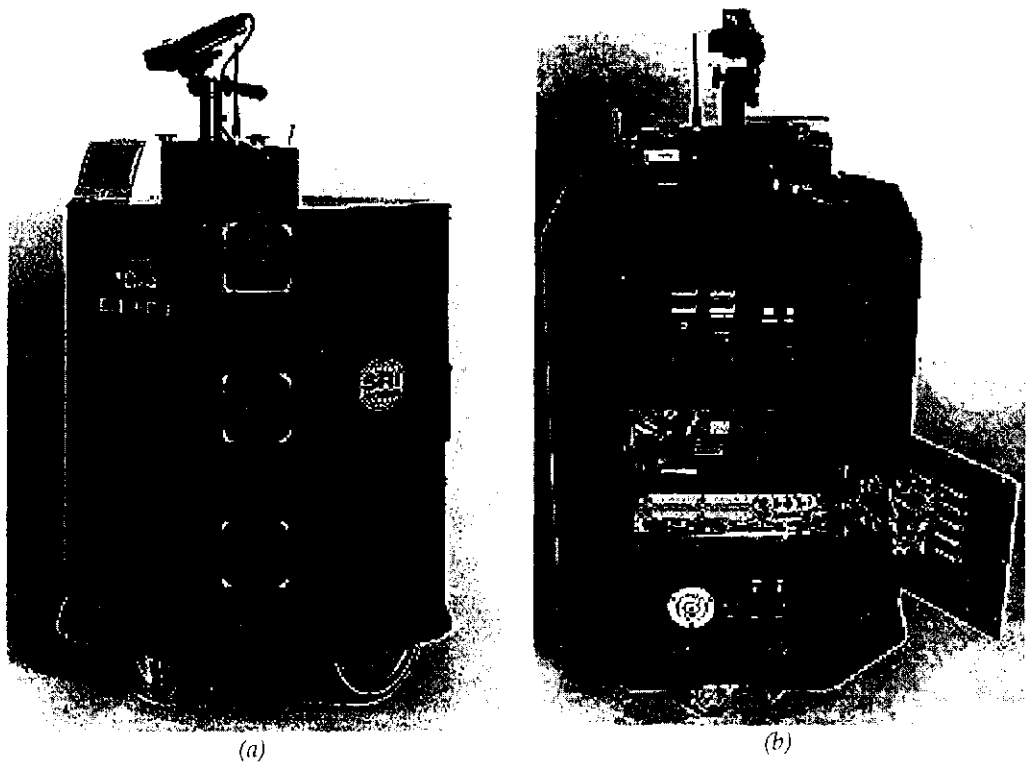
Como parte integrante do Stanford Research Institute (SRI), uma entidade sem fins lucrativos que é capaz de promover projetos de pesquisa de longo prazo, o International's Artificial Intelligence Center (AIC), em operação desde 1966, é considerado um dos maiores centros de pesquisa em inteligência artificial, tendo como objetivos a busca de entender melhor os princípios computacionais que delineiam a inteligência nos homens e nas máquinas.

Proporcionando estímulos e troca de informações através do estabelecimento de associações com universidades e grupos de pesquisa, seja dentro ou fora dos Estados Unidos, este centro tem sido de grande importância para o desenvolvimento das capacidades computacionais ligadas ao comportamento inteligente através de seus projetos.

Em 1984, o SRI iniciou um projeto denominado *Flakey* com o intuito de pesquisar quais eram as características necessárias para que um agente móvel pudesse ser o mais autônomo possível em relação a sua tarefa e o contexto da sua ocorrência.

3.1.1. O projeto Flakey

Batizado com o mesmo nome do projeto do qual faria parte, Flakey era um robô em forma de um cilindro octogonal que media cerca de um metro de altura e meio metro de diâmetro. Para se movimentar, o robô contava com duas rodas laterais de sete polegadas de diâmetro e mais uma, do tipo rodízio, para equilibrar e dar sustentação ao robô (Figura 3.1a).



(a) (b)

(a) *Vista frontal do robô Flakey, onde se pode observar parte de uma das rodas responsáveis pela tração junto a lateral do robô, a roda tipo rodízio, três dos seus doze sonares localizados na lateral e a câmera de vídeo localizada na parte superior.*
(b) *Foto de outro ângulo do Flakey, com detalhe para o compartimento que abriga as placas e o processador Sparc Station*

Figura 3.1 – Flakey: robô desenvolvido pelo SRI International

Equipado com um processador Sparcstation e uma conexão via rádio tipo ethernet, o robô interagia com o ambiente através de doze sonares localizados na base e também, através de uma câmera de vídeo móvel e estéreo instalada no topo do robô (Figura 3.1b). Possuía também uma interface de voz, comandada por sistema de reconhecimento de fala denominado Corona,

que adicionalmente provia um programa padrão para conversão texto-voz. Tratava-se de uma plataforma construída especialmente para o projeto.

O objetivo principal do projeto não era a construção do robô propriamente dito, mas sim, pesquisar e determinar quais as características seriam necessárias para uma interação autônoma do robô em um ambiente, durante a realização de alguma tarefa [KONOLIGE,97]. Em outras palavras, a proposta consistia em introduzir um robô que fosse dotado de atuadores apropriados para desempenhar uma série de tarefas específicas e que também fosse capaz de operar de forma autônoma em um ambiente típico do cotidiano humano, ou seja, que não implicasse em nenhuma preparação prévia do ambiente ou que contasse com algum tipo de sinalização especial.

Neste contexto, o cenário de atuação escolhido foi um escritório composto por algumas salas, todas interligadas por corredores e mobiliadas com mesas, cadeiras e outros objetos característicos a este tipo de lugar, sendo que cada uma destas salas servia de escritório para uma pessoa diferente.

Como metas, o robô deveria realizar as tarefas requisitadas por uma pessoa, que interagia com ele através das modalidades de comunicação típicas entre humanos, como gestos e fala. Alguns exemplos de tarefas eram, encontrar o Mr. Johnson, ir até o escritório do Mr. Kurt, dentre outras.

Em 1993, como fruto das pesquisas do projeto Flakey, Kurt Konolige, juntamente com sua equipe, propõem uma arquitetura de controle de robôs móveis capaz não somente, de satisfazer as necessidades impostas pelo projeto, mas flexível e aberta o suficiente para que qualquer um pudesse utilizá-la mesmo com outros robôs que não o Flakey, consolidando assim, cerca de dez anos de pesquisas que envolveram o tema. Batizada com o nome de *Saphira*, essa arquitetura trata-se de um sistema operacional que se destina ao controle de robôs móveis.

3.2. O sistema de controle robótico - Saphira

Kurt [KONOLIGE,97] apresenta a arquitetura Saphira como sendo um sistema integrado responsável pelo controle e sensoriamento, destinado à aplicações robóticas, mas especificamente, como um sistema para comandar robôs móveis autônomos.

Para desempenhar tais tarefas, o sistema conta com um seqüenciador de comportamentos e

planejador reativos, integra um controlador fuzzy, além de rotinas para interpretação das leituras dos sonares, mapeamento e navegação.

Concebido segundo o paradigma cliente/servidor, o sistema foi projetado para atuar juntamente com um *servidor robótico*, que é na verdade, um tipo de sistema operacional de micro-tarefas que fornece uma interface para acesso aos serviços providos pelo robô. No intuito de facilitar o entendimento, o sistema é apresentado como um conjunto de duas arquiteturas distintas, montada uma sobre a outra, chamadas de: *arquitetura do sistema* e *arquitetura de controle do robô*.

A primeira arquitetura, denominada arquitetura do sistema, é responsável por um conjunto de rotinas de controle e comunicação entre o robô e uma unidade host. A arquitetura foi projetada para ser aberta, de forma a tornar mais fácil a definição das aplicações que se conectam ao robô como um cliente, além de disponibilizar de uma gama de serviços providos por esta arquitetura. Esta abordagem permite que o usuário possa definir seu próprio sistema de controle sem ter que se preocupar com detalhes inerentes ao hardware ou a comunicação.

Acima desta, está a arquitetura de controle do robô, encarregada de comandar desde questões de baixo nível, como controle dos motores e sensores, até problemas próprios da navegação, planejamento e reconhecimento de objetos.

Nesta arquitetura também está embutido um rico conjunto de rotinas e representações para realizar o processamento de dados provenientes dos sonares, a construção de modelos de mundo e o controle das ações do robô. Assim como a arquitetura do sistema, as rotinas presentes na arquitetura de controle do robô são altamente integradas para apresentar uma base coerente para o controle do robô.

Na Figura 3.2, pode-se observar um esquema que representa os diferentes elementos que compõe uma aplicação cliente. Nela, os blocos em verde representam as rotinas que compõem a arquitetura do sistema (biblioteca do Saphira) e os blocos em amarelo, a arquitetura de controle do robô formada principalmente pelas rotinas provenientes do usuário.

Repare que a disposição dos elementos é intencional, pois do lado esquerdo da figura, formando um grande bloco, estão todas as rotinas síncronas executadas a cada ciclo do kernel. Adicionalmente, ao lado direito, temos as rotinas do usuário que são executadas como threads assíncronas, compartilhando o mesmo espaço de endereçamento.

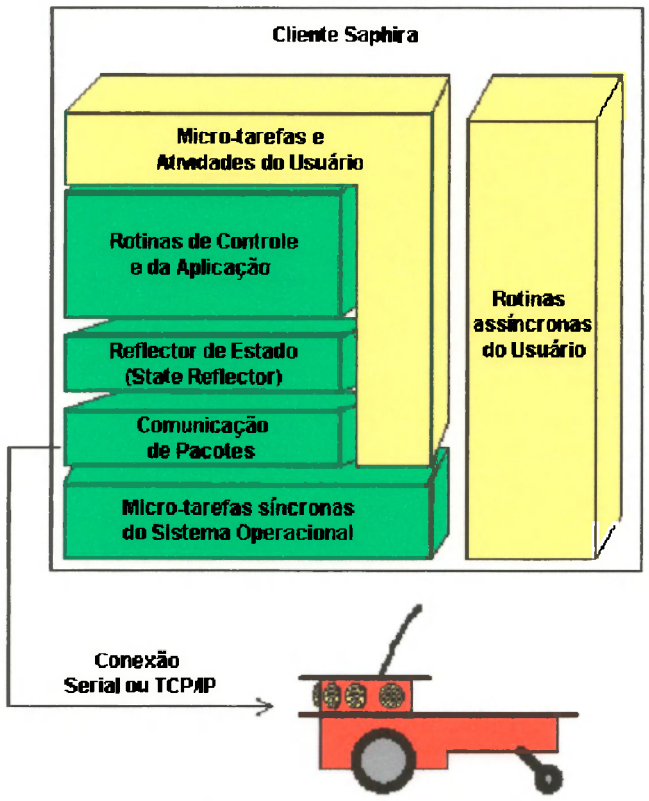


Figura 3.2 – Arquitetura do sistema Saphira

A arquitetura de controle é suficientemente flexível para proporcionar ao usuário a possibilidade de escolher entre vários métodos para se alcançar um determinado objetivo, por exemplo, usar um controlador fuzzy para controlar a direção dos motores mais diretamente. A arquitetura é considerada também uma arquitetura aberta, pois é possível aos usuários substituir rotinas pré-definidas por métodos próprios, ou adicionar novas funcionalidades e distribuir entre outros grupos de pesquisa.

3.2.1. Arquitetura do Sistema

Denominam-se rotinas do Saphira todas as micro-tarefas que são invocadas a cada ciclo do kernel (100 ms) pelo montador de micro-tarefas do SO. O conjunto formado por essas rotinas compõe a arquitetura do sistema e são responsáveis por manipular os pacotes de comunicação com o robô, construir uma representação interna do estado do robô e desempenhar tarefas mais complexas como navegação e interpretação dos dados dos sensores.

3.2.1.1. Micro-tarefas do SO

A arquitetura Saphira foi fundamentada em um sistema operacional dirigido a interrupções

síncronas. As chamadas micro-tarefas são rotinas curtas encaradas como máquinas de estado finito (MEF) que são registradas no sistema operacional.

A cada 100 ms, o fluxo de execução atravessa todas as MEFs, executando um passo em cada uma delas. É importante ressaltar que, devido as instruções serem executadas em intervalos de tempo fixos, todas as MEFs operam concomitantemente como threads.

Esta abordagem tem algumas vantagens, uma vez que, proporciona intervalos precisos e fixos a cada ciclo de tempo, é algo freqüentemente útil no controle robótico. E não apenas, mas o fato de operar em pequenos ciclos de tempo torna o sistema Saphira capaz de suportar o desenvolvimento de um controlador reativo para o robô, aplicação a qual exige uma resposta imediata às rápidas mudanças que ocorrem nas condições do ambiente.

Um SO baseado em micro-tarefas também implica em algumas limitações: cada micro-tarefa deve completar seu trabalho em um pequeno intervalo de tempo e ceder novamente o controle ao SO. Mas com a capacidade computacional dos equipamentos de hoje em dia, onde um processador Pentium de 100MHz é o comum, mesmo as tarefas mais complicadas, como por exemplo, os cálculos de probabilidades para o processamento dos sinais dos sonares, podem ser realizados em milisegundos.

O emprego de micro-tarefas também ajuda a distribuir o problema de controlar o robô através de muitas, pequenas e modificáveis rotinas. Isto torna mais fácil projetar e depurar um sistema de controle complexo.

3.2.1.2. Rotinas do Usuário

As rotinas definidas pelo usuário podem ser classificadas em dois tipos. O primeiro se refere a rotinas pequenas, similares as micro-tarefas do SO e o segundo tipo são rotinas que demandam uma maior complexidade e requerem um tempo maior que 100ms para a sua execução.

Na realidade, as rotinas do usuário enquadradas no primeiro grupo, definem suas próprias MEFs e são alocadas na mesma thread de execução que as micro tarefas do Saphira, operando sincronamente a cada ciclo de execução do kernel. Também é correto dizer que essas rotinas são extensões das rotinas do Saphira e desse modo podem acessar o sistema em qualquer nível da arquitetura. Por estarem na mesma thread de execução, também compartilham o mesmo espaço de endereçamento o que proporciona acesso a todas as informações disponíveis as rotinas do

kernel.

Nem todas as rotinas são pequenas o suficiente para serem executadas em 100ms, o que é uma obrigatoriedade para o sincronismo com as rotinas do kernel. Tendo em vista esses casos, a arquitetura provê um segundo tipo para a definição de rotinas do usuário. Neste grupo, as rotinas são executadas assincronamente, cada uma como uma thread separada. O limite para o número de rotinas do usuário que se enquadram neste tipo fica a cargo do número de threads concomitantes que o sistema operacional é capaz de suportar.

Para que não haja problemas com o consumo de tempo de maneira que possa afetar a natureza reativa do sistema, as threads do kernel do Saphira tem prioridade de execução sobre qualquer rotina definida pelo usuário.

3.2.1.3. Comunicação de Pacotes

O sistema Saphira implementa um protocolo de comunicação bidirecional baseado em pacotes o que o torna capaz de se conectar ao servidor robótico usando um canal serial ou Ethernet com TCP/IP, ou ainda, a um simulador através do uma conexão local tipo loopback.

Uma aplicação cliente típica envia ao servidor robótico em torno de um a quatro pacotes de dados por segundo, enquanto, independentemente do cliente, este estará recebendo 10 pacotes por segundo provenientes do servidor robótico. O conjunto de dados enviado em um pacote possui em média de 30 a 50 bytes, permitindo que seja facilmente acomodado em um canal de 9600 baund.

3.2.1.4. Refletor de Estados

De maneira a mostrar o atual estado do robô no computador host, o sistema implementa uma visão completa de todas as características que descrevem o estado robô em uma estrutura interna chamada *refletor de estados*. Esta estrutura atua como um espelho que reflete o estado do robô, fornecendo informações sobre o seu movimento e seus sensores.

A fidelidade do refletor de estados para descrever o estado do robô deve ser máxima para fornecer com precisão os valores necessários às rotinas do sistema e as rotinas do usuário. Entretanto, um outro aspecto importante é o fato de que o controle do robô está diretamente associado a ele, isto é, quando uma rotina deseja enviar uma ordem para o robô, basta que esta altere a propriedade correspondente no refletor de estados para que as rotinas de comunicação enviem o comando adequado ao robô para que execute a ação.

3.2.2. A arquitetura de controle do robô

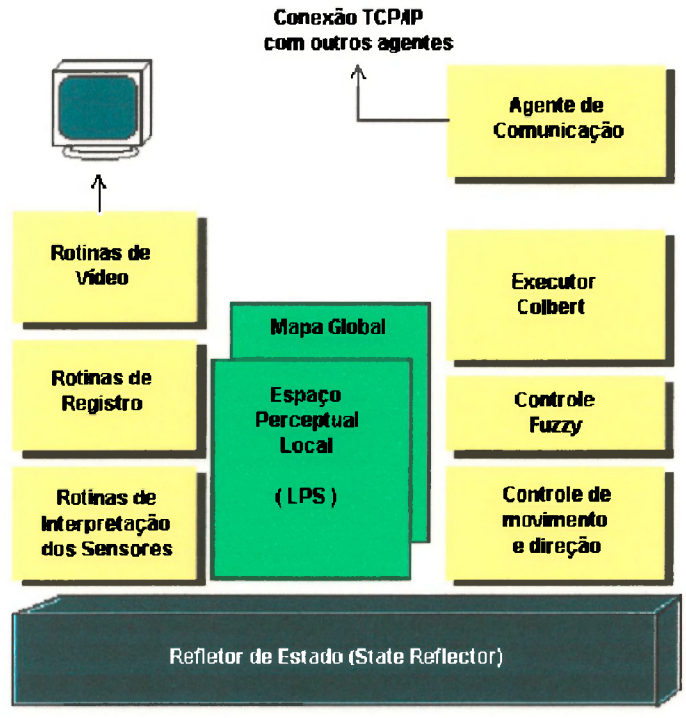


Figura 3.3 – Arquitetura de controle

A arquitetura de controle do robô está toda fundamentada sobre o refletor de estados e compreende todo o conjunto de micro-tarefas que implementam o sistema navegação para um ambiente de escritórios. Habitualmente, uma aplicação cliente emprega um subconjunto destas rotinas. Na Figura 3.3, pode-se observar um desenho esquemático dos elementos que compõem esta arquitetura.

Os subitens que se seguem foram dedicados a detalhar um pouco mais o que esses componentes fazem, entretanto, por não ter este trabalho como objetivo principal a investigação e exploração do sistema Saphira, optou-se então por não detalhar os elementos que não são diretamente relevantes para o mesmo. Outros detalhes a respeito do Saphira que não tenham sido abordados aqui, podem ser encontrados em [SAPHIRA,97].

3.2.2.1. Representação do Espaço

Em qualquer sistema robótico a representação do ambiente é um quesito crítico para o desempenho da aplicação. No sistema Saphira a reprodução do ambiente ocorre através de um modelo geométrico que representa o espaço no qual o robô está imerso e é composto por duas

partes: *Local Perceptual Space* e *Global Map Space*.

- *LPS (Local Perceptual Space):* trata-se de uma visão egocêntrica do robô em relação ao espaço. O sistema LPS fornece uma visão local, percebendo apenas uma sub-área do ambiente correspondente à posição derredor ao robô. Esta representação pode ser muito útil para tarefas como pequenos ajustes no movimento do robô, monitoramento de obstáculos para evitar colisões e fusão de sensores.
- *GMS (Global Map Space):* trata-se do mapa global do ambiente, onde são representados os objetos que o compõem em um sistema de coordenadas absolutas. Esses objetos são chamados de *artefatos*, e o conjunto deles, como corredores, portas, junções e salas são agrupados e armazenados sob a forma de um *mapa*, podendo ser utilizado novamente em uma outra execução da aplicação.

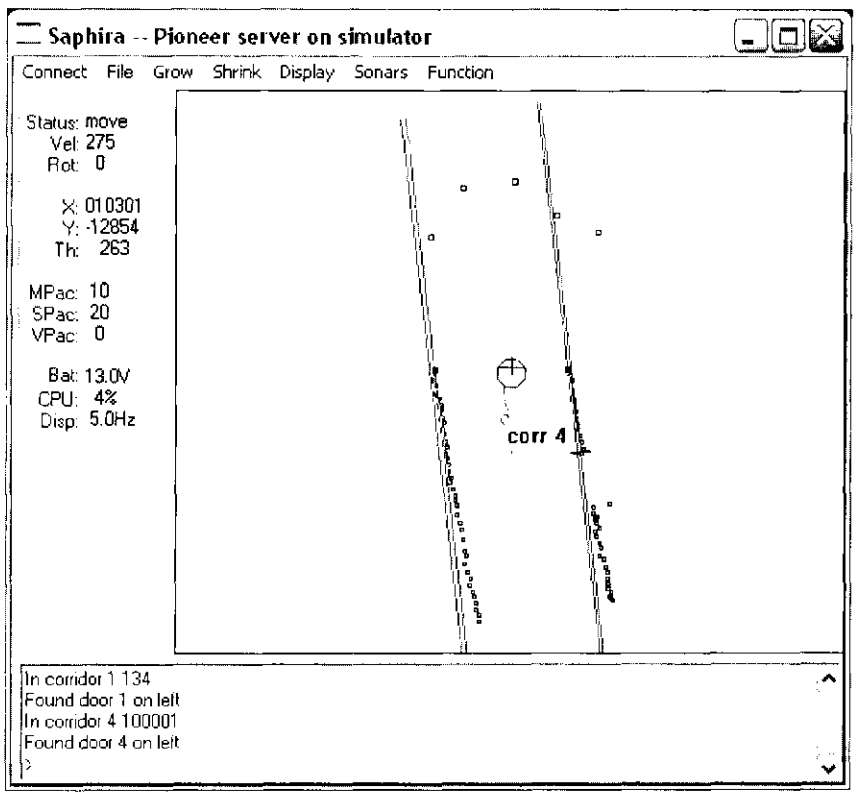


Figura 3.4 – Representação do espaço ao redor do robô no sistema Saphira.

Na Figura 3.4 podem ser observadas as duas representações do espaço sendo apresentada simultaneamente no host que executa a aplicação cliente. O LPS é apresentado como uma visão do espaço próximo ao robô, com as leituras dos sonares que são exibidas como um conjunto de pontos azuis. Já o sistema GMS, após a analise de consecutivas leituras locais, cria uma

hipótese, como a de estar navegando em um corredor, e a representa em um mapa através de artefatos, que na figura são representados pelas linhas vermelhas paralelas, equivalentes as paredes que formam o corredor.

3.2.2.2. Controle de Direção e Movimento

O sistema Saphira adota um método simples para controlar o movimento do robô, uma vez que dispõem do refletor de estados e dos sistemas LPS e GMS. A abordagem consiste em modificar os valores das variáveis de controle presentes no refletor de estados e, no próximo ciclo de execução, as rotinas de comunicação atualizarão o servidor do robô, que então, enviará os comandos necessários para que este tente ao máximo possível alcançar o novo estado.

Como mencionado anteriormente, o sistema é bastante flexível para que se realize uma mesma ação através de diferentes instruções. De qualquer forma, todo movimento realizado pode ser descrito e controlado, basicamente, por apenas duas componentes: a *translação* e a *rotação*.

A translação refere-se aos deslocamentos lineares para frente e para trás, controlável através das primitivas `sfSetPosition` e `sfSetVelocity`, ou ainda, através da manipulação direta dos valores do refletor de estados.

Já a rotação descreve os movimentos angulares, ou seja, os movimentos que o robô realiza para a direita e para a esquerda, e podem ser manipulados pelas primitivas `sfSetHeading`, `sfSetDHeading` e `sfSetRVelocity`, bem como, na intervenção direta nas variáveis do refletor de estados.

3.2.2.3. Rotinas para Interpretação dos Sensores

São procedimentos responsáveis por extrair dados a partir dos sensores ou do simulador e enviá-los para o LPS. Existem algumas rotinas disponíveis para a interpretação dos dados, que podem ser ativadas pelo usuário através da primitiva `sfInitInterpretationProcess` a fim de se detectar obstáculos, reconstruir superfícies, construir mapas e reconhecer objetos. Ou ainda, primitivas como `sfOccBox`, `sfOccBoxRet` ou `sfOccPlane` para fusão de sensores.

3.3. O robô Pioneer 1

A empresa ActivMedia Robotics, juntamente com a iRobot e em colaboração do Laboratório de Inteligência Artificial de Stanford - SRI, lançavam no mercado, no ano de 1995, uma família de robôs chamados Pioneer que dispunham do software de controle Saphira projetado por Kurt Konolige em 1993.

Essa proposta vinha de encontro com o desejo de vários pesquisadores que aspiravam desenvolver pesquisas nas mais diferentes seguimentos da inteligência artificial ligadas à robótica, e que por ventura, não dispunham de uma solução robótica para esse desenvolvimento, quer fosse por razões de pouca verba para a aquisição de um robô sofisticado, ou mesmo, por questões de tempo para construir uma nova plataforma.

O Laboratório de Inteligência Computacional (LABIC), do Instituto de Ciências Matemáticas e da Computação da Universidade de São Paulo (ICMC - USP), adquiriu um exemplar desta família, o robô Pioneer 1 (Figura 3.5).

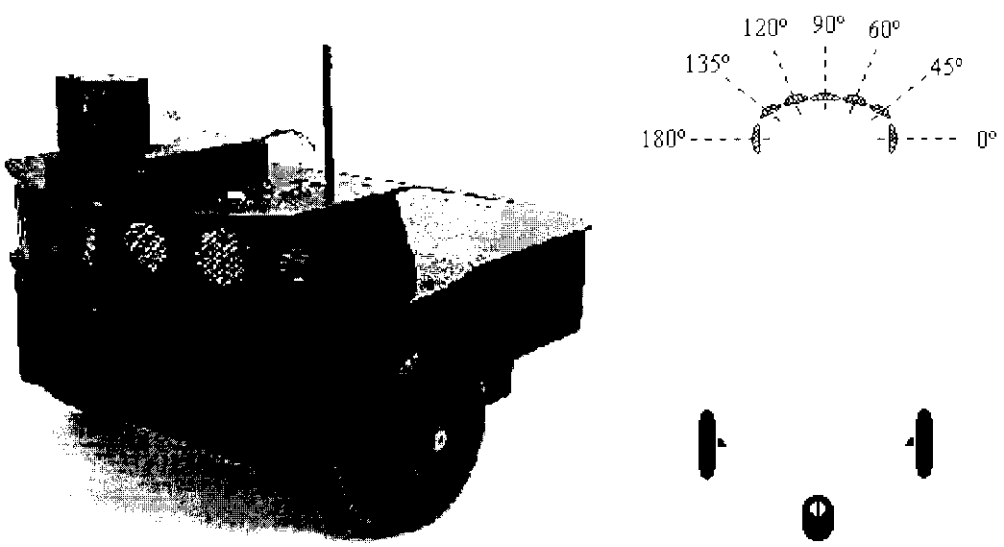


Figura 3.5 - O robô Pioneer 1, equipado com uma câmera de vídeo.

Trata-se de um robô de pequeno porte, com dimensões nominais de 46 x 35,5 x 23 centímetros. Seu sistema de locomoção é constituído por duas rodas fixas, localizadas lateralmente, que são responsáveis pela tração, e mais uma terceira roda móvel, localizada na parte traseira do robô, de modo que seja possível realizar movimentos de rotação.

As rodas de tração estão conectadas a motores DC reversíveis, permitindo que o robô desenvolva velocidades de até 60 cm/s, ou seja, pouco mais de 2 km/h. A fim de auxiliar no processo de controle, o robô vem equipado com dois decodificadores nestas rodas, proporcionando um mecanismo para a detecção de colisões e possíveis travamentos. Para atuar no ambiente, além da capacidade de locomover-se, o projeto do Pioneer 1 permite que uma garra simples, do tipo pinça, seja acoplada na parte frontal do robô, capacitando-o a manipular pequenos objetos, como por exemplo, uma lata de refrigerante que esteja a sua frente. Já, para perceber o meio ao seu redor, o robô dispõe de sete sonares Polaroid 6500, cinco localizados frontalmente e os outros dois nas laterais.

Apesar da pouca capacidade do sonar em fornecer informações mais ricas, o Pioneer 1 pode ter suas funcionalidades estendidas através da conexão de uma câmera de vídeo, uma vez que tem suporte para o Cognachrome Vision System, um sistema de visão desenvolvido pela Newton Research Lab's, composto por um pequeno hardware de processamento de imagens que realiza o rastreamento de objetos a uma taxa de 60 Hz, com resolução de 200x250 pixels, através de um método de discriminação de cores.

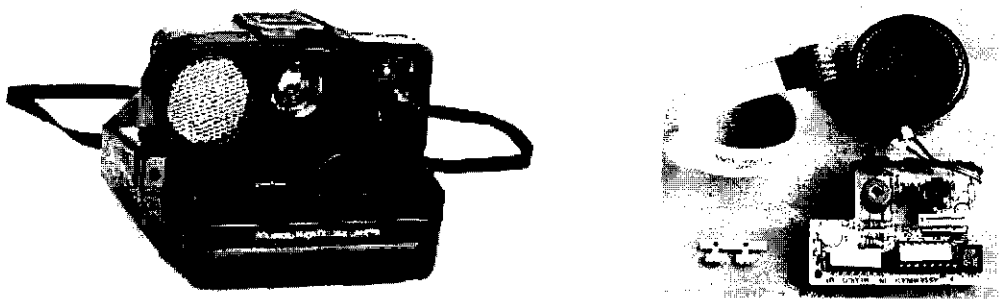
Por fim, o robô é controlado por uma aplicação cliente executada em um computador remoto que pode se conectar a ele diretamente por uma porta serial, padrão RS-232, caso seja um notebook como host, ou através de um modem à rádio, ou ainda, utilizando uma conexão TCP/IP.

3.4. O sonar Polaroid 6500

A palavra sonar é na verdade uma abreviatura para *SOund NAvigation and Ranging*, ou seja, medição e navegação por som. Este termo é usualmente empregado como sinônimo para um sistema ou um dispositivo (sensor) que utiliza a propagação de ondas sonoras como método para a determinação de medidas e detecção de obstáculos.

Alguns fatores como custo, disponibilidade e popularidade contribuíram para que os modelos de sonares fabricados pela Polaroid fossem os mais adotados até hoje. Entretanto, talvez uma contribuição inusitada para essa difusão tenha sido uma descoberta, por parte de alguns pesquisadores pioneiros da robótica, quando encontraram um modo de reciclar velhas câmeras fotográficas da Polaroid, que utilizava um sistema de sonar para ajustar o foco da máquina e o adaptaram em seus "robôs caseiros" (Figura 3.6a). Atualmente é possível de se

encontrar tutoriais na Internet, como por exemplo [REMINGTON] e [RECYCLING], fornecendo informações, passo a passo, para que qualquer pessoa possa realizar as modificações necessárias.



(a) um dos modelos de câmera fotográfica instantânea com ajuste automático, que empregava o sistema de medição por sonar série 6500, fabricada pela Polaroid na década de 80.

(b) esse mesmo sistema de sonar na sua atual forma de comercialização (cerca de US$ 38.00 incluindo o módulo de controle).

Figura 3.6 – Polaroid Série 6500

O sistema de sonar é composto por um módulo de medição e um único transdutor eletrostático, que é empregado tanto para a transmissão quanto para a recepção das ondas de alta freqüência. Com temperatura de operação recomendada entre 0 e 40° C, este sistema é capaz de medir distâncias entre 150 e 10660mm, ou seja, aproximadamente de 15 centímetros a 10 metros e meio.

O módulo de medição possui componentes para: (1) gerar o sinal digital para o transdutor, (2) funções para controle de tempo, (3) receber, amplificar e filtrar a onda de eco retornada, (4) processar o sinal e prover uma saída TTL.

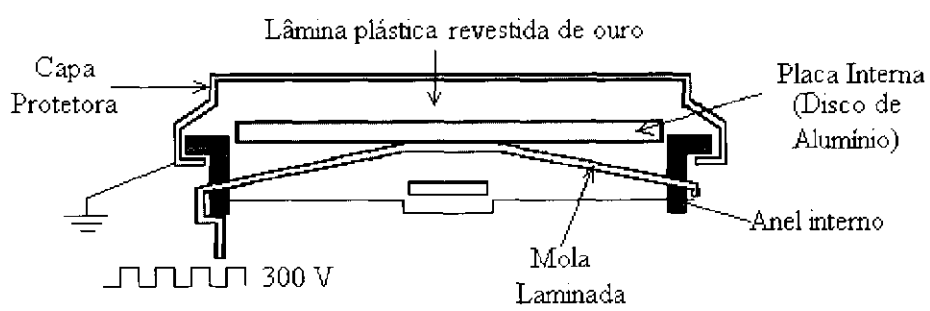


Figura 3.7 – Desenho esquemático de um transdutor eletrostático.

Completando o conjunto, temos o transdutor eletrostático que é um dispositivo similar a um capacitor elétrico, formado por duas placas, sendo uma fixa e uma móvel. A placa fixa é, na verdade, um disco de alumínio com ranhuras concêntricas em uma de suas faces. A outra placa,

mais fina e móvel, é confeccionada com um material plástico chamado kapton, e revestida com uma fina película de ouro, que durante a montagem, fica assentada sobre uma face da contra-chapa de alumínio (Figura 3.7).

O material plástico age como um isolante entre o ouro e o alumínio, então, quando um potencial elétrico (um sinal em forma de onda quadrada com 16 ciclos à freqüência de 50 kHz) é colocado entre estas duas placas, uma força eletrostática age sobre a lâmina e a atrai em direção a contra-chapa de alumínio. O movimento da lâmina desloca o ar gerando um estampido ultra-sônico, nesta mesma freqüência, que será transmitido por 0,32 ms (16 ciclos).

A onda sonora produzida viaja através do ar, até que colida com algum objeto, quando então, sofrem reflexão produzindo um eco. Essas ondas de retorno são captadas pelo transdutor que novamente as converte em um sinal elétrico. Dadas as características da viajem e da forma como o eco é refletido, este sinal necessita ser amplificado, filtrado e discretizado antes que possa ser devidamente processado (figura 3.8).

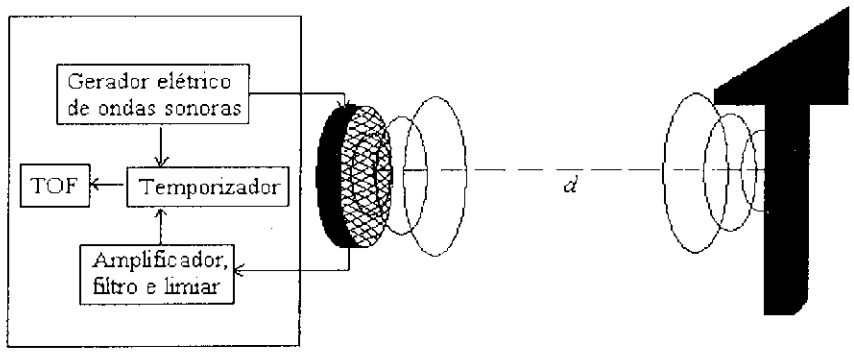


Figura 3.8 – Sistema de medição por ultra-som da Polaroid série 6500.

Por fim, calculando o intervalo de tempo entre a emissão e a recepção das ondas, determina-se o que chamamos de tempo de vôo (TOF – Time-of-flight), a partir do qual uma medida indireta da distância entre o sensor e o objeto é obtida a partir do simples cálculo:

$$d = \frac{c_s \times \Delta t}{2}$$

onde c_s é a velocidade de propagação do som no ar ($\approx$ 340 m/s), Δt é o intervalo de tempo transcorrido entre a emissão e a recepção do pulso ultra-sônico (TOF). Lembrando que o pulso viaja até o objeto e depois realiza o mesmo deslocamento durante o retorno, o valor desejado é a metade da distancia total percorrida pela onda sonora.

A velocidade com a qual o pulso ultra-sônico se propaga pelo ar pode sofrer pequenas variações em virtude das condições de temperatura, de umidade e do vento. Em um ambiente fechado, esses fatores são pequenos causando poucas ou nenhuma alteração. Desse modo então, a velocidade do feixe é próxima à velocidade do som (≈ 340 m/s).

O alcance do dispositivo é diretamente proporcional a energia transmitida pelas ondas de ultra-som, P_{trans}. Segundo Wijk [WIJK,02], essa energia apresenta um decréscimo proporcional ao quadrado da distância percorrida, bem como, sofre uma perda exponencial devido à atenuação ou absorção do som pelo ar. A perda na intensidade do sinal pode ser obtida por:

$$P_{trans} = \frac{e^{-ad}}{d^2}$$

onde a é o coeficiente de absorção do meio no qual a onda se propaga.

Para se compensar a energia perdida pelo pulso ultra-sônico até o momento da detecção da onda de eco, o sistema amplifica os sinais recebidos em um dos 16 diferentes níveis de ganho, o qual é selecionado com base no tempo transcorrido entre a emissão e o retorno da onda sonora. Estes valores são, na verdade, uma aproximação discretizada dos valores obtidos pela função inversa a dada na equação acima.

É importante notar ainda que a energia emitida pelo transdutor não é uniformemente distribuída para todas as direções, mas sim, este concentra a maior parcela de energia na direção normal a ele.

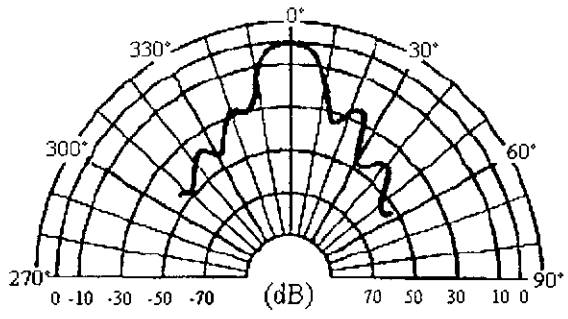


Figura 3.9 – Gráfico da amplitude máxima à 50kHz.

Traçando um gráfico da amplitude da onda captada (eco) pelo ângulo de abertura (Figura 3.9), pode-se observar a formação de um lóbulo principal com aproximadamente 25° de largura (de 347° à 12°). Ao lado deste, formam-se outros dois lóbulos, bem menos significativos que o primeiro, mas ainda assim, fortes o suficiente para serem detectados pelo transdutor. Por fim,

mais dois laterais, com sinais muito baixos, mas em uma angulação relativamente grande.

Para se evitar que falsos ecos sejam indevidamente interpretados como respostas reais, o módulo receptor é equipado com um circuito responsável pela filtragem das freqüências captadas, desprezando todas as outras freqüências que não estejam próximas a de transmissão.

Na prática, a maior parte dos ecos captados pelo transdutor são provenientes do lóbulo principal, apesar de que em se tratando de objetos grandes, como paredes por exemplo, os sinais provenientes dos lóbulos laterais também acabam sendo detectados.

É bem sabido que alguns sensores são muito sensíveis a ruídos, quer sejam de natureza externa ou inerente ao próprio dispositivo. Talvez uma categoria de sensores no qual esse problema pode ser evidenciado sejam os sonares, que apesar da precisão de suas estimativas estar geralmente dentro de 1% da distância verdadeira, apresenta outros de fatores, que ao longo do tempo, deram a eles uma reputação um tanto quando má. São estes:

- *Ecos fracos:* ondas de eco com baixa intensidade levam muito tempo para ser integrada até o ponto em que alcancem o limiar de detecção, dificultando a interpretação e obtenção das estimativas de medida.
- *Reflexões múltiplas:* uma onda de som pode ser refletida em mais de um objeto até que seja finalmente refletida na direção do receptor. Isto causa um erro na estimativa da medida, uma vez que o pulso ultra-sônico realiza um percurso maior do que a distancia real entre o dispositivo e o objeto.
- *Conversa transversal:* em caso de múltiplos sonares emitindo ondas de mesma freqüência, um problema de interpretação dos ecos pode surgir, uma vez que, um sinal de retorno pode ser captado por outro sensor que não necessariamente seja o emissor daquele sinal. Este problema poderia ser facilmente resolvido se os sonares emitissem sinais em freqüências diferentes, porém, isso requer a manipulação do hardware padrão, sem mencionar as modificações nos processamentos dos sinais.
- *Má definição angular:* devido ao largo ângulo de abertura do lóbulo principal (25°), a origem de um eco é incerta quanto à determinação da localização exata do objeto. A onda refletida pode basicamente se originar de qualquer ponto da parte longitudinalmente de uma superfície esférica (Figura 3.10), ou pior, ser proveniente de um dos lóbulos laterais.

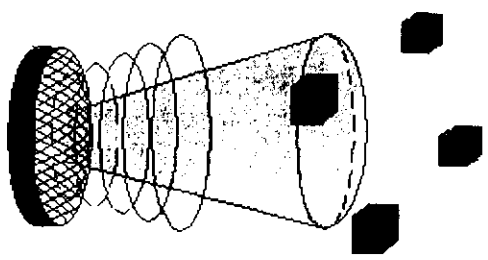


Figura 3.10 - Cone de propagação do sinal de ultra-som.

- *Limitação na escala de detecção:* as especificações técnicas do Polaroid relatam que os dispositivos da série 6500 são capazes de medir distâncias de 0,2 à 10m. Entretanto, constatações empíricas provam que, para uma estimativa mais fiel, o dispositivo deve operar entre 0,2 e 5m.

- *Cadência dos disparos:* por ser a velocidade de som muito mais lenta que a velocidade de luz, os sonares levam mais tempo que os sensores lasers para estarem disponíveis para um novo disparo ultra-sônico. Este fato impõe limitações na taxa de amostras que podem ser obtidas por um sonar. Por exemplo, se pulso for emitido de encontro a um objeto à 5m de distância, não há meios do dispositivo processar o sinal em menos 29ms, uma vez que este é o tempo mínimo necessário para que a onda de som viaje até o objeto e retorne.

3.5. Filtro de Kalman

O Filtro de Kalman foi enunciado em 1960 por Rudolph E. Kalman [KALMAN,60]. Este filtro é um *algoritmo recursivo ótimo para processamento de dados*, ou seja, trata-se de um algoritmo matemático computacional baseado na teoria de sistemas dinâmicos e nos conceitos estatísticos.

O termo *recursivo* indica que, para o seu funcionamento, o filtro não tem a necessidade de armazenar todas as entradas utilizadas anteriormente, uma vez que, apesar de levar em conta todas essas informações, ele não reprocessa os dados antigos quando uma nova de entrada é fornecida.

Quanto a expressão *ótimo*, sabemos que são inúmeras as maneiras pelas quais se define este termo, principalmente porque depende diretamente dos critérios utilizados para se medir o desempenho. De qualquer forma, alguns dos aspectos que justificam essa qualificação seriam,

segundo [MAYBECK,79]: o fato de o filtro incorporar toda a informação disponibilizada, processar todas as medições independentemente da precisão, estimar o valor das variáveis desejadas utilizando o conhecimento sobre a dinâmica do sistema e do dispositivo de medida, descrições estatísticas dos ruídos do sistema e dos erros de medidas, grau de incerteza nos modelos, e ainda, qualquer outra informação sobre as condições iniciais e as variáveis de interesse.

O funcionamento do filtro baseia-se na propagação da densidade probabilística condicional, sendo normalmente indicado para problemas que envolvem modelos lineares e que sofrem interferência de ruídos brancos nas medidas e ou no sistema.

Embora ao contrário do que se pode pensar inicialmente, o emprego do filtro não se restringe apenas a modelos lineares, mas quando se obedece a estas especificações, a média[1], a moda[2], a mediana[3] ou qualquer outra escolha razoável para ser uma estimativa ótima, terá valores coincidentes, o que implica que haja somente uma única "melhor" estimativa para cada um dos valores medidos.

3.5.1. Funcionamento do Filtro de Kalman

Para entender o seu funcionamento, vamos desenvolver um exemplo. Suponha que um indivíduo A esteja perdido no mar durante a noite e que não tenha nenhuma idéia da sua posição (para efeitos de simplicidade, vamos considerar uma localização unidimensional).

Para obter a sua localização, este indivíduo observa uma estrela em um exato instante de tempo t_1 e determina que a sua posição é igual a z_1. Entretanto, devido a própria natureza da medição, ao erro humano e outros fatores deste gênero, a medida obtida apresenta um certo grau de incerteza, levando o indivíduo A, a assumir que a precisão de sua estimativa deva ser de tal forma que o desvio padrão envolvido seja igual a σ_{z_1}.

Dessa maneira, podemos estabelecer a probabilidade condicional de $x(t_1)$, que é a sua posição real no tempo t_1, condicionando-a ao valor observado em sua medição inicial z_1 e obtendo a densidade da probabilidade condicional, conforme representado na Figura 3.11.

---

[1] *Centro de massa das probabilidades*
[2] *Valor de x que tem probabilidade máxima, localizado no pico da densidade.*
[3] *Valor de x que divide os pesos das probabilidades a esquerda e a direita igualmente.*

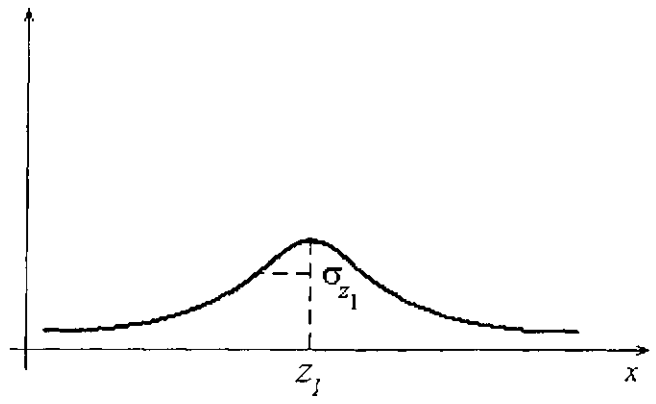


Figura 3.11 - Densidade condicional da posição baseada no valor z_1.

A partir da curva traçada pela densidade da probabilidade condicional, descrita por $f_{x(t_1)|z(t_1)}(x \mid z_1)$, a melhor estimativa que se pode obter para a posição do indivíduo A, denotada por $\hat{x}$, é:

$$\hat{x}(t_1) = z_1 \tag{1}$$

uma vez que z_1 é a mediana e corresponde a moda da densidade (pico da função), bem como, ao valor médio. Por conseguinte, a variância do erro na estimativa é dada por:

$$\sigma_x^2(t_1) = \sigma_{z_1}^2 \tag{2}$$

Após esses passos, o indivíduo A pede a um indivíduo B para que realize a mesma tarefa, obtendo, igualmente a ele, um ponto qualquer no céu como referência, entretanto agora em um instante de tempo t_2 (sendo $t_2 \cong t_1$ para que a posição real ainda não tenha sido mudada). Então, o indivíduo B obtém uma medida z_2 com uma variância σ_{z_2}.

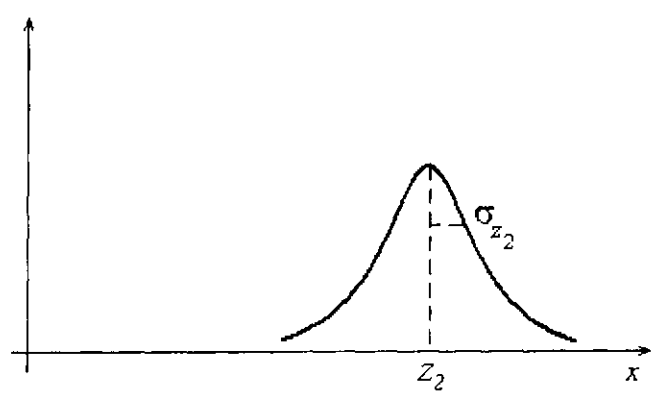


Figura 3.12 - Densidade condicional da posição baseada apenas no valor z_2.

Supondo que esta outra pessoa seja mais experiente que A, podemos assumir que a

variância na medida obtida por ele seja relativamente menor do que a variância da medida obtida por A, como poder ser notado na Figura 3.12.

Neste ponto dispomos de duas medidas para estimar a posição do indivíduo A, mas a questão é, como combinar esses dados? Isto pode ser observado subseqüentemente, uma vez que, baseado nas suposições feitas, a densidade condicional da sua posição no tempo $t_2 \cong t_1$, ou seja, $x(t_2)$ dado z_1 e z_2, é a densidade Gaussiana com média μ e variância σ^2 como indicado na Figura 3.13.

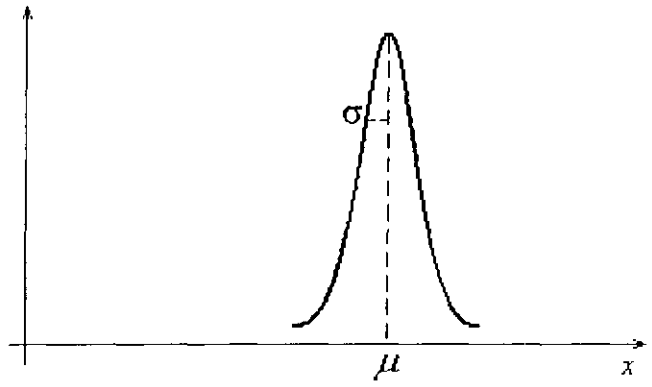


Figura 3.13 - Densidade condicional da posição baseada nos valores z_1 e z_2.

Conseqüentemente, temos:

$$\mu = \left[\frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\right] z_1 + \left[\frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\right] z_2 \qquad (3)$$

$$\frac{1}{\sigma^2} = \frac{1}{\sigma_{z_1}^2} + \frac{1}{\sigma_{z_2}^2} \qquad (4)$$

Observe que, a partir da equação (4), σ é menor que σ_{z_1} ou σ_{z_2}, indicando-nos que o grau de incerteza na estimativa da posição tem diminuído por conta da combinação das informações de A e B. Assim sendo, a nossa melhor estimativa é:

$$\hat{x}(t_2) = \mu \qquad (5)$$

com uma variância associada ao erro igual a σ^2.

Pode-se observar com um pouco mais de atenção a maneira como μ é expresso na equação (3) e pode-se perceber que, se σ_{z_2} for igual a σ_{z_1}, isto indica que pensamos nas medidas z_1 e z_2 como sendo igualmente precisas e, desta maneira, a equação que melhor expressaria a estimativa da posição atual é igual ao valor médio entre essas duas medidas.

Em contrapartida, se σ_{z_2} for maior que σ_{z_1}, a incerteza envolvida na medida z_1 é maior que em z_2, e portanto, a equação deverá dar maior credibilidade a medida z_2 uma vez que esta apresenta uma maior precisão.

Finalmente, se a variância da estimativa de z_2, que é σ_{z_2}, for muito grande ou seu valor for maior que σ_{z_1}, entende-se que os dados fornecidos ao filtro não são de boa qualidade e deveriam merecer pouca atenção, embora, dada a característica robusta do filtro, ele ainda nos permite que estes dados sejam processados, a fim de contribuir para o aumento da precisão da saída do filtro.

Baseado nas equações (3) e (5), podemos reescrever $\hat{x}(t_2)$ como sendo:

$$\hat{x}(t_2) = \left[\frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\right] z_1 + \left[\frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\right] z_2 =$$

$$= z_1 + \left[\frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\right][z_2 - z_1] \qquad (6)$$

ou ainda, na forma que é convencionalmente encontrada nas implementações do Filtro de Kalman, considerando a equação (1):

$$\hat{x}(t_2) = \hat{x}(t_1) + K(t_2)[z_2 - \hat{x}(t_1)] \qquad (7)$$

onde

$$K(t_2) = \sigma_{z_1}^2 \left[\sigma_{z_1}^2 + \sigma_{z_2}^2\right]^{-1} \qquad (8)$$

As duas equações 7 e 8, denotam que a nossa melhor estimativa, $\hat{x}(t_2)$, no tempo t_2, é igual a melhor predição deste valor antes da medida z_2 ser realizada, ou seja, igual a $\hat{x}(t_1)$, acrescido de um fator de correção que pondera, de modo ótimo, a diferença entre a medida z_2 e a melhor predição obtida antes da medição de z_2, ou seja, o valor $\hat{x}(t_1)$.

É importante observar a estrutura "predição-correção" presente no filtro de Kalman. Baseado em toda a informação previamente fornecida, o filtro realiza a "predição" dos valores das variáveis desejadas no exato momento em que antecede a próxima leitura a ser realizada. Então, quando uma nova medida é obtida, a diferença entre ela e o valor predito será usada para "corrigir" a predição das variáveis de estado.

Se utilizarmos $K(t_2)$ como é expresso na equação (8), a variância dada pela equação (4)

pode ser reescrita da seguinte maneira:

$$\sigma_x^2(t_2) = \sigma_x^2(t_1) - K(t_2)\sigma_x^2(t_1) \qquad (9)$$

Assim sendo, observe que tanto $\hat{x}(t_2)$ como $\sigma_x^2(t_2)$ incorporam todas as informações contidas na densidade condicional da posição, no instante de tempo t_2, dado z_1 e z_2. Através da propagação dessas duas variáveis através de diferentes índices, a densidade condicional pode ser completamente especificada para qualquer instante de tempo t.

Desta maneira, resolve-se o problema da estimativa estática, apesar de que ainda resta uma questão: E se houver um deslocamento entre as tomadas das medidas z_1 e z_2? Deve-se então considerar também a questão da dinâmica no problema sugerido.

Suponha que o indivíduo A tenha se deslocado por algum tempo antes de obter uma segunda medida. Além disso, suponha que o melhor modelo para descrever esse movimento é dado pela equação da física que descreve o movimento retilíneo uniforme MRU:

$$\frac{dx}{dt} = u + w \qquad (10)$$

onde dx corresponde ao deslocamento espacial, dt ao intervalo de tempo, u denota a velocidade nominal e w denota o termo que representa a incerteza sobre o nosso conhecimento a respeito da velocidade atual, assim como, perturbações, condições anormais e efeitos não considerados por uma equação de ordem simples.

Estes fatores, aos quais são chamados simplesmente de ruído, serão modelados como um único ruído branco gaussiano, w, ou seja, significa que assumirá valores independentemente dos valores anteriores obtidos ao longo do tempo, o que também implica, que terá média igual a zero e variância igual a σ_w^2.

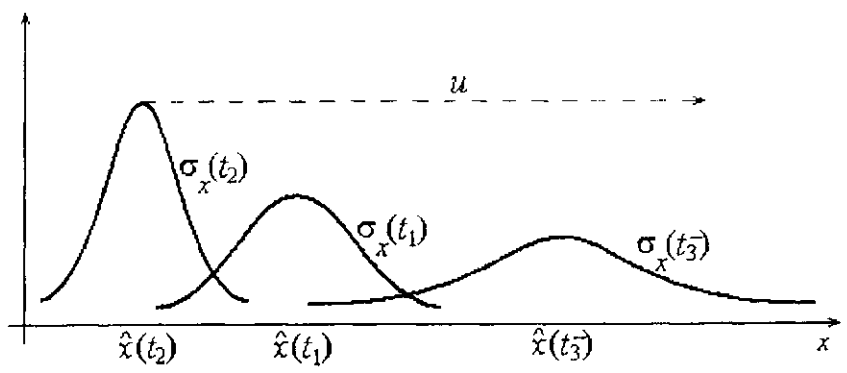


Figura 3.14 – Propagação da densidade condicional.

Se a densidade condicional da posição é propagada no transcorrer do tempo, tem-se um resultado gráfico como o que se apresenta na Figura 3.14. Ao analisá-lo, pode-se observar que, com o transcorrer do tempo, a densidade viaja ao longo do eixo x, a uma velocidade u, sofrendo um achatamento de sua moda devido ao aumento no grau de incerteza envolvido na determinação da posição ao longo do tempo.

No tempo t_3^- (onde o sinal menos que aparece sobrescrito indica um instante de tempo que precedente a realização da medida no tempo t_3), a densidade da probabilidade condicional pode ser matematicamente descrita como a densidade gaussiana, cuja a média e a variância são dadas pelas seguintes equações:

$$\hat{x}(t_3^-) = \hat{x}(t_2) + u[t_3 - t_2] \tag{11}$$

$$\sigma_x^2(t_3^-) = \sigma_x^2(t_2) + \sigma_w^2[t_3 - t_2] \tag{12}$$

assim, $\hat{x}(t_3)$ é a melhor predição do valor de x no tempo t_3^-, que é o instante antes da medida obtida no tempo t_3, e $\sigma_x^2(t_3^-)$ é a variância esperada na predição.

Agora uma medição é realizada e o valor retornado é z_3, sendo que a variância assumida para esta medida será $\sigma_{z_3}^2$. Do mesmo modo que anteriormente, dispõe-se de duas densidades gaussianas contendo informações sobre a posição, sendo que, uma agrega toda a informação provida antes da realização da medida, e uma outra, contém a informação provida pela medida em si.

Empregando o mesmo princípio que foi utilizado na predição estática, realiza-se uma combinação entre a densidade com média $\hat{x}(t_3^-)$ e variância $\sigma_x^2(t_3^-)$ e a densidade com média z_3 e variância $\sigma_{z_3}^2$, resultando em uma terceira densidade gaussiana, cuja média é dada por:

$$\hat{x}(t_3) = \hat{x}(t_3^-) + K(t_3)[z_3 - \hat{x}(t_3^-)] \tag{13}$$

e a variância por:

$$\sigma_x^2(t_3) = \sigma_x^2(t_3^-) - K(t_3)\sigma_x^2(t_3^-) \tag{14}$$

sendo que o ganho $K(t_3)$ é dado por:

$$K(t_3) = \sigma_x^2(t_3^-)[\sigma_x^2(t_3^-) + \sigma_{z_3}^2]^{-1} \tag{15}$$

Assim tem-se que a estimativa ótima, $\hat{x}(t_3)$, é igual a melhor predição do valor de x, antes

da medida z_3 ser obtida, ajustada através de um termo corretor, que é ponderado pelas diferenças entre z_3 e o valor da melhor predição da posição, levando-se em conta as medidas aferidas ao longo do tempo.

Relacionando esses resultados, percebe-se que a estimativa ótima, $\hat{x}(t_3)$, satisfaz a mesma forma que a dada anteriormente através da equação (7). Seguindo essa mesma característica, as equações da variância e do ganho apresentam-se como quando dadas nas equações (8) e (9).

Finalmente, analisando a maneira com $K(t_3)$ é dado, pode-se perceber algumas implicações para os diferentes valores assumidos pelas variâncias. Se o valor da variância do ruído da medida for grande, ou seja, se $\sigma_{z_3}^2$ tiver um valor alto, então o valor de $K(t_3)$ será pequeno, sugerindo que uma medida muito ruidosa foi obtida e, portanto, deve-se atribuir pouca relevância ou confiança a ela. Em outras palavras, se o limite de $\sigma_{z_3}^2 \to \infty$, isto implica que $K(t_3)$ tenderá a zero, e portanto, teremos $\hat{x}(t_3)$ igual a $\hat{x}(t_3^-)$; desta maneira uma medida infinitamente ruidosa será completamente ignorada.

Por outro lado, se a variância no ruído do sistema dinâmico, σ_w^2, for grande, então $\sigma_x^2(t_3^-)$ também será grande (vide equação (12)), assim como $K(t_3)$; por essa situação, entende-se que não há muita incerteza no valor estimado pelo filtro. Note que se o limite $\sigma_w^2 \to \infty$, então $\sigma_x^2(t_3^-) \to \infty$ e $K(t_3) \to 1$, implicando em $\hat{x}(t_3) = z_3$. Deste modo, se a saída do modelo atingir seu limite absoluto de incerteza, a política adotada será a de ignorar totalmente a saída do filtro e utilizar a nova medida diretamente como o estado ótimo.

Em última instância, se $\sigma_x^2(t_3^-)$ tornar-se zero, assim também acontecerá a $K(t_3)$. Na verdade, se tal fato ocorrer, significa que há absoluta convicção de que a estimativa antes da medida z_3 ser realizada está correta e, deste modo, pode-se descartar a medição.

3.5.2. Filtro de Kalman em Modelos Estocásticos

Em linhas gerais, a teoria matemática para a modelagem de sistemas dinâmicos, também chamados de modelos estocásticos, é uma maneira de estimar um estado desconhecido a partir de um conjunto de medidas, levando-se em conta a natureza das perturbações (ruído) que ocorrem durante o processo de medição [BROOKS,98].

O processo é governado por duas equações lineares:

$$\vec{x}_{i+1} = A\vec{x}_i + Gu_i \tag{15}$$

$$\vec{y}_{i+1} = H_i\vec{x}_i \tag{16}$$

Sua importância se dá pelo fato que estas equações servem de base para a definição de métodos de estimativa linear, como é o caso do Filtro de Kalman. Este filtro endereça o problema geral de se tentar estimar o estado $x \in \Re^n$ de um processo controlado por uma equação diferencial linear estocástica em tempo discreto:

$$\hat{x}_{k+1} = A\hat{x}_k + Bu_k + w_k \tag{17}$$

com uma medição $z \in \Re^m$ que é:

$$z_{k+1} = Hx_k + v_k \tag{18}$$

onde w_k e v_k são variáveis randômicas que representam, respectivamente, o ruído do processo e o ruído da medição.

A matriz $A_{n\times n}$, na equação (17), relaciona o estado atual com o estado anterior. A matriz $B_{n\times l}$ relaciona a entrada de controle adicional $u \in \Re^l$ com o estado x, enquanto a matriz $H_{m\times n}$, na equação (18), relaciona o estado a medida z_k.

Aplicando esses princípios e fazendo as considerações necessárias para a aplicação em nosso trabalho, como por exemplo, o fato de que não temos uma entrada adicional, podemos reescrever a equação (17) da seguinte maneira:

$$x_{k+1} = Ax_k + w_k \tag{19}$$

Da mesma forma, modela-se o ruído w como a diferença entre a leitura z e a melhor estimativa no tempo k, ponderada por um fator de ganho K, denominado ganho de Kalman. Assim tem-se:

$$H_k = P_k + R \tag{20}$$

$$K_k = AP_kH_k^{-1} \tag{21}$$

$$P_{k+1} = AP_kA^T + Q - AP_kH_k^{-1}P_kA^T \tag{22}$$

$$\hat{x}_{k+1} = A\hat{x}_k + K_k\left(z_{k+1} - A\hat{x}_k\right) \tag{23}$$

Nas equações acima, o sobrescrito -1 indica matriz inversa enquanto o sobrescrito T indica a transposição da matriz. H é chamado de covariância de inovação, K é a matriz de ganho e P

é chamado de covariância do valor do erro da predição.

O quadro a seguir ilustra o processo recursivo de execução do algoritmo:

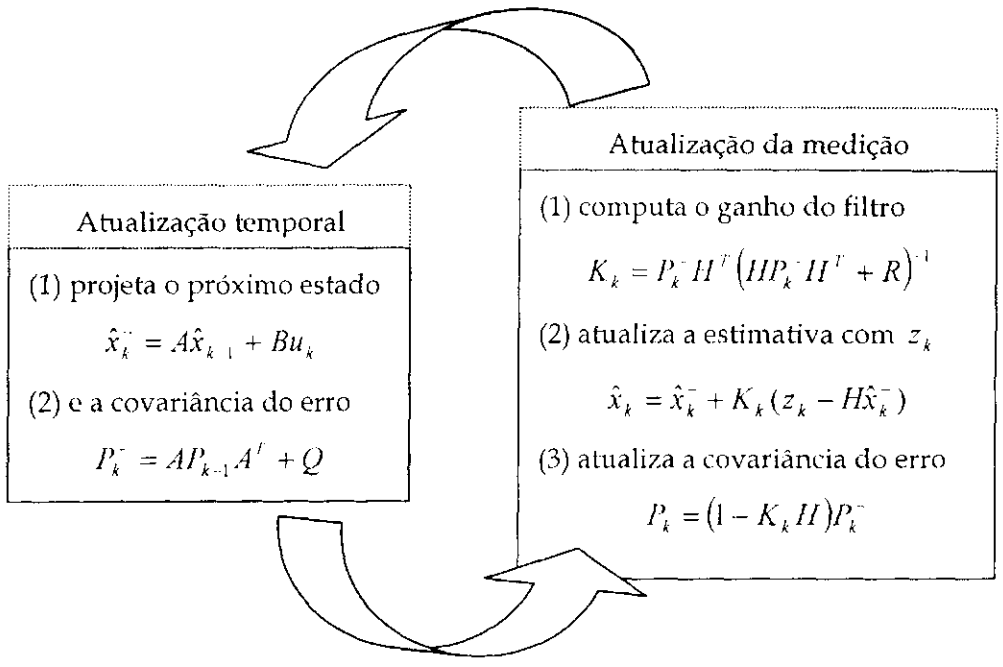


Figura 3.15 - Algoritmo do Filtro de Kalman

As equações de atualização temporal podem ser entendidas como equações de previsão enquanto as equações de atualização da medição podem ser vistas como equações de correção.

A equação (23) é razoavelmente intuitiva, pois o primeiro termo usado para derivar o estado estimado no tempo $k+1$ é apenas A vezes do estado estimado no tempo k. Assim sendo, este pode ser o estado estimado se não se tem nenhuma medida. Em outras palavras, o estado estimado propaga no tempo apenas o vetor do estado. O segundo termo na equação (23) é chamado de termo de correção, representando o quão correta a estimativa propagada implica para nossa medida. A análise da equação (22) nos indica que se o ruído da medida for muito maior que o ruído do processo, K será pequeno, significando que nos não se deve dar muito crédito à medida; entretanto, se o ruído da medida for muito menor que o ruído do processo, K será grande, o que significa que deve-se dar mais crédito à medida do que à estimativa.

3.6. Redes Neurais Artificiais

Chamamos de Redes Neurais Artificiais - RNA, ou simplesmente de Redes Neurais,

mecanismos matemático-computacionais que, inspirados na arquitetura cerebral humana, são capazes de realizar tarefas tais como aprendizado, reconhecimento de padrões, classificação, dentre outras aplicações. As redes neurais modelam estruturalmente o cérebro humano, além de simular o seu comportamento funcional quanto ao aspecto do armazenamento informações, que se dá através do estabelecimento de sinapses (conexões neurais) entre os elementos que a forma.

Tão qual um cérebro jovem, uma RNA precisa aprender, ou seja, ganhar conhecimento através da formação de conexões sinápticas. Para executar essa tarefa contamos com dois diferentes paradigmas de aprendizado:

- *Aprendizado supervisionado*: consiste em treinar a rede através da utilização de vários pares de informação, cada um composto por um valor de entrada e resposta esperada. O valor de entrada é então apresentado para a rede que, após processá-lo, fornece uma resposta. A partir disto, acontece o ajuste das conexões sinápticas de acordo com a diferença entre a resposta esperada e aquela que fora resultada pela rede, determinando assim uma margem de erro entre as repostas. O processo se repete, até que o erro fornecido pela rede esteja em níveis aceitáveis.
- *Aprendizado não supervisionado*: essa modalidade de aprendizado recebe esse nome por, diferentemente do anterior, não apresentar uma resposta esperada para comparação com o valor gerado pela rede. Na verdade, este paradigma de aprendizado consiste em uma abordagem que permite a RNA aprender através de um processo de auto-organização, que se baseia nos valores fornecidos como entrada, agrupando-as de acordo com características semelhantes.

3.6.1. Mapas Auto-Organizáveis (Self-Organizing Maps)

Neste modelo de RNA, os neurônios são distribuídos em forma de um mapa bi-dimensional, através do qual podemos endereçar um único neurônio, de maneira exclusiva, simplesmente utilizando um par de valores que representam a linha e a coluna onde o neurônio se encontra posicionado no mapa. Complementarmente, o modelo apresenta ainda um vetor de neurônios, cada um ligado a todos os outros neurônios que compõem o mapa. A figura 3.X apresenta uma ilustração do que seria uma visão gráfica de uma Rede de Kohonen.

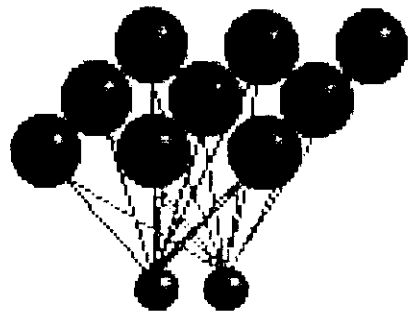


Figura 3.16 - Modelo estrutural de um Mapa Auto-Organizável.

As ligações entre os neurônios do vetor e os do mapa, representam as conexões sinápticas que ocorrem entre os neurônios naturais e são modelados matematicamente, como um vetor numérico, chamado de vetor de pesos, com a finalidade de representar o grau de acoplamento entre os neurônios, ou melhor dizendo, a intensidade da conexão sináptica.

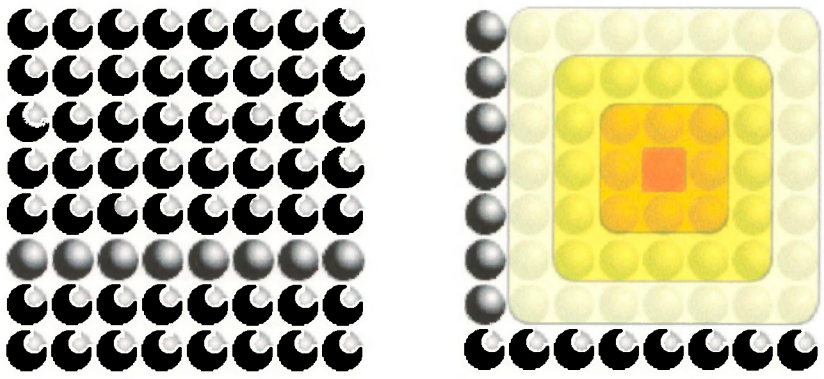
A entrada de informações para a rede neural sempre se dará por intermédio dos neurônios do vetor, que, por este motivo, a partir daqui serão referenciados como *neurônios de entrada*. São estes os neurônios responsáveis por distribuir os valores de entrada para todos os neurônios do mapa.

Diz-se que estes modelos de redes são mapas auto-organizáveis por que se baseiam no aprendizado competitivo, ou seja, dada uma entrada, os neurônios que formam o mapa bidimensional concorrem entre si até que seja eleito um, e apenas um, que melhor represente aquela entrada. A esse neurônio dá-se o nome de neurônio vencedor e ele terá seus valores (vetor de pesos) ajustados para adaptar-se melhor a entrada.

O processo de competição é dado pelo cálculo da diferença entre o vetor de entrada, $\vec{x}$, e o vetor de pesos, $\vec{w}$, para cada um dos neurônios que compõem o mapa. Habitualmente utiliza-se para isso o calculo da distância euclidiana dos vetores $\vec{x}$ e $\vec{w}$ (Equação 24). A eleição do vencedor se dá através da comparação entre as distâncias calculadas, sendo, o que apresentar a menor distância do vetor de entradas, o neurônio vencedor.

$$d_j = \sum_{i=1}^{n} \left\| x_i - w_j \right\| = \sqrt{\sum_{i=1}^{n} \left(w_j - x_i \right)^2} \qquad (24)$$

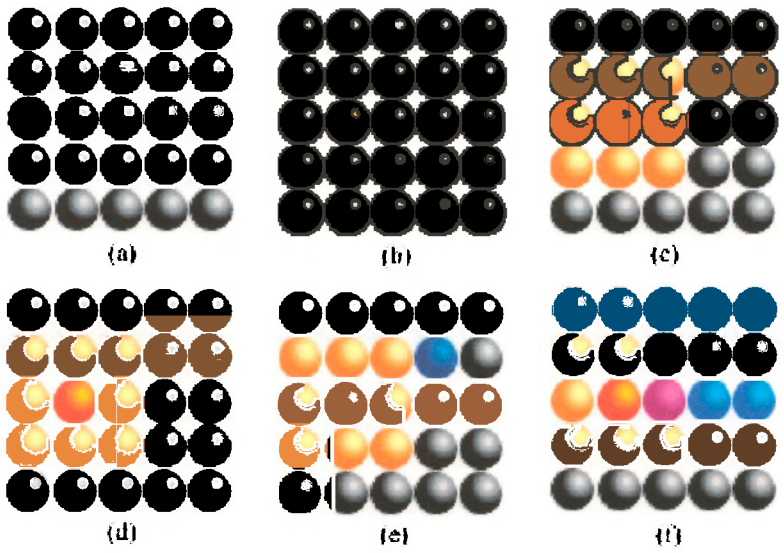
Durante o processo de aprendizado, que para este modelo de RNA é o não-supervisionado, após a apresentação de um vetor de entrada, ocorre um processo de atualização das conexões sinápticas (vetor de pesos) de alguns neurônios que formam o mapa. Isso se dá para que a rede possa modelar as características da informação de entrada.



(a) Mapa no estagio inicial, quando nenhuma entrada foi apresentada. (b) O neurônio em vermelho é eleito como vencedor, e a atualização ocorre para as vizinhanças de raio 1 (em laranja), 2 (em amarelo) e 3 (na cor mais clara).

Figura 3.17 – Determinação da vizinhança do neurônio vencedor.

A atualização dos pesos ocorre não somente para o neurônio vencedor, mas também para todos os neurônios de uma região circunvizinha. O neurônio vencedor tem seus pesos corrigidos, para melhor representar aquele padrão de entrada, enquanto os neurônios das localidades vizinhas terão seus pesos atualizados com uma taxa de aprendizado menor, proporcional à distância que estão em relação ao vencedor, como representado na Figura 3.17.



O mapa em seu estágio inicial(a), recebe uma entrada e após o processamento, o neurônio da 3ª linha, 2ª coluna, vence a competição(b), e então terá seus pesos atualizados, juntamente com os da sua vizinhança (c). Durante a apresentação de uma segunda entrada, o mapa já se apresenta com algumas características aprendidas (d). Se esta nova entrada for relativamente diferente da primeira, certamente um outro neurônio irá vencer a nova competição (e) e consequentemente criar um novo cluster. Note que com o passar do tempo, a rede terá alguns neurônios que agregarão características de vários grupos (f.

Figura 3.18 – Atualização de pesos durante a apresentação de duas entradas consecutivas.

Por meio da atualização da vizinhança os neurônios assumem características regionais, ou seja, dada uma posição do mapa, os neurônios que compõem a região ao redor desta posição possuirão algumas similaridades entre si. Desta forma, consegue-se a formação de agrupamentos (Figura 3.18) e conseqüentemente uma diminuição no grau de entropia[4] do mapa.

[4] Diz respeito ao grau de desordem de um determinado elemento.

Capítulo 4 - Um sistema para Localização

Utilizando concomitantemente o simulador do sistema Saphira e o robô Pioneer 1 do LABIC como a plataforma de testes para a realização deste trabalho, buscou-se uma solução adequada e compatível de forma a viabilizar a implementação de um sistema de localização, utilizando as técnicas descritas anteriormente. É importante salientar que a solução aqui apresentada é até certo momento uma proposta genérica, entretanto inevitavelmente esta abordagem tornou-se, em outras instâncias, mais específica aos casos abordados.

4.1. Visão geral do Sistema

Sendo o robô Pioneer 1 uma das plataformas que executam o sistema Saphira, desenvolveu-se um programa que se conecta ao controle do robô como uma aplicação cliente.

A maior parte das rotinas foram implementadas sob a forma de processos de usuário no sistema Saphira, sendo que todas foram codificadas usando apenas as primitivas mais atômicas disponíveis pelo sistema, isto é, rotinas que realizam a interface direta com o hardware do robô e não apresentam nenhum tipo de recurso de alto nível como, a definição de comportamento ou a interpretação dos sinais provenientes do sonar.

Na verdade, todo o controle de navegação e de localização, foi implementado do início ao fim, deixando a cargo do sistema Saphira apenas questões de interface de hardware e comunicação com o host. O sistema desenvolvido neste trabalho pode ser dividido em pequenos módulos, cada qual apresentando um objetivo específico e estando completamente integrado aos outros. São eles:

- *Módulo de navegação:* que tem como objetivo prover os quesitos básicos para o módulo de localização, fundamental para os estudos desse trabalho. Composto por pequenas rotinas implementadas como MEFs no sistema Saphira, este módulo é responsável pelas tarefas inerentes à navegação de robôs móveis em um ambiente fechado

(indoor), tais como, a exploração do meio e a capacidade de evitar colisões com objetos estáticos e dinâmicos. Uma visão mais detalhada e minuciosa é apresentada na seção 4.2.

- *Módulo de Filtragem:* este módulo contém as rotinas responsáveis pela filtragem dos dados de todos os sonares do robô Pioneer 1 de maneira individual, disponibilizando os valores obtidos para todos os outros módulos do sistema, que poderão acessá-los e manipulá-los de acordo com interesses diversos. Um detalhamento da sua implementação, bem como, informações sobre os parâmetros de ajuste, são descritos na seção 4.3.

- *Módulo de Apoio:* constituído por todas as rotinas auxiliares que tem por objetivo prover o apoio para os outros módulos do sistema, tais como, inicialização dos processos modelados em MEFs, funcionalidades de depuração, arquivos de log, exibição de parâmetros internos, bem como, funções para a representação gráfica das áreas de sensibilidade e pontos detectados. Todas as rotinas que fazem parte deste módulo estão codificadas em *OtherTasks.cpp* e *OtherTasks.h*. Não há um item especialmente dedicado a este módulo, uma vez que, as rotinas aqui contidas têm um caráter geral e podem ser individualmente comentadas ou citadas nos itens em que se discorre a respeito do módulo ao qual se relacionam.

- *Parâmetros de configuração:* essencial para o funcionamento do sistema, este módulo é responsável por fornecer todos os parâmetros necessários ao funcionamento dos diferentes subsistemas que compõem a aplicação. Codificado na forma de um arquivo texto dividido em sessões, uma gama de identificadores são especificados e valorados com o propósito de fornecer os valores aos parâmetros indispensáveis para o funcionamento do sistema, bem como, outros parâmetros opcionais para que podem ser úteis para processos de depuração. Cada parâmetro será individualmente comentado nos itens subseqüentes, de acordo com a sua implicação para o sistema, ou ainda, poder-se-á observá-los no apêndice A.

4.2. Sistema de Navegação

Inicialmente, motivado pelas pesquisas de [NEHMZOW,91] desenvolveu-se um sistema de navegação muito simples, no qual o robô tenta acompanhar a parede localizada a sua direita,

mantendo uma distância sempre constante durante todo o trajeto, além, é claro, de evitar colisões.

Esse sistema foi concebido para operar em um ambiente simples, tal qual se apresenta na Figura 4.1. Entretanto deve-se ressaltar uma característica adicional importante que é a ausência de elementos dinâmicos.

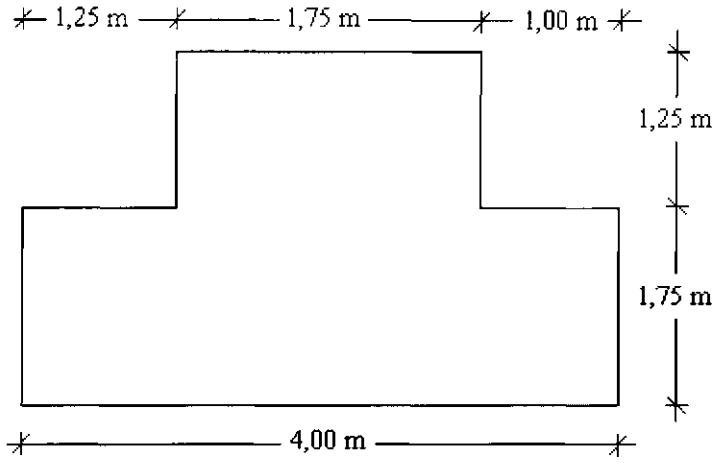


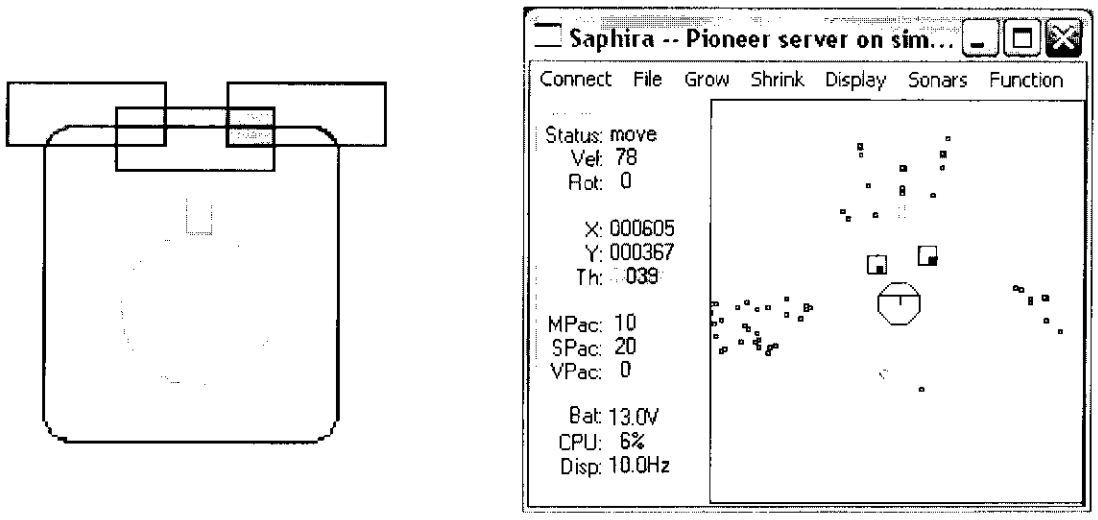
Figura 4.1 - Ambiente do ALDER

Uma vez que o robô fosse capaz de navegar sem colidir e acompanhando o layout do ambiente , havia aqui os quesitos necessários para fundamentar os experimentos com a rede de Kohonen, então fornecíamos como entrada para o Mapa de Kohonen um vetor composto por uma combinação de informações referentes à movimentação do robô, ou seja, um descritor para deslocamentos à frente, para as curvas à esquerda e à direita.

Após os testes junto ao simulador, havia o desejo de se testar o sistema em uma situação real, ou seja, no robô Pioneer 1 propriamente dito. Assim sendo, um modelo de mundo mais complexo se apresentava e conseqüentemente adaptações se fizeram necessárias. Sendo o primeiro sistema de navegação funcional mas muito simples, neste momento optou-se pela construção de um novo sistema de navegação, mais robusto e com características que independesse do layout do ambiente em que o robô estava.

Este novo sistema emprega a idéia de janela virtual, que consiste em se estabelecer uma área ao redor do robô que represente o espaço necessário para a execução de uma ação de desvio sem que haja uma colisão. Em outras palavras, trata-se de definir uma distância mínima que resguarde o robô de qualquer elemento do ambiente, servindo também como uma área de sensibilidade para a tomada de decisões relacionadas a navegação.

Na Figura 4.2, tem-se um esquema que representa conceitualmente a janela virtual. Repare que foram definidas três áreas de sensibilidade frontais, a partir das quais o robô percebe o ambiente a sua frente e, então, decide por qual direção irá optar para se desviar do elemento percebido. Note que há uma área de intersecção entre estas áreas, duas a duas, a fim de fornecer uma informação mais rica e precisa a respeito do objeto detectado pelo sonar.



A esquerda tem-se uma ilustração do conceito de janela virtual juntamente com o destaque para o posicionamento das áreas de sensibilidade que servirão para a detecção de elementos na dianteira do robô. Na imagem à direita, pode-se observar a janela do sistema Saphira durante a execução da aplicação, onde destacam-se: as áreas de sensibilidade representadas em laranja, as leituras dos sonares em azul e das marcações dos pontos detectados, como os retângulos em vermelho e em verde.

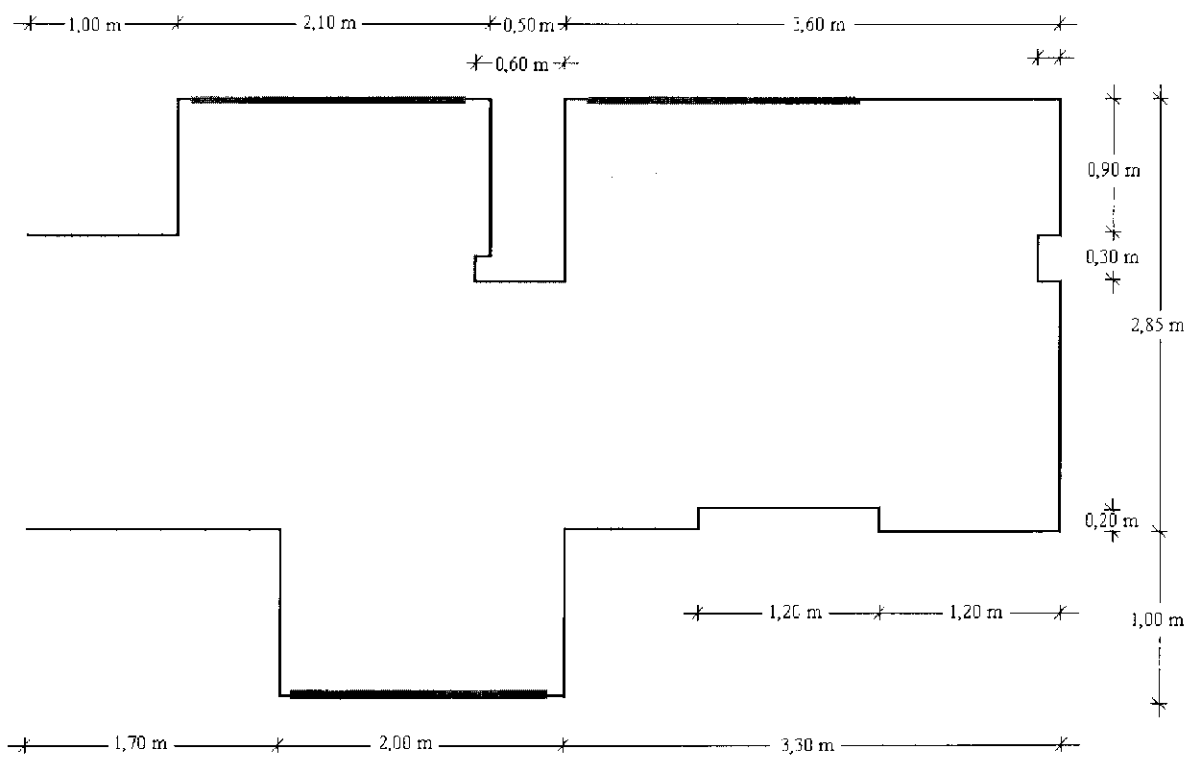
Figura 4.2 - A Janela Virtual

A detecção dentro de cada uma destas áreas de percepção pode ser realizada por qualquer um dos sonares que realizem um cone de varredura cobrindo, ainda que uma pequena porção do espaço compreendido pela área de sensibilidade. Devido a falta de mais sonares laterais no robô Pioneer 1, não puderam ser criadas áreas tão ricas em informações para as laterais da janela virtual, embora foi utilizado o único sensor lateral para evitar aproximações do robô com paredes quando descrevendo trajetórias quase paralelas com paredes, com uma distância mínima que pode ser estabelecida através do parâmetro *how_close_to* no arquivo de parâmetros.

O tamanho da janela é estabelecido durante a inicialização do sistema e tem seus valores determinados por meio dos parâmetros *window_width* e *window_height* do ítem *[Navigation System]*, que devem ser fornecidos no arquivo de configuração do sistema (para detalhes vide Apêndice A). Automaticamente, as áreas de percepção frontais são redimensionadas para se ajustarem as dimensões da janela virtual.

O sistema ainda oferece um parâmetro chamado *draw_sensing_areas* e um outro denominado *draw_detected_points*, ambos definidos no ítem *[Debugger]* do arquivo de configuração, através dos quais podem-se controlar a exibição ou não das áreas de sensibilidade e dos pontos detectados por elas na janela do sistema Saphira. Este recurso pode ser útil para a execução de ajustes finos no dimensionamento da janela virtual.

Também foi construído uma réplica do corredor que interliga alguns laboratórios do Bloco1 do Instituto de Ciências Matemáticas e de Computação - ICMC/USP (Figura 4.3), a fim de que pudessem ser realizadas experiências também utilizando o simulador presente no sistema Saphira. As especificações do arquivo que representa este bloco pode ser encontradas no Apêndice B deste trabalho.



Vista superior, corte longitudinal da porção final do corredor que abriga os laboratórios LABIC, LASD e Intermídia do Bloco 1 do ICMC – USP. Mapa obtido a partir da medição real do ambiente, onde os quadrados formados pelas linhas em cinza têm correspondência direta com piso que reveste o corredor, guardando-se, também, a devida escala.

Figura 4.3 – Corredor do Bloco 1 do ICMC

4.3. Filtragem dos dados dos sonares

Os sonares são dispositivos que, por natureza, apresentam-se muito suscetíveis a erros de

leitura, quer seja por causa de perturbações externas ou por má interpretação dos sinais de retorno. Desta forma, para se operar com uma taxa de credibilidade mínima, ou seja, obter valores que apresentem uma margem de erro aceitável, os valores obtidos por esta categoria de dispositivo necessitam de algum tipo de tratamento.

Através de uma análise experimental dos dados obtidos pelos sonares do robô, observou-se que a margem de erro apresentada pela leitura em relação ao valor real é diretamente proporcional à distância do sensor ao objeto.

Tais aferições foram realizadas com o robô parado, sendo realizadas várias tomadas a partir de cada posição e, posteriormente, repetia-se o processo para diferentes distâncias. O que se pôde perceber é que mesmo que se pudesse estabelecer um valor ou expressão para a margem de erro, esta deveria levar em consideração não somente a distância entre o sensor e o anteparo, mas o ângulo de reflexão do sinal e outras perturbações inerentes ao ambiente e ao movimento.

4.3.1. Utilizando o Valor Médio

Uma primeira idéia para se solucionar o problema das perturbações nas leituras obtidas pelos sonares foi a de se trabalhar com a média dos valores obtidos pelo sonar ao longo do tempo, ou seja, uma vez que os valores aferidos apresentavam erros, para se obter uma boa estimativa da distância real entre o robô e o objeto, se deveria realizar algumas medições consecutivas e então calcular a média dos valores obtidos.

Pensando na possibilidade de se explorar outras técnicas estatísticas, que não simplesmente a média entre os valores, a implementação desta idéia deveria ser de tal modo que pudesse oferecer, através de uma pequena e fácil adaptação, o suporte necessário ao método estatístico que se desejasse aplicar.

Por possuir características bem peculiares, como tamanho fixo, sobreposição do elemento mais antigo, e ainda, por ser capaz de manter a ordem dos elementos inseridos ao longo do tempo, como se fosse um histórico, optou-se pela construção de um buffer (uma fila circular) para o armazenamento dos dados provenientes dos sonares.

Havia a necessidade de se buscar um melhor desempenho uma vez que as MEFs são executadas a cada 100ms, e portanto, os algoritmos codificados não podem demandar um tempo muito grande de execução. Dessa maneira, a implementação deu-se através de um algoritmo recursivo que rendeu uma boa economia no tempo de execução, mesmo que em

detrimento do histórico das leituras.

4.3.2. Utilizando Filtro Kalman

Tendo em vista as características do Filtro de Kalman para o processo de estimativa, bem como, a característica recursiva de seu algoritmo onde leva-se em consideração os estados anteriores sem a necessidade de armazenamento, nos lançamos à tarefa de implementar o filtro para processar os sinais provenientes dos sonares, codificando-a sob a forma de um processo de usuário no sistema Saphira.

```
void __cdecl prcKalmanFilter(void)
{
   switch(process_state)
   {
      case sfINIT:
         for (i=0;i<7;i++)
         {
            x_[i] = sfSonarRange(i);
            eVar_[i] = 1;
         }
         process_state = sfRESUME;
         break;
      case sfRESUME:
         for (i=0;i<7;i++)
         {
            KalmanGain[i] = eVar_[i]/(eVar_[i] + KalmanParamR);
            Z[i] = sfSonarRange(i);
            x[i] = x_[i] + KalmanGain[i] * (Z[i] - x_[i]);
            eVar[i] = (1-KalmanGain[i])*eVar_[i];
            x_[i] = x[i];
            eVar_[i] = eVar[i] + KalmanParamQ;
         }
         break;
      case sfSUSPEND:
         break;
      case sfINTERRUPT:
         break;
      case sfREMOVE:
         break;
   }
}
```

Figura 4.4 – Implementação do Filtro de Kalman

Na Figura 4.5 tem-se um gráfico com os valores das leituras realizadas por um dos sonares relacionados com as respectivas saídas fornecidas pelo filtro.

(a) Detalhe da curva de ajuste do filtro para variações bruscas nas leituras. (b) Filtragem do ruído das leituras, atenuação dos valores. (c) Segmento similar à primeira porção do gráfico, embora com ruído maior.

Figura 4.5 – Gráfico: Leituras do sonar x Saída do filtro

A fim de proporcionar uma melhor visualização do impacto causado pelo filtro na atenuação dos ruidosos valores do sonar, foram confeccionados pequenos quadros que ampliam a visão de três diferentes porções que a curva do gráfico descreve.

Para a obtenção deste gráfico foram realizadas 496 leituras do sensor lateral direito do robô, enquanto ele percorria uma trajetória retilínea e paralela a uma parede do ambiente. Durante esse percurso, em um dado momento, essa parede sofre um desvio brusco de 1000mm da posição inicial, permanecendo com a mesma direção paralela à trajetória, porém com 1 metro a mais de distância além do que já possuía em relação ao robô.

A amplitude do gráfico é de 0 à 2500mm, com marcações intermediárias a cada 250mm (linhas tracejadas). Em cada uma das janelas em que apresentam destaques de regiões do gráfico a escala é preservada e dessa forma, o que se pode visualizar é uma ampliação da curva do gráfico.

Após a 296ª leitura, o sensor passa a captar a parede a uma distância de 2111mm, correspondendo a primeira leitura realizada após o desnivelamento de 1 metro da parede. Podemos notar que nas quarenta leituras seguintes, os valores retornados pelo sonar são completamente equivocados. Simultaneamente, começamos a perceber um crescimento nos valores gerados pelo filtro a fim de acompanhar as variações da entrada.

Mesmo se considerarmos que o filtro ainda levará cerca de outros quarenta ciclos para atingir o valor da nova distância, esta abordagem ainda apresenta vantagens em relação ao valor de retorno do sonar, pois a variação dos valores é gradativa e tênue, diferentemente dos valores retornados pelo sonar para a mesma situação. Outro fator relevante é o ajuste dado ao filtro pelos parâmetros que representam a covariância do erro do processo e do erro da leitura, pois, através deles, controlamos a credibilidade que o filtro dará as leituras.

4.3.3. Ajustando os parâmetros do Filtro

Como discutido na seção 3.5, o filtro apresenta dois parâmetros que correspondem aos tipos de perturbações que o filtro deve atenuar. Sendo um modelo estocástico, têm-se duas modalidades de ruídos: o do sistema e o das leituras.

Nas equações apresentadas pelo algoritmo da Figura 3.15, o parâmetro R representa o ruído nas medidas aferidas pelo sonar enquanto o parâmetro Q representa o ruído do processo como um todo.

Desta maneira, se R tender a zero (lembrando que ele obedece a uma ordem exponencial) o valor produzido será próximo à leitura obtida, ou seja, não há muita variação entre o valor medido e a saída fornecida pelo filtro.

Já se o valor de Q for zero, nenhuma perturbação na saída será gerada, mas em contrapartida, quanto mais positivo for o seu valor, maior será o impacto que causará na atenuação do sinal ajustado por R.

4.4. Localização

Fundamentalmente o elemento pelo qual se dá a interação entre o robô e o ambiente são os sensores, desta maneira a exatidão dos valores por eles fornecidos está diretamente relacionada com o desempenho do sistema, justificando assim, o uso de algum mecanismo para a filtragem dos dados fornecidos.

Para que se possa estabelecer a posição do robô em relação ao ambiente, desenvolveu-se um sistema capaz de converter as informações obtidas pelo robô a partir de seus sonares, e conseqüentemente locais a sua posição no instante da leitura, para um sistema global orientado cuja origem é coincidente com o ponto em que o sistema foi iniciado.

4.4.1. Sistema de Orientação

Quando um objeto é percebido pelo sonar, este retorna a distância em que este objeto foi percebido. Lembrando que esta informação está relacionada à posição corrente do robô, tem-se um sistema de orientação local, cujo ponto de cruzamento dos vértices consiste no centro de movimento do robô, ou seja, o plano de orientação LPS definido pelo sistema Saphira. Desta maneira, todas as coordenadas obtidas através de leituras dos sonares serão dadas em relação ao LPS e devem, conseqüentemente, serem convertidas para o sistema de representação global GPM, também definido pelo sistema Saphira.

Assim sendo, o problema consiste em buscar as coordenadas equivalentes ao ponto M desejado para o novo sistema de orientação, aplicando para isto, conceitos de geometria analítica tais como translação e rotação. A Figura 4.7 ilustra graficamente o problema da rotação dos eixos:

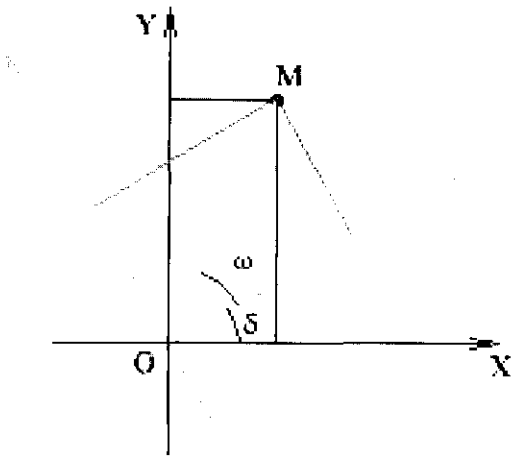


Figura 4.7 – Conversão de coordenadas entre dois sistemas de orientação.

Observando a Figura 4.7, fica perceptível que o segmento $\overline{OM}$ pode ser descrito como o a soma vetorial das projeção de $\overline{OM}$ em cada um dos eixos orientados, ou seja, pode-se escrever $\overline{OM}$, tomado suas projeções para cada sistema de orientação.

Em relação ao sistema $X\hat{O}Y$, tem-se:

$$\overline{OM} = X \cdot \cos(\omega + \delta) + Y \cdot \operatorname{sen}(\omega + \delta) \tag{25}$$

Analogamente, em relação ao sistema $x'\hat{O}y'$, tem-se que:

$$\overline{OM} = x' \cdot \cos(\omega) + y' \cdot \operatorname{sen}(\omega) \tag{26}$$

Observe que tanto a equação (25) quanto a equação dada em (26) determinam o mesmo segmento $\overline{OM}$. Logo, igualando as duas equações obtém-se:

$$X \cdot \cos(\omega + \delta) + Y \cdot \operatorname{sen}(\omega + \delta) = x' \cdot \cos(\omega) + y' \cdot \operatorname{sen}(\omega) \tag{27}$$

Desenvolvendo os termos a esquerda da igualdade da equação (27), e tão somente eles neste momento, pode-se obter:

$$X[\cos(\omega) \cdot \cos(\delta) - \operatorname{sen}(\omega) \cdot \operatorname{sen}(\delta)] + Y[\operatorname{sen}(\omega) \cdot \cos(\delta) + \operatorname{sen}(\delta) \cdot \cos(\omega)] = \ldots$$

$$X \cdot \cos(\omega) \cdot \cos(\delta) - X \cdot \operatorname{sen}(\omega) \cdot \operatorname{sen}(\delta) + Y \cdot \operatorname{sen}(\omega) \cdot \cos(\delta) + Y \cdot \operatorname{sen}(\delta) \cdot \cos(\omega) = \ldots$$

$$[X \cdot \cos(\delta) + Y \cdot \operatorname{sen}(\delta)]\cos(\omega) + [-X \cdot \operatorname{sen}(\delta) + Y \cdot \cos(\delta)]\operatorname{sen}(\omega) = \ldots$$

o que, após escrevermos juntamente com o lado esquerdo da igualdade novamente, resulta em:

$$[X \cdot \cos(\delta) + Y \cdot \operatorname{sen}(\delta)]\cos(\omega) + [-X \cdot \operatorname{sen}(\delta) + Y \cdot \cos(\delta)]\operatorname{sen}(\omega) = x' \cdot \cos(\omega) + y' \cdot \operatorname{sen}(\omega)$$

Concluindo-se que:

$$x' = X \cdot \cos(\delta) + Y \cdot \text{sen}(\delta) \quad (28)$$

$$y' = -X \cdot \text{sen}(\delta) + Y \cdot \cos(\delta) \quad (29)$$

Na Figura 4.8, tem-se uma ilustração do que acontece com os dois sistemas de orientação durante a execução de um movimento do robô.

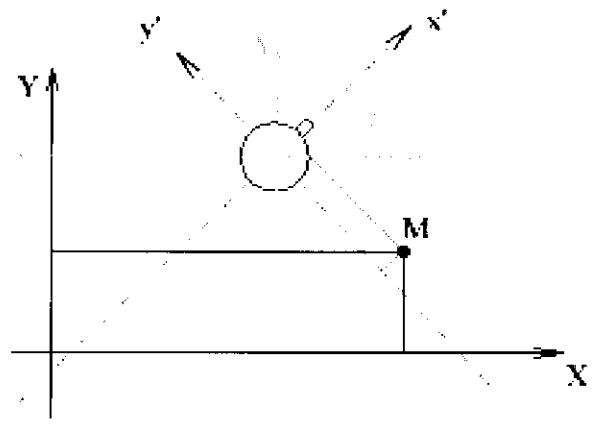


Figura 4.8 – Sistemas Cartesianos de Orientação Local e Global.

Observe que a solução tem por princípio que o ponto de intersecção do sistema $X\hat{O}Y$ seja exatamente o mesmo que o do sistema $x'\hat{O}y'$. Então, antes de tudo, devemos transladar o ponto de origem de um dos sistemas de maneira que tenhamos as origens coincidentes. Para isto, basta acrescentar a equação (28) um fator de ajuste d e para a equação (29), um fator h, que representem, respectivamente, os deslocamentos em X e os deslocamentos em Y.

$$x' = X \cdot \cos(\delta) + Y \cdot \text{sen}(\delta) + d \quad (30)$$

$$y' = -X \cdot \text{sen}(\delta) + Y \cdot \cos(\delta) + h \quad (31)$$

Para se aplicar as equações (30) e (31) empregando as variáveis do sistema Saphira e do filtro de Kalman, é imperativo que se lembre de que o valor retornado pelos sonares corresponde a distância entre o sonar e o objeto detectado.

Supondo a situação representada na Figura 4.9, a coordenada dada em relação ao eixo x' é equivalente a distância entre o sonar e o centro de movimento do robô (ponto médio entre as duas rodas de tração), que no caso do robô Pioneer 1 é de aproximadamente 10mm. Da mesma forma, o valor da coordenada no eixo y' é dada pela distância aferida pelo sonar, já devidamente ponderada pelo filtro de Kalman, o que equivale a dizer ser igual à medida $Z[0]$ para elementos a esquerda do robô e igual a $-Z[6]$ para elementos à direita.

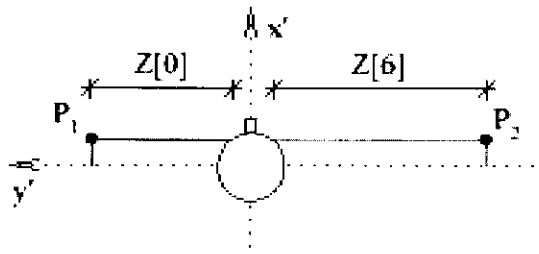


Figura 4.9 – Relação entre a detecção sensorial e LPS.

Assim, para os pontos à esquerda do robô, tais como o ponto P_1 do exemplo, as coordenadas globais serão dadas através das seguintes equações:

$$X = 10 \cdot \cos(-sfRobot.ath) + Z[0] \cdot \text{sen}(-sfRobot.ath) + sfRobot.ax \quad (32)$$

$$Y = -10 \cdot \text{sen}(-sfRobot.ath) + Z[0] \cdot \cos(-sfRobot.ath) + sfRobot.ay \quad (33)$$

e para os pontos a direita, tais como P_2, teremos:

$$X = 10 \cdot \cos(-sfRobot.ath) + (-Z[6]) \cdot \text{sen}(-sfRobot.ath) + sfRobot.ax \quad (34)$$

$$Y = -10 \cdot \text{sen}(-sfRobot.ath) + (-Z[6]) \cdot \cos(-sfRobot.ath) + sfRobot.ay \quad (35)$$

onde *sfRobot.ax*,. *sfRobot.ay* e *sfRobot.ath*, correspondem respectivamente aos termos d, h e δ dados nas equações (30) e (31).

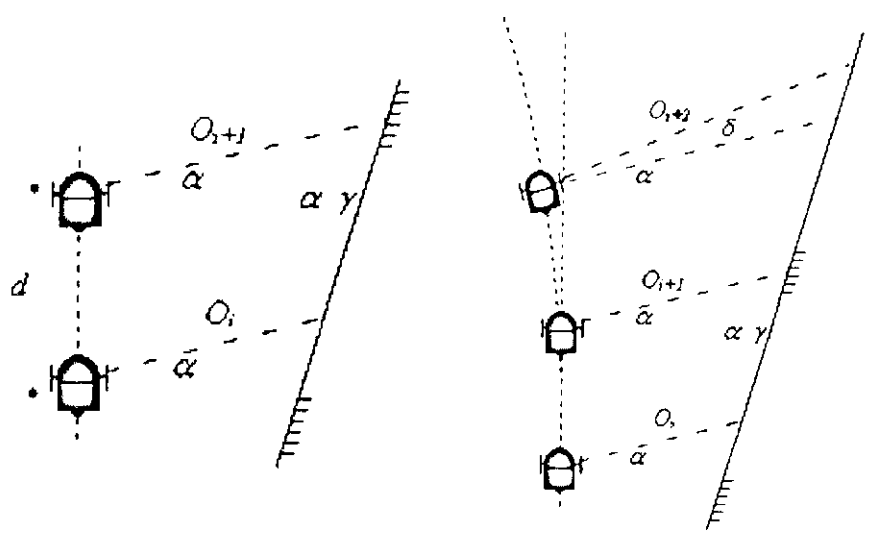
4.4.2. Odometria

O método proposto para mensurar erros odométricos durante um percurso realizado pelo robô, consiste em estimar as variações angulares formadas pela traçado da trajetória e um referencial estático do ambiente, através de leituras consecutivas executadas a partir do mesmo sensor, embora não seja empregado apenas um sensor para isto.

Chamamos de O_i a observação realizada pelo sensor S, com angulação α em relação à carenagem do robô, realizada na i-ésima iteração com o mundo, ocorrida em um instante de tempo t Após um deslocamento d, uma nova observação O_{i+1} é obtida, agora em um instante $t+1$, conforme ilustrado na Figura 4.10a.

O ângulo γ, formado pela reta paralela a trajetória do robô e o anteparo, permanece constante sempre que a trajetória descrita durante o deslocamento d for retilínea. Desta forma podemos diagnosticar possíveis distorções ou derrapagens durante o todo o percurso, simplesmente observando variações no conjunto Ω, formado pelos ângulos γ obtidos. Um

exemplo desta ocorrência é representado na Figura 4.10b.



(a) *(b)*

Figura 4.10 - Odometria.

Aplicando algumas relações trigonométricas podemos obter, a medida do ângulo γ através da seguinte equação:

$$\gamma = arctg\left(\frac{(O_{i+1} - O_i) + d.sen(\alpha)}{d.\cos(\alpha)}\right) - \alpha \qquad (36)$$

Desta forma, o sistema atualiza os valores das coordenadas globais ao longo do tempo desde a origem, segundo os valores correspondentes as componentes X e Y, cuja resultante equivalha ao deslocamento d realizado pelo robô.

4.4.3. A Rede de Kohonen

O método empregado para a localização absoluta é utilizar a rede de Kohonen para selecionar e reconhecer autonomamente elementos que serão classificados como *landmarks* do ambiente. O processo de seleção ocorre automaticamente enquanto o robô se movimenta pelo ambiente durante a fase de aprendizado da rede, enquanto o processo de reconhecimento, acontece quando o robô, agora não mais em uma fase de aprendizado da rede, passa novamente por um lugar já mapeado pela rede em uma de suas regiões (*clusters*).

A rede de Kohonen foi implementada utilizando alguns recursos de orientação a objetos fornecidos pela linguagem C++. Nela, definiu-se que cada neurônio é um objeto de uma classe que provê os métodos fundamentais deste tipo de elemento, tais como o cálculo da distância do

vetor de pesos em relação a um padrão e o de atualização dos pesos sinápticos. A definição da classe *cNeuron* é apresentada na Figura 4.11.

```
class cNeuron
{
  protected:
    TMapCoordinate Coordinate;
    TWeightVector Weight;
    TCluster BelongTo;
  public:
    double Distance;
    cNeuron();
    void MakeDistance(TInputVector VectorIn);
    void UpdateWeight(TInputVector Pattern, double LearningRatio);
    void SetCoord(int X,int Y);
    void GetCoord(int &X,int &Y);
    void SetCluster(TCluster C);
    TCluster GetCluster();
};
```

Figura 4.11 - Definição da classe que representa os neurônios.

Um recurso não recomendável, como a definição pública do atributo *Distance*, é utilizado em virtude das restrições impostas pelo tempo disponível para executar um MEF no Saphira, uma vez que praticamente todas as operações da rede envolvem a distância. Assim sendo, optou-se por não executar tantos saltos no fluxo de execução do programa além dos que já seriam inevitáveis.

A rede propriamente dita é modelada como uma outra classe, chamada *cKohonen* (Figura 4.12), constituída de uma matriz bidimensional de tamanho *MAPWIDTH* e *MAPHEIGHT* de objetos da classe cNeuron, um handle[5] para manipular um arquivo de log e outros atributos característicos do mapa de Kohonen, tais como a taxa de aprendizado e as coordenadas do neurônio vencedor.

Dentre os métodos disponibilizados por esta classe, dois algoritmos para a atualização dos pesos da rede são oferecidos, sendo a diferença básica entre eles uma questão relacionada ao critério de atualização da vizinhança do neurônio vencedor, como se explica no parágrafo que se segue a figura.

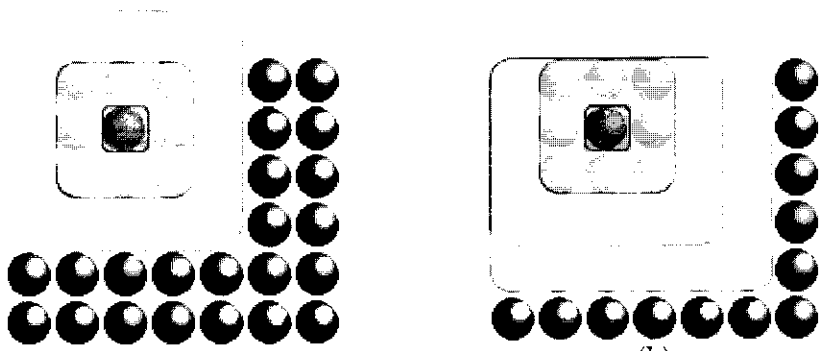
---

[5] Identificador especial designado a um arquivo que permite a um programa acessá-lo.

```
class cKohonen
{
  protected:
    FILE *hArqLog;
    unsigned char flagLearning;
    cNeuron Map[MAPWIDTH][MAPHEIGHT];
    TMapCoordinate Winner;
    double LearningRatio;
  public:
    cKohonen();
    cKohonen(double TX);
    void ProcessPattern(TInputVector Pattern);
    void UpdateMap(TInputVector Pattern);
    void UpdateMap_WaveForm(TInputVector Pattern);
    void SetCluster(TCluster C);
    void Show();
    void ShowMap();
    void StartLearing();
    void StopLearing();
    void SetLogFile(char *FileName);
    void PutIntoLogFile();
    void CloseLogFile();
};
```

Figura 4.12 - Definição da classe que modela a Rede de Kohonen..

O primeiro, codificado através de um método chamado *UpdateMap*, implementa o algoritmo padrão de atualização dos neurônios vizinhos, que consiste em estabelecer um valor máximo para o raio de atualização de modo que este não extrapole os limites do mapa enquanto se garante que uma moldura em forma de quadrado seja estabelecida. Na Figura 4.13a tem-se uma ilustração deste processo.



(a) (b)

Figura 4.13 - Atualização da vizinhança, segundo cada um dos algoritmos.

Note que para um raio de atualização igual a dois (área em amarelo) não é possível que se mantenha um quadrado, pois os limites do mapa seriam ultrapassados. Portanto, empregando

esse algoritmo apenas o neurônio vencedor (área em vermelho) e a vizinhança de raio um (área em laranja) seriam atualizados nesta situação.

Entretanto, na Figura 4.13b, pode-se observar a diferença apresentada no segundo algoritmo, que segue um princípio de propagação como se fosse uma onda, cobrindo os neurônios que distarem do vencedor independentemente de se conseguir definir uma região quadrada ou não.

É importante ressaltar que tanto o primeiro quanto o segundo algoritmo sofrem um decréscimo na taxa de atualização à medida que se distanciam do neurônio vencedor, ou seja, os vizinhos do neurônio vencedor modelarão menos intensamente as características da entrada formando, assim, pólos regionais com centro no neurônio vencedor.

Capítulo 5 - Discussão dos resultados

Neste capítulo, encontra-se uma descrição dos diversos testes realizados acompanhados de uma discussão dos resultados obtidos, bem como, a exposição de pontos positivos e negativos percebidos em relação aos diversos elementos presentes no sistema e pertinentes ao domínio do problema, havendo, ainda, uma seção dedicada a sugestões para trabalhos futuros.

Como exposto nos capítulos anteriores, a questão da localização foi abortada em três instâncias: uma usando somente odometria, outra usando redes de Kohonen para a seleção e detecção de landmarks e, por fim, uma terceira que emprega as duas técnicas anteriores através de uma abordagem híbrida. Todos os testes realizados tiveram como modelos de mundo, os ambientes apresentados nas Figuras 4.1 e 4.3.

5.1. Utilizando Odometria

Por se tratar de um método relativo, a posição atual do robô é dada através da decomposição do deslocamento espacial realizado pelo robô em suas componentes X e Y correspondente.

Os primeiros resultados nos levaram a refletir sobre o impacto que a taxa ou a cadência em que o sistema realizava esta decomposição causava sobre os valores estimados, uma vez que quanto maior o intervalo de tempo, proporcionalmente a velocidade, seria também o deslocamento.

Entretanto, notamos que muitas vezes a trajetória descrita pelo robô não era retilína e o sistema, dado o intervalo estabelecido, computava a distância linearmente causando assim discrepância entre o deslocamento registrado e o deslocamento real. Esse problema é ilustrado na Figura 5.1, note que o deslocamento aferido pelo robô (linha sinuosa) é maior que a distância linear entre os pontos P_1 e P_2, e o sistema não levar em consideração as variações na orientação da trajetória, as projeções estarão além do ponto P_2.

Figura 5.1 - Deslocamento real vs Deslocamento medido.

Observe que outro fator limitante para a nossa abordagem diz respeito, as fronteiras do anteparo, ou melhor, as bordas que determinam seu contorno das figuras detectadas. Caso estas forem irregulares ou curvas, este método pode ser comprometido uma vez que podemos realizar uma trajetória retilínea e, ainda assim, ter variações no conjunto Ω referido na seção 4.4.2, que não serão provenientes de desvios na trajetória, mas sim de desvios no anteparo.

É importante ressaltar que assumimos conscientemente as características de nossa proposta, uma vez que o percentual de elementos curvos de grande extensão em um ambiente *indoor* e estático é reduzido, restringindo-se basicamente à corredores curvos e grandes pilares.

Na Figura 5.2, seguem quatro tomadas consecutivas das telas do sistema de localização empregando odometria durante um dos testes realizados. Para este experimento, em específico, utilizou-se o simulador e o modelo de mundo apresentado na Figura 4.1.

As imagens que se vê a esquerda, são reproduções das telas do sistema Saphira (janelas windows) e do sistema odométrico (janela Dos), enquanto ao lado direito destas imagens, tem-se ampliações que destacam os valores aferidos pelo Saphira (acima) e fornecidos pelo nosso sistema (imediatamente abaixo a direita), onde cada conjunto de três imagens, representam uma tomada diferente.

Note que os valores obtidos ficaram muito próximos dos valores aferidos pelo sistema Saphira, embora, ao longo de tempo, esses valores se acumulem e as diferenças ficam maiores. Ainda assim, outros testes nos mostraram que os erros de estimativa inerentes à modelagem e a implementação podem ser desconsiderados sempre que não haja necessidade de uma maior precisão, uma vez que tratam-se de erros sistemáticos e portanto mantém uma proporção quando observados globalmente.

Entretanto, apesar de grande esforço para se buscar soluções alternativas aos problemas com odometria e mesmo diante da relativa "boa desenvoltura" do sistema odométrico apresentado, sempre que houver a necessidade de informações precisas quer seja tão somente para a localização, quer seja para uma tarefa mais complexa como mapeamento, desaconselha-se o uso único e isolado do método odométrico.

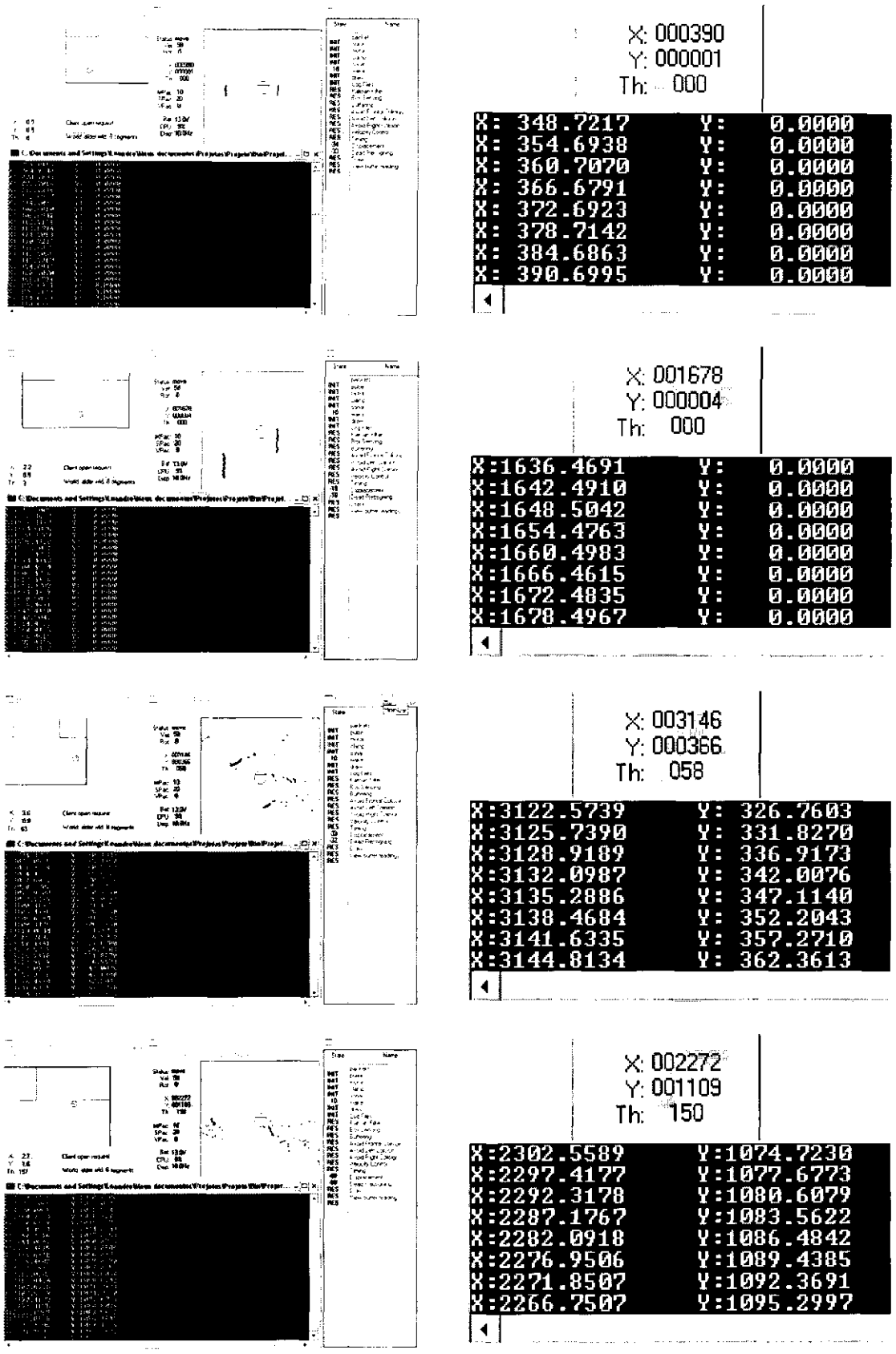


Figura 5.2 – Saída gerada pelo sistema odométrico (4 tomadas consecutivas).

Em um outro teste, forçamos uma colisão para medir o desempenho do sistema quanto a não computar as derrapagens. Os resultados se mostraram insatisfatórios para todas as tentativas. Na Figura 5.2, tem-se uma ilustração de um dos testes desta modalidade, onde destaca-se que a posição do robô permanece a mesma (circulo em destaque) enquanto duas tomadas posteriores e subseqüentes, mostram que o sistema Saphira e o nosso sistema, computaram o movimento das rodas.

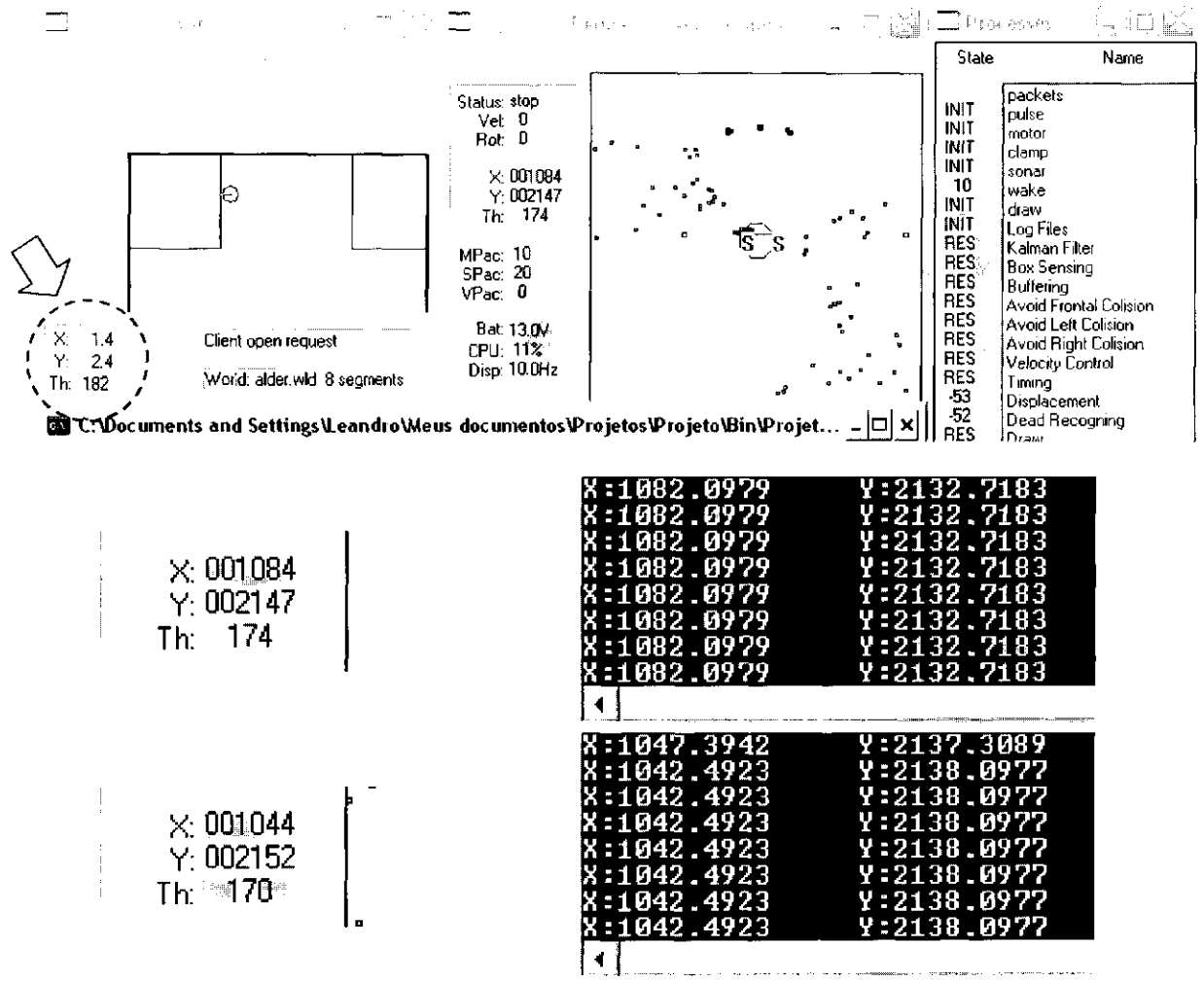


Figura 5.3 – Falha do sistema odométrico durante colisão.

A questão do choque frontal causa um bloqueio do robô e ao mesmo tempo permitem que as rodas executem pequenos movimentos, conseqüentes de derrapagens. O sensor deveria acusar que não houve mudança na distância aferida entre o robô e o anteparo, entretanto, em se tratando de um ambiente dinâmico, devemos considerar a possibilidade de que um objeto se move diante do robô em igual velocidade.

Durante alguns testes detectou-se uma falha de odometria quando o robô executava

movimentos para trás, sendo computado pelo sistema como um deslocamento positivo. Após um estudo, solucionou-se o problema através de uma análise das coordenadas dos pontos de referência. Implementada as correções, os testes passaram a empregar o novo sistema de navegação, independentemente do modelo de mundo utilizado, uma vez que este proporciona situações onde o robô realiza manobras de ré. A Figura 5.4 corresponde ao momento exato em que uma destas manobras é realizada durante um dos testes.

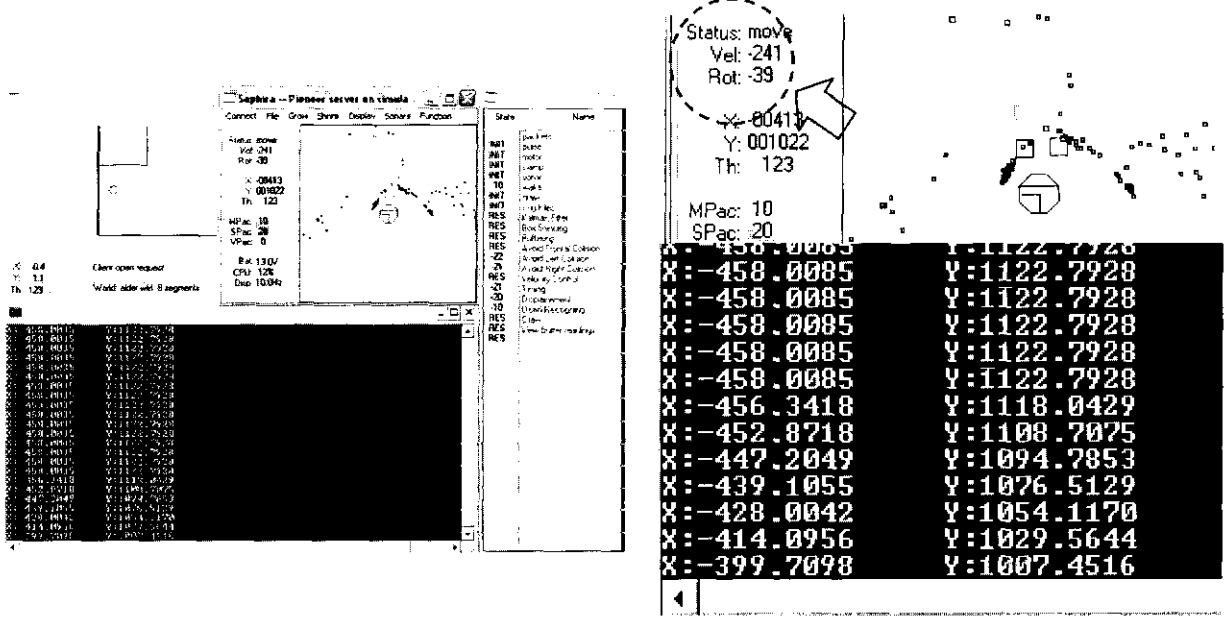


Figura 5.4 - Estimativa odométrica durante um manobra de ré.

O robô Pioneer 1 não nos possibilita acesso direto às informações provenientes dos *encoders* das rodas, conseqüentemente, optamos por trabalhar com outras informações que alternativas. O que propomos é uma solução factível mesmo com a ausência de um dispositivo capaz de mensurar os giros da roda, uma vez que os deslocamentos podem ser calculados em função das leituras dos sensores e portanto, ser aplicado em plataformas que não disponham de um mecanismo de medida da distância percorrida.

5.2. Utilizando Landmarks

Inicialmente, trabalhamos o primeiro sistema de navegação descrito, onde o robô percorria um ambiente simples, acompanhando a parede a sua direita. Sempre que esta terminasse em uma outra parede, o robô executava um giro de 90°, passando a acompanhar esta nova parede.

O vetor de entrada era formado pelas instruções fornecidas pelo sistema de navegação,

codificadas em padrões de três bits binários e concernentes aos movimentos executados pelo robô. Assim, a rede cria regiões para cada tipo de movimento realizado pelo robô, ou seja, rotação para a direita, rotação para a esquerda e movimentos à frente.

Ainda nos primeiros testes, pode-se perceber que a rede gerava a mesma saída para os conjuntos de pontos {D,G} e {B, C, E, F, H, A}, conforme exibidos na Figura 5.5. Constatamos que não havia um diferencial para estes pontos, uma vez que os pontos do primeiro conjunto correspondem à curvas à direita, e os do segundo conjunto, correspondendo a curvas à esquerda.

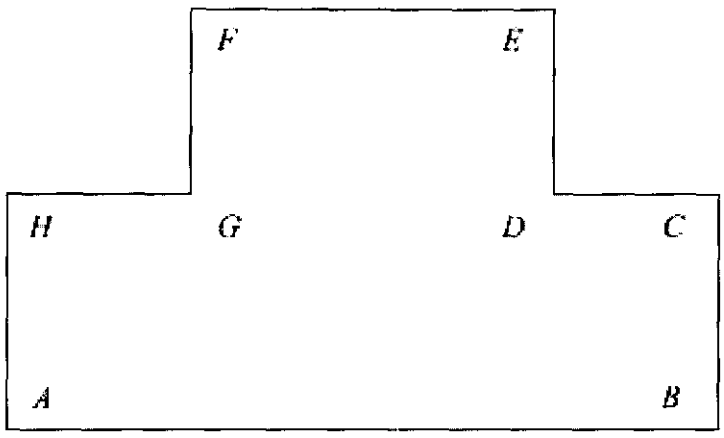


Figura 5.5 - .

Neste contexto, um diferencial a ser introduzido para estas instruções, sem a utilização dos valores odométricos ainda neste momento, é a questão do tempo. Uma diferença útil entre os pontos A e B diz a respeito de quanto tempo o robô se deslocou pra frente antes de realizar a rotação. Desta forma, agregou-se ao vetor de entrada alguns bits binários para representar uma unidade de tempo.

Para estes novos testes, o vetor foi composto pela instrução fornecida pelo sistema de navegação tal qual descrita anteriormente, acrescendo-se 7 bits para tempo. Entretanto, o intervalo de tempo que o robô precisa para realizar o trajeto A→B era maior que 7 segundos. Ao realizarmos os testes, com apresentações a rede a cada 1 segundo, as entradas variavam tanto que a rede não conseguia extrair nenhuma característica das entradas das posições correspondente ao tempo.

O vetor de entradas foi novamente modificado, passando a comportar 24 bits para representar o intervalo de tempo. Desta vez, a apresentação da rede era realizada a cada 10 segundos e também, a cada rotação, quer seja a direita ou a esquerda.

Após vários testes, percebeu-se uma certa conformidade na distribuição das regiões dentro do mapa, dada pela inicialização dos pesos da rede. Como que por hábito, sempre que se iniciava o aprendizado, os primeiros neurônios a se tornar vencedores localizavam-se na periferia do mapa. Dada as características de atualização dos neurônios vizinhos, dificilmente se atualizava mais do que o neurônio vencedor, uma vez que a na maioria das vezes a vizinhança de raio um já extrapolava os limites do mapa.

Um novo algoritmo para a atualização da vizinhança do neurônio vencedor foi proposto, nele a atualização dos neurônios vizinhos prossegue até atingir um raio pré-estabelecido, independentemente dos limites do mapa terem sido atingidos.

Com estas novas configurações, obtivemos uma melhora no aprendizado da rede, embora, tenham sido necessários vários testes para se estabelecer qual seria o melhor raio para que uma região não invadisse uma outra que tenha se formado próximo a ela (equivalente aos neurônios marcados entre o cluster azul e o laranja da Figura 3.18).

Ainda assim, tentou-se uma outra abordagem para a diferenciação dos landmarks comentados anteriormente, esta baseada em informações históricas. O princípio diferenciador é exatamente os comandos pregressos a cada nova rotação, ou seja, supondo o ponto F, dois comando de rotação a esquerda, seguido de um comando de rotação à direita precedem a instrução no ponto E.

Os testes mostraram-se positivos, embora tenha sido uma das melhores respostas da rede, considero esta uma solução particular e tão discutível quanto o uso da temporização, pois sempre haverá a necessidade de se estimar uma quantidade ideal de bits para representá-los, e ainda assim, nada se pode garantir de que não aconteçam semelhanças dentro do ambiente.

Finalmente, adotou-se uma solução que mesclava os dois conceitos, ou melhor, a composição do vetor de entrada agregava as informações sobre o histórico dos comandos, um temporizador, além da própria instrução de rotação.

5.3. Utilizando juntos Odometria e Landmarks

Apesar de não se ter obtido um desempenho satisfatório com a rede de Kohonen, e com isso entende-se que o robô foi capaz de estabelecer sua posição com uma margem de erro aceitável, os dados apresentados no trabalho de [NEHMZOW,91] ainda eram intrigantes.

Nehmzow treinou seu mapa tão somente com as informações provenientes dos *encoders* acoplados às rodas de seu robô. Mais uma vez, tentou-se solucionar o problema de diferenciação dos landmarks, entretanto, agora através de um algoritmo híbrido, o vetor de entrada para a rede era composto da instrução e dos valores aferidos pelo sistema odométrico.

Após uma normalização dos dados odométricos, um vetor contendo 5 elementos binários representando os diferentes graus de rotação, juntamente com dois elementos com valores reais entre 0 e 1, era apresentado para a rede.

Com a realização dos testes, apesar das expectativas favoráveis, esta abordagem não resultou em uma melhora significativa do comportamento da rede em relação aos resultados já obtidos com o histórico de movimentos.

5.4. Trabalhos futuros

Como trabalhos futuros, sugere-se:

A aplicação de técnicas de fusão de sensores, para melhorar a interpretação dos dados do sonar, incluindo um estudo mais aprofundado do Filtro de Kalman, sugerido a leitura dos trabalhos de [WIJK,98b] e [BROOKS,98], uma vez que todo o processo iterativo entre o robô e o ambiente ocorre por intermédio dos sensores e os sistemas de navegação e localização dependem diretamente dos valores fornecidos. Além do estudo de tipos diferentes de sensores, tais como scanners laser, acelerômetros e GPS.

Construção de sistema de navegação mais robusto, que permita a inserção de objetivos, exploração inteligente do ambiente e suporte a mapeamento, afim de que se proporcionem novos elementos a serem mesclados em técnicas híbridas para localização, tais como *map matching*.

Realização de experimentos em plataformas diferentes, que também apresentem sonares e *encoders* nas rodas, podendo incluir a exploração de técnicas de localização envolvendo inúmeros indivíduos (localização coletiva).

Estudos das relações de custo-benefício para a implementação dos métodos aqui descritos aplicados em ambientes dinâmicos.

Referências bibliográficas

[ARKIN,99] R. C. ARKIN, *"Behavior-Based Robotics - Intelligent Robots and Autonomous Agents"*, MIT Press, ISBN: 0262011654.

[ASIMOV,94] I. ASIMOV, *"I, Robot"*, July 1994 (Reprint edition), Bantam Books, ISBN: 0553294385.

[BETKE,97] M. BETKE and L. GURVITS, "Mobile Robots Localization using Landmarks" IN: *IEEE Transactions on Robotics and Automation*, 13:2, pp. 251-263, 1997

[BORENSTEIN et al,96] J. BORENSTEIN, H. R. EVERETT and L. FENG, *"Where am I? - Systems and Methods for Mobile Robot Positioning"*, versão eletrônica, URL: http://www.eng.yale.edu/ee-labs/morse/other/intro.html, visitada em Novembro 2002.

[BORESTEIN,98] J. BORESTEIN, "Experimental Results from Internal Odometry Error Correction with the OmniMate Mobile Robot", IN: *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 6, December 1998, 963-969

[BRITÂNICA] URL: http://www.britannica.com/eb/article?eu=115621&tocid=0, visitada em Abril 2001.

[BROOKS,98] R. R. BROOKS and S. S. IYENGAR, "Multi-Sensor Fusion-Fundamentals and Applicaions with Software", Prentice Hall, 1998.

[BULUSU,00] N. BULUSU, J. HEIDEMANN and D. ESTRIN, *"GPS-less Low Cost Outdoor Localization For Very Small Devices"*, Technical report 00-729, Computer Science Department, University of Southern California, Apr. 2000.

[BURGARD,96] W. BURGARD, D. FOX, D. HENNIG and T. SCHMIDT, "Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids", IN: *Proc. of the Fourteenth AAAI - National Conference on Artificial Intelligence, 1996.*

[CAPEK,21] K. CAPEK, *"R.U.R. and the Insect Play"*, 1a. Edição, 1921 Oxford Univ Press; ISBN: 0192810103.

[DUCKETT,96] T. DUCKETT and U. NEHMZOW, *"A Robust Perception-based Localisation Method for a Mobile Robot"*, Department of Computer Science, Technical Report Series, Report UMCS-96-11-1.

[DUCKETT,99] T. DUCKETT and U. NEHMZOW, "Knowing Your Place in Real World Environments" IN: *Proc. 3th European Worshop on Advanced Mobile Robots - Eurobot'99*, 1999, IEEE Computer Society.

[DUCKETT,00] T. DUCKETT and U. NEHMZOW, "Performance Comparison for Landmark Recognition Systems for Navigating Mobile Robots", IN: *Proc. AAAI - American Association for Artificial Intelligence*, 2000.

[GREINER,94] R. GREINER and R. ISUKAPALLI, "Learning to select useful landmarks." IN: *Proc. of the Twelfth National Conference on Artificial Intelligence*, 1994.

[KLEEMAN,92] L. KLEEMAN, "Optimal estimation of position and heading for mobile robots using ultrasonic beacons and dead-reckoning." IN: *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 2582--2587, Nice, France, 1992.

[KOHONEN,84] T. KOHONEN, *"Self-organization and Associative Memory"*, Springer Series of Information Science, vol. 8, Springer-Verlag: New York, 1984.

[KONOLIGE,97] K. KONOLIGE, K. MYERS, E. RUSPINI, and A. SAFFIOTTI, "The Saphira architecture: A design for autonomy", IN: *Journal of Experimental and Theoretical Artificial Intelligence*, 9(1):215--235, 1997

[MAYBECK,79] P. S. MAYBEC, *"Stochastic Models, Estimation and Control"*, Academic Press, London, Vol1, 1979, ISBN: 0124807011.

[NEHMZOW,91] U. NEHMZOW, T. SMITHERS and J. HALLAM, "Location Recognition in a Mobile Robot Using Self-Organising Feature Maps", IN: *Information Processing in Autonomous Mobile Robots*, Springer Verlag 1991

[NEHMZOW,92] U. NEHMZOW and T. SMITHERS, *"Using Motor Actions for Location Recognition"*, F. Varela and P. Bourgine (eds.),Toward a Practice of Autonomous Systems, MIT Press 1992.

[OWEN,96] C. OWEN and U. NEHMZOW, "Map interpretation in dynamic enviroments" IN: *Proc. of 5th Intenational workshop on Advanced Motion Control*, IEEE Press, ISBN 0-7803-4484-7, 1998.

[OWEN,98] C. OWEN and U. NEHMZOW, "Map interpretation in dynamic enviroments" IN: *Proc. of 5th Intenational workshop on Advanced Motion Control*, IEEE Press, ISBN 0-7803-4484-7, 1998.

[RIBEIRO,01] C. RIBEIRO, A. REALI COSTA and R. ROMERO, *"Robôs Móveis Inteligentes: Princípios e Técnicas"*, IN: Jornada de Atualização em Inteligência Artificial JAIA-2001, XXI Congresso da SBC (Mini curso), 2001.

[RVL] URL: http://rvl.www.ecn.purdue.edu/RVL/mobile-robot-nav/mobile-robot-nav.html, visitada em Abril de 2001.

[SAPHIRA,97] *"Saphira - Software Manual"*, ActivMedia Inc., October, 1997.

[SIMMON et al, 95] R. SIMMON and S. KOENIG, "Probabilistic Robot Navigation in Partially Observable Environments", Proc. Of the Int. Joint Conf. On Artificial Intelligence (IJCAI-95), Montreal, CA, pp. 1080-87, 1995.

[SIMON,98] URL: http://www.innovatia.com/software/papers/kalman.htm, Simon, D., "Kalman Filtering", visitada em Janeiro de 2003.

[STANFORD] URL: http://arl.stanford.edu/~rover/, visitada em Abril 2001.

[THRUN,98] S. THRUN, *"Bayesian Landmark Learning for Mobile Robot Localization"*, Tecnical Report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.

[THRUN,00] S. THRUN, *"Probabilistic Algorithmis in Robots"*, Tecnical Report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.

[ULRICH,00] I. ULRICH and I. NOURBAKHSH, "Appearance-Based Place Recognition for Topological Localization", IN: *IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 2000, pp. 1023-1029.

[WEBSTER] URL: http://work.ucsd.edu:5141/cgi-bin/http_webster, visitada em Novembro de 2002.

[WELCH,01] G. WELCH and G. BISHOP, *"An introduction to the Kalman Filter"*, IN: SIGGRAPH 2001, ACM Association for Computer Machinery, Los Angeles, Agosto 2001.

[WIJK,98b] O. WIJK, *"Triangulation Based Fusion of Sonar Data with Application in Mobile Robot Mapping and Localization"*, Department of Signals, Sensors and Systems, Royal Institute of Technology (KTH), SE-100 44 Stockholm, Sweden, 1998.

[WRIGHT,96] A. WRIGHT, R. SARGENT and C. WITTY, *"Cognachrome Vision System User's Guide"*, 2nd Edition, Newton Research Labs, Junho , 1996.

Apêndice A

Arquivo de parâmetros PARAMS.CFG

```
[Debugger]
console = On                    //Opcional, podendo ser valorado em On ou Off
draw_sensing_areas = On         //Opcional, podendo ser valorado em On ou Off
draw_detected_points = On       //Opcional, podendo ser valorado em On ou Off

[Connection]
connect_to = simulator          //Necessário, podendo ser valorado em
                                //simulator ou robot

[Input Files]
world = ..\\World\\blocol.wld   //Necessário ser connect_to for igual a
                                //simulator. Define o caminho e o nome do
                                //arquivo que descreve o mundo
robot = ..\\Params\\PSOS41m.p   //Necessário ser connect_to for igual a
                                //simulator. Define o caminho e o nome do
                                //arquivo com os parâmetros a respeito do modelo
                                //do robô

[Output Files]
readings = ..\\Debug\\readings.log //Opcional. Quando presente, define o caminho e o
                                //nome do arquivo de log onde serão gravadas as
                                //leituras dos sonares
filter = ..\\Debug\\kalman.log  //Opcional. Quando presente, define o caminho e o
                                //nome do arquivo de log onde serão gravadas as
                                //leituras dos sonares e os valores correspondentes
                                //de saída do filtro de Kalman
kohonen = .\\Debug\\SOFM.log    //Opcional. Quando presente, define o caminho e o
                                //nome do arquivo de log onde serão as informa-
                                //-ções relativas a rede de Kohonen.

[Navigation system]
how_close_to = 300              //Necessário, sempre valorado com um numero inteiro
window_width = 500              //Necessário, sempre valorado com um numero inteiro
window_height = 800             //Necessário, sempre valorado com um numero inteiro

[Kalman filter]
Q = 0.009                       //Necessário, sempre valorado com um número real
R = 0.6                         //Necessário, sempre valorado com um número real
```

Apêndice B

Arquivo BLOCO1.WLD, definição do ambiente para experimentos com o simulador.

```
;;; Corredor do Labic, Intermídia, LCad até o LCR
;;;
;; <------------------ 7,0 ------------------->
;;       Interm.      Labic
;;     +-======-+ +-=======----------------------+              |       |       |
;;     |  2,0   | |  |        3,30               | Labic      0,9     |       |
;;     |        ++ |1,20                          |              |       |       |
;; ---+         +--+  |                      |  +-+                    |       |
;;               0,6                        0,3 |                      |       |
;;                                              |  +-+                 2,85    |
;;                                              |                 |      |     3,85
;;                                              |                 |      |      |
;;                                    1,2       | CompUSP   1,65  |      |      |
;;                        0,9     +------+  1,2  |                 |      |      |
;; ---------+         +----------+      +--------+                 |      |      |
;;  1,70    |         |  |                                         |      |      |
;;          |  2,0    | 1,0                                       |      |      |
;;          +-=====-+  |                                          |      |      |
;;              Lcad
width  7000
height 3850
;; Perimetro
0000 0000 7000 0000
7000 0000 7000 3850
7000 3850 0000 3850
0000 3850 0000 0000
;; Paredes Internas
0000 1000 1700 1000
1700 1000 1700 0000
1700 0000 3700 0000 ;; Porta do LCAD
3700 0000 3700 1000
3700 1000 4600 1000
4600 1000 4600 1200
4600 1200 5800 1200 ;; Pillar
5800 1200 5800 1000
5800 1000 7000 1000
7000 1000 7000 2650
7000 2650 6700 2650
6700 2650 6700 2950 ;; Pilar
6700 2950 7000 2950
7000 2950 7000 3850
7000 3850 3700 3850
3700 3850 3700 2550 ;; Parede entre o LABIC
3700 2550 3100 2550 ;; e o Intermídia
3100 2550 3100 2700
3100 2700 3150 2700
3150 2700 3150 3850
3150 3850 1150 3850 ;; Intermídia
1150 3850 1150 2550
1150 2550 0000 2550
;; Posição do Robô
position 300 1300 0
```