NAVEGAÇÃO DE ROBÔS MÓVEIS

Dissertação submetida ao
departamento de engenharia electrotécnica
da Universidade de Coimbra
satisfazendo parcialmente os requisitos
para a obtenção do grau de
mestre

Por
Paulo Jorge Carvalho Menezes
Licenciado em Engenharia Electrotécnica
Setembro 1999

Dissertação realizada sob a supervisão de
Aníbal Traça de Almeida
Professor Catedrático
e
Jorge Manuel Miranda Dias
Professor Auxiliar
do
Departamento de Engenharia Electrotécnica
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Agradecimentos

Ao Professor Aníbal Traça de Almeida, pelo apoio tanto científico como moral ao longo do desenvolvimento deste trabalho.

Ao Professor Jorge Dias, pelo apoio científico, pela paciência e pelas dicas oportunas.

Aos restantes colegas do DEE, que me apoiaram na resolução de problemas que foram surgindo ao longo deste trabalho.

Aos colegas e amigos Paulo Ventura, Rui Bernardes e Fernando Moita, pelo incentivo.

À Ana, pelo apoio e paciência.

Aos amigos, Seco, Vasco, Pedro Alexandre, J.P., Garcia, Inês, Paixão, Susana, Guida, João Luís, Ester e muitos outros, por me lembrarem que a vida não é só trabalho.

A meus pais, por tudo.

Ao meu irmão, por ter evitado que eu fosse filho único.

A Leslie Lamport, por ter criado o LaTeX.

A Ken Thompson, por ter aproveitado os tempos livres para criar o UNIX.

Ao Pilecas, por ser o meu cavalo de estimação.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

1.1 Da Ficção à Realidade

KAREL Čapek, foi quem utilizou pela primeira vez a palavra *robô* ao escrever a peça teatral R.U.R. (Rossum's Universal Robots). Esta peça, que se estreou em Janeiro de 1921 na cidade de Praga, tinha como personagens dispositivos mecânicos com aparência humana que, por serem desprovidos de qualquer tipo de sensibilidade, podiam apenas executar operações rotineiras de forma automática.

Karel Čapek, para designar estes dispositivos, escolheu inicialmente "*Laborí*". No entanto, como o termo não lhe agradava por lhe parecer demasiado intelectual, acabou por adoptar a palavra Checa "*robota*", que significa trabalhador "à força".

O termo "robótica", que refere o estudo e uso de robôs, foi introduzido pelo cientista e escritor Isaac Asimov.

Os robôs são, ainda hoje, muitas vezes associados às personagens de Čapek, de Asimov ou de outros escritores que os tornaram nos "super-homens-de-metal" que nos habituámos a ver nas telas de cinema. No entanto, a robótica está ainda longe de conseguir reproduzir algo que se assemelhe a alguma dessas personagens e as razões prendem-se com o facto de existirem muitos problemas para os quais ainda não foram encontradas soluções satisfatórias. Vejamos então alguns exemplos desses problemas:

- *A fonte de energia* - Um sistema, que se quer que seja autónomo, deve possuir uma fonte de energia que suporte o seu funcionamento durante períodos de tempo consideráveis, de forma a que as tarefas a realizar não sejam constantemente interrompidas devido as falhas de energia. A fonte de energia é normalmente composta por conjuntos de baterias e, para resolver o problema da autonomia, seria necessário a utilização de baterias que fossem simultaneamente leves, de pequenas dimensões e que possuíssem uma grande capacidade de armazenamento. Ora, como se sabe este tipo de baterias ainda não existe, introduzindo logo a partida um problema que limita a utilização dos robôs móveis.

- Recepção de ordens e interpretação das mesmas (interface homem/máquina) - O robô deveria ser capaz de receber ordens em linguagem natural, tanto escrita como falada por forma a facilitar a sua utilização. Deveria também ter capacidade para interpretar uma ordem e desta determinar o conjunto de acções que permitem a sua execução. Dada a seguinte ordem, *Vai para o Laboratório de Visão*, seria necessário em primeiro lugar associar o nome Laboratório de Visão com uma localização no espaço e de seguida determinar quais as acções a realizar para atingir essa posição.

- *Interacção com o mundo real* - Capacidade de detectar alterações no ambiente de trabalho é uma capacidade fundamental para permitir o funcionamento num ambiente dinâmico. A criação de ambientes estáticos destinados à operação de robôs apresenta desvantagens tanto a nível da flexibilidade de produção como em termos económicos. Para permitir o funcionamento em ambientes dinâmicos, torna-se necessária a introdução de sistemas sensoriais adequados, de forma a que estes permitam a obtenção de informações sobre o ambiente onde o robô opera. Dando como exemplo um ambiente fabril, onde a cada momento poderá surgir um novo obstáculo no caminho do robô. A detecção de novos obstáculos é importante pois permite que estes sejam evitados automaticamente e, desta forma, completar com sucesso a missão em curso seja ela a de inspecção, transporte ou limpeza. Os sistemas de visção artificial poderão vir a desempenhar um papel muito importante na resolução destas dificuldades mas, apesar dos avanços que têm sido feitos nesta área e do elevado desempenho de alguns sistemas já construídos, há ainda um longo caminho a percorrer para se conseguir desenvolver sistemas com um desempenho global semelhante à visão humana.

- Flexibilidade na execução de tarefas - O comportamento do robô terá de ser ajustado para permitir a ultrapassagem de novas situações. Por exemplo, dada a indicação da presença de um obstáculo no caminho escolhido para atingir a posição objectivo, uma nova trajectória deverá ser escolhida.

É já comum encontrar em aplicações industriais sistemas semi-autónomos os quais

são normalmente utilizados em operações de transporte de peças ou componentes em linhas de montagem. Os mais comuns são os conhecidos por *AGV (Automated Guided Vehicle)* que consistem em plataformas móveis que se deslocam seguindo um cabo metálico metálico, uma faixa reflectora pintada no pavimento ou outro tipo de guia, sendo o primeiro o mais comum. As trajectórias são estáticas e uma vez definidas basta "desenhá-las", instalando o fio metálico para que estas possam ser executadas. O seu seguimento é efectuado usando sensores que detectam o afastamento do robô em relação ao fio e com essa informação é feita a correcção dos movimentos.

É comum encontrar outros sistemas integrados de forma a tornar os AGV mais versáteis. Um exemplo comum é a colocação de códigos de barras em determinadas posições da trajectória que juntamente com um sensor de infravermelhos colocado no robô, permite determinar em qual das etapas da trajectória este se encontra em cada momento. Isto permite a rentabilização da utilização destes sistemas pois permite a construção de tarefas compostas, como por exemplo a distribuição de componentes diversos ao longo de uma linha de montagem.

Os AGV's apresentam no entanto grandes limitações devido ao facto de serem filoguiados; uma das desvantagens é a incapacidade de se desviar de quaisquer obstáculos que possam existir no seu caminho, nestes casos a única solução é a de parar, dar um qualquer sinal de alarme e esperar que o obstáculo seja retirado do seu caminho. Uma outra desvantagem é a necessidade de instalar novos fios quando se pretende adicionar ou modificar uma trajectória.

Perante estas limitações torna-se evidente a necessidade de criar sistemas com um maior grau de autonomia. O aumento do grau de autonomia passa pelo desenvolvimento de novas técnicas, nomeadamente para a geração automática de trajectórias e para a detecção e caracterização de obstáculos.

1.2 Navegação Autónoma

Quando se fala em robôs autónomos surge-nos imediatamente a ideia de um sistema que executa as tarefas sozinho, sem intervenção humana.

Para conseguir um tal sistema é necessário antes de mais introduzir-lhe a capacidade de navegação autónoma, sobretudo se falamos de robôs móveis. A execução de uma tarefa tão simples como ir de um ponto a outro no espaço, passa por um conjunto de fases distintas. Antes de mais é necessário um conhecimento da localização dos pontos de partida e de chegada para se poder estabelecer uma relação entre eles, o caminho. O estabelecimento da relação caminho entre dois pontos depende do conhecimento que se tem do ambiente. Assim, a definição de um caminho pode ser feita com base em primitivas geométricas (segmentos de recta, arcos, etc.) que são unidas através de pontos de passagem bem definidos, desde que o ambiente seja também descrito através de primitivas geométricas num espaço cartesiano. Por outro lado, se

a descrição do ambiente existente com base em informação topológica, o planeamento do caminho implica a definição de um conjunto de acções de navegação e percepção. Esta última abordagem é a normalmente utilizada por nós próprios quando descrevemos o caminho entre quaisquer dois sítios sendo as descrições geométricas utilizadas apenas em alguns casos particulares.

A primeira abordagem produz, de uma forma geral, caminhos muito mais exactos do que a segunda, requer no entanto uma descrição mais exaustiva do ambiente e não é particularmente ajustada para ambientes não estruturados dado que estes estão normalmente em constante mutação o que torna quase impossível manter uma descrição pormenorizada dos mesmos.

Por outro lado, na abordagem topológica os caminhos planeados são normalmente descritos através de relações entre acções a executar e características do ambiente, como sejam "**atravessar** a *porta*", "**ir até** ao *fim do corredor*", "**entrar** na *segunda porta à esquerda*". Para possibilitar a execução de um caminho descrito desta forma é necessário dispor de sistemas sensoriais que permitam a detecção de todos estes elementos, bem como a identificação de obstáculos novos para permitir a sua ultrapassagem.

Ambas as aproximações ao problema da navegação apresentam dificuldades para as quais ainda não existem soluções óptimas. Por esta razão é muitas vezes necessário optar por um compromisso entre as duas ou pela a adaptação do ambiente de forma a facilitar a utilização de uma delas.

Criar um sistema com capacidade para navegar de forma autónoma num ambiente conhecido ou desconhecido é só por si um desafio considerável devido ao conjunto de problemas a resolver. Colocando o problema de uma forma simples podemos dizer que o que pretendemos é, dado um ambiente conhecido, que qualquer que seja o ponto de partida onde se encontre o robô este deve ser capaz de atingir um ponto destino que lhe seja dado.

Um robô móvel capaz de resolver este problema simples necessita de:

1. **[Mapas]** Dispôr de uma representação do ambiente à qual chamamos habitualmente mapa que poderá ser fornecido "a priori" ou construído pelo próprio robô após ter explorado o ambiente inicialmente desconhecido. Diversos autores se têm debruçado sobre o problema da contrução automática de mapas geométricos ou utilizando estruturas do tipo grelha [9],[41].

2. **[Planeamento]** Utilizando mapas, estabelecer uma trajectória entre o ponto de partida e o ponto de chegada. O planeamento de uma trajectória num ambiente é abordado muitas vezes através de métodos de pesquisa global[8] o que o torna esta operação muito morosa para ambientes "complicados". Actualmente muitos autores utilizam métodos de amostragem aleatória [18] para reduzir o tempo de computação necessário à obtenção da trajectória. O planeamento da

trajectória poderá ainda produzir não apenas um caminho mas uma sequência de pares acção-percepção do género "ir em frente até detectar uma parede" ou "seguir o corredor até encontrar a segunda porta à direita".

3. [**Sistema de Controlo**] Seguir a trajectória planeada convertendo o caminho em comandos de movimento e verificar a sua execução. Deverá ainda evitar a colisão com quaisquer obstáculos inesperados e sempre que possível contorná-los de forma a regressar à execução da trajectória. Sempre que se verificar que não é possível transpor um obstáculo é necessário invocar novamente o planeamento para obter uma trajectória alternativa que conduza ao objectivo. É comum controlar a execução da trajectória evitando a colisão com obstáculos inesperados através da utilização de campos de potencial artificiais [16]

4. [**Localização**] Embora muitos dos robôs utilizem sistemas odométricos para estimar a posição do robô em cada instante, estes, devido a factores como o escorregamento das rodas ou variação do diâmetro das mesmas, vão acumulando erros ao longo da execução de uma trajectória. Os erros de posicionamento vão fazer com que o robô termine a execução num ponto do espaço diferente do ponto objectivo ou aborte a execução devido a uma parede de um corredor se transformar num obstáculo intransponível devido a um erro de orientação acumulado. Por estas razões é frequente utilizar sistemas de localização complementares destinados a compensar os erros de odometria ou dos sistemas inerciais. Os sistemas de localização baseiam-se na comparação da informação sensorial obtida com a que deveria ter sido obtida de acordo com o mapa e a posição estimada [12].

5. [**Percepção**] Sistemas sensoriais que permitam identificar características no ambiente que rodeia o robô e que uma vez detectadas as suas posições relativas ao robô possam ser utilizadas pelo sistema de localização [40], ou obstáculos cuja posição relativa deve ser enviada ao sistema de controlo para evitar a colisão. São os sistemas sensoriais que permitem ao robô a sua adaptação a alterações no ambiente conhecido sendo por isso peças fundamentais para obter a desejada autonomia.

O trabalho desenvolvido, e aqui apresentado, aborda os seguintes aspectos: a utilização de um método capaz de efectuar a detecção de obstáculos através de uma estrutura semelhante à grelha de ocupação, o controlo reactivo aplicando campos de potencial sobre esta estrutura o que possibilita a navegação local (nas imediações do objectivo) e a suavização de trajectórias pré-planeadas e a adaptação das mesmas à presença de novos obstáculos sem necessidade de efectuar o replaneamento, utilizando para isso o princípio das bandas elásticas.

1.3 Resumo da Dissertação

Nesta dissertação é abordado o problema da "navegação de uma plataforma móvel em presença de obstáculos utilizando sensores de ultra-sons". Como iremos ver ao longo do texto este conjunto de problemas divide-se em dois, que são a detecção e representação dos obstáculos e a navegação ultrapassando os obstáculos detectados.

No capítulo 2.1, é feita a descrição do sistema experimental, começando por descrever a plataforma Robuter na secção 2.1, a forma de locomoção da mesma (secção 2.1.1), a forma com é estimada a posição através de odometria (secção 2.1.2), o sistema operativo de tempo-real (secção 2.1.3) e os sensores de ultra-sons utilizados (secção 2.1.6). É de seguida feita uma descrição dos sistema de desenvolvimento e de um conjunto de ferramentas desenvolvidas para permitir a mais fácil utilização desta plataforma e desenvolvimento de aplicações na secção 2.2.

No capítulo 3 são descritos os métodos utilizados para a construção de mapas e detecção de obstáculos.

No capítulo 4 é abordado o problema da navegação bem como alguns métodos testados, nomeadamente os métodos baseados em campos de potencial artificial.

O capítulo 5 começa com uma análise dos resultados comparando os diversos métodos utilizados. De seguida é feita uma proposta de alteração à plataforma Robuter para melhorar o seu desempenho e aumentar a sua capacidade de integrar sensores. Sendo as actividades a desenvolver no futuro orientadas para robôs com pernas, são apresentados ainda: um robô móvel com pernas que está em fase de desenvolvimento, algum do trabalho já desenvolvido nesta área e por último o trabalho que se espera vir a desenvolver no futuro.

Capítulo 2

Sistema de Exploração

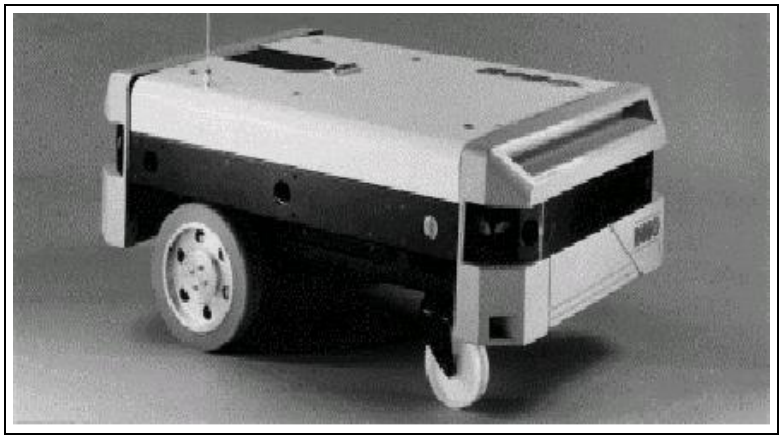

Figura 2.1: A plataforma Robuter

2.1 A plataforma Robuter

O TRABALHO aqui apresentado, foi desenvolvido sobre uma plataforma móvel Robuter I, de base rectangular[1] (ver figura 2.1), desenvolvida pela empresa francesa RobotSoft.

A locomoção desta plataforma baseia-se na utilização de duas rodas motrizes, comandadas de forma independente, e duas rodas livres. Isto permite não só controlar a velocidade da plataforma mas também a direcção dos movimentos, através da aplicação de velocidades diferentes a cada uma das rodas. A obtenção de movimentos rectilíneos ou circulares depende da aplicação de velocidades iguais ou diferentes às rodas e da geometria do próprio veículo, que neste caso é rectangular (não-holonómico).

O controlo da posição e velocidade da plataforma utiliza um sistema de estimação odométrico que por sua vez utiliza como entradas dois codificadores ópticos incrementais ligados a cada uma das rodas motrizes.

A parte sensorial é composta por 24 sensores de ultra-sons, os quais se encontram distribuídos em torno da plataforma e por sensores de colisão ("on-off") e colocados como "pára-choques" na frente e na traseira da mesma.

Existe ainda um sistema de localização baseado na detecção de 3 faróis de infravermelhos [37] mas que, por requerer a paragem da plataforma durante as medições, não se mostrou de grande utilidade para este trabalho.

[1]Existe também uma outra versão da plataforma Robuter, cuja base é circular. A vantagem da base circular é a capacidade de manobrar com maior facilidade, devido à sua geometria e ao facto das rodas motrizes estarem localizadas sobre o diâmetro do círculo da base.

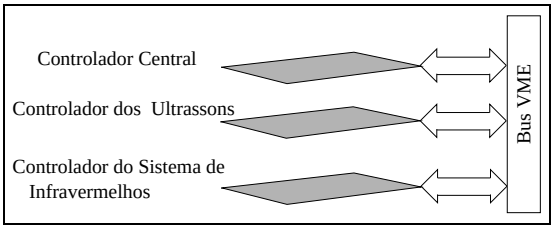

Figura 2.2: Controladores integrados na plataforma móvel

O controlo do robô é baseado num sistema computacional baseado no "bus" VME composto por três unidades controladoras. Um dos controladores efectua o controlo dos motores, da odometria e dos sensores de colisão, sendo os restantes controladores, destinados ao controlo dos sensores de ultra-sons e do sistema de localização absoluta, respectivamente. Todos os controladores utilizam o sistema operativo de tempo-real designado por Albatros [35].

Vamos nas secções seguintes analisar cada um destes sistemas para melhor compreender o seu funcionamento e limitações.

2.1.1 Modo de Locomoção

A locomoção da plataforma Robuter assenta na utilização de duas rodas fixas que estando ligadas a dois motores podem ser controladas de forma independente, e por duas rodas do tipo castor. O controlo dos movimentos é então conseguido definindo as velocidades de cada uma das rodas fixas.

O software de controlo do robô permite-nos definir em cada instante a velocidade linear $\mathbf{v_l}$ e velocidade angular ω a aplicar ao robô.

Como se pode ver na figura 2.3 a velocidade linear $\mathbf{v_l}$ corresponde à velocidade tangencial do ponto P situado a meio do eixo que une as duas rodas motrizes e a velocidade angular corresponde à velocidade angular de qualquer ponto do robô. Utilizando o conceito de *centro de rotação instântaneo*, que permite descrever o movimento de um corpo rígido em cada instante como uma rotação pura em torno de um ponto no espaço[2], obtemos da relação entre a velocidade angular e a velocidade linear a distância r do ponto P ao centro de rotação instantâneo (C_{RI}). Uma vez que todos os pontos sobre o robô têm a mesma velocidade angular, temos a seguinte relação para as velocidades lineares da roda esquerda ($\mathbf{v_e}$) e da roda direita ($\mathbf{v_d}$):

$$\frac{\mathbf{v_e}}{r_e} = \frac{\mathbf{v_d}}{r_d} \tag{2.1}$$

[2]Note-se que o centro de rotação instantâneo vai variando ao longo da trajectória do corpo rígido, sendo fixo apenas nos casos em que estamos em presença de uma rotação pura.

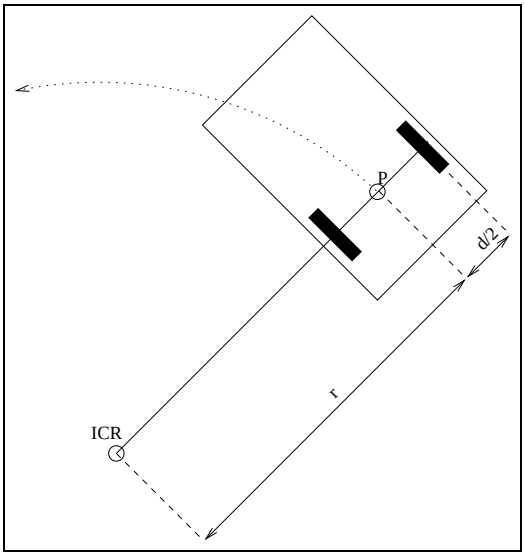

Figura 2.3: Rotação da plataforma em torno do ponto ICR

em que r_e e r_d correspondem respectivamente às distâncias da roda esquerda e da roda direita ao ponto C_{RI}.

Por outro lado se o ponto P está situado exactamente a meio das duas rodas motrizes então a distância do ponto P a cada uma das rodas será $d/2$ e obtemos facilmente a velocidade linear de cada uma das rodas,

$$\begin{aligned} \mathbf{v_e} &= \omega \times (\mathbf{r} - \mathbf{d}/2) \\ \mathbf{v_d} &= \omega \times (\mathbf{r} + \mathbf{d}/2) \end{aligned} \tag{2.2}$$

onde $\mathbf{d}$ é um vector coincidente com o eixo de rotação das rodas e que parte da roda esquerda para a roda direita e $\mathbf{r}$ é o vector que une o ponto C_{RI} ao ponto P. Obtidos estes valores é possível calcular a velocidade angular de cada uma das rodas em torno do seu eixo de rotação

$$\begin{aligned} \omega_\mathbf{e} &= \mathbf{v_e}/R \\ \omega_\mathbf{d} &= \mathbf{v_d}/R, \end{aligned} \tag{2.3}$$

onde R é o raio das rodas motrizes.

2.1.2 Estimação da Posição

A forma mais utilizada de obter a posição de um robô móvel é através da estimação da mesma com base no conhecimento da posição inicial, no caminho percorrido e no perfil de velocidade utilizado ao longo desse mesmo caminho. Este tipo de localização

é muitas vezes vezes referido na literatura pela expressão inglesa "dead reckoning" que derivou de "deduced reckoning"[24] ou simplesmente por odometria. Este é um dos métodos mais simples e mais utilizados, sendo na maior parte das vezes implementado acoplando codificadores ópticos incrementais ao eixo das rodas motrizes. Estes sensores permitem detectar qualquer deslocamento de cada uma destas rodas. Vejamos a seguir como é feita a estimação da posição com base nos incrementos obtidos dos sensores.

Em primeiro lugar é necessário dispor, para cada uma das rodas, de um factor de conversão entre os impulsos obtidos do codificador e o deslocamento linear associado. Sendo C_m esse factor de conversão, D_m o diâmetro da roda (mm), C_e o número de impulsos emitidos pelo codificador durante uma revolução completa e n o factor de redução entre o eixo da roda e o eixo a que está ligado o codificador, temos

$$C_m = \frac{\pi D_m}{nC_e} \tag{2.4}$$

Sendo N_e e N_d o número de impulsos contados desde a última amostragem para cada uma das rodas, podemos calcular a variação incremental na posição destas.

$$\Delta U_{e,i} = C_m N_{e,i} \tag{2.5}$$

$$\Delta U_{d,i} = C_m N_{d,i} \tag{2.6}$$

Note-se que $N_{e,i}$ e $N_{d,i}$ podem tomar valores positivos ou negativos se as rodas rodarem no sentido directo ou inverso, respectivamente e de acordo com o que for convencionado. Assim, os valores $\Delta U_{e,i}$ e $\Delta U_{d,i}$ corresponderão a deslocamentos lineares "para a frente" ou "para trás".

Utilizando estes valores, o deslocamento do centro do robô (ponto P) vem então

$$\Delta U_i = \frac{\Delta U_d + \Delta U_e}{2} \tag{2.7}$$

A variação incremental da orientação do robô vem

$$\Delta \theta_i = \frac{\Delta U_d - \Delta U_e}{b} \tag{2.8}$$

em que $b = ||\mathbf{d}||$ é a distância entre os pontos de contactos de cada uma das rodas com o solo.

A orientação (relativa) do robô é calculada por

$$\theta_i = \theta_{i-1} + \Delta \theta_i \tag{2.9}$$

sendo a posição (relativa) do ponto central dada por

$$\begin{aligned} x_i &= x_{i-1} + \Delta U_i \cos \theta_i \\ y_i &= y_{i-1} + \Delta U_i \sin \theta_i \end{aligned} \tag{2.10}$$

Este método de localização tem como vantagem a simplicidade de , sendo por isso bastante utilizado em robôs móveis terrestres. No entanto, dado o seu funcionamento em malha aberta, os valores de posição fornecidos estão sujeitos a erros cumulativos que advêm de factores como sejam o escorregamento das rodas, a variação do centro de curvatura ao longo da superfície de contacto das rodas com o chão (deslocamento do ponto de contacto) e mesmo da variação do diâmetro das rodas com a carga do veículo.

2.1.3 Sistema Operativo

Tal como já foi referido anteriormente, a plataforma Robuter utiliza 3 controladores interligados por um bus VME e cujas funções se repartem entre o controlo de sensores, o controlo dos movimentos e a estimação odométrica. É através de um destes controladores que é possível enviar comandos para o robô ou colocar programas em execução para o controlo do mesmo.

Todos os controladores se baseiam em processadores da série 68000 da Motorola e utilizam um sistema operativo de tempo-real designado por Albatros.

Mas afinal "o que é um sistema de tempo-real"? Se fizermos uma pesquisa encontramos um grande número de definições todas elas diferentes mas que no entanto todas têm algo em comum. Vejamos algumas delas.

- **Oxford Dictionary of Computing** "Qualquer sistema no qual é importante o instante em que a saída é produzida. Isto, porque normalmente a entrada corresponde a uma qualquer modificação no mundo físico e a saída também terá de se relacionar com a mesmo modificação. O atraso entre o instante de entrada e o de saída deverá ser suficientemente pequeno para ser aceitável".

- **Young (1982)**

  "Qualquer actividade ou sistema de processamento de informação que tenha de responder a estímulos externos dentro de períodos de tempo finitos e conhecidos".

- **Randel (1995)**

  "... é um sistema ao qual se requer que reaja a estímulos do ambiente (o que inclui a própria passagem do tempo) dentro de intervalos de tempo ditados pelo próprio ambiente."

Com base nestas definições podemos dizer que um sistema de tempo-real é qualquer sistema de processamento de informação que ao receber um estímulo externo produz uma resposta a este, dentro de um período de tempo finito bem definido.

Podemos ainda acrescentar que num sistema de tempo-real a validade de um resultado não depende apenas da precisão numérica com que este é calculado mas também do instante em que é produzido. Ou ainda, que a ausência de uma resposta atempada é tão má como uma resposta errada.

Há a ideia mais ou menos generalizada de que um sistema de tempo-real é aquele em que o processamento é feito da forma mais rápida possível. Isto não corresponde inteiramente à verdade pois o que é realmente importante é que as operações sejam feitas dentro de certos limites pré-estabelecidos. Exemplos disto são os sistemas de controlo para os quais os limites para os tempos de resposta variam de caso para caso. Por exemplo, no caso do controlo da posição de um braço robotizado, o tempo de resposta deverá ser inferior ao período de amostragem que rondará alguns milisegundos, ao passo que o controlo da temperatura de um forno necessitará de tempos de resposta não tão rápidos devido à inercia térmica do forno.

Sistemas Operativos de Tempo-Real

A principal diferença entre os sistemas de tempo-real e os sistemas convencionais multi-programados reside na capacidade que normalmente só os primeiros apresentam em fornecer respostas rápidas a eventos, e assim, satisfazerem restrições temporais no escalonamento de processos, permitindo a sua utilização em malhas de controlo de sistemas industriais, centrais telefónicas, ou qualquer outro que sistema que apresente restrições temporais. Para atingir estes objectivos, muitas vezes são relegados para segundo plano alguns factores como sejam: a facilidade de utilização, e a uniformização e gestão de recursos. Estas aparentes falhas surgem como resultado de um compromisso que muitas vezes é necessário fazer, pois é normal exigir-se a um sistema deste tipo a capacidade de processar milhares de interrupções por segundo, durante determinados períodos de tempo, sem perder um único evento. São também, em muitos casos, relegados para segundo plano, mecanismos de protecção, de gestão de memória ou ainda os sistemas de ficheiros, por razões ligadas ao facto destes normalmente introduzirem complexidade e peso computacional adicionais, reduzindo o desempenho do sistema, ou simplesmente por questões de incompatibilidade com as restantes especificações.

Os sistemas de tempo-real podem ser construídos com base em aplicações que correm sobre um sistema operativo, ou utilizando um núcleo de multi-programação que é utilizado para construir a aplicação. No primeiro caso, a fase de desenvolvimento é sem dúvida facilitada pois o "debugging" é feito com o recurso a ferramentas convencionais. No segundo caso, o desenvolvimento de aplicações é dificultado essencialmente pela falta de ferramentas de "debugging" que possam ser utilizadas nos sistemas embebidos. Em ambos os casos é normalmente utilizado um escalonador

preemptivo que utiliza as prioridades dos processos para seleccionar o processo a executar em cada instante. Estes sistemas permitem utilizar mecanismos de comunicação inter-processo, bem como primitivas de sincronização (que podem ser semáforos) e sinalização, além das imprescindíveis primitivas de temporização.

"Albatros"

O "Albatros" utiliza um mecanismo baseado em dois escalonadores, um não-preemptivo e outro preemptivo, sendo o escalonamento dos processos feito com base em eventos juntamente com um mecanismo de prioridades. Os processos são divididos em 3 classes básicas que, de acordo com as designações adoptadas pela RoboSoft, são: processos do sistema, processos do utilizador não interruptíveis, processos do utilizador interruptíveis [35, 38].

Os processos do sistema não podem ser controlados pelo utilizador de forma directa, embora possam ser activados ou desactivados indirectamente através do envio de determinados comandos. Um exemplo é o comando *SERV ON* que coloca 3 novos processos em execução: um processo responsável por efectuar as leituras dos sensores, um segundo que implementa a malha de controlo e um último que vai escrever os valores calculados nos registos dos conversores digital-analógicos.

Os processos não interruptíveis do utilizador podem ser acrescentados ou retirados a partir de uma aplicação desenvolvida numa linguagem de alto nível. Estes processos quando colocados em execução não são interrompidos por outros processos, com excepção dos processos do sistema. Este tipo de processos deve ser utilizado com muito cuidado pois dado que estes obedecem a uma disciplina de escalonamento do tipo "First-come-first-served" (FCFS) conduzem normalmente a valores médios de "Turnaround Time[3]" bastante elevados sobretudo quando existem processos com tempos de execução longos. Por outras palavras, estando um processo longo em execução em determinado instante, se um segundo processo de curta duração entra na fila de execução, este só irá ser activado quando o primeiro terminar (voluntariamente). Esta forma de escalonamento faz com que o segundo processo apresente um tempo de resposta igual à soma do tempo em que esteve em fila de espera mais o tempo que realmente demorou a sua execução. Uma outra consequência óbvia desta disciplina de escalonamento é que se um processo entra num ciclo fechado, não terminando a execução, todos os restantes processos ficam bloqueados, exceptuando-se, no caso do "Albatros", os processos do sistema.

O terceiro tipo de processos são controlados por um escalonador do tipo "event driven". Por isso cada processo recebe no instante da criação uma prioridade, um período e um desfasamento. A prioridade é utilizada pelo escalonador para seleccionar

[3]Turnaround Time é definido como o tempo que vai desde o instante em que um processo (job) fica pronto a executar (ready) até ao instante em que este completa a sua execução.

em cada instante o processo a executar de entre os processos "prontos". O período serve para definir de quanto em quanto tempo é que cada processo é colocado em execução. O parâmetro de desfasamento serve para o caso em que dois processos têm o mesmo período, para definir a diferença temporal na activação de cada um em relação ao início do período.

2.1.4 Facilidades de desenvolvimento

O sistema operativo "Albatros", tal como acontece com a maioria dos sistemas embebidos, está longe de ser a base ideal para o desenvolvimento de aplicações de tempo-real.

O problema principal prende-se com a dificuldade de fazer o "debugging" de uma aplicação num sistema embebido. Em alguns casos esta operação poderá ser mesmo impossível se não existirem dispositivos de visualização que possam ser utilizados para indicar que o valor de uma variável está correcto ou que um fio de execução passou em determinado ponto do código.

Por outro primitivas fornecidas pelo núcleo do "Albatros" resumem-se à criação e terminação de processos, não existindo também quaisquer mecanismos de comunicação ou de sincronização entre processos, o que não é justificável por questões de rapidez, pois "mailboxes", semáforos ou mesmo sinais se implementados de forma adequada não introduzem peso computacional significativo, sendo estes mecanismos classificáveis como pertencendo ao conjunto de requisitos mínimos para qualquer sistema operativo.

Podemos ainda referir como limitação a não existência de um sistema fiável de comunicação com estações remotas de forma a permitir efectuar remotamente o controlo do robô e obter leituras dos sensores. A configuração original utilizava dois "modems" sem fios que permitia a comunicação via portos RS232C, no entanto este sistema era demasiado sensível a interferências e não dispunha de qualquer mecanismo de detecção/correcção de erros pelo que as comunicações eram frequentemente corrompidas.

2.1.5 Ambiente de desenvolvimento original

A criação de aplicações baseadas neste sistema de desenvolvimento pode ser feita através do uso de uma linguagem de alto nível-"C" [36]. O desenvolvimento de programas é efectuado usando um computador externo e depois de compilados são enviados para o computador de bordo através de uma linha série RS232. A comunicação entre os programas desenvolvidos e o "Albatros" é feita, com a ajuda de uma pequena biblioteca de rotinas, de uma forma muito semelhante ao que é feito a partir de um terminal remoto, pois esta comunicação é efectuada enviando comandos ou

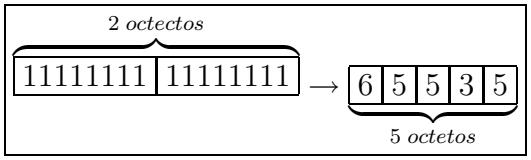

Figura 2.4: Representação do número 65535 usando os códigos Binário e ASCII

recebendo os resultados sob a forma de cadeias de caracteres *ASCII*.

Este modo de funcionamento tem a vantagem de o desenvolvimento de programas destinados a serem executados no computador de bordo da plataforma ou numa máquina remota ser semelhante; enquanto no primeiro caso os comandos são enviados usando a função *sendf*, no segundo caso pode-se utilizar uma qualquer função que permita enviar uma cadeia de caracteres para o porto série. No entanto, a vantagem acima apontada é apenas aparente, pois em ambos os modos de funcionamento, local ou remoto, o método apresenta alguns problemas como se pode ver a seguir. Uma vez que todos os comandos, e respectivos resultados, têm obrigatoriamente que ser enviados sob a forma de cadeias de caracteres isso implica o uso de uma rotina de formatação para a sua criação e de um interprete de comandos após a sua recepção. Na prática vai aumentar o peso computacional e além disso aumentar o tempo de resposta a estes comandos.

O uso da ligação série para enviar os comandos a partir de um computador remoto sujeita-os a possíveis corrupções por ruído externo, ou ainda por falhas de ligação no caso de se utilizar uma ligação via rádio. Ora se em alguns casos isto se traduz apenas na recepção de um comando inválido que vai gerar uma mensagem de erro, noutros casos isto pode ter consequências bem mais catastróficas que provoquem a destruição do equipamento devido a uma colisão, como resultado da recepção de um comando cujo parâmetro foi corrompido.

Uma segunda desvantagem tem a ver com o fraco aproveitamento que este método permite fazer de uma ligação série, que apresenta à partida uma velocidade de transmissão baixa, sobretudo em casos em que se pretende efectuar um controlo ou uma monitorização remotos, com um tempo de amostragem pequeno. Os comandos do "Albatros" são normalmente compostos por quatro caracteres, que os identificam, e por valores numéricos que fazem parte dos parâmetros. Para os transmitir, os valores numéricos têm que ser convertidos numa representação *ASCII*, o que se traduz no envio de mensagens longas. O aumento do tamanho das mensagens, tem como consequência a redução no número de mensagens que podem circular num ou noutro sentido num determinado período de tempo. Como se pode ver na figura 2.4, um inteiro representável por 2 octetos pode necessitar de 5 caracteres ao ser representado

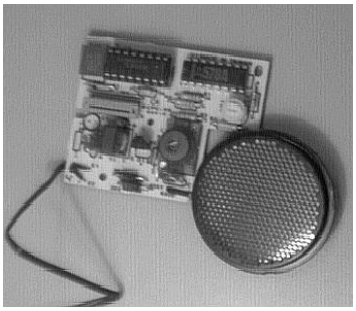

Figura 2.5: Transdutor e circuito do sensor de ultra-sons modelo 6500 da Polaroid

em *ASCII*.

2.1.6 Sensores de Ultra-sons

Os sensores de ultra-sons foram utilizados pela primeira vez, por volta de 1918, no desenvolvimento do sistema SONAR (SOund Navigation And Ranging)[24]. O objectivo deste sistema, era permitir a detecção da posição, velocidade e orientação de objectos em ambiente subaquático.

Foi nos primeiros sistemas fotográficos dotados de focagem automática que os ultra-sons começaram a ser utilizados ao "ar livre", tendo como função medir a distância entre a câmara fotográfica e o alvo a fotografar. No entanto, este sistema obteve pouco sucesso, devido ao facto de o ar ser um meio menos favorável à propagação das ondas sonoras do que a água, tendo sido maioritariamente substituído por métodos de focagem baseados na utilização de medições directas sobre a imagem.

Actualmente, a utilização de sistemas de ultra-sons para sistemas ao ar livre, tem recebido alguma atenção por parte de muitos investigadores devido ao baixo preço destes sistemas e pela simplicidade de utilização dos mesmos. No entanto, a sua utilização para extrair informação detalhada sobre o ambiente ainda não obteve o sucesso desejado, por razões que irão ser analisadas mais à frente.

O sistema Polaroid

O sistema de ultra-sons usado neste trabalho é baseado nos sensores da marca Polaroid da série 6500 (figura 2.5), sendo possivelmente o mais utilizado em aplicações de robótica móvel. O primeiro sensor de ultra-sons que a Polaroid desenvolveu, com vista a aplicação em câmaras fotográficas de focagem automática, emitia quatro frequências discretas por volta dos 50 kHz. O modelo que se seguiu (SN28827) utilizava um número menor de componentes e apresentava um menor consumo bem como um

interface simplificado. Este segundo modelo utilizava uma única frequência de 49.41 kHz. O modelo 6500, utilizado no desenvolvimento deste trabalho, foi introduzido em 1990 com um circuito ainda mais simplificado que os anteriores tendo ainda a capacidade de detectar ecos múltiplos[4].

O transdutor

Este tipo de sensores utiliza um único transdutor que é funciona alternadamente como emissor de uma onda ultra-sonora e como receptor dos ecos das mesmas ondas. A parte principal deste transdutor é uma lâmina fina, na qual é efectuada a conversão da energia eléctrica em ondas sonoras e vice-versa[32]. Esta lâmina é feita de um material plástico (kapton), sobre o qual é depositado uma fina camada condutora, sendo esticada sobre um suporte metálico. Obtém-se, deste modo, um condensador cujo dieléctrico é a lâmina plástica, sendo as armaduras constituídas pelo suporte metálico e pela fina camada condutora depositada sobre a lâmina. Quando aos terminais deste dispositivo é aplicada uma tensão, a lâmina sofre a acção de uma força electrostática que a leva a curvar-se. Se em vez de uma tensão contínua, for aplicado um trem de impulsos estes irão fazer com que a lâmina vibre emitindo uma onda sonora. Por outro lado, uma onda sonora ao embater na lâmina irá fazer com que esta vibre, provocando uma variação da capacidade do dispositivo permitindo-lhe funcionar como receptor das ondas ultra-sonoras.

É com base no que se acaba de referir que se baseia o funcionamento dos sensores de ultra-sons utilizados neste trabalho. Estes sensores usam um único transdutor que é usado de forma alternada como emissor e receptor, sendo as medições de distância efectuadas com base no tempo de voo de uma onda de ultra-sons, ver figura 2.6.

Em cada medição, o transdutor é inicialmente excitado com um trem de 16 impulsos com uma frequência de 49.4kHz, que o faz vibrar emitindo uma onda sonora perpendicularmente à superfície do sensor. Findo este período, e após um pequeno período de pausa destinado a evitar falsas detecções que poderiam ser provocadas por vibração residual, é posto em funcionamento um contador ao mesmo tempo que o sensor passa a funcionar como receptor. A onda emitida, ao embater num obstáculo sofre um efeito de reflexão, sendo enviada, em certas circunstâncias, de volta para o sensor.

O transdutor ao receber as ondas sonoras reflectidas vai gerar um sinal eléctrico correspondente o qual é aplicado a uma etapa de amplificação cujo ganho varia de forma inversamente proporcional ao tempo de propagação da onda. O efeito desta de amplificação de ganho variável é compensar as atenuações sofridas devido à distância

[4] A capacidade da série Polaroid 6500 de detectar ecos múltiplos não foi aproveitada na sua integração na plataforma Robuter, pelo que não foi possível testar a sua utilidade na detecção de obstáculos.

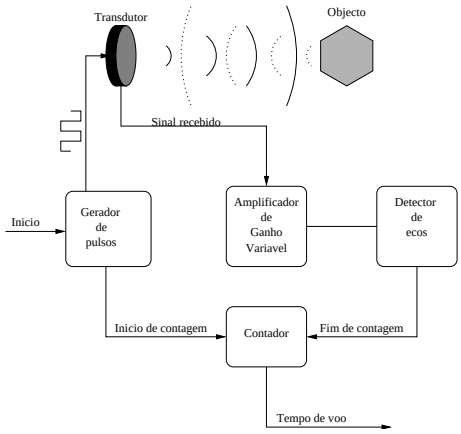

Figura 2.6: Funcionamento do Sensor de Ultra-Sons Polaroid

percorrida.

Quando a energia da onda reflectida sobe acima de um limiar pré-estabelecido, é parado o contador ficando no seu registo o número de períodos correspondentes ao tempo de voo da onda. Deste tempo de voo pode-se determinar a distância do sensor até ao alvo que provocou a sua reflexão usando a seguinte expressão.

$$R_0 = \frac{ct_0}{2} \tag{2.11}$$

onde c representa a velocidade de propagação do som no ar (m/s) e t_0 o tempo de voo medido.

Vantagens e desvantagens dos sensores de ultra-sons

A primeira vantagem da utilização deste tipo de sensores advêm da simplicidade de aplicação e o seu custo reduzido. Existem disponíveis no mercado sistemas completos que incluem o transdutor, o circuito de controlo assim como conjuntos de desenvolvimento por preços bastante acessíveis.

Outras vantagens surgem da forma como é efectuada a medição das distâncias. A distância absoluta do sensor a um ponto observado é obtida directamente do tempo de voo da onda não necessitando de recorrer a qualquer outro tipo de análise do sinal. Por fim podemos apontar como vantagem o facto de os sensores baseados no tempo de voo, quando utilizados num meio homogéneo a temperatura constante, manterem a precisão com a distância (desde que o eco possua energia suficiente para

ser detectado), ao passo que nos métodos baseados em triangulação a precisão vai diminuindo à medida que a distância ao alvo aumenta.

Apesar das vantagens que acabámos enunciar a aplicação de sistemas de SONAR baseados no tempo de voo, tem-se resumido a casos simples de medição de distâncias, por estes sistemas apresentarem diversos problemas, como os que são enumerados a seguir:

1. **Velocidade de Propagação** - A velocidade de propagação das ondas sonoras é fortemente influenciada por variações de temperatura pois uma variação na temperatura ambiente de 30^0F (aproximadamente 16.45^0C) pode produzir um erro de $0.3m$ numa medida de $10m$. Um outro factor que influencia a velocidade de propagação, se bem que de uma forma mais suave, é a humidade atmosférica.

2. **Incerteza na Detecção** - A energia reflectida por um alvo depende do tipo de material do qual é composta a superfície deste, o que vai influenciar a intensidade do sinal reflectido. Isto por sua vez, vai influenciar o tempo de subida dos ecos recebidos, e no caso de se utilizar um limiar de detecção fixo, vai fazer com que alvos mais reflectores pareçam estar mais próximos do que os menos reflectores. Por esta razão, são aplicados em muitos casos, *discriminadores de fracção de tempo constante*, para estabelecer o limiar de detecção numa determinada fracção do eco recebido.

3. **Interacção com as Superfícies** - O eco recebido contém apenas uma pequena fracção da energia contida no sinal enviado. A energia restante, é perdida em efeitos de absorção, difracção ou refracção. É frequente, neste tipo de sistemas, o sensor não receber qualquer sinal reflectido mesmo existindo um objecto na frente deste. Este fenómeno deve-se ao facto de muitas superfícies se comportarem como espelhos para as ondas ultra-sonoras. Assim, se o ângulo de emissão, em relação à superfície do alvo, exceder um determinado valor crítico, a energia reflectida será desviada para fora da zona de detecção do receptor. Também poderá acontecer que estes sinais, sendo multiplamente reflectidos regressem ao receptor produzindo uma medida errada que é igual à soma das distâncias percorridas pelas ondas.

Modelo Utilizado

O modelo comummente utilizado para o sensor de ultra-sons baseia-se em aproximá-lo a um pistão plano de raio r, que vibra a uma frequência f.

Se o raio deste pistão for muito maior que o comprimento de onda λ do sinal sonoro emitido, este propaga-se sob a forma de um feixe dirigido.

Podem-se considerar duas zonas: a zona de *Fresnel* também chamada zona próxima e a zona de *Fraunhofer* ou zona distante. Na zona de Fresnel o feixe apresenta uma

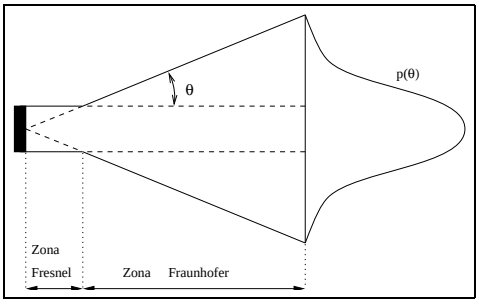

Figura 2.7: Padrão de radiação para o modelo do pistão plano

forma cilíndrica de raio $2r$ e estende-se até uma distância do pistão de aproximadamente r^2/λ. Na zona de Fraunhofer, o feixe apresenta uma forma cónica, divergindo com uma abertura de $2\theta_0$, sendo este ângulo dado por:

$$\theta_0 = \sin^{-1} \frac{0.61\lambda}{r} \tag{2.12}$$

Como se pode verificar pela expressão 2.12, o ângulo θ_0 depende da frequência do sinal emitido assim como do raio do pistão r.

Para o caso do sensor Polaroid, cujo pistão apresenta um raio de $18.9mm$ e a frequência do sinal emitido é de $49.9kHz$, o comprimento da zona de Fresnel é de $51.4mm$ e o ângulo de divergência é de 12.96^o.

Como se mostra na figura 2.7, a pressão espacial provocada pela onda radiada não é uniforme. Se considerarmos uma linha paralela à superfície do pistão, o valor da pressão é máximo no ponto de intersecção desta linha com uma perpendicular ao pistão e que passa pelo seu centro. Para os restantes pontos situados sobre esta linha, a pressão decai de forma exponencial com a distância ao ponto de intersecção. Kuc[7] mostrou com base num modelo para o padrão de radiação do pistão plano (equação 2.13) que os pontos situados na parte central do feixe, recebem e consequentemente emitem muito mais energia do que quaisquer outros colocados na periferia deste.

$$p(\theta) = p_{max} e^{\frac{-2\theta^2}{\theta_0^2}} \tag{2.13}$$

Analisando a interacção das superfícies com as ondas de ultra-sons, verifica-se que estas podem ser divididas em duas categorias: superfícies reflectoras cujas dimensões são maiores que o comprimento de onda do sinal emitido, e as superfícies que provocam difracção e cujas dimensões são menores que o comprimento de onda. Como,

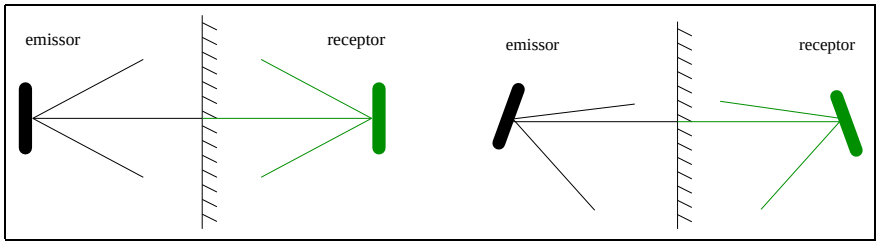

Figura 2.8: Um único sensor pode ser visto como um emissor e um receptor virtual

no caso do sensor Polaroid em que o comprimento de onda é de aproximadamente $6mm$, a maior parte das superfícies existentes nos ambientes humanos pertencem à categoria reflectora, e, por esta razão comportam-se como espelhos para as ondas de ultra-sons.

Considerando que os transdutores emissor e receptor apresentam padrões de emissão e recepção semelhantes, então o produto dos referidos padrões dará a seguinte expressão para a amplitude do sinal recebido,

$$A(\theta_1, \theta_2) = A_{max} e^{\frac{-2\theta_1^2}{\theta_0^2}} e^{\frac{-2\theta_2^2}{\theta_0^2}} \tag{2.14}$$

onde θ_1 e θ_2 são os ângulos entre as superfícies dos sensores emissor e receptor e uma linha que passa pelos seu centros, A_{max} é a amplitude do sinal recebido quando os sensores estão dirigidos um para o outro, ou seja $\theta_1 = \theta_2 = 0$.

Usando as propriedades reflectoras apresentadas pela maior parte das superfícies, um único transdutor funcionando como emissor e receptor alternadamente pode ser visto como um par de sensores, um emissor E e um receptor virtual R como se mostra na figura 2.8.

Neste caso os ângulos referidos na expressão 2.14 são iguais em módulo e o sinal depende do número de reflexões (n_r) que o sinal sofre antes de atingir a superfície do receptor. A relação entre estes ângulos é dada por

$$\theta_1 = (-1)^{n_r} \theta_2 \tag{2.15}$$

Fazendo $\theta_2 = \theta$ obtemos $\theta_1 = (-1)^{n_r}\theta$. Substituindo estes valores em 2.14 vem

$$A(\theta) = A_{max} e^{\frac{-4\theta^4}{\theta_0^2}} \tag{2.16}$$

Daqui se pode confirmar que a amplitude do sinal recebido $A(\theta)$ é máxima quando a superfície do sensor é paralela à superfície, o que corresponde a ter o emissor e

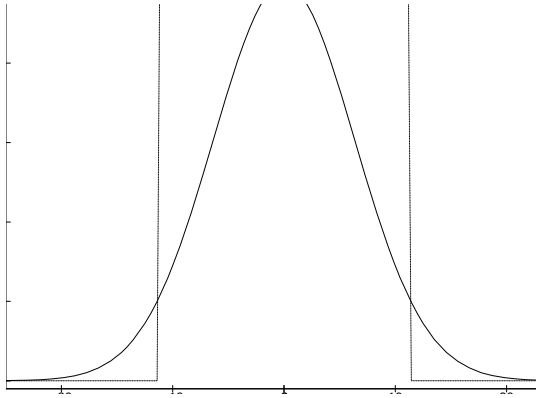

Figura 2.9: Variação angular da potência recebida e correspondente detecção discreta

o receptor virtual perfeitamente apontados um para o outro. Note-se que nestas expressões não se tomou em conta o efeito da distância na amplitude do sinal recebido, mas, dado que a maioria dos sensores usam amplificadores de ganho variável com o tempo, o valor de A_{max} não depende da distância R.

Na figura 2.9 podemos ver a variação angular da potência recebida bem como a aplicação de uma função de limiar à mesma. No caso da figura escolheu-se como limiar 20% da potência emitida, obtendo-se uma zona onde o sinal recebido tem uma potência acima do limiar e duas zonas laterais, onde o ângulo de incidência na superfície reflectora é tal que faz com que a potência reflectida seja inferior ao limiar.

Na realidade o feixe apresenta um padrão constituído por vários lobos como se pode verificar pela figura 2.10 e não apenas um em torno do eixo acústico como se poderia pensar. Isto vai ter uma grande influência nas medidas obtidas, pois como se pode verificar pela figura 2.11, ao variarmos o ângulo entre o eixo acústico e a normal a uma superfície reflectora, vamos obter medidas correctas para valores do ângulo pertencentes aos seguintes intervalos $[-12^0, +12^0]$,$[-22^0, -18^0]$ e $[18^0, 22^0]$. Para ângulos fora destes intervalos obtêm-se valores que não têm nada a ver com a distância entre o sensor e a superfície reflectora, ou seja a superfície deixa de ser detectada pelo sensor.

Análise Estocástica

Perante o que foi apresentado na secção anterior, facilmente se conclui que as medidas obtidas por este tipo de sensores não podem ser analisadas de forma isolada, dado que a informação contida nas mesmas é bastante limitada. Esta limitação deve-se por

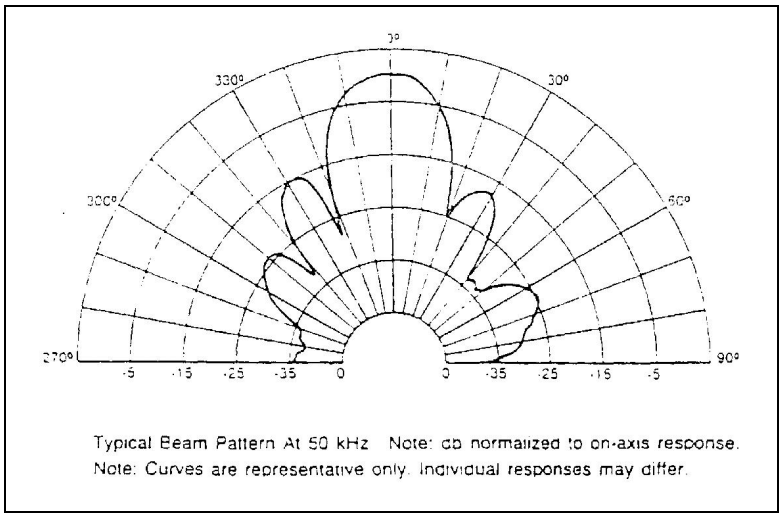

Figura 2.10: Padrão de radiação típico (fornecido pela Polaroid)

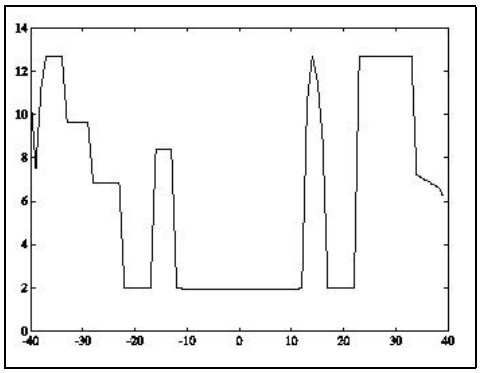

Figura 2.11: Medidas de SONAR obtidas (em metros) para uma superfície plana a uma distância de 2 metros versus o ângulo (em graus) entre o eixo acústico do sensor e a normal à superfície.(Cortesia de F. Moita)

um lado, ao facto de os valores medidos corresponderem à distância entre o sensor e o ponto mais próximo que reflectiu energia suficiente para ser detectada pelo sensor, por outro lado o padrão de radiação destes sensores, que, por apresentar uma grande abertura, conduz a um elevado grau de incerteza no que diz respeito à localização do obstáculo que produziu dos ecos medidos.

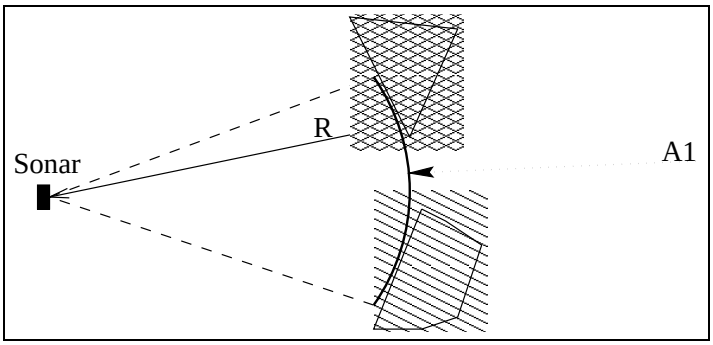

Figura 2.12: Uma medida não permite identificar o alvo que a produziu

A figura 2.12 ilustra o problema que se acabou de referir, onde dois alvos colocados em posições diferentes produzem a mesma leitura por parte do sensor. Daqui se pode concluir, que para qualquer superfície, desde que tangente ao arco A_1, o valor de distância medido pelo sensor é o mesmo.

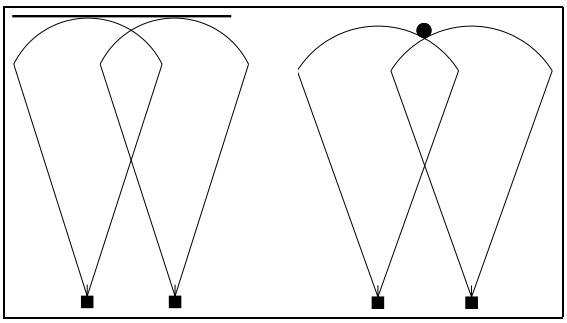

Figura 2.13: Mesmo usando dois sensores por vezes não é possível identificar o alvo detectado

Este tipo de incerteza quanto ao alvo detectado surge mesmo quando se utilizam sensores múltiplos, como se pode verificar pelo exemplo da figura 2.13. Neste caso uma superfície plana, ou uma superfície cilíndrica produziriam o mesmo tipo de leituras para dois sensores de ultra-sons, ou para um sensor que se deslocasse e efectuasse as leituras em pontos diferentes do espaço. Podemos então concluir que para qualquer

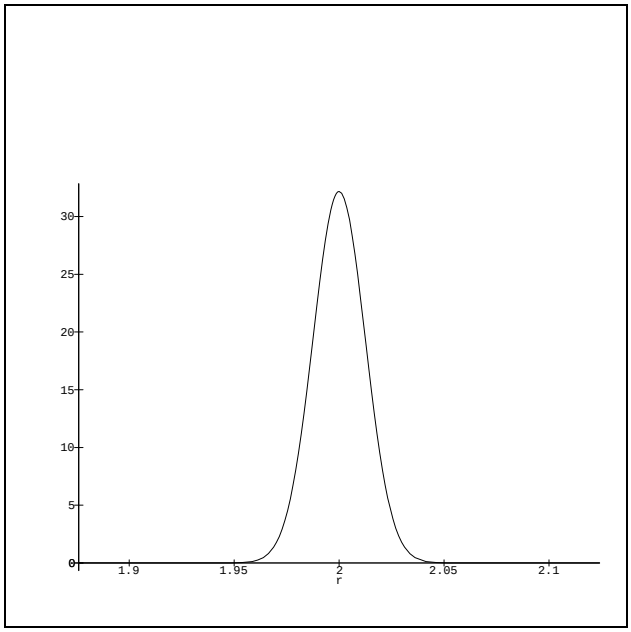

Figura 2.14: Densidade de probabilidade Gaussiana para as medidas de um sensor de ultra-sons (alvo a uma distância do sensor $z = 2m$)

medida que se obtenha com este tipo de sensores, existe um elevado grau de incerteza no que diz respeito à localização do obstáculo que lhe deu origem. Se ignorarmos os efeitos provocados por reflexões múltiplas, podemos dizer que o alvo está certamente localizado sobre um dos pontos do arco A_1, sendo qualquer destes pontos um possível candidato.

Usando os resultados das experiências de calibração sobre este tipo de sensores efectuados nos nossos laboratórios[11], podemos aproximar os sensores Polaroid como sendo sensores ideais cujas saídas são corrompidas por ruído Gaussiano de média zero e desvio padrão dado por:

$$\sigma_D(r) = 0.0052 \times r + 0.002[m] \tag{2.17}$$

onde r é a distância entre o sensor e o alvo expressa em metros.

Tomando estes resultados como ponto de partida podemos escrever a seguinte expressão para o modelo do sensor (ver figura 2.14), o qual define a densidade de probabilidade de obter uma medida r com a existência de um alvo a uma distância z.

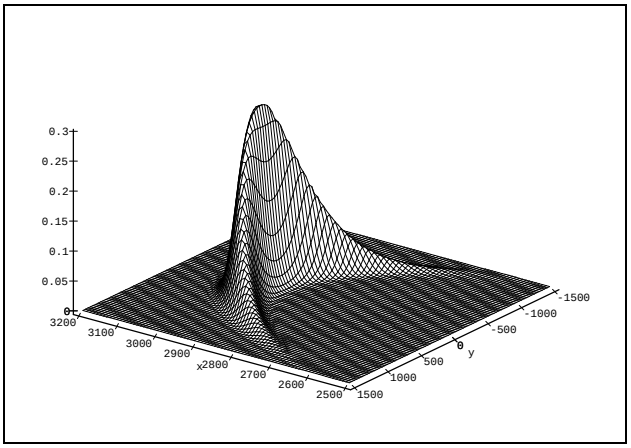

Figura 2.15: Densidade de probabilidade Gaussiana para as medidas de um sensor de ultra-sons dependente da distância e do ângulo.

$$p(r|z) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(r-z)^2}{2\sigma^2}} \tag{2.18}$$

Podemos agora expandir o modelo do sensor para o espaço a duas dimensões caracterizando-o com base na distância radial entre o alvo e o sensor (z) e o ângulo formado pela recta perpendicular ao obstáculo e que passa pelo centro do sensor e a perpendicular ao sensor. Desta forma utilizamos σ_r^2 e σ_θ^2 como sendo a variância com a distância e a variância angular, respectivamente, obtendo a expressão 2.19 para a densidade de probabilidade em coordenadas polares.

$$p(r|z,\theta) = \frac{1}{2\pi\sigma_r\sigma_\theta} e^{-\frac{1}{2}\left(\frac{(r-z)^2}{\sigma_r^2}+\frac{\theta^2}{\sigma_\theta^2}\right)} \tag{2.19}$$

A figura 2.15 mostra o gráfico da densidade de probabilidade, definido pela expressão 2.19, para $z = 3000mm$, $\sigma_r = 17mm$ e $\sigma_\theta = 0.1rad$, com o sensor colocado no ponto de coordenadas $x = 0$, $y = 0$ e orientado ao longo do eixo dos yy.

2.2 Criação de um suporte para controlo distribuído

Com vista a ultrapassar as limitações da plataforma Robuter referidas anteriormente, no que diz respeito à criação de aplicações, optou-se por desenvolver um conjunto de

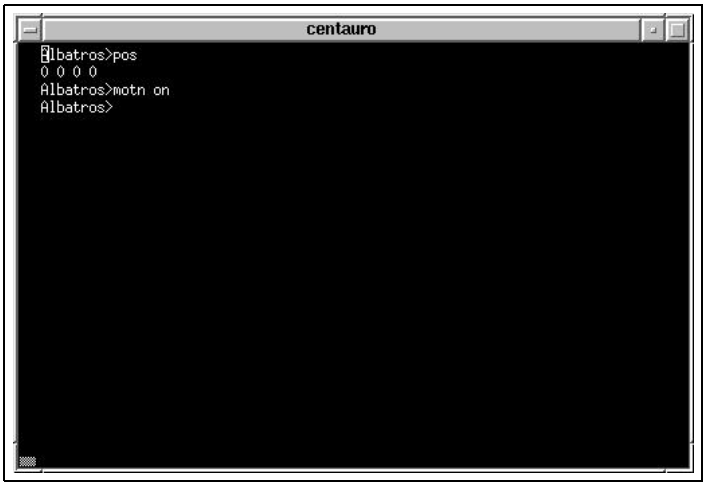

Figura 2.16: Exemplo de envio de comandos usando um terminal remoto

ferramentas que permitissem uma utilização mais simples do robô e o desenvolvimento de aplicações.

Uma vez que as ferramentas já existentes se destinavam a ser utilizadas em ambiente UNIX, procurou-se tornar este último como o sistema preferencial de desenvolvimento e suporte de futuras aplicações. Para isto era necessário desenvolver um sistema de comunicação que permitisse às aplicações executadas numa estação de trabalho comandar remotamente o robô assim como obter leituras dos seus sensores.

Para isto começou-se por analisar as capacidades de comunicação disponíveis tendo-se verificado que a placa principal do robô dispõe de quatro portos série sendo apenas dois suportados pelo sistema operativo "Albatros". Este suporte foi concebido de forma a possibilitar o uso de um terminal externo para enviar comandos e receber os resultados de uma forma idêntica à de um terminal ligado a uma estação de trabalho.

Como consequência, as funções disponíveis para a utilização dos portos série limitam o seu uso ao envio e recepção de cadeias de caracteres imprimíveis,i.e. caracteres cujo código esteja entre os valores decimais 32 e 126. Deste modo, quando se pretende enviar um qualquer valor binário este terá de ser previamente convertido numa representação ASCII antes do envio e a conversão inversa após a recepção. A principal desvantagem deste método, é a sobrecarga ("overhead") protocolar introduzida ao efectuar as conversões de binário para ASCII visto este tipo de representação necessitar, em média, de um número maior de bits para representar o mesmo valor, como poderemos verificar pelo seguinte exemplo.

Considerando que os parâmetros de comunicação são: $9600bps$, 8 bits por caracter, sem paridade e 1 stop bit, verifica-se que por cada octeto a transmitir são utilizados

$$N = 8 \text{ bits de informação} + 1 \text{ start bit} + 1 \text{ stop bit} = 10 \text{ bits}. \tag{2.20}$$

Calculando o valor da velocidade de transmissão efectiva, ou seja a velocidade média a que são transmitidos os bits de informação, vem

$$C_e = \frac{C \times N_i}{N} \tag{2.21}$$

em que C_e é a velocidade efectiva, C a velocidade de transmissão e N_i o número de bits de informação e N o número total de bits transmitidos. Substituindo nesta expressão os valores acima definidos vem que $C_e = 7680bips$ (bits de informação por segundo) o que corresponde a uma taxa de transmissão de 960 caracteres por segundo. Considerando agora a transmissão de números inteiros representáveis por 16 bits

Se se utilizar a codificação ASCII para representar números inteiros entre 0 e 65535, os quais são representáveis por 16 bits em numeração binária ($N_i = 16$), verifica-se que o número de octetos utilizados varia de 1 a 5 ($N = 5 \times 8 = 40$). Substituindo estes valores na equação (2.21) , obtém-se o seguinte valor para a velocidade de transmissão efectiva

$$C_{e2} = \frac{C_{e1} \times 16}{40} = \frac{2}{5} \times 7680 = 3072. \tag{2.22}$$

De forma a ultrapassar as limitações acima referidas foi criado um módulo de software que acede directamente aos registos das DUARTs (Dual Asynchronous Receiver Transmitter) permitindo o envio de qualquer octeto de forma transparente[5], não sendo necessário efectuar qualquer encapsulamento ou codificação.

Este módulo foi concebido de forma a eliminar a possibilidade de ocorrência de erros nas comunicações através da utilização de um protocolo que permite a sua detecção e recuperação.

[5]Por envio transparente de caracteres significa que qualquer que seja o valor do octeto, e independentemente do significado normalmente atribuído ao caracter correspondente, este pode ser enviado sem necessitar de qualquer pré-processamento. Num sistema de transmissão "não transparente" existem caracteres que têm significados especiais e esses caracteres para serem transmitidos têm que sofrer uma codificação. Por exemplo num sistema que utilize os caracteres XON e XOFF para controlar o fluxo de informação, estes caracteres terão de ser retirados dos dados a transmitir. Isto é normalmente feito definindo um caracter DLE (data link escape) que modifica o significado do byte que lhe sucede. Desta forma os caracteres XON e XOFF podem ser convertidos em DLE+A e DLE+B a título de exemplo. O próprio caracter DLE, se ocorrer nos bytes a transmitir, será convertido em DLE+DLE. Após a recepção é feita a reconversão dos caracteres, substituindo as sequências de escape (é assim que são normalmente designadas) pelos bytes originais.

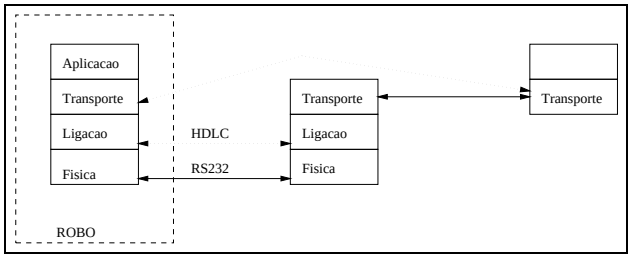

Figura 2.17: Estrutura por camadas do sistema de comunicações desenvolvido

2.2.1 Um ambiente proposto

Com vista a proporcionar um interface simples e ao mesmo tempo eficiente, para o desenvolvimento deste tipo de aplicações, procurou-se criar um ambiente que ocultasse todos os problemas intrínsecos às comunicações, tornando esta parte completamente transparente para o utilizador. Optou-se pela adopção de um modelo estruturado, baseado em camadas protocolares, com vista a obter um sistema de fácil manutenção e reconfiguração.

Esta estrutura por camadas, é comum em sistemas de comunicação, devido às vantagens referidas e que são conseguidas definindo quais os serviços que cada camada fornece à seguinte. Desta forma a substituição, por exemplos, das camadas física e de ligação por outras duas correspondentes, não deverá perturbar as restantes camadas, desde que o interface (serviços fornecidos) seja o mesmo. Neste projecto foi utilizada uma estrutura de quatro camadas que, por comparação com o modelo de referência OSI da ISO (International Standards Organization), são: camada física, camada de ligação, camada de transporte e camada de aplicação, as quais vão ser analisadas de seguida.

Camada Física

No caso presente a camada física consiste numa ligação série que baseada na norma RS232C. A transmissão é efectuada usando um par de modems sem fios, sendo este o equipamento que limita a velocidade de transmissão a 9600 bits por segundo (bps), visto que os portos série aos quais estes estão ligados e que equipam tanto o robô como a estação de trabalho utilizada suportam velocidades superiores, podendo em ambos os casos atingir velocidades de 38400bps.

Camada de Ligação

É esta camada, a responsável por garantir a fiabilidade da transmissão dos dados através de um determinado meio físico. Deverá, através da utilização de um qualquer

protocolo, detectar eventuais erros introduzidos nos dados e proceder à sua recuperação. Deverá ainda fornecer um conjunto de serviços às camadas superiores que lhes permita enviar dados entre os seus pares, ocultando o mais possível as particularidades inerentes à camada inferior.

A escolha do protocolo a implementar foi feita com base nos seguintes parâmetros, por ordem decrescente de prioridade:

1. Robustez
2. Possibilitasse obter um coeficiente de eficiência[6] próximo de 100%
3. Baixo "overhead" protocolar e computacional
4. Simplicidade de implementação

O protocolo HDLC (High-level Data Link Control) é neste momento um protocolo de referência aprovado pela ISO[17] e que é bastante utilizado na implementação de ligações ponto-a-ponto, um exemplo é o protocolo PPP (Point-to-Point-Protocol) [25] que se baseia no primeiro. Este foi o protocolo escolhido pelas razões referidas e por apresentar as seguintes vantagens:

- Transparência de dados.
- Eficiência elevada.
- Possibilidade de funcionar em modo assíncrono.
- Detecção de erros robusta e rápida.

Este protocolo foi originalmente concebido como sendo um protocolo orientado para o bit, ou seja, tratando os bits numa sequência são tratados como entidades independentes e não agrupados em caracteres, tendo-se mostrado bastante eficaz com esta forma de funcionamento No entanto dado o seu sucesso, têm surgido muitas aplicações, em que este é adaptado para funcionar com conjuntos de caracteres como é o caso das implementações de PPP assíncrono já referidas.

Na figura 2.18 pode-se ver o formato típico dos quadros utilizados pelo protocolo HDLC. Todos os quadros começam e terminam com o octeto 0x7e (em hexadecimal) que representa o marcador de início de quadro, isto permite ao receptor detectar de forma simples o início e o fim dos quadros, visto que este padrão não se pode repetir em mais sítio algum. Para tornar mais eficiente e para em casos extremos

[6] O coeficiente de eficiência corresponde à razão entre o tempo de transmissão de um quadro e o tempo de transmissão entre quadros, ou seja dá uma medida do aproveitamento da largura de banda disponível.

Flag	Endereço	Controlo	Informação	FCS	Flag
01111110	11111111	*	*	*	01111110

Figura 2.18: HDLC: Formato dos quadros

procurar reduzir o peso introduzido pelo protocolo, nas situações em que os quadros transmitidos são imediatamente consecutivos, utiliza-se a marca de fim de um quadro para sinalizar o início do quadro seguinte. A razão para isto, é que, uma vez que se utiliza o mesmo padrão para sinalizar ambas as situações (início e fim), se torna desnecessário o envio de duas marcas iguais consecutivas, em que uma sinaliza o fim de um quadro e a segunda o início de outro, como consequência reduz-se o número de octetos adicionais introduzidos pelo nível protocolar, obtendo-se assim um melhor aproveitamento da largura de banda disponível.

O campo de endereço destina-se principalmente a utilizações do protocolo em configurações multi-ponto, em que é usado o modo de funcionamento "Normal Response Mode". No caso presente, dado que se trata de uma ligação ponto-a-ponto, entre o robô e a estação de trabalho, torna-se desnecessária a utilização deste campo, visto não haver problemas de ambiguidade dadas as características de ligação.

O campo de controlo define o tipo de quadro recebido ou enviado, e que poderá pertencer a um dos grupos que se seguem:

- **Quadros Não-Numerados** Este tipo de quadros é utilizado para funções como o estabelecimento e o corte da ligação.

- **Quadros de Informação** São estes os quadros que efectivamente transportam a informação. Uma das características destes quadros é o facto de um quadro poder confirmar a recepção de quadros enviados em sentido contrário.

- **Quadros de Supervisão** Estes, tal como o nome indica, servem para controlo de fluxo e de erros, apesar de não transportarem informação, contêm números de sequência de envio e recepção.

O campo FCS (Frame Check Sequence) contém dois octetos para a detecção de possíveis erros. Este valor de 16 bits é um código polinomial conhecido por CRC (Cyclic Redundancy Check) que permite uma detecção de erros mais robusta do que a simples soma de todos os octetos da mensagem. O algoritmo CRC encontra-se descrito no apêndice C.

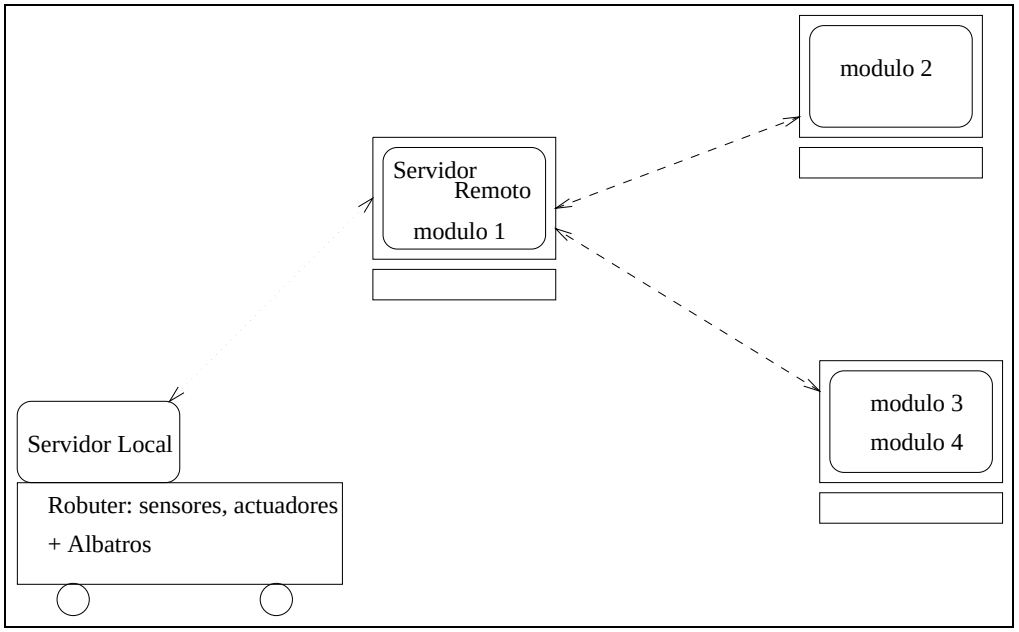

Figura 2.19: Sistema de exploração distribuído

Camada de Transporte

Tendo como base uma ligação fiável fornecida pela camada de ligação, interessa agora poder aceder a determinados serviços na máquina remota. No caso presente os serviços dividem-se em: controlo de movimentos, leitura de sensores de ultra-sons, da odometria e dos detectores de colisão, embora na presente fase estes últimos sejam processados localmente interrompendo qualquer operação quando é detectado o contacto com um destes sensores. O uso desta camada intermédia tem por objectivo simplificar e permitir estruturar melhor a camada seguinte. Em caso de necessidade, permite implementar novos serviços ou simplesmente activar ou desactivar serviços em determinada altura.

Camada de Aplicação

Nesta camada procurou-se criar um interface o mais simples possível para o desenvolvimento de aplicações de robótica móvel. Este interface fornece ao utilizador um conjunto de primitivas equivalentes às existentes no robô móvel. Não tendo sido feita uma implementação exaustiva de todas as funções do robô, visto que não era esse o objectivo principal deste trabalho, procurou-se que a adição de quaisquer funções pudesse ser efectuada de forma simples. A implementação foi feita utilizando a linguagem C++ por esta facilitar a estruturação dos programas através da utilização

do conceito de objecto, tirando-se partido do encapsulamento de pormenores de realização, apresentando-se ao utilizador um conjunto de classes facilmente extensíveis. Esta biblioteca de funções oculta a utilização de sockets de Berkeley para comunicar entre a aplicação desenvolvida e o servidor que efectua a comunicação com o robô. O software desenvolvido possibilita a criação de um conjunto de módulos que interaccionam com o robô, distribuídos por várias máquinas, o que permitirá obter, em caso de necessidade, maior potência de calculo (ver figura 2.19).

A possibilidade de criação de vários módulos independentes que comunicam de forma concorrente com a plataforma móvel, permite obter, além de maiores potências de processamento, uma melhor estruturação das aplicações e simplificar as fases de desenvolvimento e de testes. Um exemplo disto é a criação de módulos independentes para analisar os valores dos sensores para efectuar a detecção de obstáculos enquanto outros módulos efectuam o planeamento de trajectórias e controlam as mesmas através do envio de comandos de movimento para a plataforma.

A figura 2.19 mostra uma das situações possíveis, em que quatro módulos são distribuídos por três estações de trabalho, para tirar partido de uma maior capacidade de processamento, podendo estes aceder de forma concorrente, às funções fornecidas pelo servidor que reside no computador do robô. Os módulos, uma vez criados, podem ser distribuídos pelas estações de trabalho que se desejar, não necessitando para isso de qualquer reconfiguração ou recompilação, bastando apenas informar através da linha de comando qual o nome da máquina onde reside o servidor de comunicações.

2.2.2 Dificuldades

A principal dificuldade na realização deste sistema de comunicações surge, mais uma vez, a nível do robô. Esta dificuldade prende-se de, estranhamente, o envio e recepção de caracteres pelos portos série não serem efectuados de forma completamente assíncrona utilizando interrupções. Isto requeria que a implementação das rotinas de envio/recepção de caracteres através de ciclos de espera onde se aguardava a recepção ou o fim da transmissão de um caracter. As consequências deste método são o desperdício de ciclos de processador a fazer trabalho inútil. Se bem que num sistema mono-programado isto nem sempre seja crítico, em sistemas multi-programados como o que utilizamos estas rotinas teriam que ser chamadas com uma frequência elevada e terem uma prioridade elevada de forma a não se perderem caracteres. Como resultado as rotinas de comunicação utilizariam grande parte do tempo de processamento restando muito pouco para as restantes. A solução passa pela implementação dos mecanismos de comunicação de forma assíncrona ou seja baseados em interrupções. Aqui surgiram muitas dificuldades pois dada a inexistência de informação sobre o assunto foi necessário efectuar um estudo completo que envolveu os dispositivos da placa de processamento (BUS, controlador de interrupções, DUART) bem como o

processador e respectiva linguagem assembly.

2.2.3 Realização

A base deste sistema, tal como já foi anteriormente referido, requeria a construção de um mecanismo de suporte às comunicações entre o robô e a estação de trabalho que serve de suporte às aplicações. Uma vez que o robô dispõe de um conjunto de 4 portos série assíncronos foi utilizado um destes para efectuar a comunicação. A utilização dos portos série do robô levantou alguns problemas dado que o sistema operativo e as bibliotecas fornecidas pela Robosoft não permitem a utilização destes de forma transparente o que impossibilita o envio e a recepção de dados na forma binária. Dado que isto impunha uma grande restrição como já foi referido optou-se por utilizar os portos série directamente tirando partido da possibilidade de utilização de rotinas de interrupção para a recepção e envio dos dados. Desta forma conseguiu-se reduzir a carga imposta ao processador pelo mecanismo de comunicação uma vez que o sistema operativo baseia a utilização dos portos série em mecanismos de espera activa.

Esta realização requereu um estudo do hardware utilizado na placa de controlo principal do robô, especialmente o controlador de interrupções, o controlador dos portos série e o próprio micro-processador. Feito este estudo verificou-se que havia necessidade de efectuar duas pequenas ligações na placa que serviram para permitir ao controlador dos portos série enviar pedidos de interrupção ao processador e as quais estão descritas no Apêndice A.

O servidor do robô baseia-se em 3 tarefas concorrentes: uma de comunicações que utiliza o protocolo HDLC, uma que comanda os movimentos do robô e outra que controla as leituras dos sensores. Tendo sido identificados os períodos de activação requeridos para cada uma destas tarefas, as prioridades foram atribuídas às mesmas utilizando um ordenamento "rate monotonic". Foi possível verificar que esta distribuição de prioridades permitia um melhor funcionamento, pois o sistema continuava a responder de forma correcta mesmo quando o fluxo de pedidos aumentava, o que não acontecia para outros esquemas de prioridades utilizados.

2.2.4 Resultados e Limitações

Com este sistema é possível realizar um sistema distribuído de controlo para a plataforma Robuter. Um tal sistema permite a construção por exemplo de um sistema de controlo da plataforma no princípio "visual servoing" em que todo o processamento de imagem pode ser efectuado numa estação remota que envia os comandos de movimento para o computador de bordo da plataforma. Este conceito foi utilizado no desenvolvimento de várias aplicações que serão referidas mais à frente.

O interface de desenvolvimento (API - Aplication Programming Interface), juntamente com o sistema de comunicações criados estão de acordo com os objectivos inicialmente estabelecidos. As aplicações para controlar o robô são desenvolvidas como qualquer outra aplicação para ambiente UNIX podendo tirar partido disso, por exemplo dispor de um interface gráfico em Xwindows. O controlo das funções do robô é feito através de um conjunto de funções disponibilizadas numa biblioteca ficando todos os pormenores relativos às comunicações ocultadas por essas mesmas funções. Uma das principais vantagens da utilização deste API é o facto de permitir ultrapassar as limitações impostas pela fraca capacidade de processamento do computador de bordo do robô, podendo-se colocar as aplicações numa máquina UNIX mais ou menos poderosa em termos de capacidade de processamento.

Uma desvantagem que deriva da utilização do UNIX como ambiente para executar as aplicações é a de este não ser um sistema operativo de tempo-real, no entanto, têm surgido recentemente algumas versões do UNIX ou similares que já dispõem de algum suporte para o desenvolvimento de aplicações de tempo-real.

2.3 Simulador

A plataforma Robuter existente no ISR é partilhada por vários projectos e também porque a sua autonomia energética é bastante limitada, sentiu-se a necessidade de dispor de uma ferramenta que simulasse e substituísse a referida plataforma nos períodos de indisponibilidade da mesma. Foi com este objectivo que se desenvolveu um simulador ao qual se deu o nome de MoRoS (Mobile Robot Simulator).

O MoRoS permite simular execução de movimentos da plataforma Robuter, interpretando o mesmo tipo de comandos, e os sensores de ultra-sons Polaroid que são utilizados por esta.

O interface com o utilizador, que se pode ver na figura 2.20, foi desenvolvido para Xwindows [26, 2] utilizando o pacote de software Motif [15, 14] e permite a selecção da descrição do ambiente onde o robô se situa bem como da configuração dos sensores do robô. Isto permite modificar a localização dos sensores sobre a plataforma e definir as características relativas à abertura e alcance dos mesmos.

2.3.1 Realização

O coração do simulador é um escalonador de eventos, que utiliza uma lista de eventos ordenada por ordem de ocorrência. Assim, de cada vez que expira o tempo do primeiro evento, é executada a função de processamento do evento e este é apagado da lista ou , se se tratar de um evento periódico, este é recolocado na lista no local apropriado de acordo com o tempo que falta até à sua próxima ocorrência. Utilizando esta filosofia, cada movimento é transformado numa sequência de eventos que vão sendo

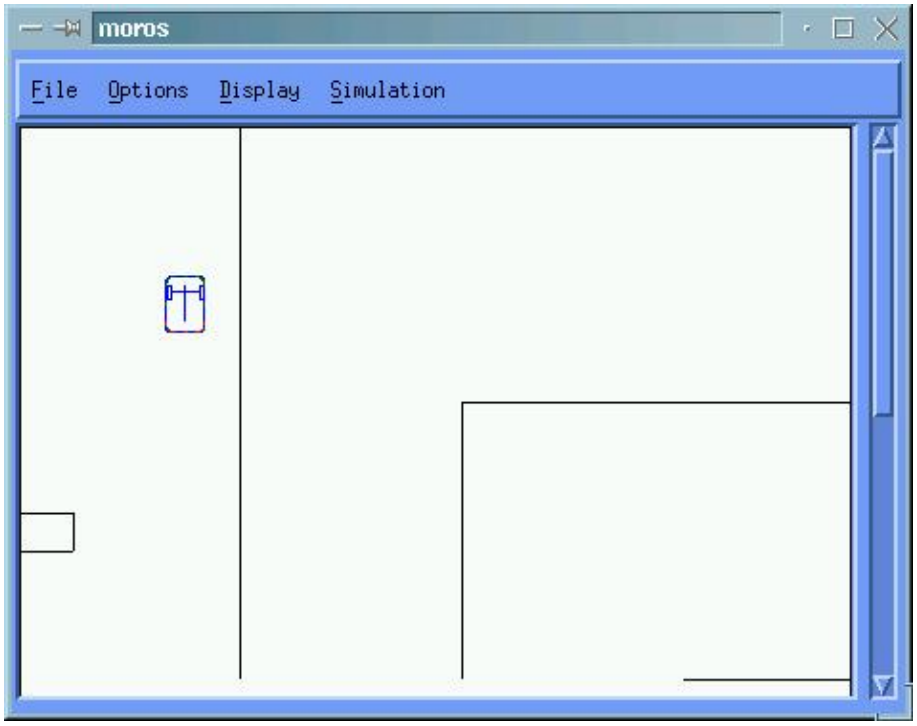

Figura 2.20: MoRoS: Simulador da Plataforma Robuter

executados por ordem. Por exemplo um movimento linear "eterno" é transformado num evento periódico de incremento da posição do robô. Este escalonador é suficientemente genérico para processar qualquer tipo de evento bastando para isso definir o evento e as funções de processamento do mesmo. Daí que a utilização de leituras dos sensores periódica possa ser utilizada, bastando, para isso, construir uma função que efectue a leitura dos sensores e transmita os valores para a aplicação remota.

A simulação das leituras dos sensores de distância é feita de forma simples, determinando a menor das distâncias entre o sensor e os obstáculos definidos no mapa do ambiente em que o segmento de recta que une o sensor e um ponto desses obstáculos se encontre forme um ângulo com a direcção de busca do sensor menor ou igual à abertura definida para o mesmo.

2.3.2 Interface com Aplicações

O interface entre o simulador e as diversas aplicações de robótica que o utilizam foi pensado de forma a manter o interface que as aplicações utilizam quando controlam directamente o robô. De facto, se a criação do simulador pretendia reduzir o esforço de desenvolvimento, possibilitando uma fase de testes iniciais seguros sem os perigos

de danificar a plataforma e disponibilizando uma plataforma virtual sempre que o robô esteja indisponível para utilizar, seria muito pouco conveniente se o interface com o simulador fosse diferente daquele apresentado pelo sistema de comunicações do robô uma vez que implicava um esforço adicional na criação das referidas aplicações. Assim, o interface foi desenvolvido utilizando mais uma vez "sockets" de Berkeley e mantendo a compatibilidade protocolar com as bibliotecas existentes para a utilização do sistema de comunicações referido na secção anterior.

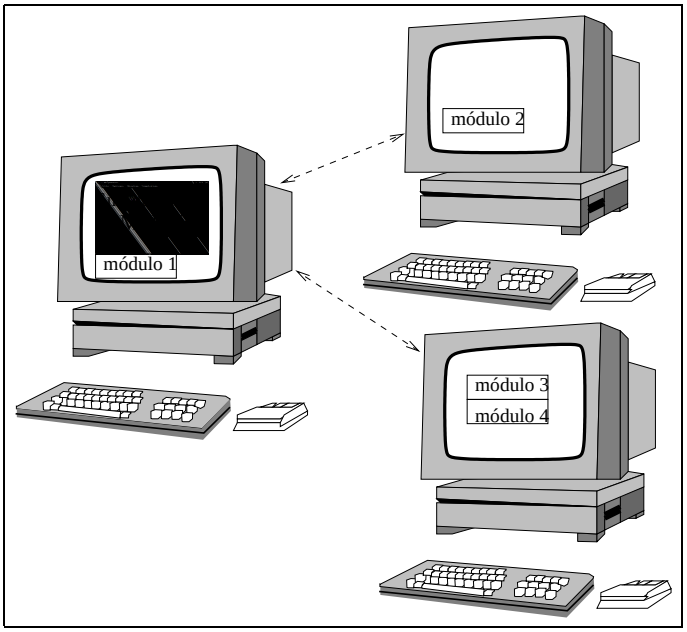

Figura 2.21: O simulador pode substituir o sistema de comunicações do robô, uma vez que o protocolo de comunicações com as aplicações é o mesmo do sistema de comunicações

Como se pode ver na figura 2.21, uma aplicação criada para controlar o robô, pode controlar o robô simulado na mesma máquina sem sofrer qualquer tipo de alteração ou re-compilação. Se o simulador for utilizado numa máquina diferente daquela em que são executadas as aplicações, basta definir a variável de ambiente **ROBOT** com o nome ou endereço da máquina onde é executado o simulador. A definição da variável de ambiente em UNIX pode ser efectuada utilizando o comando `export ROBOT=maq.dee.uc.pt` no caso de se utilizar a Bourne-Shell ou BASH como interprete de comandos, ou `setenv ROBOT maq.dee.uc.pt` no caso de se utilizar a C-Shell.

2.3.3 Resultados

Este simulador mostrou-se de grande utilidade para o desenvolvimento de aplicações de robótica móvel baseadas na plataforma Robuter, pois possibilitou o trabalho simultâneo de vários grupos de pessoas, apesar de existir apenas uma plataforma móvel. Foi possível ainda utilizado para testar em ambiente simulado a utilização de sensores de distância baseados em pares emissor-receptor de infravermelhos, tendo bastado para isso a criação de um ficheiro de configuração do robô onde as características de abertura e alcance desse sensores foram especificados.

Como desvantagem podemos apontar o facto de os sensores simulados serem sensores ideais, desprovidos por isso de ruído nas medições. No entanto isto poderá ser substancialmente melhorado, colocando um módulo gerador de ruído na função de leitura dos sensores.

Capítulo 3

Construção de Mapas

OS mapas ou modelos do ambiente são ingredientes indispensáveis em aplicações de robótica onde seja necessário efectuar planeamento de caminhos. Esses mapas são muitas vezes construídos previamente e fornecidos "a priori", no entanto uma das aplicações da robótica móvel consiste exactamente na exploração de ambientes desconhecidos e criação de modelos desses mesmos ambientes. Um dos primeiros problemas a resolver é o de definir qual a forma que deve ser utilizada para a descrição do referido ambiente. Para tentar encontrar uma resposta a este problema comecemos

por analisar os tipos de representações de ambientes que utilizamos no dia a dia e a que vamos designar genericamente por mapas.

Existem diversos tipos de mapas; sabemos que as cartas topográficas utilizam certo tipo de linhas para descrever estradas e outro tipo para unir zonas de igual altitude (curvas de nível), por outro lado quando se trata de descrever um edifício utilizamos muitas vezes plantas que, mais uma vez, não são mais do que descrições geométricas do espaço mas onde a simbologia é diferente da anterior, até porque o que interessa descrever também é diferente.

Claro que quando falamos em mapas, surge-nos imediatamente a ideia do mapa geométrico onde os objectos que queremos representar são descritos pela sua forma geométrica à custa de linhas, polígonos ou curvas. No entanto, este tipo de representação nem sempre é o mais adequado para certo tipo de aplicações. Um exemplo disso são as provas de *rally* onde o navegador utiliza uma descrição topológica da estrada conhecida como "*road book*" onde a informação contida é a das relações geométricas entre os elementos que constituem a estrada. Ex. "curva à direita, seguida de contra-curva à esquerda". A razão da utilização deste tipo de mapas tem a ver com a maior rapidez na sua interpretação e relacionamento com o que é visível tanto ao piloto como ao navegador. Um mapa é na maior parte das vezes um modelo que durante sua utilização é comparado com os dados sensoriais sobretudo quando o objectivo é a estimação da posição. Por isso um mapa deve conter informação que possa ser comparada com o ambiente através da utilização dos sensores. Para reforçar este ideia voltemos ao exemplo do "*road book*" para as provas de "*rally*", a informação "virar à direita a seguir à casa amarela" não serve de nada se o piloto sofrer de daltonismo. Neste caso deve-se utilizar outro tipo de informação como por exemplo "virar à direita a seguir à segunda casa".

Da mesma forma, quando pensamos em utilizar robôs dotados de sensores para construir mapas, temos que começar por decidir qual a forma de representação do ambiente que desejamos obter e que melhor se adequa à aplicação em vista e que é de mais fácil construção face ao tipo de sensores utilizado.

3.1 Grelha de Ocupação

Na secção 2.1.6 analisámos o comportamento do sensor de ultra-sons Polaroid e verificámos que a informação fornecida por este tipo de sensores sobre o meio ambiente é muito pouca e sujeita a erros, tendo no final da secção sido apresentado um modelo estocástico para o mesmo. Dado isto, percebemos facilmente que é necessário efectuar muitas medições com estes sensores a partir de diferentes localizações para se poder ter uma ideia sobre a estrutura do meio que rodeia o robô. Para integrar as várias medições e construir uma representação do meio ambiente de forma incremental, Alberto Elfes[10] utilizou uma estrutura a que chamou grelha de ocupação. A grelha

de ocupação consiste na divisão do espaço 2D (ou 3D consoante o caso), em porções iguais chamadas células e onde cada célula armazena uma estimação probabilística do seu estado (vazio ou ocupado). Para estimar o estado das células da grelha são feitas medições com os sensores de ultra-sons e os valores resultantes são interpretados de acordo com o modelo estocástico dos sensores.

3.1.1 Estimação da grelha de ocupação

A grelha de ocupação baseia-se na discretização do espaço (2D) em quadrículas (ou células). Assim, podemos representar uma porção do mundo por uma matriz, onde cada elemento corresponde a uma das quadrículas do espaço.

Os elementos da matriz, serão preenchidos com uma estimação probabilística do estado das quadrículas correspondentes, ou seja a probabilidade desta estar ocupada ou a probabilidade desta estar livre. A figura 3.1 representa um exemplo de uma grelha onde a probabilidade 0 (zero) de ocupação é representada pela cor branca, a probabilidade 1 (um) de ocupação) pela cor preta e as probabilidades intermédias por níveis de cinzento.

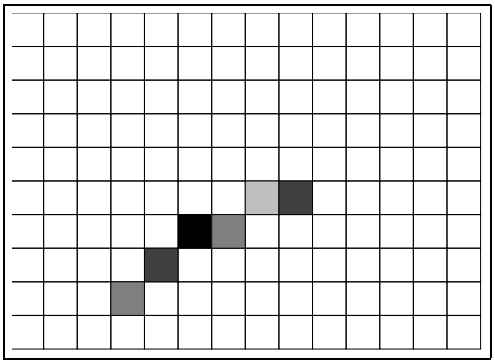

Figura 3.1: Exemplo de uma grelha de ocupação

Os estados ocupado e livre são mutuamente exclusivos e exaustivos o que significa que uma célula terá obrigatoriamente que se encontrar num destes estados. Por consequência a soma das probabilidades $P[s(C) = \mathfrak{va}] + P[s(C) = \mathfrak{oc}] = 1$, onde $s(C)$ representa o estado da célula C, $\mathfrak{va}$, $\mathfrak{oc}$ e $\mathfrak{de}$ representam os estados *vazio*, *ocupado* e *desconhecido*, respectivamente.

Para a obtenção de mapas do ambiente do robô com base nesta estrutura (a grelha) e em sensores de distância, teremos que utilizar em primeiro lugar um modelo estocástico do sensor definido por uma função densidade de probabilidade do tipo $p(r|z)$, em que r é a medida obtida e z a distância real ao objecto detectado. Em

segundo lugar, usando como base as medidas obtidas, é efectuada a estimação da grelha onde, por razões de simplicidade de realização e pelo facto de não existir uma relação de causalidade entre os estados das várias células, se assume que cada o estado de cada célula é independente do das restantes.

O que queremos obter é uma estimação para cada uma das células da probabilidade de estas estarem ocupadas

$$O(C_i) = P[s(C_i) = \mathfrak{oc}]$$

e da probabilidade de estarem vazias

$$V(C_i) = P[s(C_i) = \mathfrak{va}] = 1 - O(C_i).$$

Como a estimação é feita com base nas medidas dos sensores, o que pretendemos é obter

$$O(C_i|r) = P[s(C_i) = \mathfrak{oc}|r]$$

ou seja, a probabilidade da célula C_i estar ocupada dada uma medida r. O mesmo acontecendo para a estimação do facto de uma célula estar vazia.

Uma vez que temos os sensores caracterizados por uma função densidade de probabilidade que relaciona a probabilidade de obter uma medida com a existência de um alvo ($p(r|z)$), e que o que pretendemos saber é se numa determinada posição (correspondente a uma célula) existe um alvo dado uma medida ($P(C_i|r)$), isto sugere-nos a aplicação do teorema de Bayes

$$P[s(C_i) = \mathfrak{oc}|r] = \frac{P[r|s(C_i) = \mathfrak{oc}]P[s(C_i) = \mathfrak{oc}]}{\sum_{s(C_i)} P[r|s(C_i)]P[s(C_i)]}. \tag{3.1}$$

No entanto, e apesar da ideia ser bastante atractiva existem algumas dificuldades nesta abordagem que são analisadas a seguir. Para esta análise vamos considerar o caso uni-dimensional por ser de mais simples compreensão, podendo facilmente aplicar os resultados obtidos a casos 2D ou mesmo 3D.

Em primeiro lugar, os termos $P[r|s(C_i) = \mathfrak{oc}]$ não correspondem exactamente ao modelo dos sensores de distância $p(r|z)$, visto que estes relacionam-se com a detecção de um único alvo, o que não é o caso dos referidos termos pois podem existir células ocupadas entre o sensor e a célula a estimar. Relacionando então o modelo do sensor com a ocupação das células da grelhas teríamos

$$p(r|z) = p[r|s(C_i) = \mathfrak{oc} \wedge s(C_k) = \mathfrak{va}, k < i], \tag{3.2}$$

considerando que o sensor está situado sobre a célula C_0 e apontado no sentido das células de índice crescente.

Por consequência para a obtenção dos termos $p[r|s(C_i)]$ é necessário considerar todas as configurações do mundo vindo,

$$P[r|s(C_i) = \mathfrak{oc}] = \sum_{\{G_{s(C_i)}\}} \left(P\left[r|s(C_i) = \mathfrak{oc}, G_{s(C_i)}\right] P\left[G_{s(C_i)}|s(C_i) = \mathfrak{oc}\right]\right) \tag{3.3}$$

onde $G_{s(C_I)}$ é uma das possíveis configurações do mundo tendo fixado o estado da célula $s(C_i) = \mathfrak{oc}$, da mesma forma $\{G_{s(C_i)}\}$ representa o conjunto de todas as configurações possíveis do mundo com a mesma restrição.

Os termos $P[r|s(C_i) = \mathfrak{oc}, G_{s(C_i)}]$ podem ser pré-calculados, utilizando o modelo dos sensores, e armazenados numa tabela. Por seu lado as probabilidades das configurações $P[G_{s_(C_i)}|s(C_i) = \mathfrak{oc}]$ são determinadas a partir dos valores previamente estimados dos estados das células como se pode ver a seguir.

Determinação das probabilidades das configurações

Se $\{G_{s(C_i)}\}$ representa o conjunto de todas as configurações da grelhas com a restrição de que a célula C_i está no estado ocupado (ou vazio), $G_{s(C_i)}$ representa um elemento desse conjunto. Assim se a grelha for composta por n células e dado que os estados das células são independentes entre si

$$P[G_{s(C_i)}|s(C_i) = \mathfrak{oc}] = \prod_{j=1, j\neq i}^{n} P[s(C_j) = s_j] \tag{3.4}$$

onde s_j corresponde ao estado da célula C_j na configuração considerada e os valores $P[s(C_j) = s_j]$ utilizados na equação 3.4 são, como já foi referido, obtidos a partir dos valores previamente armazenados nas células da grelha. Assim, se partirmos de uma situação em que não sabendo nada sobre o espaço representado pela grelha iniciamos as células da grelha com $P[C_i = \mathfrak{oc}] = P[C_i = Va] = 1/2$, obtemos o valor esperado para a probabilidade de uma configuração específica igual a

$$P[G_{s(C_i)}|s(C_i) = \mathfrak{oc}] = \prod_{j=1, j\neq i}^{n} \frac{1}{2} = \frac{1}{2^{n-1}}.$$

3.1.2 Integração na Grelha de Novas Medidas

O que se pretende é, como se sabe, utilizar os sensores de ultra-sons (ou eventualmente outro tipo de sensores) para estimar o estado das células que compõem a grelha. No entanto essa estimação deverá ser efectuada, de forma incremental, à medida que as medidas são obtidas. Para isso basta-nos reescrever a expressão 3.1 de forma a incluir

os valores previamente estimados o que resulta na formulação recursiva da formula de Bayes (equação 3.5) aplicada à estimação das células da grelha.

$$P[s(C_i) = \mathfrak{oc}|\{r\}_t] = \frac{P[r_t|s(C_i) = \mathfrak{oc}]P[s(C_i) = \mathfrak{oc}|\{r\}_{t-1}]}{\sum_{s(C_i)} P[r_t|s(C_i)]P[s(C_i)|\{r\}_{t-1}]} \tag{3.5}$$

O valor obtido para $P[s(C_i) = \mathfrak{oc}|\{r\}_t]$ é armazenado na célula C_i e é usado para o cálculo das estimativas futuras. Os termos $P[s(C_i)|\{r\}_{t-1}]$ são obtidos directamente da grelha e resultaram de observações prévias. O termo $P[r_t|s(C_i) = \mathfrak{oc}]$, que define a probabilidade de acontecer a medida obtida, e $P[r_t|s(C_i)]$ são obtidos a partir da expressão 3.3, sendo r_t o resultado da última observação efectuada pelo sensor.

3.1.3 Fusão das grelhas e decisão quanto ao estado das células

O descrito até aqui permite-nos estimar as probabilidades das células da grelha estarem ocupadas ou vazias, no entanto há casos em que é necessário, com base nestes valores, decidir se se considera que as células estão ocupadas ou vazias. Uma forma imediata de efectuar essa decisão poderá baseada nas seguintes regras:

$$\begin{cases} s(C_i) = \mathfrak{oc} & se \quad P(s(C_i) = \mathfrak{oc}) > P(s(C_i) = \mathfrak{va}) + \delta \\ s(C_i) = \mathfrak{va} & se \quad P(s(C_i) = \mathfrak{oc}) < P(s(C_i) = \mathfrak{va}) - \delta \\ s(C_i) = \mathfrak{de} & se \quad P(s(C_i) = \mathfrak{va}) - \delta < P(s(C_i) = \mathfrak{oc}) < P(s(C_i) = \mathfrak{va}) + \delta. \end{cases} \tag{3.6}$$

O que corresponde a optar pelo estado cuja probabilidade estimada é superior. Note-se que o valor de δ estabelece uma banda de incerteza e se fizermos $\delta = 0$ reduzimos a incerteza ao caso em que a probabilidade de uma célula estar ocupada é igual à probabilidade de esta estar vazia. Este valor de δ pode ser ajustado como um factor de segurança em algumas aplicações desde que considerem as células de estado desconhecido como possivelmente ocupadas.

3.1.4 Sobre a utilização do método

Para utilizar este método é necessário começar por definir as dimensões das células da grelha. Esta escolha representa um compromisso entre a resolução dos obstáculos e o peso computacional pois, se por por exemplo no nosso caso, seria desejável que as células tivessem dimensões da ordem de $1cm^2$, por outro lado a escolha de células muito pequenas implica a utilização de uma grelha com um número de elementos muito grande. A utilização de um grande número de células pode-se tornar inviável do ponto de vista de armazenamento assim como do ponto de vista computacional, uma vez que quanto maior for o número de elementos da grelha maior é o número

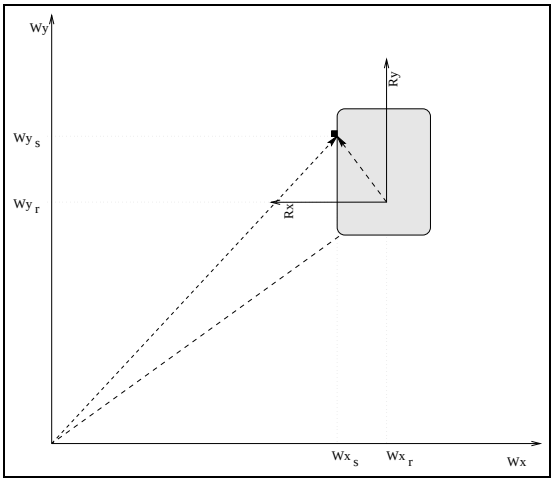

Figura 3.2: Obtenção das coordenadas de um sensor no mundo

de operações a efectuar por cada operação de estimação. As dimensões das células dependem também da resolução espacial dos sensores utilizados, pois se esta resolução for baixa, a resolução obtida para a descrição do ambiente não é melhorada pelo uso de células muito pequenas.

Por estas razões o tamanho escolhido para as células na nossa implementação foi de $10cm \times 10cm$, pois representa uma densidade de células relativamente baixa o que permite um funcionamento em tempo-real com o sistema computacional e processadores utilizados. Torna-se evidente que com estas dimensões para as células não é possível efectuar certo tipo de manobras que exijam muita precisão como seja a transposição de portas ou a navegação em corredores "apertados". Por outro lado os sensores de ultra-sons utilizados não permitem a medição segura de distâncias inferiores a $30cm$, pelo que seria necessário utilizar outro tipo de sensores para este tipo de aplicações.

3.1.5 Localização dos sensores

Obtida uma medida r a partir de um determinado sensor, é necessário antes de mais saber qual a posição que esse sensor ocupava no espaço no instante da medição. A determinação da posição do sensor é efectuada de forma simples (figura 3.2 uma vez que é conhecida a sua localização sobre o robô e a localização do robô no mundo.

Assim se ${}^{R}\mathbf{x}$ for a posição do sensor no referencial do robô, então

$$ {}^{W}\mathbf{x} = {}^{W}T_{R}.{}^{R}\mathbf{x}. \tag{3.7} $$

onde ${}^{W}\mathbf{x}$ é a posição do sensor no mundo e ${}^{W}T_R$ é a matriz de transformação dada por

$$T = \begin{bmatrix} 1 & 0 & 0 & {}^{W}x \\ 0 & 1 & 0 & {}^{W}y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos^{W}\theta & -\sin^{W}\theta & 0 & 0 \\ \sin^{W}\theta & \cos^{W}\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

Como se pode ver na expressão 3.8 a matriz de transformação ${}^{W}T_R$ corresponde a uma rotação em torno do eixo dos zz seguida de uma translação no plano xy. O vector $[{}^{W}x, {}^{W}y, 0, 1]$ corresponde às coordenadas homogéneas do robô no referencial do mundo e ${}^{W}\theta$ à sua orientação relativamente ao eixo dos xx deste mesmo referencial. Assim, sendo ${}^{R}\theta_S$ a orientação do sensor em relação ao referencial do robô e que corresponde à direcção de detecção do mesmo, então a orientação do sensor em relação ao referencial do mundo é dada por

$${}^{W}\theta_S = {}^{R}\theta_S + {}^{W}\theta_R, \tag{3.9}$$

em que ${}^{W}\theta_R$ representa a orientação do robô relativamente a um referencial global.

3.1.6 Pré-processamento das medições obtidas

As medidas dos sensores de ultra-sons estão sujeitas a diversos tipos de erros. Alguns erros devem-se a variações de temperatura ou ruído gaussiano e podem ser incluídos no modelo ou compensados. Existem no entanto outras formas de erro que poderão ser devidas a "*cross-talk*" quando um sensor recebe um eco proveniente de outro sensor ou efeitos especulares. No caso dos erros devidos a "*cross talk* o resultado é a obtenção de uma medida inferior à medida real a que se encontra o alvo, isto porque antes de chegar o eco reflectido nesse alvo é recebido o eco do sinal emitido por outro sensor (ver figura 3.3).

Os efeitos especulares das superfícies podem contribuir para os erros anteriores e podem ainda produzir medições de valor superior à distância aos alvos por o sinal enviado ser reflectido entre diversas superfícies antes de voltar ao sensor (ver figura 3.4).

Seria desejável eliminar as medidas que contivessem este tipo de erros, no entanto estas situações são difíceis de detectar. Seria necessário utilizar um mapa do ambiente e conhecer a localização do sensor com exactidão, mas tal mapa não pode existir pois é o que pretendemos construir. Por outro lado, e ao contrário do que acontece com o ruído gaussiano, o uso de observações múltiplas a partir do mesmo ponto não iria resolver o problema uma vez que a configuração geométrica se mantém e com ela as condições para a ocorrência deste tipo de erros. Resta-nos a solução de

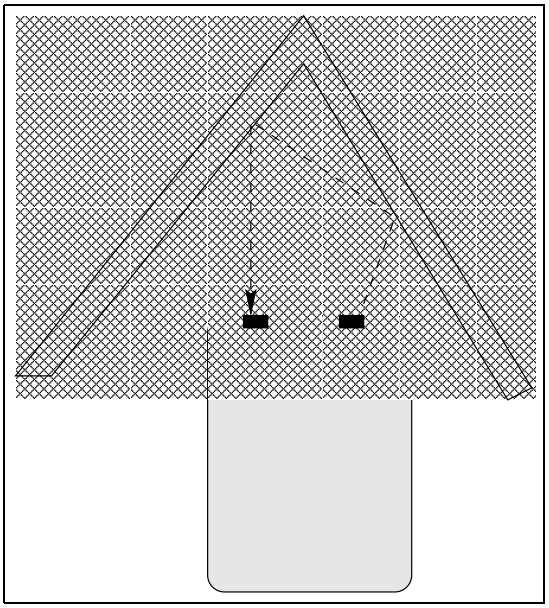

Figura 3.3: Exemplo de "Cross Talk" entre dois sensores

eliminar medidas que não sejam consideradas úteis e que no nosso caso são as medidas superiores a 3 metros e inferiores a 30cm.

Podemos ainda utilizar um filtro do tipo passa baixo mas se o robô se deslocar com uma velocidade considerável de forma a não ser possível obter um conjunto considerável de leituras a partir de pontos próximos do espaço, poderemos perder características importantes como portas ou outro tipo de características do ambiente.

3.1.7 Análise do Método

Este método apresenta a possibilidade de integrar de forma incremental os valores obtidos dos sensores à medida que estes que estes ficam disponíveis. Por outro lado, este processo de estimação permite reduzir os efeitos dos erros presentes nas medições efectuadas, desde que estes erros apresentem uma distribuição centrada em torno de um valor médio nulo.

A principal desvantagem deste método é a carga computacional que dificulta a sua utilização "on-line", como se pode ver a seguir.

Um dos passos do método para a estimação do estado de uma célula dada uma nova medida de SONAR, a estimação da probabilidade de obter essa medida (rever expressões (3.3) e (3.4)). Esta última estimação introduz um peso computacional bastante elevado, pois para um mapa com 10 por 10 células teremos que para cada uma das $2^{10\times10-1}$ configurações possíveis da grelha, efectuar 99 multiplicações cujo resultado será multiplicado pelo resultado da estimação da probabilidade de obter uma

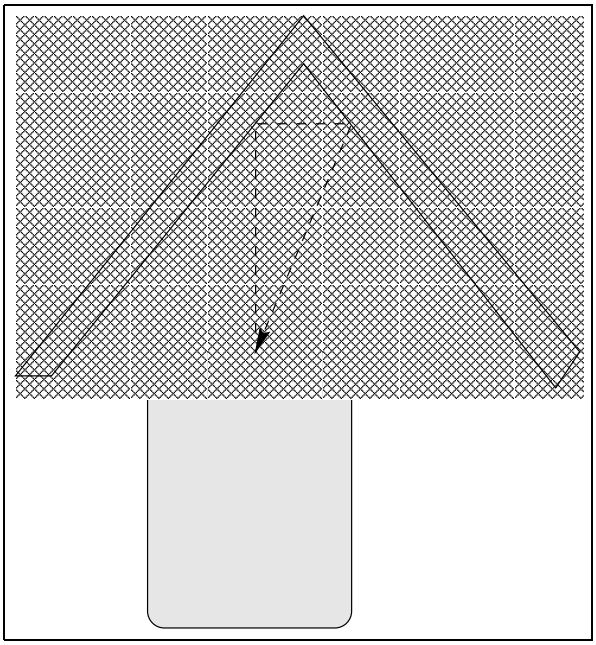

Figura 3.4: Exemplo da obtenção de uma medida superior à distância ao alvo devido a reflexões múltiplas

determinada medida pelo sensor nessa mesma configuração. Ou seja, após receber uma medida teremos de efectuar mais de,

$$2^{99} \times (99 + 1) = 6338253001141147007483516026880 0 \text{ multiplicações.}$$

Este valor mostra, só por si, que o método não se presta a uma utilização em tempo-real.

3.2 Método Alternativo

Dadas as limitações encontradas na utilização do método da grelha de ocupação em tempo real foi necessário encontrar outras formas que permitissem efectuar o mapeamento do ambiente ou a detecção de obstáculos não conhecidos *a priori*, e que cujo peso computacional não seja proibitivo para aplicações em tempo-real. Por esta razão, e tomando como base a divisão do espaço em células criando uma grelha mas no qual cada célula, em lugar de conter uma estimação probabilística da sua ocupação, contém a acumulação das observações de acordo com o modelo estocástico dos sensores. Como será mostrado mais adiante, este método permite de uma forma mais rápida que o método da grelha de ocupação efectuar a integração da informação

dos sensores permitindo o seu uso em tempo-real tendo sido utilizado para a detecção de obstáculos fixos não conhecidos previamente.

3.2.1 Análise do método

Para perceber o funcionamento do método vamos começar por considerar um sensor ideal e analisar os dados fornecidos por este e em seguida utilizar o mesmo tipo de análise para os sensores de ultra-sons utilizados.

Consideremos então um caso uni-dimensional onde um sensor ideal dá a medida exacta da distância a um alvo colocado à sua frente. O comportamento deste sensor pode ser caracterizado pela seguinte densidade de probabilidade

$$p(r|z) = \delta(z - r). \tag{3.10}$$

onde a função δ corresponde à função delta de Kronecker. A expressão para a probabilidade de existir um alvo a uma distância z dada uma medida r é intuitiva visto que o sensor é determinístico e pode ser dada por

$$P(z|r) = \begin{cases} 0 & \text{se } z < r \\ 1 & \text{se } z = r \\ 1/2 & \text{se } z > r \end{cases} \tag{3.11}$$

que de uma forma simplificada se pode traduzir pela seguinte afirmação: se um sensor ideal dá uma medida de distância r, então a probabilidade de existir um objecto a uma distância r do sensor é 1; por outro lado a probabilidade de existir objecto a uma distância inferior é 0 e a de existir a uma distância superior é de 50%, dado que a medida efectuada não fornece qualquer informação sobre a zona que se encontra além do objecto detectado.

Para o modelo discreto, se dividirmos o espaço em células, podemos escrever uma expressão que relaciona a probabilidade de cada uma das células estar ocupada dada a medida obtida, e que é dada por

$$P\left(s(C_i) = \mathfrak{oc}|r\right) = \begin{cases} 0 & \text{se } z < r \wedge z \in C_i \\ 1 & \text{se } z, r \in C_i \\ 1/2 & \text{se } z > r \wedge z \in C_i \end{cases} \tag{3.12}$$

Neste caso, a célula correspondente à posição r contém certamente o objecto detectado, as precedentes estão vazias e sobre as restantes não há informação.

Consideremos agora um caso a duas dimensões, onde um sensor omnidireccionjal, dá a medida (exacta) da distância ao objecto mais próximo (figura 3.5) que se encontre em seu redor. Neste caso, temos que dada uma medida r, a probabilidade de o objecto detectado estar sobre uma circunferência de de raio r é 1 e a densidade de probabilidade sobre a circunferência é $\frac{1}{2\pi r}$.

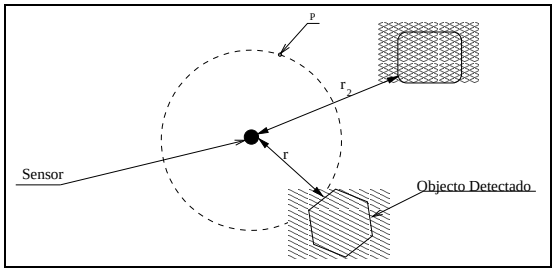

Figura 3.5: Sensor ideal com capacidade de detecção em qualquer direcção

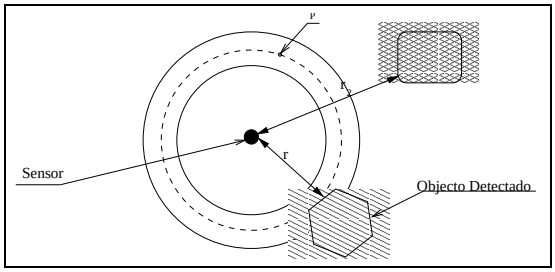

Figura 3.6: Sensor com uma distribuição de erros uniforme e com capacidade de detecção em qualquer direcção

Se se considerar agora que as medidas deste sensor são corrompidas com ruído de distribuição uniforme, então a função distribuição de probabilidade correspondente será:

$$P(z|r) = \begin{cases} 0 & \text{se } z < r - \delta \\ \frac{1}{\pi(r+\delta)^2 - \pi(r-\delta)^2} & \text{se } r - \delta < z < r + \delta \\ 1/2 & \text{se } z > r + \delta \end{cases} \tag{3.13}$$

Neste caso, dada uma medida, sabe-se que o ponto detectado estará localizado no interior de um anel centrado no sensor e com raio interior $r - \delta$ e raio exterior $r + \delta$, como se pode ver na figura 3.6.

Na secção 2.1.6 foi apresentado um modelo uni-dimensional para os sensores de ultra-sons dado por uma função densidade de probabilidade que relaciona a probabilidade de obter uma medida r sabendo que existe um alvo a uma distância z (eq: 2.18). O modelo bidimensional foi obtido adicionando um grau de incerteza que advêm do facto de o alvo poder estar situado fora da recta que passa pelo centro do sensor e é normal à superfície deste.

No entanto o que pretendemos obter é a probabilidade de existir um obstáculo a uma distância z e orientação θ relativamente ao sensor dado que este devolveu uma medida r, e este problema já foi largamente debatido na secção anterior. Como encontrar então uma forma rápida de resolver este problema?

3.2.2 Simplificação

Para simplificar este problema teremos que utilizar um conjunto de restrições e simplificações que, de forma computacionalmente expedita, permitem alcançar desempenhos temporais bons sem sacrificar em demasia a previsão dos mapas. Os métodos propostos vão ser analisados de seguida.

Observações consecutivas obtidas com o sensor estático permite apenas filtrar os erros existentes nas medidas, não fornecendo novas informações que permitam identificar a forma do obstáculo.

As medidas superiores a 2 metros são ignoradas uma vez que tendo os sensores uma abertura que ronda os 30^0, o comprimento do arco para uma medida de 2 metros é de 1,047m o que representa um grau de incerteza bastante elevado sobre a localização do obstáculo. Por outro lado, com esta restrição, os erros provocados por reflexões múltiplas são praticamente eliminados uma vez que nestes casos as medidas obtidas apresentam normalmente valores muito maiores que o máximo considerado. Uma outra vantagem desta limitação imposta, surge na redução do tempo de processamento necessário uma vez que o número de células afectadas é relativamente pequeno para as medidas de distância consideradas. Com base nesta restrição, podemos substituir na expressão (2.17), que relaciona a variância com a distância ao alvo, o valor de r pelo valor máximo considerado, uma vez que segundo esta expressão existe uma relação directa entre o valor de r e σ, chegando-se a um valor máximo para o desvio padrão de $1.24cm$.

Como as dimensões escolhidas para as células da grelha ($10cm \times 10cm$) são aproximadamente 8 vezes maiores que o desvio padrão das medidas dos sensores de ultrasons, optou-se por simplificar o processo de estimação dos valores das células da seguinte forma: Se, para um determinado valor de distância obtido r, existe um conjunto de células que contém todas as possíveis localizações do alvo que produziu este valor, então a probabilidade de neste conjunto de células existir o alvo é de 100% e consequentemente a probabilidade de cada célula conter o alvo será de $1/n$ em que n é o número de células candidatas. Utilizando este princípio, a actualização das células da grelha face a uma nova medida consiste identificar as células sobre as quais se estende um arco de4 ângulo θ equivalente à abertura do sensor e em incrementar as células que correspondem à posição do arco do valor de $1/n$, em que n é o número de células abrangidas.

Com este método, embora de cada medida se possa apenas extrair a existência de

uma zona onde o objecto detectado poderá estar, a integração ao longo do tempo de leituras sucessivas vai realçar as células que "realmente" estão ocupadas.

Este método é utilizado por diversos autores devido à sua simplicidade de implementação e baixos custos computacionais que permitem a sua utilização em tempo real. Verifica-se facilmente que a sua utilização requer a existência de actividade por parte dos sensores, que é explicada facilmente da seguinte forma: Se de determinado ponto do espaço verificarmos a existência de um objecto a uma distância R, observações sucessivas efectuadas do mesmo ponto não vão introduzir novas informações sobre o estado das células e apenas poderão ser utilizadas para confirmar a primeira destas observações. Uma vez que de cada medida se pode apenas identificar um arco com as possíveis localizações e respectivas probabilidades é necessário utilizando as medidas seguintes identificar quais as zonas que realmente estão ocupadas e quais as que foram marcadas como possivelmente ocupadas devido a falta de informação.

A questão que se coloca neste momento é a seguinte: como detectar quais as células que estão livres? Dado que as medidas obtidas do sensor de ultra-sons correspondem à distância deste ao objecto mais próximo, logo não poderá existir qualquer objecto mais próximo. Infelizmente esta afirmação nem sempre é verdadeira, pois dado o comportamento especular de muitas superfícies, se o ângulo entre a superfície reflectora e a normal à superfície do sensor for superior a um determinado valor crítico a onda não é reflectida de volta para o sensor. Como consequência, nestes casos, um objecto colocado em frente do sensor torna-se "invisível". Este problema é ignorado pela maior parte dos autores, por um lado devido a facto de ser muito difícil identificar a sua ocorrência e por outro lado por se registar apenas com alguns tipos de objectos em condições especiais.

No nosso caso optámos por considerar que as células que se interpõem entre o sensor e a zona supostamente ocupada têm uma probabilidade de 2/3 de estar vazias. Isto justifica-se pelo efeito especular das superfícies da seguinte forma. Consideremos uma superfície plana colocada entre o sensor e o alvo. Se o ângulo que essa superfície faz com a perpendicular ao eixo acústico do sensor se situar entre $[-30^0, 30^0]$, as ondas reflectidas nessa superfície são devolvidas ao sensor o que leva a que se obtenha uma medida correspondente à existência dessa superfície e não a medida r obtida. Caso o referido ângulo se situe entre $[-90^0, -30^0[$ ou entre $]30^0, 90^0]$, esta superfície será invisível para o sensor. A figura 3.7 mostra um gráfico onde se pode ver a sombreado as orientações que uma superfície plana pode tomar produzindo uma reflexão dirigida ao sensor e a branco as orientações que a tornam invisível para o mesmo sensor. Nesta figura é ainda visível que as áreas varridas pelas possíveis orientações da superfície correspondem a 1/3 para a zona visível (ou que produz eco) e 2/3 para a zona invisível.

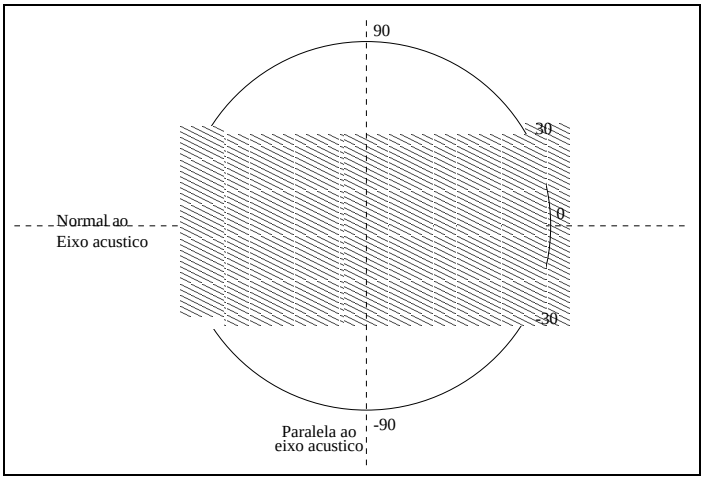

Figura 3.7: A zona sombreada representa o conjunto de orientações que uma superfície plana pode tomar produzindo uma medida num sensor de ultra-sons.

3.2.3 Construção da grelha de acumulação

A grelha é actualizada sempre que se obtém uma medida de distância. Com base nessa medida e na posição do sensor no mundo são determinadas as células onde há a probabilidade de existir o alvo, bem como as células pertencentes ao sector circular que fica entre o sensor e o referido arco. Uma vez determinados estes conjuntos de células, os respectivos valores são incrementados ou decrementados consoante pertençam ao arco ou ao sector circular. Os valores que são somados ou subtraídos às células são estabelecidos de acordo com o que foi apresentado na secção anterior. Os acumuladores das células são limitados de forma a que o seu valor não possa exceder um determinado limite superior ou descer abaixo de um limite inferior.

A decisão sobre o estado das células é feita através da utilização de uma função histerese, estabelecendo dois limites. Um limite superior que sempre que o valor acumulado sobe acima deste o estado passa ao estado ocupado e um limite inferior que define o valor abaixo do qual é feita a transição ocupado-livre.

3.2.4 Utilização para a detecção de obstáculos

Este método pode ser utilizado para detectar obstáculos não conhecidos *a priori*. Para este tipo de utilização, é necessário poder extrair a informação sobre a posição e forma dos obstáculos e enviá-lo a um módulo que controle a navegação do robô. Claro que a descrição pormenorizada dos obstáculos nem sempre é possível pois isso requereria que o sistema sensorial fizesse uma pesquisa em torno de cada obstáculo detectado. Em primeiro lugar é necessário distinguir entre obstáculos conhecidos, por

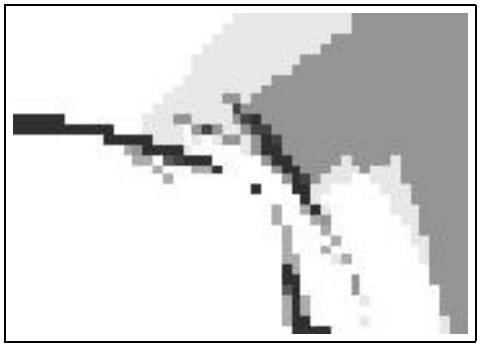

Figura 3.8: Exemplo de resultado obtido após a integração de algumas leituras

estarem descritos no mapa do ambiente e os que não são conhecidos. Os primeiros não interessam pois já foram tomados em conta no planeamento do trajecto a ser efectuado pelo robô, os segundos são muito importantes pois permitem ao sistema de navegação evitar as colisões e se possível ultrapassá-los.

A solução encontrada passa por integrar na grelha os obstáculos descritos no mapa. Esses obstáculos são integrados marcando como ocupadas as células correspondentes, no entanto estas células não podem ser alteradas como consequência de leituras dos sensores. Em segundo lugar as células adjacentes mesmo que fiquem marcadas como ocupadas não são consideradas como obstáculos pois podem ter sido marcadas como resultado de erros nas medidas dos sensores ou na localização do robô. Todas as outras células ocupadas vão fazendo parte de obstáculos que são definidos por uma primitiva geométrica que as envolva. A forma mais adequada para efectuar este agrupamento de células ocupadas em obstáculos é através da determinação de um polígono que as envolva. No nosso caso, por simplicidade de implementação, esse polígono foi reduzido a um rectângulo, sendo as coordenadas de dois vértices opostos desse rectângulo passadas ao módulo de navegação.

3.2.5 Resultados

A figura 3.2.5 apresenta um exemplo de utilização de uma grelha de acumulação onde as zonas livres correspondem às células brancas, as ocupadas estão marcadas com a cor preta e os níveis de cinzento correspondem a zonas sobre as quais ainda não há informação suficiente, sendo no entanto a sua tonalidade um indicador da tendência das células correspondentes.

A figura 3.9 mostra os níveis acumulados durante a execução de um trajecto. Note-se que o mapa previamente conhecido está devidamente marcado através das linhas rectas exteriores.

É possível verificar que a detecção da parede lateral e que corresponde ao mapa, no entanto há um desfasamento devido a um erro no estabelecimento da posição inicial do robô.

A figura 3.10 corresponde a uma aplicação onde é possível ver a evolução do robô bem como as leituras de ultra-sons. Esta aplicação foi modificada para receber a descrição dos obstáculos e desenhá-los sobre o mapa. Assim, nesta figura podemos ver o mapa inicial e que corresponde às linhas exteriores, à direita o robô e os arcos correspondentes às medidas de ultra-sons e um conjunto de rectângulos que correspondem a obstáculos detectados.

Os resultados obtidos mostraram que este método pode ser utilizado em tempo-real devido ao facto de a estimação ser feita através da acumulação simples das observações. Como qualquer método destinado a fazer mapeamento (não topológico) necessita de um sistema de localização eficiente pois quaisquer erros que sejam acumulados na localização do robô vão distorcer completamente o mapa obtido, ou a localização dos obstáculos.

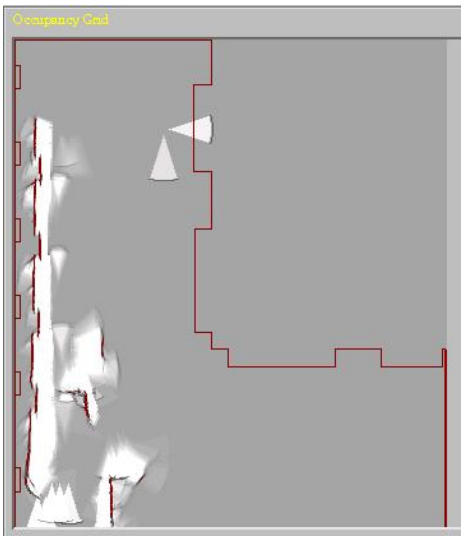

Figura 3.9: Grelha de ocupação obtida, no laboratório de Robótica Móvel do ISR Polo de Lisboa, com um mapa do ambiente sobreposto

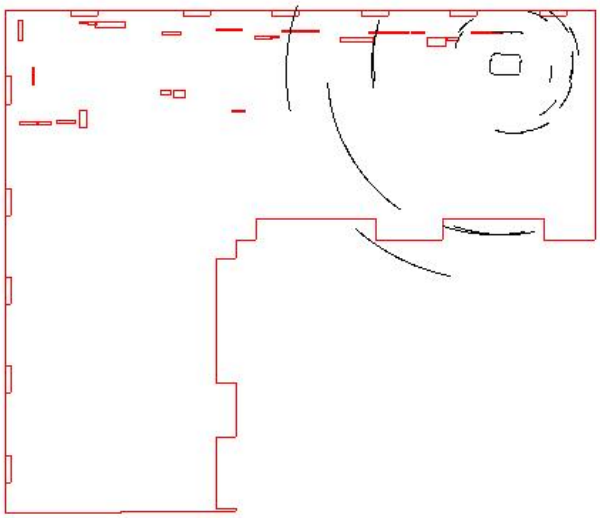

Figura 3.10: Envolventes dos obstáculos recebidos por uma aplicação que mostra as últimas leituras dos sensores de ultra-sons sobre um mapa do ambiente

Capítulo 4

Navegação Autónoma

CONSEGUIR desenvolver métodos de navegação autónoma é um dos principais objectivos da investigação actual em robótica móvel, uma vez que sem esta capacidade as plataformas móveis vêem a sua utilização limitada a sistemas guiados ou telecomandados. Os sistemas guiados, onde as plataformas móveis seguem um

fio ou marcas no pavimento, têm sido bastante utilizados em linhas de produção para o transporte de peças ou produtos. No entanto, dado que os trajectos são pré-fixados através da instalação dos guias referidos, o aparecimento de um obstáculo na trajectória constitui um problema uma vez que apesar de este poder ser detectado pela plataforma (com o auxílio de sensores apropriados), esta não possui, normalmente, capacidade de o ultrapassar. Uma outra desvantagem dos sistemas filo-guiados tem a ver com a constante alteração da disposição das linhas de montagem em muitas das fábricas actuais. A utilização de plataformas filo-guiadas requeria que os "guias" fossem alterados sempre que a disposição da linha de montagem fosse alterada o que é em muitos casos impossível visto que estes se encontram enterrados no pavimento.

No que diz respeito aos sistemas telecomandados tem-se verificado a necessidade de os dotar de um certo grau de autonomia por forma a possibilitar a sua utilização quando existem grandes atrasos nas comunicações ou para reduzir o "stress" do operador. Um exemplo dessa necessidade foi a recente missão a Marte da plataforma Sojourner que se a mesma não dispusesse de um certo grau de autonomia seria completamente impossível a sua utilização dado que as comunicações apresentavam um atraso bastante elevado (20 minutos)[1] e uma largura de banda extremamente reduzida (de 2400 bits por segundo).

Apesar de as tarefas de navegação parecerem simples, dada a facilidade com que os seres humanos as executam, a criação de sistemas capazes de as executar está longe de ser um problema trivial devido à complexidade das operações envolvidas.

Para melhor perceber quais os mecanismos requeridos vamos analisar as diferentes fases necessárias para conduzir uma plataforma entre dois pontos arbitrários: A primeira fase é a identificação das posições de partida e destino. Feito isto é necessário estabelecer um caminho que una as duas posições. Para o planeamento de uma trajectória que conduza o veículo até à posição objectivo é necessário existir um "mapa" onde possam ser identificadas as zonas livres (sem obstruções) e onde o veiculo caiba.

Definido o caminho é necessário converter os diversos segmentos que o compõem em primitivas de comando da plataforma e executá-las de forma sequencial.

Até aqui não temos mais do que os sistemas guiados uma vez que o planeamento das trajectórias também antecede a instalação dos fios ou das pinturas de guiamento, e as saídas dos sensores que detectam os desvios em relação ao guia são utilizadas para gerar os comandos para o sistema de tracção.

Basicamente o que se pretende introduzir de novo é a capacidade de reacção à presença de qualquer tipo de obstáculos. Para satisfazer este requisito é necessário utilizar sistemas sensoriais que permitam não só detectar a presença dos obstáculos mas também descrevê-los pelo menos parcialmente por forma a permitir a ultrapassagem dos mesmos.

[1] De acordo com informações obtidas em http://mars.jpl.nasa.gov/MPF/rover/faqs_sojourner.html

Assim, a execução de movimentos pelos robôs moveis podem ser divididos em três tipos:

- **Planeados** - Os deslocamentos a executar pelo robô foram previamente estabelecidos com base numa configuração conhecida do ambiente.
- **Reactivos** - Por forma a não colidir com eventuais obstáculos, os deslocamentos executados pelo robô dependem da informação sobre o ambiente que o rodeia, adquirida através de sistemas sensoriais.
- **Planeados-Reactivos** - Os movimentos planeados são adaptados ao ambiente através da utilização de informação sensorial.

A secção seguinte descreve uma parte do trabalho desenvolvido na qual foram explorados alguns métodos de navegação baseados essencialmente no conceito de campo de potencial artificial.

4.1 Campos de Potencial

A utilização de campos de potencial artificiais foi sugerida por O.Khatib[27]. Este método baseia-se na colocação de cargas repulsivas, geradoras de um campo, nos obstáculos e uma carga atractiva na posição objectivo. O robô escolhe a trajectória do gradiente do potencial gerado em direcção ao seu mínimo. O controlo da plataforma pode ser feito tanto a nível cinemático como dinâmico usando o equivalente à força aplicada no robô por este campo para induzir valores de velocidade e aceleração. Este método tem sido largamente utilizado devido à sua simplicidade e capacidade de reacção. No entanto a sua maior falha é a possibilidade de obter mínimos locais para determinadas configurações de obstáculos e posição objectivo.

4.1.1 Método Clássico

Comecemos por analisar o método dos campos de potencial clássico [27]. Tal como já foi referido, a posição destino (ou objectivo) gera um campo de potencial atractivo e os obstáculos por sua vez geram um campo potencial repulsivo. Como se assume que o campo gerado é função linear das cargas, podemos sobrepor linearmente os campos de mais de uma carga. Começando por analisar o caso onde existe apenas um único obstáculo, o potencial resultante sendo a soma destes dois é dado por:

$$U(\mathbf{x}) = U_g(\mathbf{x}) + U_o(\mathbf{x}) \tag{4.1}$$

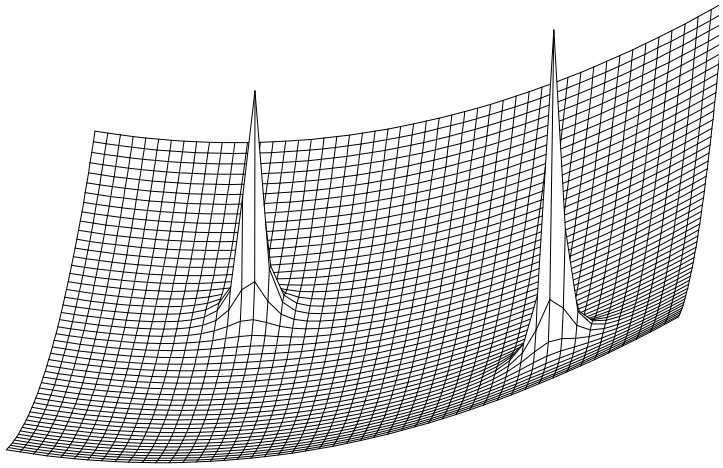

Figura 4.1: Distribuição do potencial gerado por uma posição objectivo (canto inferior esquerdo) e dois obstáculos

onde $\mathbf{x}$ é a posição do robô, $U(\mathbf{x})$ é o potencial resultante, $U_g(\mathbf{x})$ é o potencial atractivo produzido pelo objectivo em x, $U_o(\mathbf{x})$ é o potencial repulsivo induzido pelo obstáculo em $\mathbf{x}$.

A força resultante $\mathbf{f_r}$ é dada por:

$$\mathbf{f_r} = \mathbf{f_g} + \mathbf{f_o} \tag{4.2}$$

em que

$$\mathbf{f_g} = -\nabla U_g(\mathbf{x}) \tag{4.3}$$
$$\mathbf{f_o} = -\nabla U_o(\mathbf{x}) \tag{4.4}$$

$\mathbf{f_g}$ é uma força atractiva que guia o robô até ao objectivo e $\mathbf{f_o}$ é uma força repulsiva produzida pela superfície do obstáculo.

A função que expressa o potencial atractivo num ponto $\mathbf{x}$ é definido por

$$U_g(\mathbf{x}) = \frac{1}{2}k_g(\mathbf{x} - \mathbf{x_g})^T(\mathbf{x} - \mathbf{x_g}) \tag{4.5}$$

onde $\mathbf{x_g}$ é a posição objectivo e k_g é uma constante.

A força aplicada no robô devido a este campo é

$$\mathbf{f_g}(\mathbf{x}) = -\nabla U_g(\mathbf{x}) = -k_g(\mathbf{x} - \mathbf{x_g}) \tag{4.6}$$

Ou seja, é proporcional à distância do robô à posição objectivo e aponta no sentido desta.

Por outro lado o potencial repulsivo é definido como

$$U_o(\mathbf{x}) = \begin{cases} \frac{1}{2}k_o(\frac{1}{d} - \frac{1}{d_0})^2 & \text{se } d < d_0 \\ 0 & \text{se } d \geq d_0 \end{cases} \tag{4.7}$$

onde d é a distância entre o robô e o obstáculo, d_0 é a distância máxima de influência do obstáculo e k_o uma constante de ganho. A força repulsiva induzida por este campo é

$$\mathbf{f_o}(\mathbf{x}) = \begin{cases} k_o(\frac{1}{d} - \frac{1}{d_o})\frac{1}{d^2}\frac{\mathbf{dd}}{\mathbf{dx}} & \text{se } d < d0 \\ 0 & \text{se } d \geq d_0 \end{cases} \tag{4.8}$$

onde $\frac{\mathbf{dd}}{\mathbf{dX}}$ representa o vector unitário segundo o qual a força é aplicada.

Generalizando para um qualquer número n de obstáculos, obtemos a força total produzida pelos obstáculos $\mathbf{f_{o_t}}$ através da expressão

$$\mathbf{f_{o_t}} = \sum_{i=0}^{n} \mathbf{f_{o_i}} \tag{4.9}$$

A força resultante de dos campos atractivos e repulsivos é

$$\mathbf{f_R} = \mathbf{f_g} + \mathbf{f_{o_t}} \tag{4.10}$$

A força $\mathbf{f_R}$, uma vez determinada, poderá ser aplicada ao controlo dos movimentos do robô. Esse controlo depende do tipo de robô e das restrições holonómicas do mesmo.

4.1.2 Poços de Potencial

O principal problema da utilização dos campos de potencial é aparecimento de mínimos locais, ou poços de potencial, em determinados pontos do espaço para certas configurações de obstáculos. Isto tem como consequência a anulação da força resultante aplicada ao robô quando este se encontra num destes pontos levando-o a parar. Uma forma de recuperar destas situações seria a detecção da paragem fora da posição objectivo. No entanto este método não é suficiente pois em vez de parar o robô pode oscilar em torno do mínimo local o que torna difícil a detecção(figura 4.2). Uma solução para este problema é a verificação de que a soma das forças aplicadas durante um conjunto de iterações é próxima de zero [23], no entanto mantém-se a dificuldade em escolher uma direcção que nos afaste deste mínimo e nos conduza ao objectivo ou seja é necessário usar um método complementar ao campo de potencial.

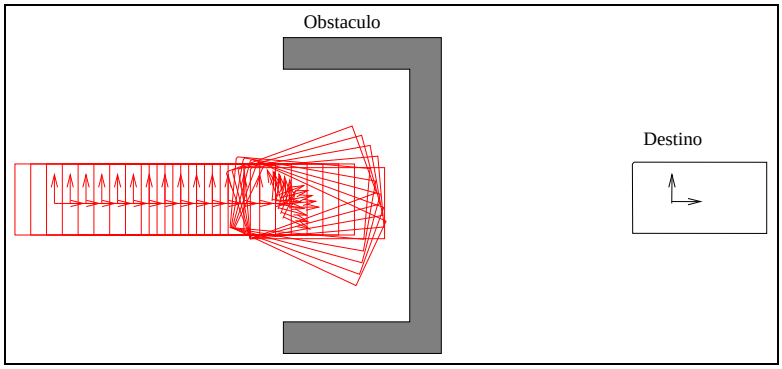

Figura 4.2: Poço de Potencial

4.1.3 Controlo dos movimentos de um robô rectangular através da aplicação de forças virtuais

Quando os métodos de campos de potencial artificial são aplicados em robôs sem restrições holonómicas, a sua aplicação é bastante simples pois a direcção da força resultante indica a direcção do movimento a efectuar. Isto já não é possível nos casos onde o robô apresenta restrições holonómicas, pois ao contrário dos anteriores este não pode efectuar translações em todas as direcções. Nestes casos é necessário começar por efectuar a rotação do robô, orientando-o segundo a direcção na qual ele se deve deslocar, efectuando só então a translacção pretendida.

Para este tipo de robôs (Robuter por ex.) começa-se por escolher um ponto de aplicação das forças sobre o eixo longitudinal que não o centro de rotação do próprio robô, como se pode ver na figura 4.3.

A direcção da força resultante aplicada no ponto C_f indica a direcção a seguir pelo robô. O passo seguinte é o calculo do comando a utilizar para obter a translação e/ou rotação necessárias. Aqui é utilizado um método simples que utiliza o conceito de *centro de rotação instantânea*, já referido na secção 2.1.1. Assim, uma vez conhecido o vector força, temos a direcção segundo a qual o robô se deve deslocar e, dado que se trata de um robô com restrições holonómicas, esse movimento será o resultado de uma rotação em torno do centro de rotação do robot e de uma translacção o que corresponde a uma rotação em torno de um ponto do espaço (centro de rotação instantãneao).

O centro de rotação instantânea é então determinado pela intersecção entre uma recta, que passa pelo ponto de aplicação das forças (C_f) e que é perpendicular à força resultante, e a recta que passa pelo centro das rodas motrizes (figura 4.4). Uma vez determinado esse ponto é possível determinar a relação entre as velocidades lineares aplicadas a cada uma das rodas motrizes por forma a obter a rotação pretendida e que é dada pela expressão 2.1.

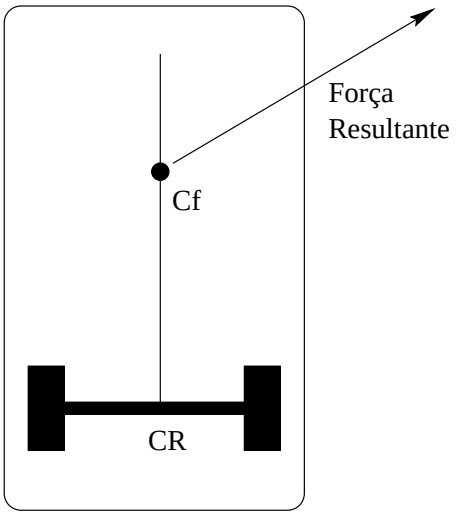

Figura 4.3: Ponto de aplicação das forças num robô não holonómico

No entanto esta relação não pode ser aplicada directamente uma vez que o comando da plataforma se faz especificando os valores para a velocidade linear (v_l) e angular (ω) e não pela atribuição directa das velocidades de cada uma das rodas motrizes.

Para obter estes valores começamos por definir um valor para velocidade linear (v_l) do robô . Com base neste valor e na distância d entre centro de rotação do robô (ponto ao meio da distância entre as duas rodas motrizes) e o ponto C_{RI} definimos a velocidade angular da seguinte forma:

$$\omega = v_l/d \tag{4.11}$$

Obtidos os valores v_l e ω é necessário tomar precauções quanto à ocorrência de alguns casos particulares destes valores.

O primeiro destes casos surge quando o ponto C_{RI} se situa no infinito, ou seja a força resultante é perpendicular ao eixo das rodas motrizes. Neste caso consideramos apenas a velocidade linear, uma vez que a velocidade angular é nula.

O segundo caso surge quando o ponto C_{RI} se situa sobre o centro de rotação do robô. Neste caso consideramos apenas a existência de velocidade angular sendo a velocidade linear nula.

O terceiro e último caso, não surgindo de uma indeterminação matemática, tem a ver com os limites físicos ou mecânicos do veículo. Quando a distância entre o centro de rotação do veículo e o ponto C_{RI} toma valores muito pequenos, a expressão 4.11 produz valores muito elevados para a velocidade angular. Existindo um limite para o valor desta é necessário garantir que o valor calculado seja inferior a este limite. Para os casos em que isto não se verifica é necessário fixar um valor (admissível) para a

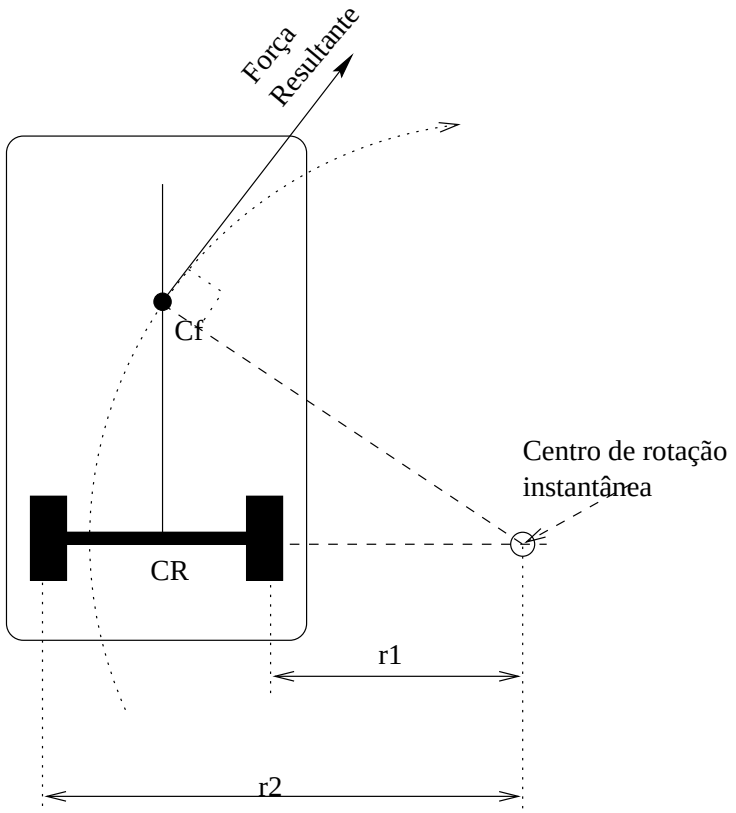

Figura 4.4: Determinação do Centro de rotação instantânea com base na força aplicada ao robô.

velocidade angular e recalcular a velocidade linear a partir desta, usando novamente a relação 4.11.

4.2 VFF - Campo de Forças Virtuais

A utilização de campos de potencial artificiais a navegação de robôs móveis tem sido adoptada por diversos autores devido à simplicidade dos conceitos envolvidos, no entanto o método básico apresentado na secção 4.1.1 apresenta algumas dificuldades para uma implementação que se destine a um funcionamento "em linha". Em primeiro lugar a obtenção de uma expressão para o potencial é difícil pois é necessário tomar em conta com a contribuição de todos os obstáculos podendo existir alguns cuja geometria seja mais ou menos complexa. Por outro lado a introdução de novos obstáculos, obtidos através dos sistemas sensoriais, vai introduzir uma dificuldade adicional pois também estes contribuem para o potencial e consequentemente para as forças que

actuam sobre o robô.

Borenstein [20] procurou resolver este tipo de problemas utilizando uma grelha para representar os obstáculos detectados. Esta grelha é construída, tal como no método apresentado na secção 3.2, como um histograma. No entanto, esse histograma é obtido de uma forma mais simples, pois não toma em conta a dispersão do feixe de ultra-sons mas apenas a medida obtida ao longo do eixo acústico do sensor, o que leva a que para cada medida seja actualizada uma única célula da grelha. Em cada uma das células da grelha vai ficando registado o número de vezes que um sensor registou a presença de um obstáculo na zona correspondente. Neste histograma bidimensional vão surgindo picos e vales que correspondem a zonas onde existem (ou não existem) obstáculos. Por forma a reduzir o efeito de certo tipo de erros como o "*crosstalk*" na detecção de obstáculos consideram-se zonas ocupadas as que apresentarem um valor (frequência) superior a um limiar pré-definido.

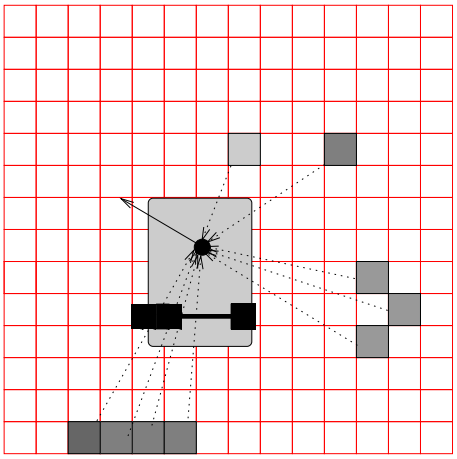

Figura 4.5: Ponto de aplicação das forças sobre o robô, forças geradas pelos obstáculos e força resultante.

Uma vez registados os obstáculos na grelha é necessário determinar as forças repulsivas produzidas pelos mesmos sobre o robô o que é feito como se descreve a seguir. Cada célula ocupada, exerce uma força repulsiva sobre o robô. Esta força é proporcional ao valor do histograma e inversamente proporcional à distância entre a célula e o robô como se pode ver pela expressão

$$\mathbf{F}(i,j) = \frac{F_{cr}C(i,j)}{d^2(i,j)}\left[\frac{x_t - x_o}{d(i,j)}\hat{\mathbf{x}} + \frac{y_t - y_o}{d(i,j)}\hat{\mathbf{y}}\right], \tag{4.12}$$

onde $\mathbf{F_{cr}}$ é uma força constante repulsiva, $d(i,j)$ a distância entre a célula (i,j) e o robô, $C(i,j)$ valor da célula, x_o, y_o coordenadas do robô e x_t, y_t coordenadas da célula.

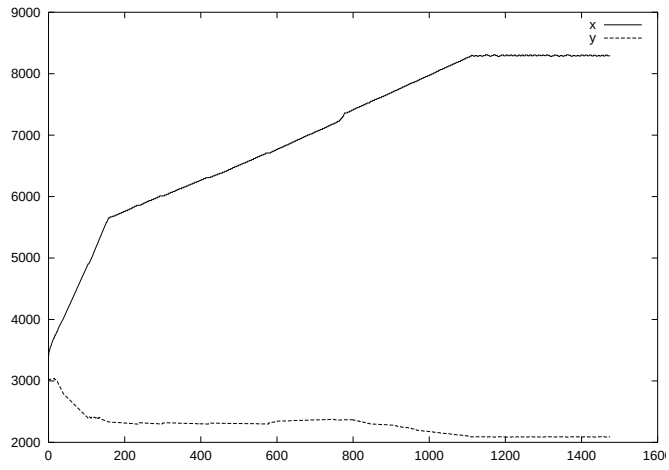

Figura 4.6: Evolução dos valores das coordenadas (x,y) do robô (eixo vertical, unidades: mm) ao longo do tempo (eixo horizontal, unidades: ciclos de processamento).

As células que circundam o robô vão, desta forma, exercer forças repulsivas sobre este, sendo a força resultante dada por:

$$\mathbf{F_r} = \sum_{ij} \mathbf{F}(i,j) \tag{4.13}$$

Esta força (F_r) tende a "empurrar" o robô para longe dos obstáculos, tal como se pode ver na figura 4.5. É ainda aplicada ao robô uma força constante que o "puxa" em direcção à posição destino, que é definida por:

$$\mathbf{F_a}(i,j) = F_{ca}\left[\frac{x_t - x_o}{d_t}\hat{\mathbf{x}} + \frac{y_t - y_o}{d_t}\hat{\mathbf{y}}\right] \tag{4.14}$$

Onde F_{ca} é o valor da força constante atractiva, d_t a distância entre o robô e o objectivo e x_t, y_t as coordenadas do objectivo.

A força resultante aplicada ao robô será a soma das forças repulsivas e da força atractiva:

$$\mathbf{F} = \mathbf{F_r} + \mathbf{F_a} \tag{4.15}$$

4.2.1 Resultados

A implementação mostrou que o método é eficaz no que diz respeito à utilização dos dados sensoriais para conduzir o veículo sem colidir com os obstáculos como se pode

ver pelas figuras 4.6, 4.7 e 4.8, que foram obtidas por simulação partindo da posição (3400,3000) e cujo objectivo era a posição (10000,10000). Na situação inicial nenhum dos obstáculos estava descrito na grelha e à medida que o robô se ia deslocando eram feitas medições de distância com os sensores de ultra-sons e os registos correspondentes sobre a grelha.

Como se pode ver pelas figuras mencionadas o robô começou por se deslocar em direcção ao objectivo, mas assim que houve um número suficiente de leituras dos obstáculos a direcção foi alterada seguindo de forma quase paralela a estes. Como se pode ver no gráfico da figura 4.6 onde sensivelmente a partir da iteração 1100 não há qualquer evolução na coordenada x ou y do robô o que indica que este terá caído num mínimo local.

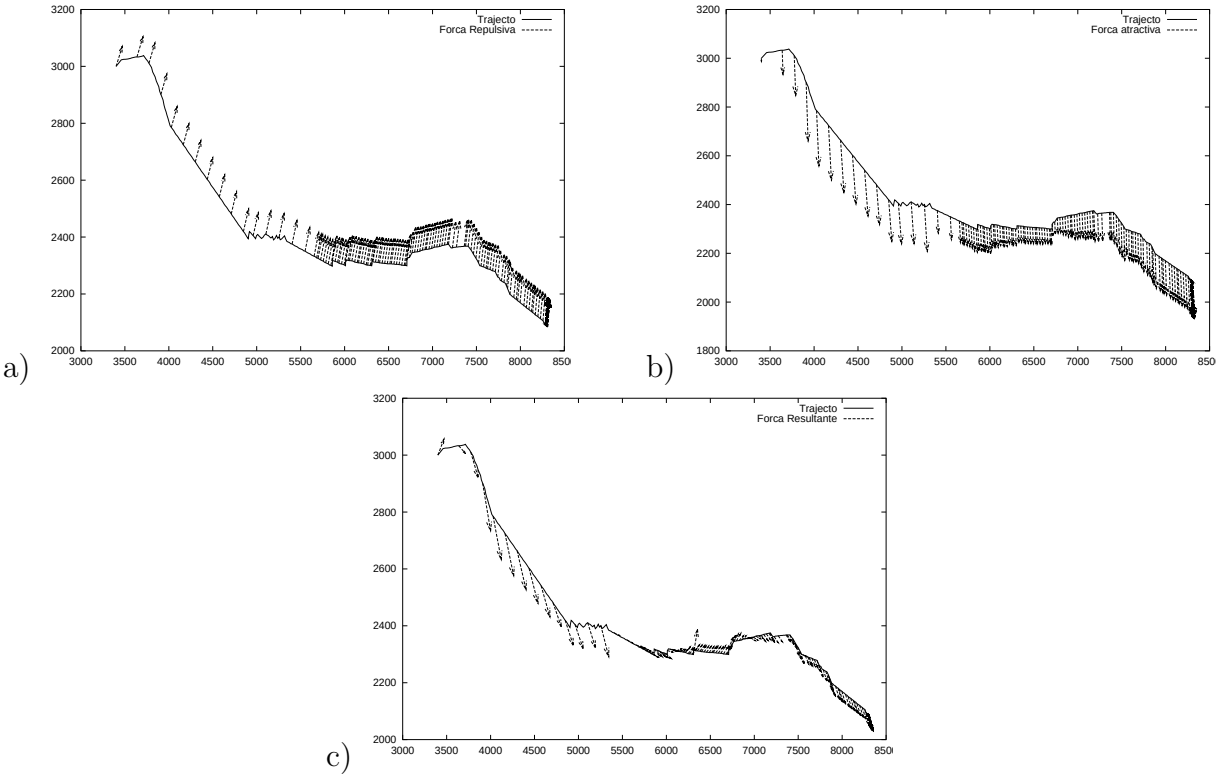

Figura 4.7: Trajecto percorrido utilizando o método VFF, partindo do ponto (3400,3000) com destino à posição (10000,10000): a) força repulsiva, b) força atractiva produzida pelo objectivo, c) força resultante.

Este método baseando-se apenas na soma vectorial das forças que cada uma das cargas gera sobre o robô, não utiliza directamente a informação sobre a disposição dos obstáculos no espaço para escolher uma direcção a seguir. Isto faz, tal como o método clássico dos campos de potencial, que o robô possa ficar preso zonas onde a

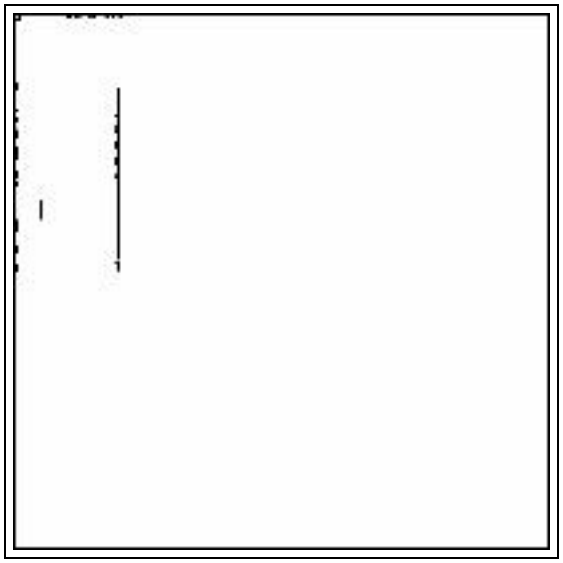

Figura 4.8: Grelha obtida durante o percurso

soma das forças repulsivas anulem a força atractiva produzida pela posição objectivo. Estas zonas não são mais do que os poços de potencial de que falámos na secção 4.1.2.

4.3 VFH - Histograma de Campo Vectorial

Este método, também desenvolvido por Borenstein[19], tenta resolver o problema dos mínimos locais existente nos métodos de campo virtual. Aqui os dados são processados em duas etapas: a primeira consiste na construção de uma grelha de ocupação tal e qual como no método VFF, consistindo a segunda etapa na construção de um histograma polar relativo à distribuição de obstáculos em torno do robô.

Tal como no método VFF é utilizada uma janela centrada no robô. Essa janela é dividida em células onde cada célula corresponde a uma zona do espaço nas imediações do robô. O preenchimento das células é feito de forma semelhante ao utilizado da grelha de ocupação, sendo no entanto mais simples por só ser actualizada a célula situado sobre o eixo do sensor de ultra-sons a uma distância fornecida por este sensor. Como o pretendido é construir um histograma de observação, a célula em questão é incrementada de uma unidade (ou outro valor de escala). As restantes células que se situam sobre o eixo acústico do sensor entre esta e o sensor poder ser colocadas a zero ou simplesmente ver o seu valor decrementado de uma quantidade predefinida.

A figura 4.9 ilustra a construção do histograma de ocupação. Nela podemos ver que a janela activa da grelha (histograma) se encontra centrada no robô. É apenas

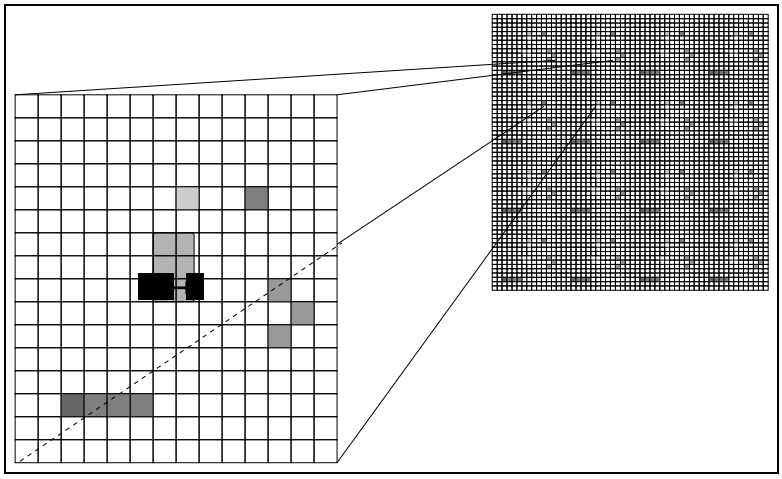

Figura 4.9: Em cada instante é utilizada, na actualização e calculo das forças, apenas uma porção do histograma de ocupação a que chamamos janela activa.

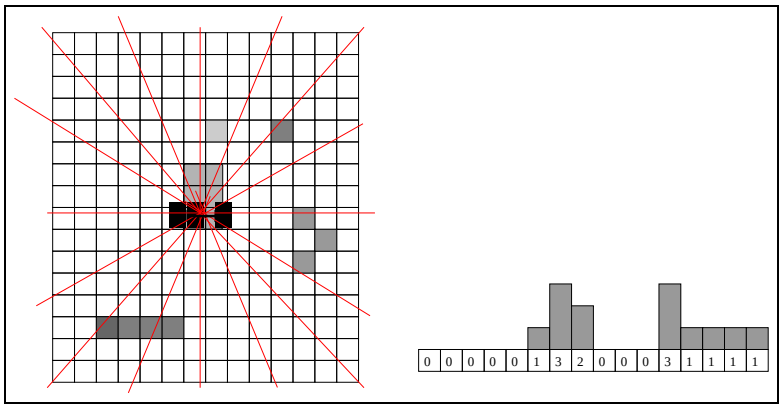

Figura 4.10: Construção do histograma polar de densidade de obstáculos

sobre as células que se encontram nesta janela que são feitos os registos correspondentes às observações dos sensores. Obtido o histograma é feita nova redução de dados, desta vez dividindo a janela em sectores centrados no robô e contabilizando o número de células ocupadas em cada sector, como se mostra na figura 4.10.

Obtemos desta forma um histograma onda cada elemento contém o número de células ocupadas que em princípio poderíamos encontrar se nos deslocássemos em cada direcção.

Se analisarmos este histograma verificamos que dele poderemos saber qual ou quais as direcções que o robô pode tomar sem haver perigo de colidir com obstáculos.

Para a construção do histograma polar determina-se para cada célula do histograma de ocupação a direcção (ângulo) em relação ao centro do robô e um valor de

amplitude.

O ângulo vem dado por

$$\beta_{i,j} = arctan\frac{y_j - y_0}{x_i - x_0} \tag{4.16}$$

onde (x_i, y_j) são as coordenadas da célula (i,j), (x_0, y_0) são as coordenadas do centro do robô.

A amplitude $m_{i,j}$ obtém-se por

$$m_{i,j} = (c^*_{i,j})^2 \times (a - bd_{i,j}) \tag{4.17}$$

onde a, b são constantes positivas escolhidas de forma que a amplitude se anule para a célula mais distante ao centro do robô (e que se encontre no interior da janela), $c^*_{i,j}$ é o valor de certeza de ocupação da célula e $d_{i,j}$ a distância entre a célula e o centro do robô.

O histograma polar é construído somando para cada sector k os valores da amplitude calculados para cada célula no interior desse sector. Assim para cada sector a densidade polar de obstáculos vem

$$h_k = \sum_{i,j} m_{i,j} \tag{4.18}$$

Por razões que têm a ver com a natureza discreta do histograma polar, Borenstein sugere a utilização de um filtro para suavizar o histograma obtido. O filtro sugerido baseia-se nos valores das frequências adjacentes e é dado por

$$h'_k = \frac{h_{k-l} + 2h_{k-l+1} + ... + 2h_{k+l-1} + h_{k+l}}{2l + 1} \tag{4.19}$$

sendo necessário escolher o valor para l que dita o número de sectores adjacentes que são utilizados para obter o valor filtrado de determinado sector.

O histograma resultante contém picos que correspondem a direcções com maior densidade de obstáculos e vales que correspondem a direcções onde existem poucos ou nenhuns obstáculos. Note-se ainda que a contribuição de cada obstáculo para o histograma depende da distância entre este e o robô, pelo que obstáculos muito distantes pouco ou nada contribuem para o histograma, ao passo que obstáculos próximos vão surgir como picos no histograma.

Construído o histograma polar, estamos em condições de dele extrair a informação que necessitamos para conduzir o robô e que é o par direcção, velocidade a utilizar.

Sendo conhecida a posição final desejada e a posição actual do robô, define-se uma direcção objectivo, sendo escolhido do histograma o vale que mais se aproxima dessa direcção objectivo.

Depois de seleccionado o vale é necessário escolher o sector ao longo do qual o robô se irá deslocar.

Os sectores extremos do vale servem como referência para a escolha da direcção de navegação a escolher. Assim se o primeiro sector tiver uma direcção θ_i e o último θ_f, a direcção de condução é dada por $\theta = (\theta_i + \theta_f)/2$.

Para a selecção dos vales candidatos é necessário definir um valor de limiar que permite decidir se um determinado sector pertence ou não a um vale. A escolha deste valor é importante pois um valor muito elevado pode levar a que sejam ignorados valores de densidade elevados e desta forma o robô se aproxime demasiado dos obstáculos. Por outro lado um valor muito baixo pode levar ao desaparecimento total dos vales, impossibilitando o robô de navegar em corredores apertados ou mesmo "fechando-o" numa zona sem vales.

4.3.1 Resultados

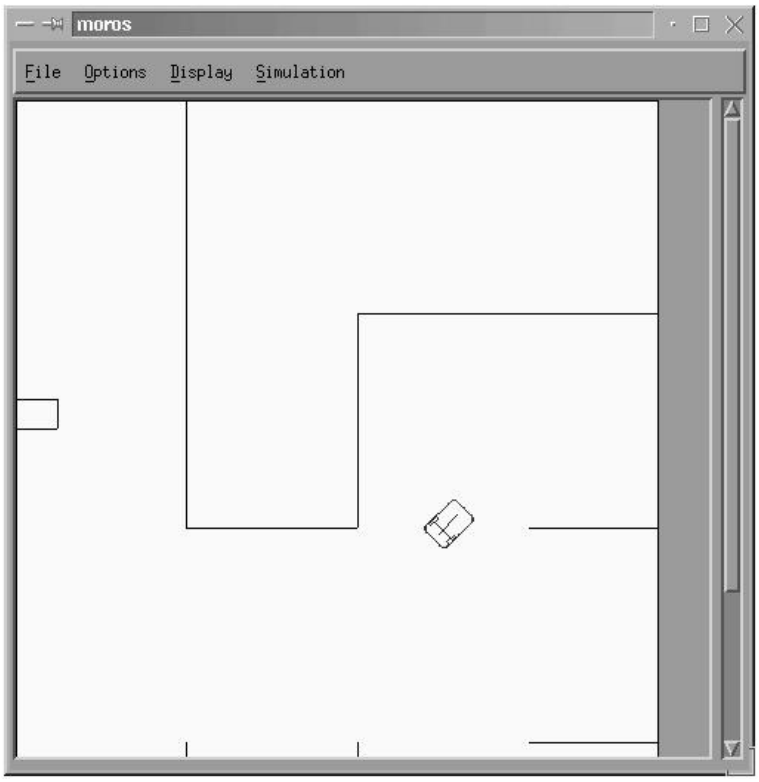

Figura 4.11: Simulador com o robô no ponto de chegada

Os resultados obtidos demonstram que este método apresenta um melhor comportamento do que o VFF. Assim para um ambiente da figura 4.11 o VFH consegue

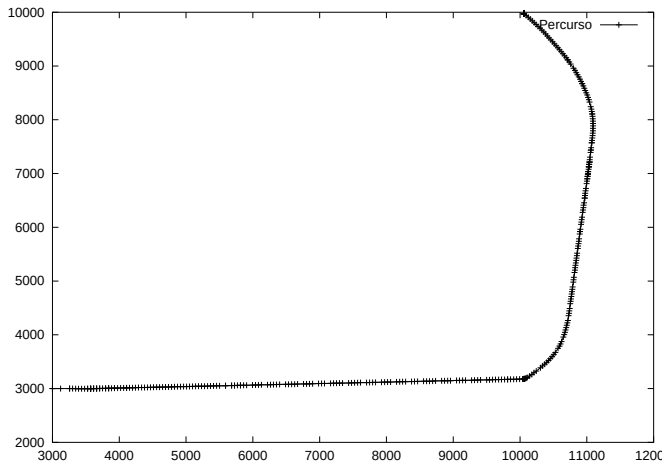

Figura 4.12: Caminho percorrido usando o método VFH

atingir a posição objectivo (10000,10000) ao passo que o método VFF fica preso devido à existência de mínimos locais onde as forças se anulam (rever figura 4.6). Isto deve-se devido ao facto de, no método VFH, o robô ser conduzido não pela direcção da força resultante da soma da força atractiva produzida pelo objectivo com as forças repulsivas produzidas pelos obstáculos como acontece no método VFF, mas, pela escolha da direcção livre (sem obstáculos) que mais se aproxima da direcção do segmento de recta que une o centro do robô à posição objectivo.

A figura 4.12 mostra o trajecto percorrido e é possível observar que se trata de um trajecto suave e sem oscilações perceptíveis.

Na figura 4.13 podemos observar um histograma de ocupação obtido durante a execução do trajecto referido bem como o mesmo histograma depois de aplicado o filtro da expressão 4.19.

A figura 4.14 mostra o histograma de ocupação que obtido durante o percurso e ao ser comparado com o mapa utilizado pelo simulador (figura 4.11) se pode ver quais os obstáculos que produziram leituras nos sensores de ultra-sons. Note-se que o histograma de ocupação está rodado 90^0 em relação ao mapa do simulador.

4.3.2 Vantagens e Limitações

Este método por utilizar uma estrutura em grelha onde são integradas consecutivamente as leituras dos sensores, dispõe de memória que actua como filtro para quaisquer erros que esporadicamente surjam nas medidas dos sensores. O método possibilita ainda a navegação em corredores e passagens estreitas. As oscilações presentes em outros métodos como o VFF, não se verificam quando utilizamos o VFH.

O VFH sendo essencialmente um método reactivo, não utilizando qualquer tipo de

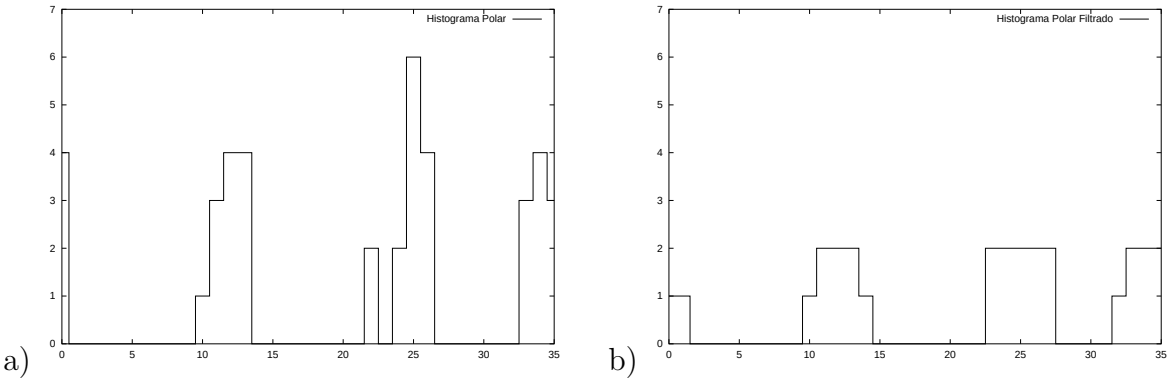

Figura 4.13: Exemplos de: a) histograma polar obtido. b) histograma polar depois de filtrado.

planeamento, não consegue detectar as armadilhas que impossibilitam a progressão do robô. Neste caso também vamos ter um comportamento oscilatório pois se existirem dois vales com direcções opostas a escolha num instante pode cair sobre um deles, fazendo com que o robô avance nesse sentido, e após esse deslocamento o vale preferencial passa a ser o oposto.

Para ultrapassar este tipo de problemas é necessário introduzir formas de detecção de armadilhas, como aquela já referida de verificar que os deslocamentos do robô acumulados ao longo de um período de tempo tendem para o deslocamento nulo, e na sua presença utilizar métodos alternativos para conduzir o robô para fora destas, ou simplesmente efectuar o replaneamento e gerar um ponto objectivo intermédio que já seja possível de atingir com o VFH.

4.4 Grelha de Potencial

Para a criação de um campo de potencial com base no conhecimento das posição dos obstáculos, Plumer [31] sugeriu a utilização de uma estrutura matricial sobre a qual eram calculados valores de potencial resultantes da presença desses obstáculos. Tendo sido inicialmente pensado para ambientes estáticos onde as posições dos diversos obstáculos eram conhecidas "a priori", verificou-se que este método também poderia ser adaptado a sistemas onde a informação sobre os obstáculos é obtida a partir de sistemas sensoriais, desde que se disponha de poder computacional suficiente.

Tal como nos outros casos, a ideia consiste em criar uma barreira de potencial em torno dos obstáculos o que vai evitar que o robô se aproxime demasiado dos obstáculos e consequentemente não colida com estes. Utilizando como base a estrutura da grelha de ocupação referida anteriormente, é construída uma nova grelha que corresponde a uma discretização do campo de potencial gerado pelos obstáculos. Assim, para cada

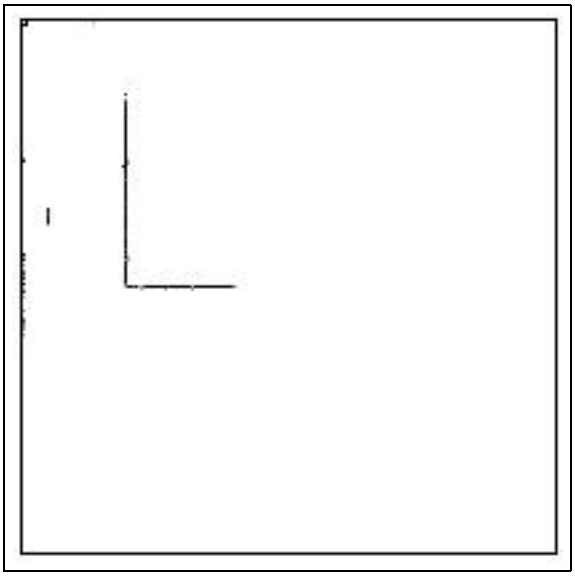

Figura 4.14: Histograma de ocupação obtido no final do percurso

célula $C(i,j)$ (ver figura 4.15) é obtido o potencial $\Phi_{i,j}$ com base no potencial das células vizinhas, de acordo com as seguintes equações:

$$
\begin{aligned}
\Phi_{i,j}(t=0) &= \text{um qualquer valor} \\
\Phi_{i,j}(k+1) &= max \left\{ \begin{array}{r} \alpha^{\sqrt{2}}\Phi_{i+1,j-1}(k) \\ \alpha^{\sqrt{2}}\Phi_{i-1,j-1}(k) \\ \alpha^{\sqrt{2}}\Phi_{i+1,j+1}(k) \\ \alpha^{\sqrt{2}}\Phi_{i-1,j-1}(k) \\ \alpha\Phi_{i+1,j}(k) \\ \alpha\Phi_{i-1,j}(k) \\ \alpha\Phi_{i,j+1}(k) \\ \alpha\Phi_{i,j-1}(k) \end{array} \right\} && \text{se } s(C(i,j)) \neq \mathfrak{oc} \\
\Phi_{i,j}(k+1) &= \Phi_{max} && \text{se } s(C(i,j)) = \mathfrak{oc}
\end{aligned}
$$

onde $s(C(i,j)$ refere-se ao estado da célula $C(i,j)$ e $\mathfrak{oc}$ denota o estado ocupado.

Como se pode depreender, esta aproximação do campo de potencial é feita de forma iteractiva, varrendo em cada iteração todas as células da grelha. O resultado é a obtenção de valores de potencial máximos nas células que correspondem aos obstáculos e que este potencial vai decrescendo geométricamente com a distância a estas. O raio de influência dos obstáculos pode ser controlado através do parâmetro α. A figura 4.16 representa uma grelha de potencial obtida em que os niveis de potencial máximos são representados pela cor branca, os mínimos pela cor preta e os cinzentos valores intermédios.

aa	ab	ac
ad	ae	af
ag	ah	ai

Figura 4.15: O potencial de uma célula é calculado com base no estado desta e do potencial das células vizinhas.

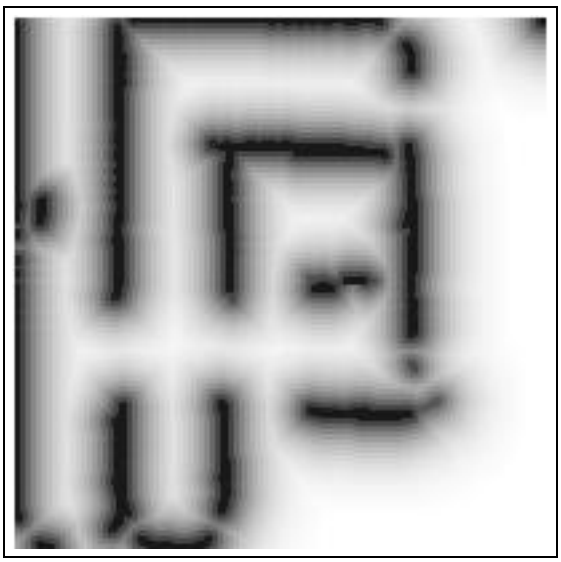

Figura 4.16: Mapa de potencial obtido através das equações de relaxação

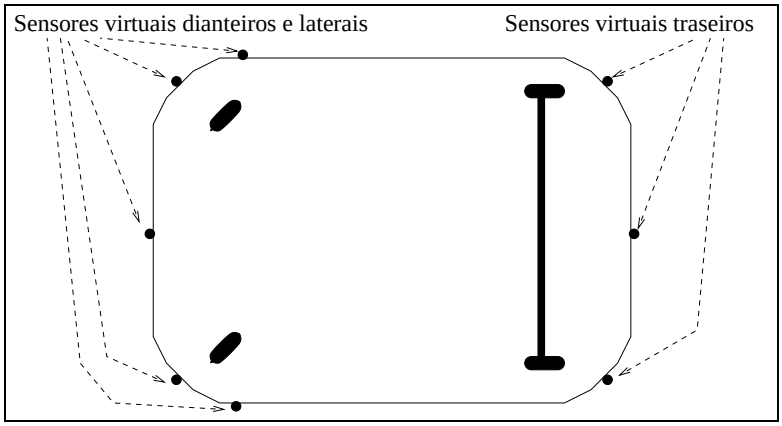

Figura 4.17: Distribuição dos sensores virtuais em torno do robô

Uma vez obtido o potencial, como a força é dada por: $F_o = -\nabla U_o(X)$, esta é aproximada através do uso de sensores virtuais, como se pode ver na figura 4.17, que "medem" o potencial em torno do robô. Uma vez encontrado o par de sensores que apresentam a maior diferença de potencial, a força repulsiva é dada por $F_o = k.\Delta U$ sendo a direcção a da recta que une os dois sensores.

Tal como acontece com outros métodos referidos, para conduzir o robô até à posição objectivo é introduzida uma força constante atractiva que o "puxa" para esta posição.

4.4.1 Resultados obtidos

Este método foi aplicado na plataforma Robuter com sucesso. Sempre que a densidade de obstáculos não era demasiadamente elevada, o robô conseguia atingir a posição objectivo. No entanto, verificou-se que em alguns casos o robô ficava preso em poços de potencial. Isto deve-se ao facto de neste método não existir qualquer tipo de planeamento global sendo o seu comportamento puramente reactivo. As figuras 4.18, 4.19 e 4.20 ilustram os resultados obtidos, podendo-se ver a evolução das coordenadas do robô bem como o mapa de potencial obtido para diferentes posições de partida e destino.

4.5 Limitações dos métodos de campo de potencial

Estes métodos apresentam diversos tipos de limitações as quais podem ser facilmente identificadas e que são:

- Armadilhas provocadas pelos mínimos locais.

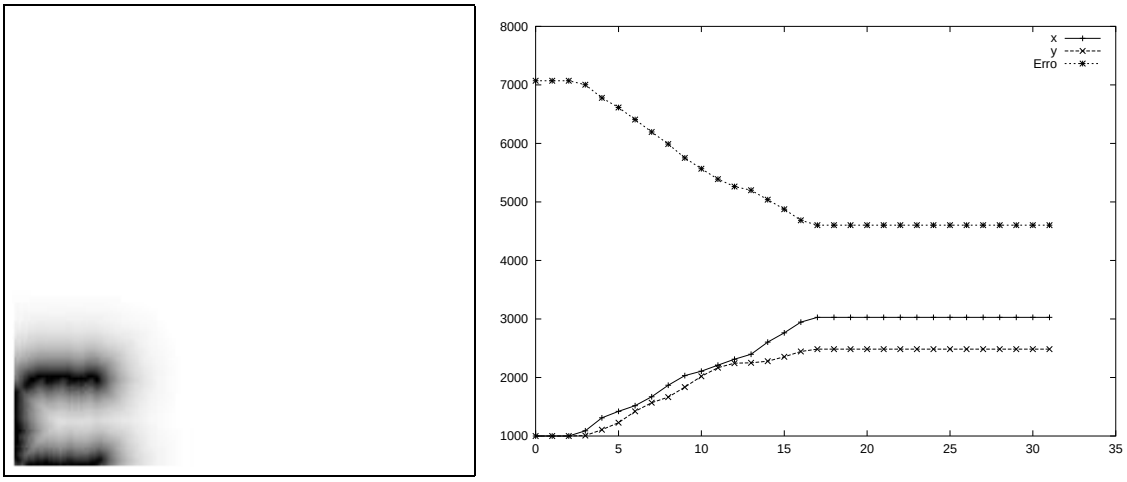

Figura 4.18: Potencial (esquerda) e evolução das coordenadas (direita) do robô partindo do ponto $(1000, 1000)$ com destino ao ponto (6000,6000), tendo caído num poço de pontencial a uma distância de aproximadamente 4700 unidades do objectivo.

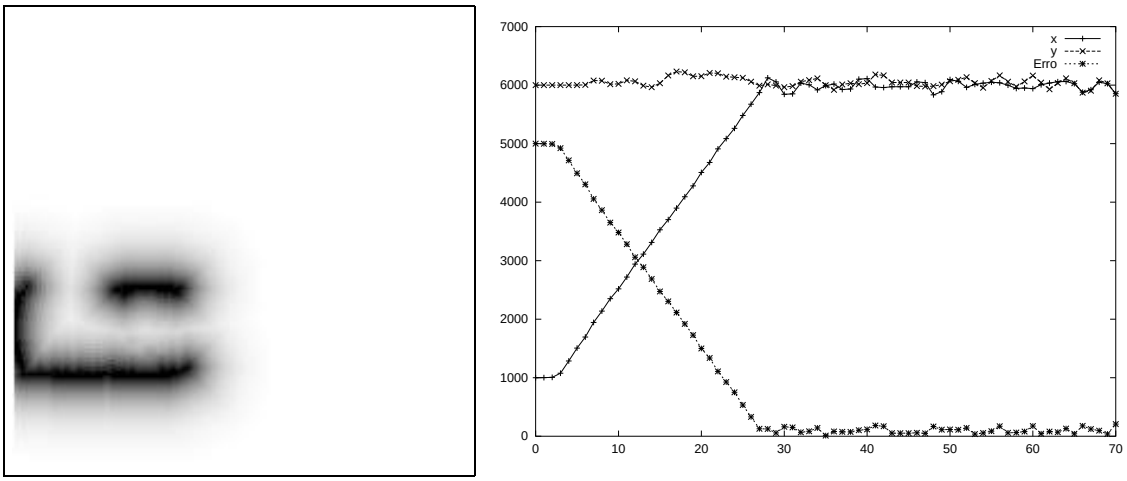

Figura 4.19: Potencial (esquerda) e evolução das coordenadas (direita) do robô partindo do ponto $(1000, 6000)$ com destino ao ponto (6000,6000), tendo atingido as imediações do objectivo

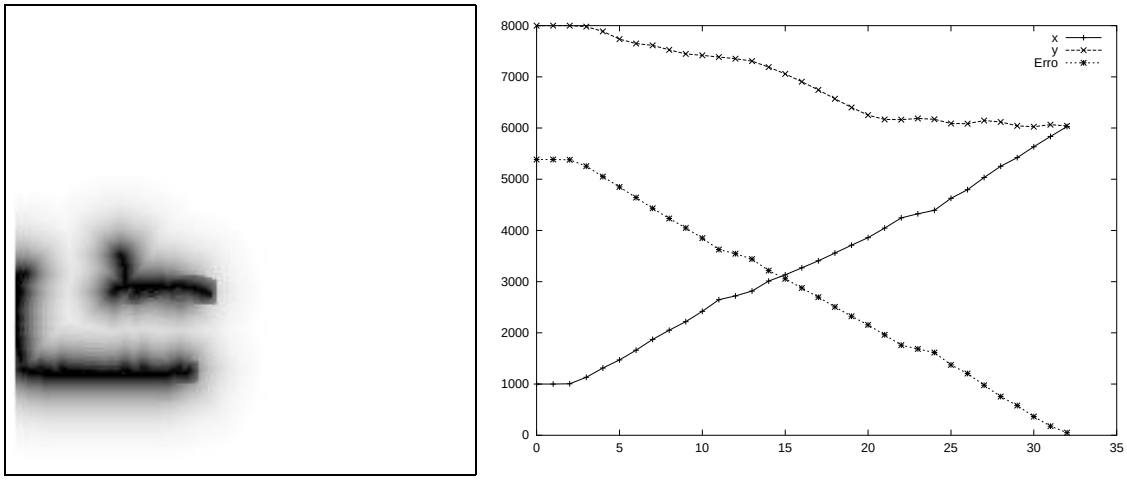

Figura 4.20: Potencial (esquerda) e evolução das coordenadas (direita) do robô partindo do ponto $(1000, 8000)$ com destino ao ponto (6000,6000), tendo atingido o objectivo.

- Não permitem a passagem entre obstáculos muito próximos, ainda que exista espaço físico para o robô passar.
- Comportamento oscilatório nas imediações dos obstáculos e em passagens estreitas.

Mínimos locais

Tal como foi já discutido anteriormente, este é o principal problema dos métodos de navegação baseados em campos de potenciais.

Dado que este tipo de navegação não se baseia no planeamento global das trajectórias, mas sim na procura do mínimo da função de potencial, é muito frequente o trajecto conduzir o robô para um mínimo local. Uma vez atingido um destes pontos, o gradiente anula-se e consequentemente a força que "conduzia" o robô. Quando o robô cai numa destas "armadilhas", a recuperação pode ser feita de duas formas: tentativa-e-erro ou por um replaneamento global.

Dado que as soluções heurísticas apresentam normalmente custos elevados, devido à geração de trajectórias não-óptimas, procura-se muitas vezes combinar o planeamento local e global. Assim, o planeamento global gera um conjunto de pontos intermédios que conduzem à posição objectivo. Esses pontos são então fornecidos ao planeamento local, que pode utilizar os referidos campos de potencial. Estes pontos intermédios funcionam como pontos objectivo durante uma fase da trajectória, ou seja dada uma sequência de pontos objectivo $\{P_1..P_n\}$ e partindo do ponto inicial P_0 o planeador local (muitas vezes referido como controlador) vai tentar conduzir o robô

até às imediações do ponto P_1. Uma vez atingido esse ponto (ou uma vizinhança pré-fixada) é escolhido o ponto P_2 como nova posição objectivo.

Nos casos em que o robô cai num mínimo local, pode-se invocar o planeador global por forma a que este gere um novo conjunto de pontos intermédios que permitam ultrapassar esta situação.

Um dos problemas que se colocam é o de como detectar se o robô está ou não num mínimo local. Não existem soluções óptimas pelo menos em termos de velocidade de detecção pois se bem que se pode facilmente detectar um mínimo local se o robô ficou parado fora da posição objectivo, o mesmo já não se passa quando este passa a oscilar em torno de um mínimo local.

Uma das formas possíveis para detectar estas situações é analisando a soma das forças aplicadas no robô durante os últimos n ciclos pois em qualquer dos casos o valor desta soma tenderá para zero se o robô caiu num mínimo local. No entanto, e tal como já foi referido, este método só permite a detecção destas situações ao fim de determinados períodos de tempo (ao fim de alguns ciclos) o que vai fazer com que o tempo de execução das trajectórias seja muito maior que o que se poderia esperar (sem obstáculos).

Impossibilidade de passar entre obstáculos próximos

Uma vez que cada obstáculo gera em torno de si um campo de potencial repulsivo podem existir situações onde dois obstáculos se encontrem suficientemente espaçados para permitir a passagem do robô entre eles. No entanto os potenciais gerados por ambos obstáculos podem gerar uma barreira de potencial entre estes que impossibilite o robô de a atravessar. Estas situação são frequentes em portas e entradas de corredores. A resolução destas situações pode passar por aumentar a "coragem" do robô reduzindo a contribuição do campo repulsivo produzido pelos obstáculos na força aplicada ao robô o que pode levar a que este colida com alguns obstáculos, uma outra solução é a escolha de outros métodos de navegação para estas situações.

Oscilações nas imediações de obstáculos ou corredores

As oscilações que se fazem sentir na presença de obstáculos (como o são as paredes dos corredores) podem provocadas por um ganho demasiado elevado na malha de controlo de direcção do robô o que se traduz na utilização de velocidades demasiado elevadas. Por outro lado a natureza discreta da grelha de potencial ou do próprio sistema de coordenadas pode levar a que entre duas posições onde são feitas as amostragens do campo de potencial este apresente variações bruscas de valor. Se no primeiro caso a solução pode passar por reduzir o ganho da malha de controlo já no segundo a solução pode-se tornar mais difícil pois o aumento da resolução seja ela espacial seja temporal pode implicar alterações profundas tanto a nível de software como hardware.

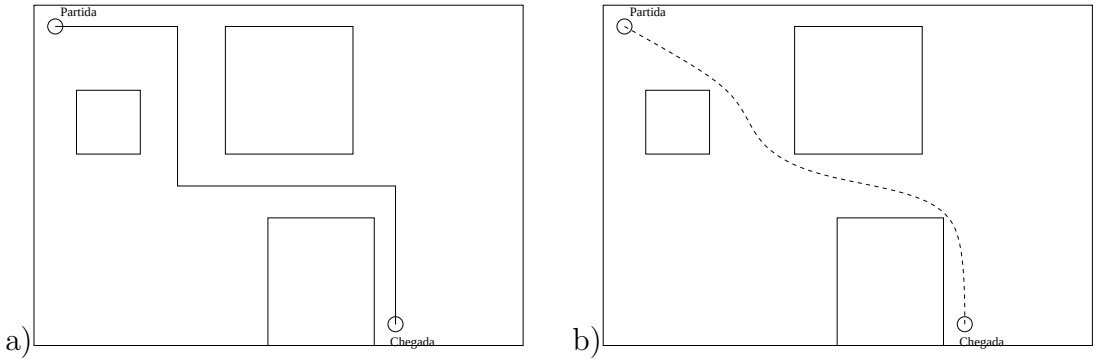

Figura 4.21: Trajectórias: a) inicial, b) adaptada pelo método das tiras elásticas.

4.6 Bandas Elásticas

Sean Quinlan desenvolveu um novo conceito para o problema da navegação face a um ambiente em mudança [39]. Partindo de um caminho definido por qualquer planeador de trajectórias e do conhecimento da posição dos obstáculos este novo método permite optimizar o caminho reduzindo o número de vezes que o robô tem de "parar para virar", ao mesmo tempo que é possível efectuar correcções ao caminho escolhido quando houver deslocação de alguns obstáculos.

Este novo método tem como base o modelo físico de uma tira elástica, embora algumas propriedades tenham sido modificadas para uma melhor adaptação à resolução do problema em causa. O caminho inicialmente definido (possivelmente com segmentos de recta) dá a forma a um elástico que ficando apenas seguro pelas extremidades (ponto de partida e ponto de chegada), vai alterando a sua forma, devido às forças internas, até se adaptar à forma dos obstáculos, cujo contacto leva ao aparecimento de forças externas que tendem a "segurar o elástico" em torno dos objectos.

A propriedade elástica do novo caminho permite que este se adapte, sem necessitar de um replaneamento, a pequenas mudanças nas posições dos obstáculos.

Como o elástico define o caminho a percorrer pelo ponto central do robô, a menos que se considere um robô pontual, este caminho deverá manter-se afastado dos obstáculos. Por outro lado é desejável que a trajectória do robô seja contínua e suave, que seja não contenha pontos onde o robô tenha de "parar para rodar"como acontece na figura 4.22.

Para evitar que isto aconteça, considera-se que todos os obstáculos criam um campo de potencial que vai exercer uma força sobre o elástico, afastando-o assim das suas fronteiras. Vejamos a seguir a definição deste modelo.

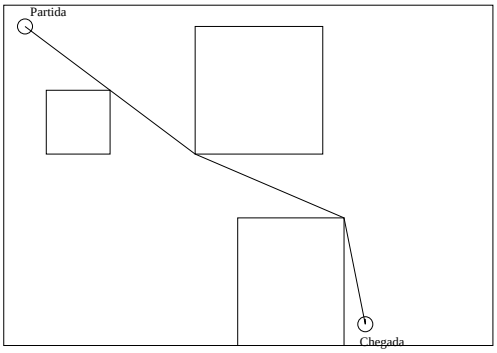

Figura 4.22: Trajectória adaptada às fronteiras dos obstáculos

4.6.1 Modelo das tiras elásticas

Começando por definir caminho, podemos dizer que este é uma sequência contínua de configurações para um robô, definindo com exactidão as configurações que o robô vai tomar durante o movimento. Usando uma definição mais formal podemos dizer que um caminho entre duas configurações $q_{inicial}$ e q_{final} é definido como uma função paramétrica $c(s)$ que mapeia os valores do intervalo $[0, 1]$ no espaço das configurações e que $q_{inicial} = c(0)$ e que $q_{final} = c(1)$. O parâmetro s não tem qualquer significado físico e a gama de valores de s foi escolhida arbitrariamente como sendo $[0, 1]$.

Por outro lado define-se trajectória como sendo um caminho cuja parametrização é feita com base no tempo. Um caminho pode ser convertido numa trajectória definindo a variável s como sendo uma função do tempo $s(t)$.

Ao dizermos que um caminho é definido por uma tira elástica estamos a considerar antes de mais que este se pode deformar ajustando-se aos contornos dos obstáculos.

Utilizando o modelo físico de uma tira elástica ideal para um caminho, então a força de contracção é dada por $f = k\Delta x$, onde k é a constante de elasticidade e Δx a deformação. Considerando que o elástico é composto por partículas infinitesimais então a força que une duas dessas partículas está igualmente relacionada pela constante de elasticidade e pelo afastamento dessas partículas.

Assim para uma determinada posição s no caminho, uma partícula que se encontre na posição correspondente sofre a influência das forças internas e externas

$$f(s) = f_{internas}(s) + f_{externas}(s).$$

Com base nesta expressão podemos dizer que a força externa sofrida por uma partícula é produzida pelo campo de potencial gerado pelos obstáculos enquanto que a força interna depende das características do material e corresponde à força de contracção do elástico. Assim a força interna aplicada a uma determinada partícula, resulta da interacção entre esta e as que se encontram na sua vizinhança. Esta

interacção depende da posição relactiva das particulas ou seja depende da forma local do elástico e não da posição da partícula em questão. Isto significa que a força interna pode ser definida como sendo uma função das derivadas do caminho em relação ao parâmetro s, ou seja

$$f_{interna}(\frac{dc}{ds}, \frac{d^2c}{ds^2}, \frac{d^3c}{ds^3}, ...).$$

Por outro lado a força externa depende apenas da posição da partícula no campo de potencial gerado pelos obstáculos, pelo que

$$f_{externa}(c(s)).$$

Movimento do Elástico

Se para uma determinada partícula do elástico a soma das forças internas e externas não for nula, e se considerarmos que cada partícula tem uma massa infinitesimal $\mu(s)$, então a partícula vai sofrer uma aceleração e por consequência vai-se deslocar. Por outro lado à medida que a partícula se desloca, a força externa, provocada pelo campo de potencial, vai variar. Isto significa que tanto a forma do elástico como a força aplicada podem ser descritos como sendo função de s e do tempo t, ou seja $c(s,t)$ e $f(s,t)$. O movimento das partículas do elástico está relacionado com as forças exercidas sobre estas, por esta razão podemos utilizar a segunda lei de Newton para obter uma expressão para esse movimento através de

$$f = \mu \frac{\partial^2 c}{\partial t^2}. \tag{4.20}$$

Esta equação diferencial pode ser integrada duas vezes em relação ao tempo para obter descrição dinâmica do comportamento do elástico sob a influência das forças. Quinlan optou por uma aproximação pseudo-estática para determinar a localização das partículas do elástico sob a influência das forças. Nesta abordagem, considera-se que as partículas do elástico estão em repouso, logo têm velocidade nula, expandindo $c(s, t + \Delta t)$ em série de Taylor

$$c(s, t + \Delta t) = c(s,t) + \Delta t \frac{\partial c}{\partial t} + \frac{1}{2}\Delta t^2 \frac{\partial^2 c}{\partial t^2} + ...$$

e considerando que em $t = 0$ a partícula está em repouso, então $dc/dt = 0$ e ignorando os termos de ordem superior a 2, temos que o movimento instantâneo da partícula será na direcção da aceleração $\partial^2 c/\partial t^2$.

Por outro lado a aceleração tem a direcção da força aplicada o que faz com que a variação na posição da partícula seja na direcção da força.

$$\Delta c \propto f$$

Elástico Ideal

Consideremos um elástico ideal o qual tem comprimento em repouso zero e que quando deformado a energia interna é dada por $\frac{1}{2}kx^2$, onde k é a constante de elasticidade e x a deformação. Para este modelo podemos dizer que a densidade de potencial interna v é dada por

$$v_{int} = \frac{1}{2}k||c'||^2 \tag{4.21}$$

que é obtida considerando que $||c'||$ corresponde à deformação da partícula infinitésimal s; por isso quanto maior for o valor de $||c'||$ mais alongada estará a partícula.

Como v é uma função de uma função não podemos utilizar o operador gradiente, uma vez que este apenas pode ser aplicado a funções com um número finito de parâmetros, mas, como se pode ver em [39], é possível obter um operador ($\overline{\nabla}$) com propriedades semelhantes o qual pode ser aplicado nos casos de funções de funções que tomam a forma de potenciais, vindo

$$\overline{\nabla}v = \frac{\partial v}{\partial c} - \frac{d}{ds}\frac{\partial v}{\partial c'} + \frac{d^2}{ds^2}\frac{\partial v}{\partial c''} - ... + (-1)^n\frac{d^n}{ds^n}\frac{\partial v}{\partial c^{(n)}} + ...$$

Aplicando este operador a 4.21 obtemos a força interna

$$\begin{aligned} f_{int}(s) &= -\overline{\nabla}v_{int} \\ &= \frac{d}{ds}\frac{\partial v_{int}}{\partial c'} \\ &= \frac{d}{ds}\frac{\partial}{\partial c'}\left(\frac{1}{2}k||c'||^2\right) \\ &= \frac{d}{ds}(kc') \\ &= kc'' \end{aligned} \tag{4.22}$$

A força externa exercida sobre cada uma das partículas depende, como já foi referido, apenas da posição da mesma no espaço das configurações, logo a densidade de energia externa depende apenas da posição o que faz com que neste caso o operador $\overline{\nabla}$ seja idêntico ao operador ∇, vindo

$$f_{ext}(s) = -\nabla v_{ext}(c(s)) \tag{4.23}$$

Daqui tiramos que a força exercida sobre cada uma das partículas infinitesimais do elástico é dada por

$$f(s) = kc''(s) - \nabla v_{ext}(c(s)) \tag{4.24}$$

Limitações na aplicação deste modelo

Este modelo apresenta algumas dificuldades na sua aplicação para a resolução do problema da obtenção de caminhos eficientes a que nos propomos. Isto deve-se à propriedade do próprio elástico de aumentar a tensão interna à medida que é "esticado". Se pensarmos no que acontece quando esticamos um pedaço de material elástico, percebemos que à medida que este é esticado maior é a força necessária para o deformar.

No caso do modelo considerado também percebemos facilmente que à medida que o comprimento deste é aumentado, a força externa necessária produzida pelo campo de potencial vê os seus efeitos anulados pelas forças internas e consequentemente a deformação é menor.

Por outro lado percebemos também que mudanças locais têm efeitos globais, ou seja, se deslocarmos um obstáculo de forma a que este "empurre"o elástico em determinada zona isto irá fazer com que o elástico seja alongado e consequentemente, devido ao aumento das forças de contracção, este se aproxime de todos os obstáculos, o que pode mesmo levar ao contacto com algum (ou alguns) obstáculo(s).

Um Elástico de Tensão Constante

Se usarmos um modelo para o elástico em que a força de contracção é constante independentemente da deformação sofrida por este, poderemos resolver o problema acima descrito. De facto este modelo não corresponde a nenhum material elástico conhecido mas serve o objectivo a que nos propomos.

Este tipo de elástico apresenta uma densidade de energia que toma a seguinte forma

$$v_{int} = k||c'||.$$

Se determinarmos a força aplicando o operador $\overline{\nabla}$ obtemos

$$\begin{aligned}
\mathbf{f_{int}} &= -\overline{\nabla} v_{int} \\
&= \frac{d}{ds}\frac{\partial v_{int}}{\partial c'} \\
&= k\frac{d}{ds}\frac{c'}{||c'||} \\
&= k\left(\frac{c''}{||c'||} - \frac{c'.c'.c''}{||c'||.||c'||^2}\right) \\
&= \frac{k}{||c'||}\left(c'' - \frac{c''.c'}{||c'||^2}c'\right). \qquad (4.25)
\end{aligned}$$

Esta expressão apresenta certas semelhanças com a do vector curvatura $\mathbf{k}$, cuja definição pode ser consultada em [39] e que é dado por

$$\mathbf{k} = \frac{1}{||c'||^2}\left(c'' - \frac{c''.c'}{||c'||^2}c'\right). \tag{4.26}$$

Para qualquer ponto de uma curva, este vector aponta para o centro do circulo que é tangente à curva nesse ponto e tem comprimento igual ao raio desse mesmo circulo.

Podemos então reescrever 4.25

$$\mathbf{f}_{int} = k||c'||\mathbf{k}(s). \tag{4.27}$$

Daqui podemos concluir que a força $\mathbf{f}_{int}$ têm a direcção d $\mathbf{k}$ mas é escalada pelo módulo da deformação local do elástico, $||c'||$. Isto significa que a grandeza da força ainda depende do alongamento sofrido pelo elástico e o problema que pretendíamos resolver ainda se mantém. Isto pode parecer um contra-senso, pois se a tensão no elástico é constante como é que a força interna varia? Isto justifica-se da seguinte forma: as forças exercida sobre uma partícula pelas partículas vizinhas são iguais em módulo mas quando o elástico é "dobrado"estas não apresentam sentidos opostos o que faz com que a soma não seja nula.

Movimentação de partículas

Como já vimos temos ainda uma força que varia com a deformação do elástico e que não tem qualquer componente tangencial ao caminho como se pode verificar pelo produto interno entre a força num ponto e c'

$$\begin{aligned}\mathbf{f}_{int}.\mathbf{c}' &= \frac{k}{||c'||}\left(c''.c' - \frac{c'.c''}{||c'||^2}.c'.c'\right)\\ &= \frac{k}{||c'||}\left(c''.c' - \frac{c'.c''}{||c'||^2}.||c'||^2\right)\\ &= 0\end{aligned} \tag{4.28}$$

Isto faz com que não exista uma configuração estável em regime permanente para o elástico. A força externa pode empurrar continuamente partículas do elástico ao longo do caminho tornando-o cada vez mais fino em algumas regiões.

Para restringir o movimento das partículas ao longo do elástico podemos adicionar uma força que contrarie este efeito. Assim esta força deverá ser igual e de sentido contrário à componente da força externa ao longo do elástico.

$$\mathbf{f}_{cont} = -\frac{\mathbf{f}_{ext}.c'}{||c'||^2}.c' \tag{4.29}$$

Esta força não vai alterar a energia do elástico pois não efectua trabalho uma vez que as partículas não se movem por acção desta.

Modelo Final

O modelo final consiste na descrição de uma trajectória através de uma função contínua $c(s)$ cujo parâmetro $s \in [0, 1]$. A força aplicada a cada partícula do elástico é dada por

$$f(s) = \mathbf{f}_{int}(s) + \mathbf{f}_{ext}(s) + \mathbf{f}_{cont}(s),$$

e que consiste na soma das três componentes $\mathbf{f}_{int}, \mathbf{f}_{ext}, \mathbf{f}_{cont}$ e que são as funções força interna, externa e restrictiva, respectivamente. Estas funções são definidas por

$$\begin{aligned} \mathbf{f}_{int} &= k||c'||\mathbf{k}(s), \\ \mathbf{f}_{ext} &= -\nabla v_{ext}(c), \\ \mathbf{f}_{cont} &= -\frac{\mathbf{f}_{ext}.c'}{||c'||^2}c', \end{aligned}$$

onde k é a constante de elasticidade, $\mathbf{k}(s)$ é o vector curvatura em s e v_{ext} é uma função potencial definida sobre o espaço das configurações do robô.

As forças interna e externa são definidas com base em funcões (de funções) potencial e controlam o movimento das partículas do elástico de acordo com as propriedades deste e mantendo o afastamento desejado dos obstáculos. A força de restrição anula a componente tangencial da força externa evitando desta forma que as partículas sejam deslocadas ao longo do elástico.

O movimento das partículas é aproximado por um movimento pseudo-estático que é dado pela relação

$$\Delta c \propto \mathbf{f}$$

.

É ainda utilizado um pequeno truque para manter o alongamento das partículas independente da força externa aplicada e que consiste em adicionar partículas em zonas onde o elástico "esteja muito esticado"e remover partículas nas zonas onde a densidade das mesmas é muito elevada.

4.6.2 Bolhas de Espaço Livre

Para a navegação de um robô móvel é necessário ter uma noção de quais as zonas do espaço circundante que não estão ocupadas por quaisquer obstáculos ou objectos e que, por isso, são normalmente designadas por zonas livres. Essa necessidade é importante sobretudo quando pretendemos fazer um robô, com restrições holonómicas, seguir uma trajectória, pois permite-nos saber se é possível efectuar num determinado ponto a rotação necessária (por exemplo) para que se possa continuar a seguir a trajectória.

A determinação de todo o espaço livre pode, dependendo da representação utilizada e da dimensão do espaço de configurações utilizado, ser extremamente pesada do ponto de vista computacional o que impedirá a sua utilização em aplicações de tempo-real com tempos de reacção curtos.

Quinlan optou por uma solução mais simples que consistiu em determinar e representar apenas uma parte do espaço livre em vez do espaço total. Esse subconjunto do espaço livre é composto por porções chamadas bolhas que são determinadas em torno de algumas configurações do robô.

Assim, sendo β_q uma bolha em torno da configuração q, esta é determinada com base na distância entre o robô na configuração q e os obstáculos existentes no ambiente circundante. O cálculo das bolhas para um caminho $c(s)$ pode ser obtido começando por determinar a bolha correspondente à configuração inicial β_{q_0}. Para isso determina-se a menor distância entre a referida configuração e os obstáculos circundantes, esta distância será o raio do círculo que limita as fronteiras da bolha β_0. É fácil perceber que existe pelo menos um ponto do caminho que está contido em β_{q_0}. A determinação da bolha seguinte é feita partindo de $c(0)$ até encontrar um ponto da trajectória $c(s_i)$ tal que, $c(s_i) \notin \beta_{q_0}$. As restantes bolhas são determinadas de forma semelhante partindo sempre da última bolha encontrada. É fácil perceber-se também que, a menos que algumas das bolhas pudessem ter raio zero, a intersecção entre duas bolhas consecutivas é sempre não nula.

4.6.3 Implementação e Resultados

Dado um caminho definido por pontos de passagem normalmente unidos por segmentos de recta começa-se por definir as bolhas destes pontos com base na distância ao obstáculo mais próximo. De seguida começando do ponto de partida são criadas bolhas intermédias de forma a que todo o caminho seja coberto por estas. Tal como já foi explicado anteriormente, cada uma das bolhas representa uma porção de espaço livre em torno da configuração respectiva.

Para calcular o movimento das partículas é necessário determinar antes de mais a força aplicada sobre cada uma delas, sendo esta a soma vectorial da força de contracção interna, da força externa produzida pelos obstáculos e da força de restrição ao movimento das partículas ao longo do elástico.

Como o elástico é representado discretamente por um número finito de pontos, os potenciais utilizados para derivar as forças são funções de um número finito de parâmetros pelo que é possível utilizar o operador gradiente ∇ para determinar essas forças.

Força de Contracção

No modelo contínuo o potencial interno é definido por

$$V_{int}[c] = k_c \int_0^1 ||c'||ds \tag{4.30}$$

onde k_c é a constante de elasticidade. Se os valores de s para as várias configurações $q_1, q_2, ..., q_n$ estiverem igualmente espaçados de um valor h, a formulação discreta poderá ser

$$\begin{aligned} V_{int}(q_1, q_2, ..., q_n) &= k_c \sum_{i=1}^{n-1} h \frac{||\mathbf{q}_{i+1} - \mathbf{q}_i||}{h} \\ &= k_c \sum_{i=1}^{n-1} ||\mathbf{q}_{i+1} - \mathbf{q}_i||. \end{aligned} \tag{4.31}$$

A força de contracção exercida sobre na partícula $\mathbf{q}_i$ pode ser determinada por

$$\begin{aligned} \mathbf{f}_{int}(i) &= -\nabla_{q_i} V_{int} \\ &= k_c \frac{\partial}{\partial \mathbf{q}_i} (||\mathbf{q}_{i+1} - \mathbf{q}_i|| + ||\mathbf{q}_i - \mathbf{q}_{i-1}||) \\ &= k_c \left(\frac{\mathbf{q}_{i+1} - \mathbf{q}_i}{||\mathbf{q}_{i+1} - \mathbf{q}_i||} + \frac{\mathbf{q}_{i-1} - \mathbf{q}_i}{||\mathbf{q}_{i-1} - \mathbf{q}_i||} \right) \end{aligned} \tag{4.32}$$

Como se pode ver nesta expressão sobre cada partícula é exercida duas forças de grandeza k_c em direcção a cada uma das suas partículas vizinhas.

A força externa

Tal como no caso anterior partimos da expressão do potencial no modelo contínuo

$$V_{ext}(c) = \int_0^1 v_{ext}(c) ds \tag{4.33}$$

e obtemos um equivalente discreto dado por

$$V_{ext}(q_1, q_2, ..., q_n) = \sum_{i=1}^{m} h v_{ext}(\mathbf{q}_i). \tag{4.34}$$

Note-se que neste caso e ao contrario do potencial interno, V_{ext} depende do valor de h. No entanto e porque a definição do intervalo de valores para s é arbitrária, em vez de fazermos $s \in [0, 1]$ podemos definir outro intervalo qualquer. Assim e sendo

os valores de s igualmente espaçados para configurações consecutivas podemos fazer $h = 1$ e logo vem $s \in [0, n-1]$.

Falta-nos agora definir v_{ext} que poderá ser qualquer função tal que a força derivada desta vá repelir os pontos do elástico dos obstáculos dando origem a caminhos suaves. Uma possibilidade é a de utilizar uma função potencial semelhante à do potencial eléctrico gerado por uma carga $v_e = q_e/(4\pi\varepsilon_0 d)$ onde q_e é o valor da carga eléctrica, ε_0 é a permitividade do vazio e d a distância entre a carga e o ponto em que queremos saber o valor do potencial. Neste caso a força exercida sobre uma segunda carga q_0 vem

$$\mathbf{f} = \frac{q_e q_0}{4\pi\varepsilon_0 d^2} \frac{\mathbf{d}}{||\mathbf{d}||}, \tag{4.35}$$

onde d é o vector que une a carga q_e à carga q_0 devendo notar-se ainda que $d = ||\mathbf{d}||$. Podemos concluir que utilizando um potencial deste tipo os pontos do elástico nunca tocariam nos obstáculos uma vez que à medida que a d tende para zero, o valor da força tende para infinito. Isto apresente alguns problemas de implementação devido a dificuldades numéricas, por essa razão optou-se por uma definição diferente para a função potencial dada por

$$v_{ext}(\mathbf{q}) = \begin{cases} \frac{1}{2}k_r(d_0 - d(\mathbf{q}))^2 & \text{se} \quad d(\mathbf{q}) < d_0 \\ 0 & \text{se} \quad d(\mathbf{q}) \geq d_0 \end{cases} \tag{4.36}$$

onde k_r é uma constante e d_0 é a distância máxima de influência dos obstáculos.

Podemos facilmente verificar que esta função potencial é limitada variando desde 0 a uma distância maior ou igual ao limiar de influência d_0 até $k_r \times d_0$ junto aos obstáculos.

Dado que os obstáculos não são na realidade pontuais seria necessário integrar a expressão 4.36 ao longo da superfície de cada um dos obstáculos para obter o potencial em cada ponto do elástico o que seria bastante difícil e pesado computacionalmente. Por isso considerámos apenas que o ponto mais próximo dos obstáculos e um determinado ponto do elástico é que vai ter influência sobre este. Assim $d(\mathbf{q})$ é a menor das distâncias entre o ponto $\mathbf{q}$ e os obstáculos pelo que a força f_{ext} vem

$$\begin{aligned} \mathbf{f}_{ext}(\mathbf{q}) &= -\nabla v_{ext} \\ &= -\nabla \left(\frac{1}{2} k_r (d_0 - d(\mathbf{q}))^2 \right). \end{aligned} \tag{4.37}$$

Dado que o que nos interessa são as componentes da força temos

$$\begin{cases} f_{x_{ext}} &= -\frac{1}{2}k_r \frac{\partial\left((d_0 - d(\mathbf{q}))^2\right)}{\partial x} \\ f_{y_{ext}} &= -\frac{1}{2}k_r \frac{\partial\left((d_0 - d(\mathbf{q}))^2\right)}{\partial y} \end{cases}$$

ou seja

$$\begin{cases} f_{x_{ext}} &= -\frac{1}{2}k_r\left(-2\frac{\partial d(\mathbf{q})}{\partial x}\right)(d_0 - d(\mathbf{q})) \\ f_{y_{ext}} &= -\frac{1}{2}k_r\left(-2\frac{\partial d(\mathbf{q})}{\partial y}\right)(d_0 - d(\mathbf{q})) \end{cases}$$

de onde vem

$$\begin{cases} f_{x_{ext}} &= k_r\frac{x_0-x}{\sqrt{(x_0-x)^2+(y_0-y)^2}}(d_0 - \sqrt{(x_0-x)^2+(y_0-y)^2}) \\ f_{y_{ext}} &= k_r\frac{y_0-y}{\sqrt{(x_0-x)^2+(y_0-y)^2}}(d_0 - \sqrt{(x_0-x)^2+(y_0-y)^2}). \end{cases} \tag{4.38}$$

Analisando as expressões 4.38 verificamos que os termos

$$\frac{x_0 - x}{\sqrt{(x_0-x)^2+(y_0-y)^2}}$$

e

$$\frac{y_0 - y}{\sqrt{(x_0-x)^2+(y_0-y)^2}}$$

não são mais do que as componentes de um vector unitário com a direcção do vector distância $\mathbf{d}$. Por isso podemos reescrever 4.38 como

$$\mathbf{f}_{ext} = \frac{\mathbf{d}}{||\mathbf{d}||}(d_0 - ||\mathbf{d}||) \tag{4.39}$$

Força de Restrição ao Movimento das Partículas ao Longo do Elástico

Tal como já foi referido anteriormente e se verificou na implementação efectuada, é necessário introduzir uma força que evite que a força externa transporte partículas de uma zona do elástico para outra. Por outras palavras é necessário introduzir uma força de restrição que anule a componente da força externa que tangencial ao elástico. Esta força é obtida pela expressão seguinte

$$\mathbf{f}_{rest} = -\frac{\mathbf{f}_{ext}.\mathbf{c}'}{||\mathbf{c}'||^2}.\mathbf{c}' \tag{4.40}$$

onde como sabemos $\mathbf{c}'$ é um vector tangente ao caminho, e onde, para efeitos de implementação do modelo discreto, a derivada de $\mathbf{c}$ foi aproximada por

$$\mathbf{c}' = \left(\frac{\mathbf{q}_{i+1} - \mathbf{q}_i}{||\mathbf{q}_{i+1} - \mathbf{q}_i||} + \frac{\mathbf{q}_i - \mathbf{q}_{i-1}}{||\mathbf{q}_i - \mathbf{q}_{i-1}||}\right). \tag{4.41}$$

Calculadas as forças resta agora determinar a força total aplicada a cada uma das partículas que, como sabemos, é $\mathbf{f}_{total} = \mathbf{f}_{int} + \mathbf{f}_{ext} + \mathbf{f}_{rest}$.

Modificar o Elástico

Vimos até agora como é possível construir um elástico usando o conceito de bolhas de espaço livre, assim como calcular as forças sobre cada uma das partículas que compõem o elástico, vamos ver agora como é modificado o elástico por forma a que este se vá ajustando ao contorno dos obstáculos criando um caminho suave em torno destes. Isto é feito percorrendo todas as partículas do elástico, num sentido e no outro, calculando as forças para cada uma delas e actualizando a sua posição utilizando a aproximação pseudo-estática anteriormente referida,

$$\mathbf{q}_i' = \mathbf{q}_i + \alpha \mathbf{f}_{total}. \tag{4.42}$$

Note-se que é necessário ir verificando se as bolhas vizinhas se intersectam pois caso contrário será necessário adicionar algumas para que esta condição se verifique. Por outro lado, é também necessário ir verificando se existem bolhas em excesso e neste caso removê-las para melhorar a eficiência do algoritmo.

Este tipo de implementação apresenta dois tipos de problemas. O primeiro surge devido à forma como é obtida a nova configuração através de 4.42 que tende a ser instável ou seja o elástico pode ficar a oscilar e nunca se encontrar um conjunto de configurações estáveis. O segundo problema tem a ver com a não existência de garantias de que a nova posição obtida para uma determinada partícula pertença ao espaço livre. Aqui a solução é mais simples podendo-se restringir que a nova posição seja escolhida na direcção da força, de acordo com 4.42, mas que fique no interior da bolha correspondente à posição anterior.

Resultados Obtidos

A figura 4.23 mostra a evolução de um elástico partindo de uma configuração inicial que poderia ter sido gerada por um planeador de trajectórias e que como se pode ver vai-se adaptando aos campos de potencial gerados pelos obstáculos minimizando o espaço a percorrer e produzindo um caminho suave em torno destes. Pode-se ver que o obstáculo à direita é um obstáculo que, não sendo estático, vai rodando em torno de uma das suas extremidades e que o elástico vai sendo modificado, adaptando-se às novas posições do obstáculo. Nesta figura são mostrados os vectores de distância mínima entre os obstáculos e cada uma das configurações que compõem o elástico. Como se pode verificar pela expressão 4.39, a força externa aplicada tem a direcção deste vector quando a distância é inferior a um limiar d_0, sendo nula nos restantes casos.

A figura 4.24 representa a evolução das bolhas de espaço livre correspondentes às mesmas situações ilustradas pela figura 4.23.

Como se pode verificar pelos resultados apresentados este método de adaptação de trajectórias funciona bem em ambientes onde existam obstáculos móveis desde que

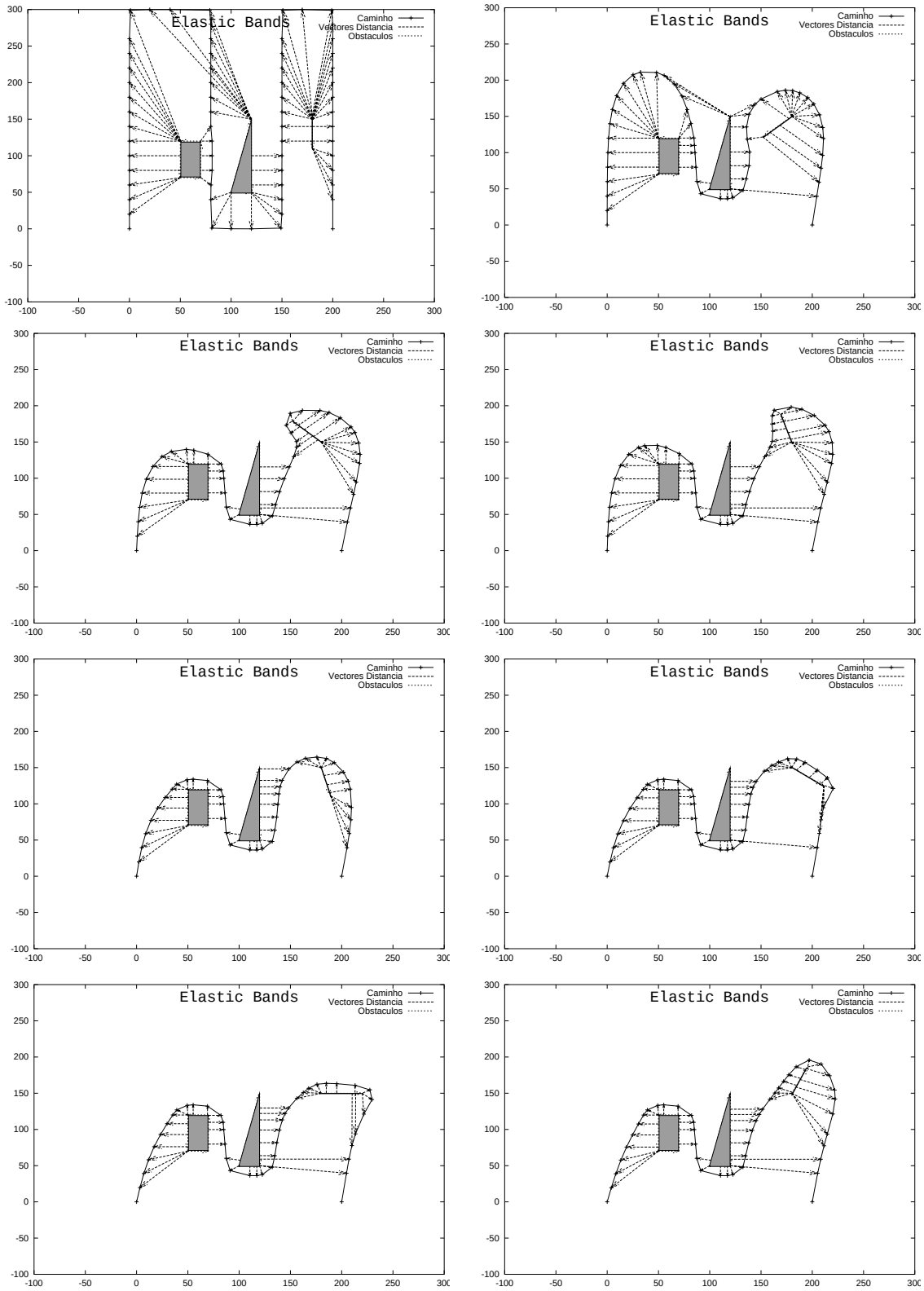

Figura 4.23: Evolução de uma banda elástica a partir de um caminho predefinido existindo um obstáculo móvel.

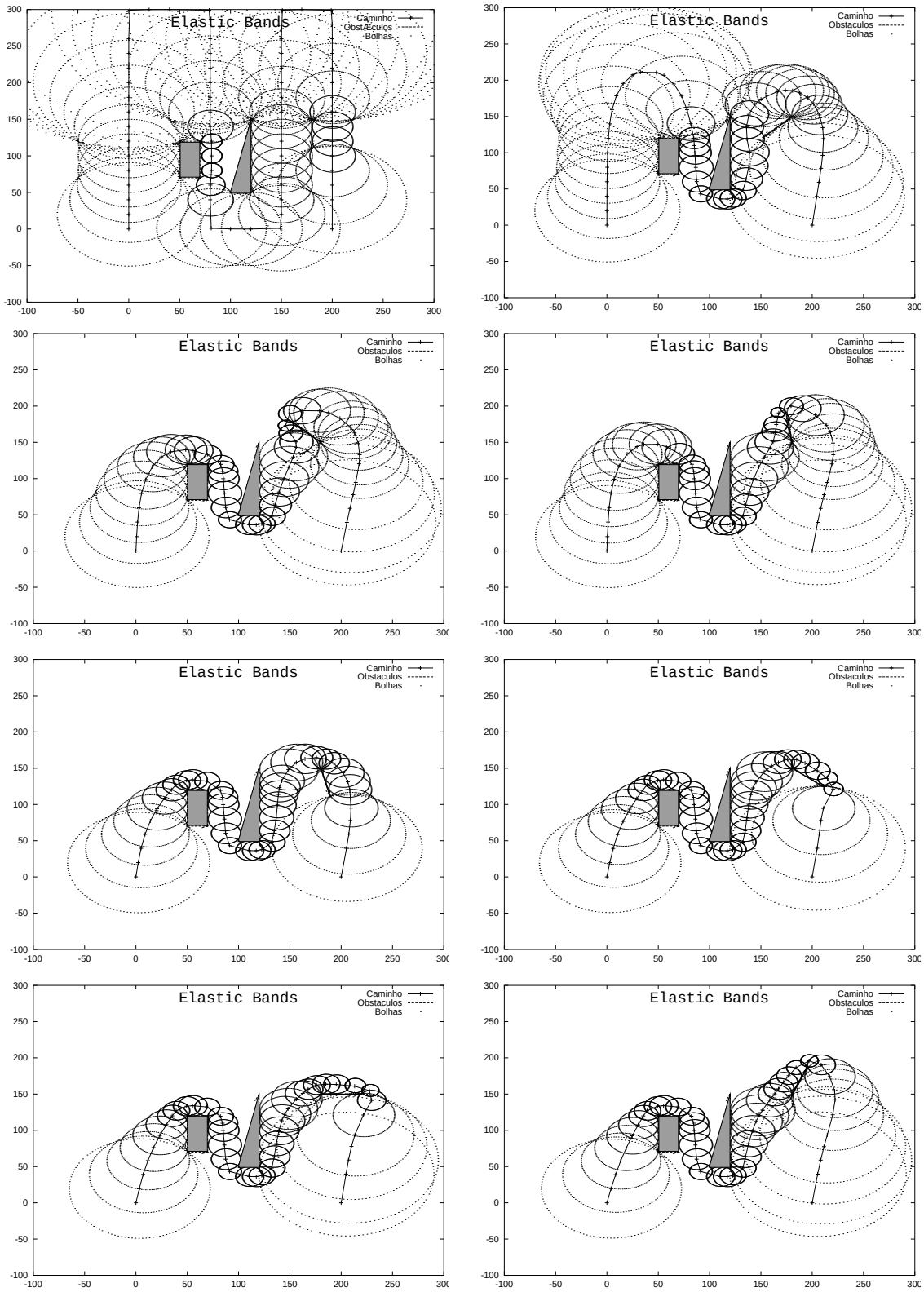

Figura 4.24: Evolução das bolhas de uma banda elástica a partir de um caminho predefinido existindo um obstáculo móvel

esses possam ser detectados. Sempre que um obstáculo seja deslocado ou introduzido de forma a bloquear completamente uma passagem o que se irá traduzir pela quebra do elástico é necessário reutilizar um planeador para gerar um caminho alternativo. Verificou-se ainda que a implementação das bandas elásticas é relativamente simples e o peso computacional imposto por este método não coloca problemas de maior para um funcionamento em tempo-real com os processadores actualmente no mercado.

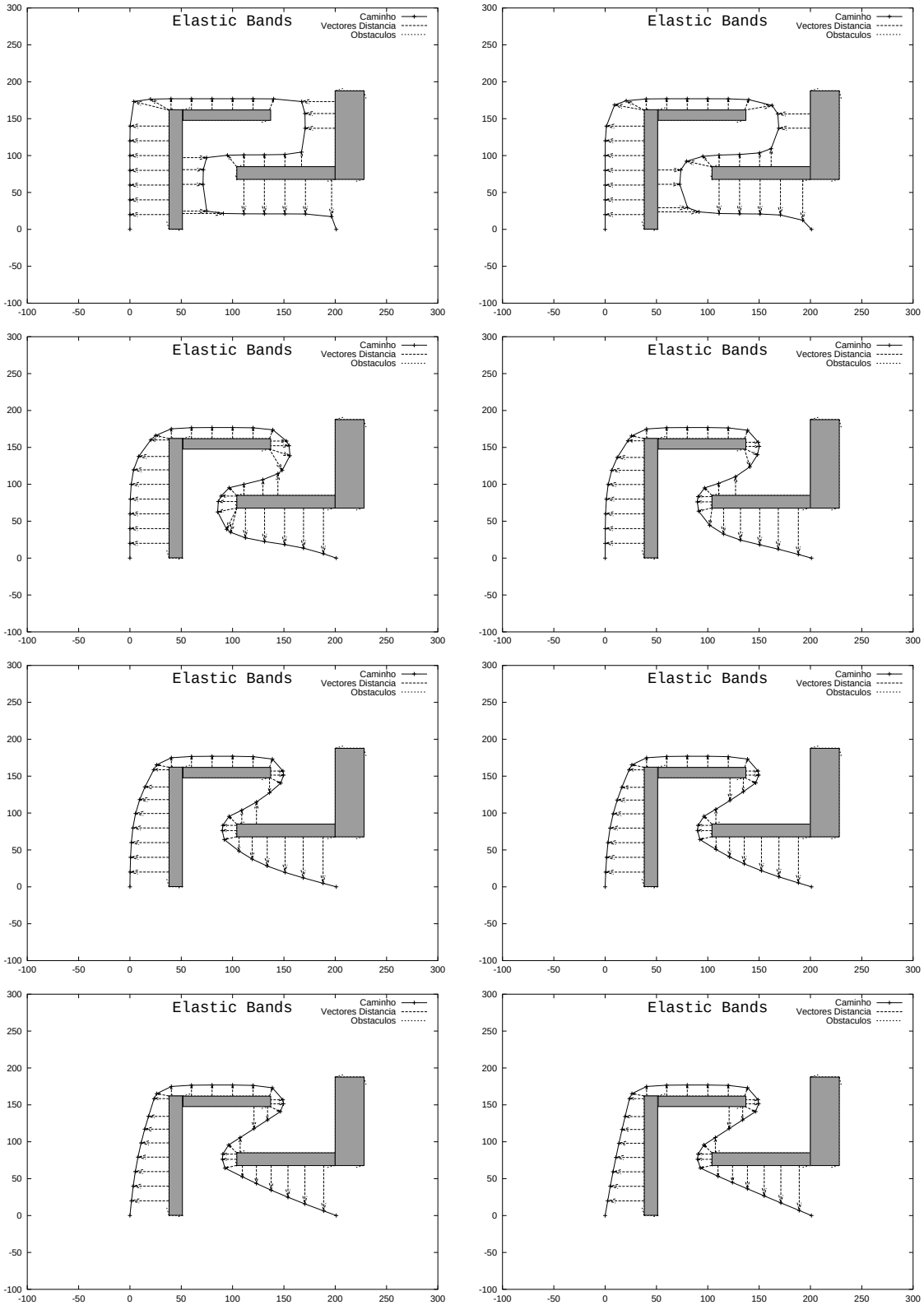

Figura 4.25: Evolução de uma banda elástica a partir de um caminho predefinido.

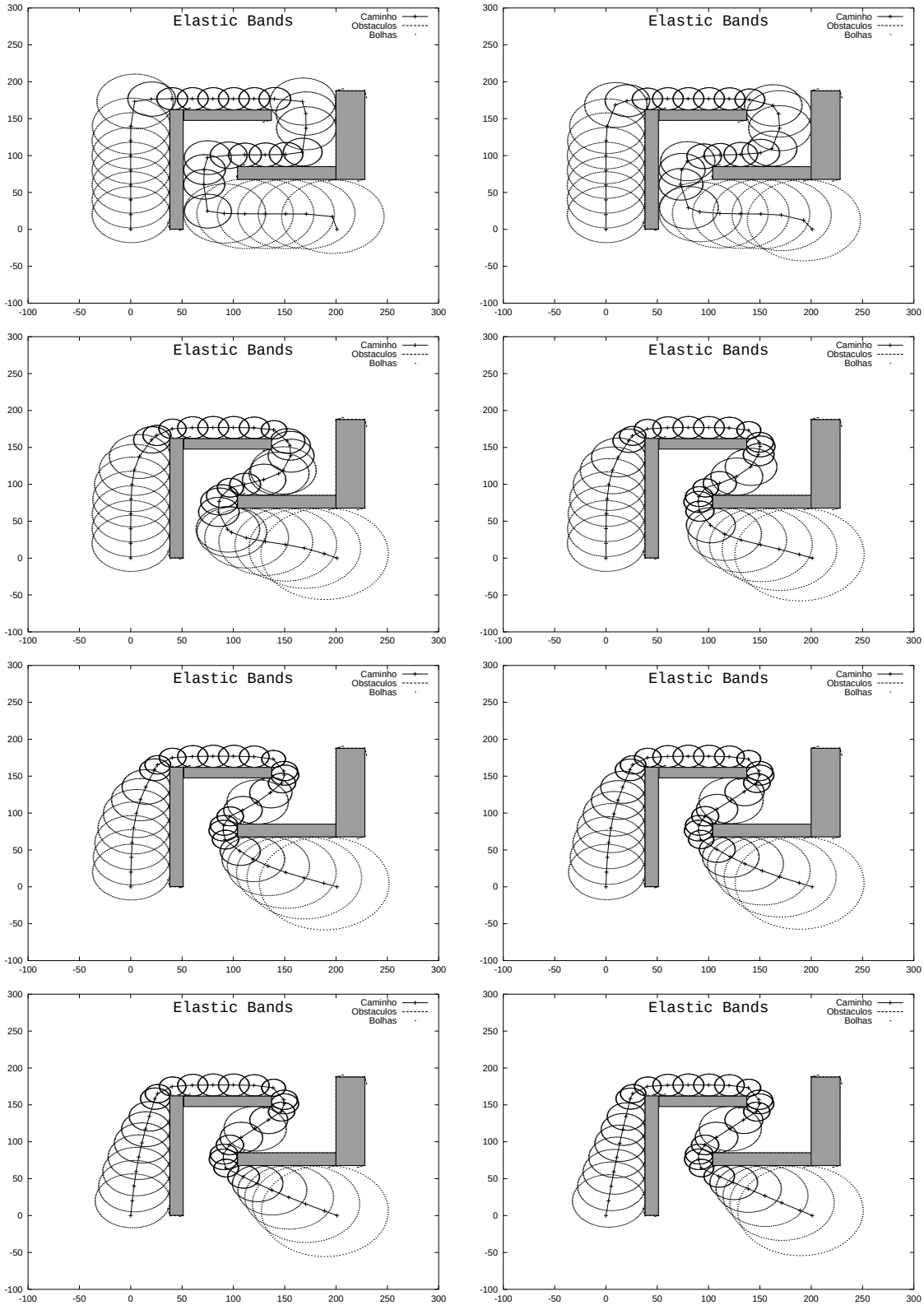

Figura 4.26: Evolução das bolhas de uma banda elástica a partir de um caminho predefinido.

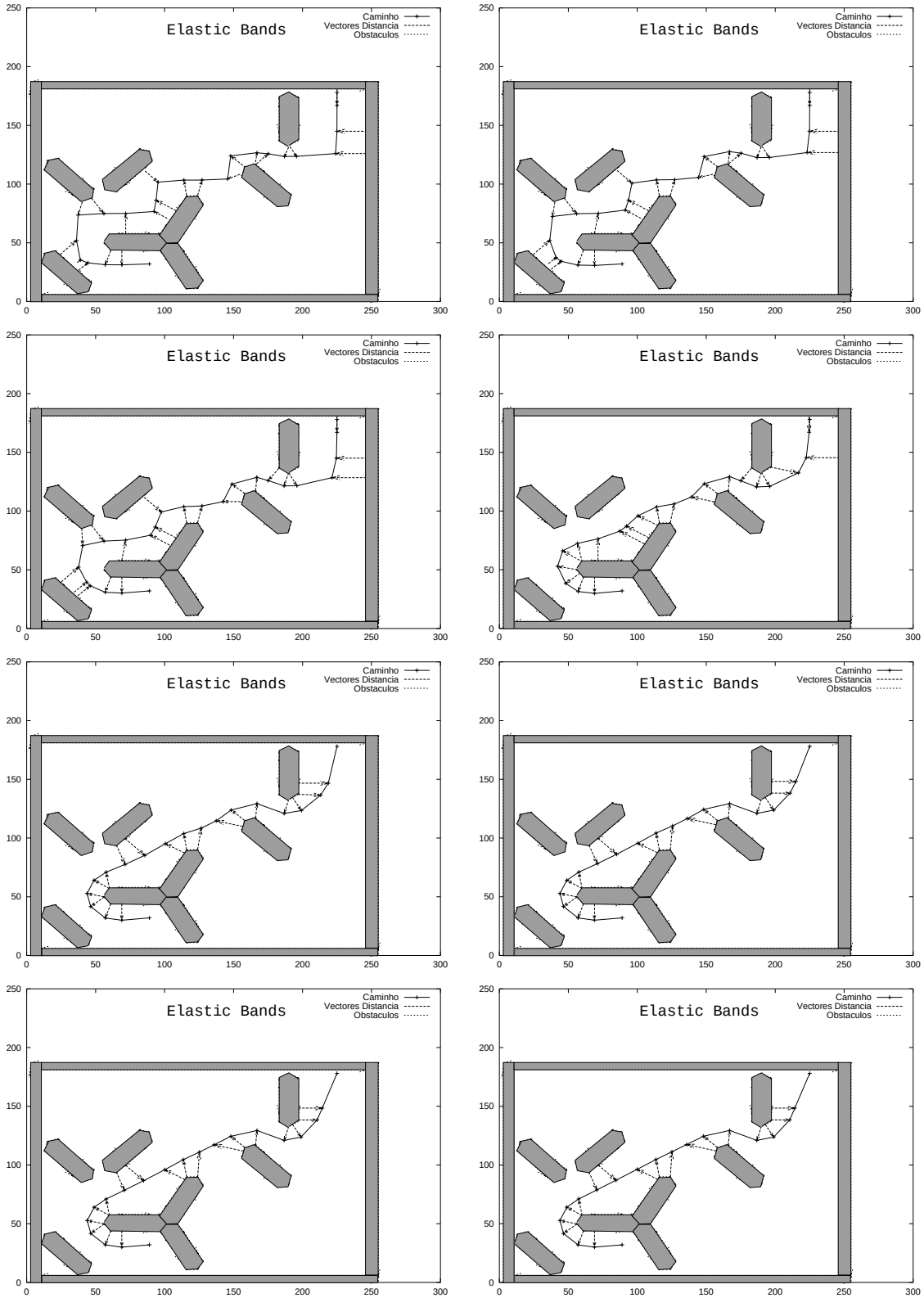

Figura 4.27: Evolução de uma banda elástica a partir de um caminho predefinido.

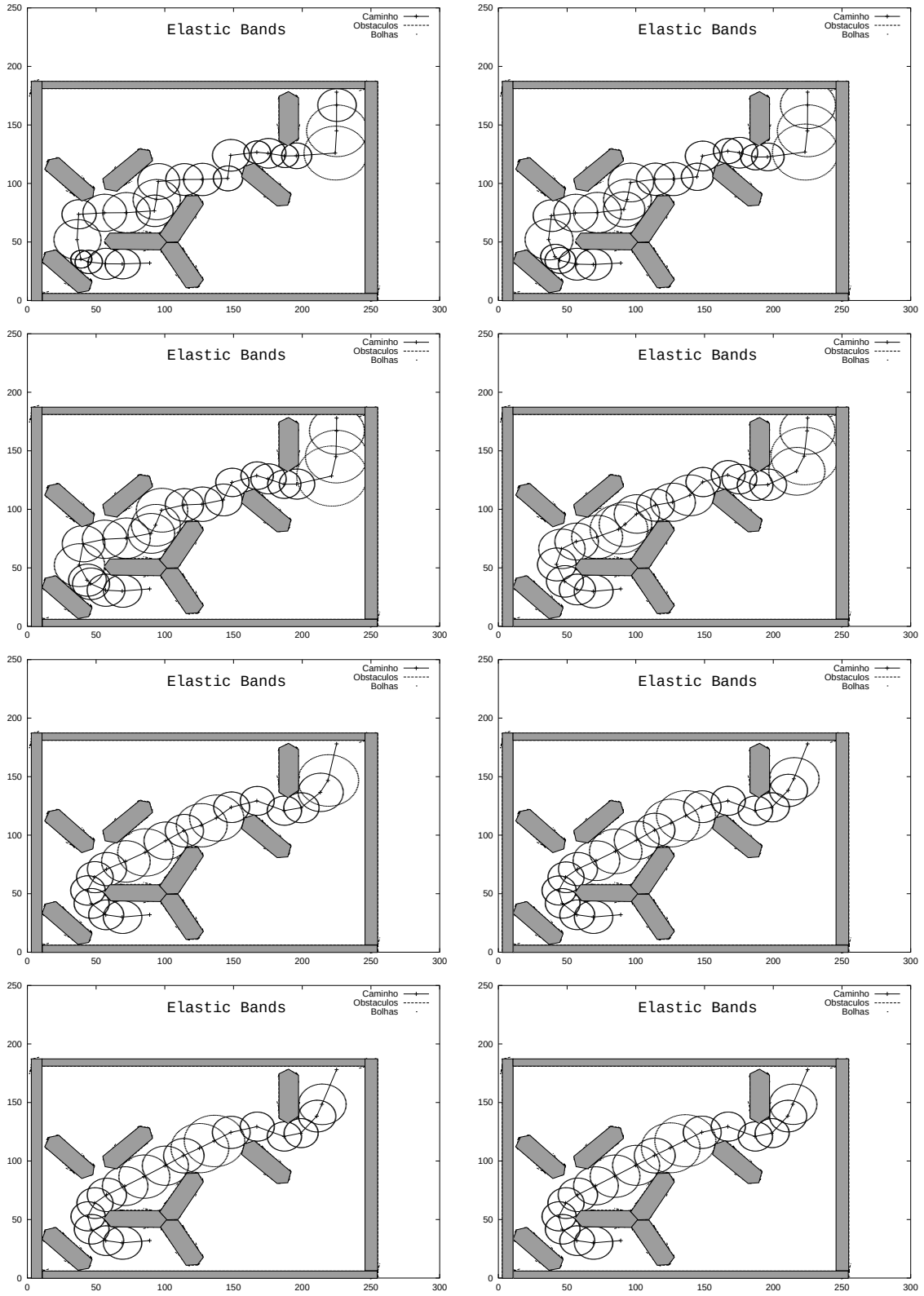

Figura 4.28: Evolução das bolhas de uma banda elástica a partir de um caminho predefinido.

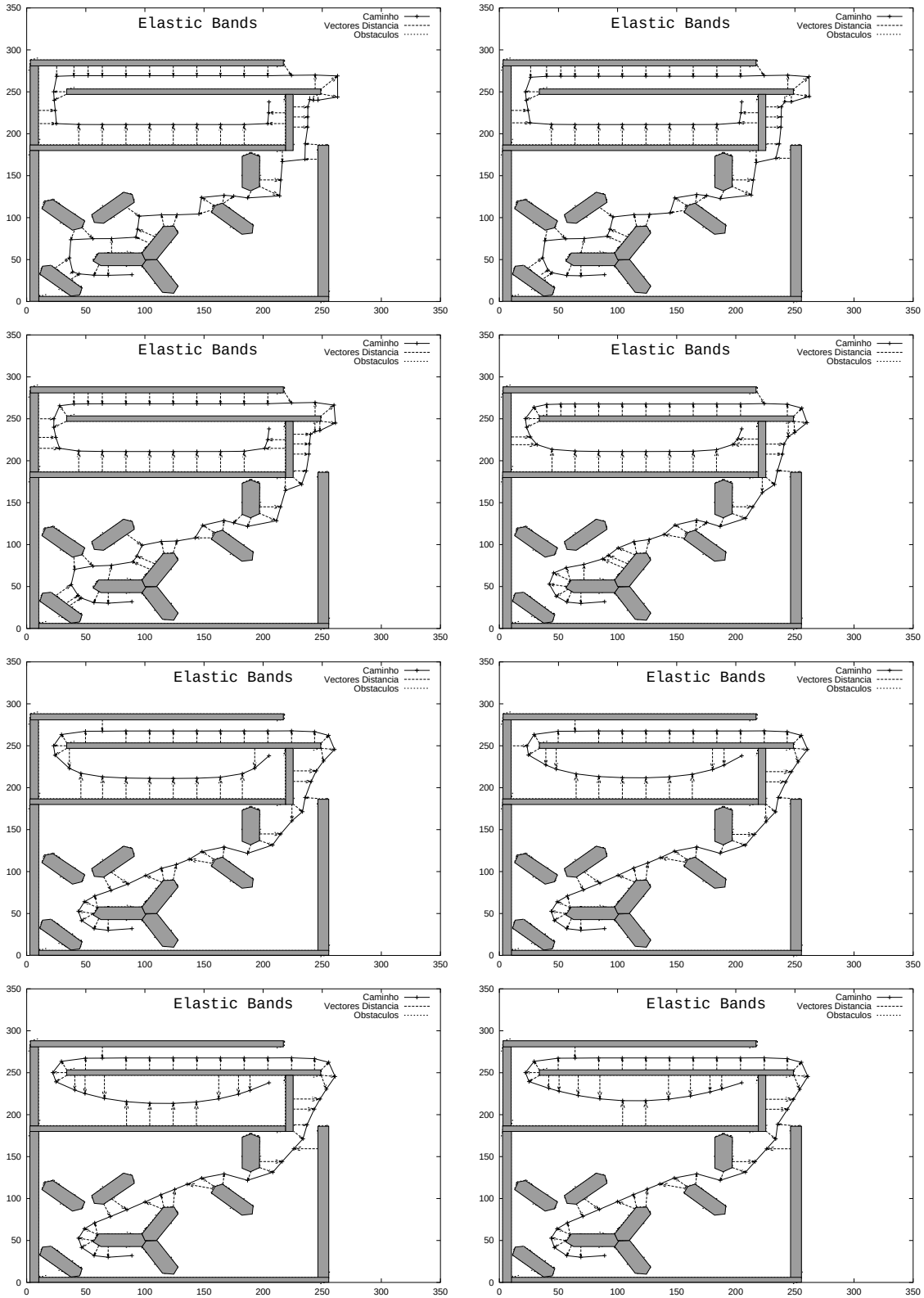

Figura 4.29: Evolução de uma banda elástica a partir de um caminho predefinido.

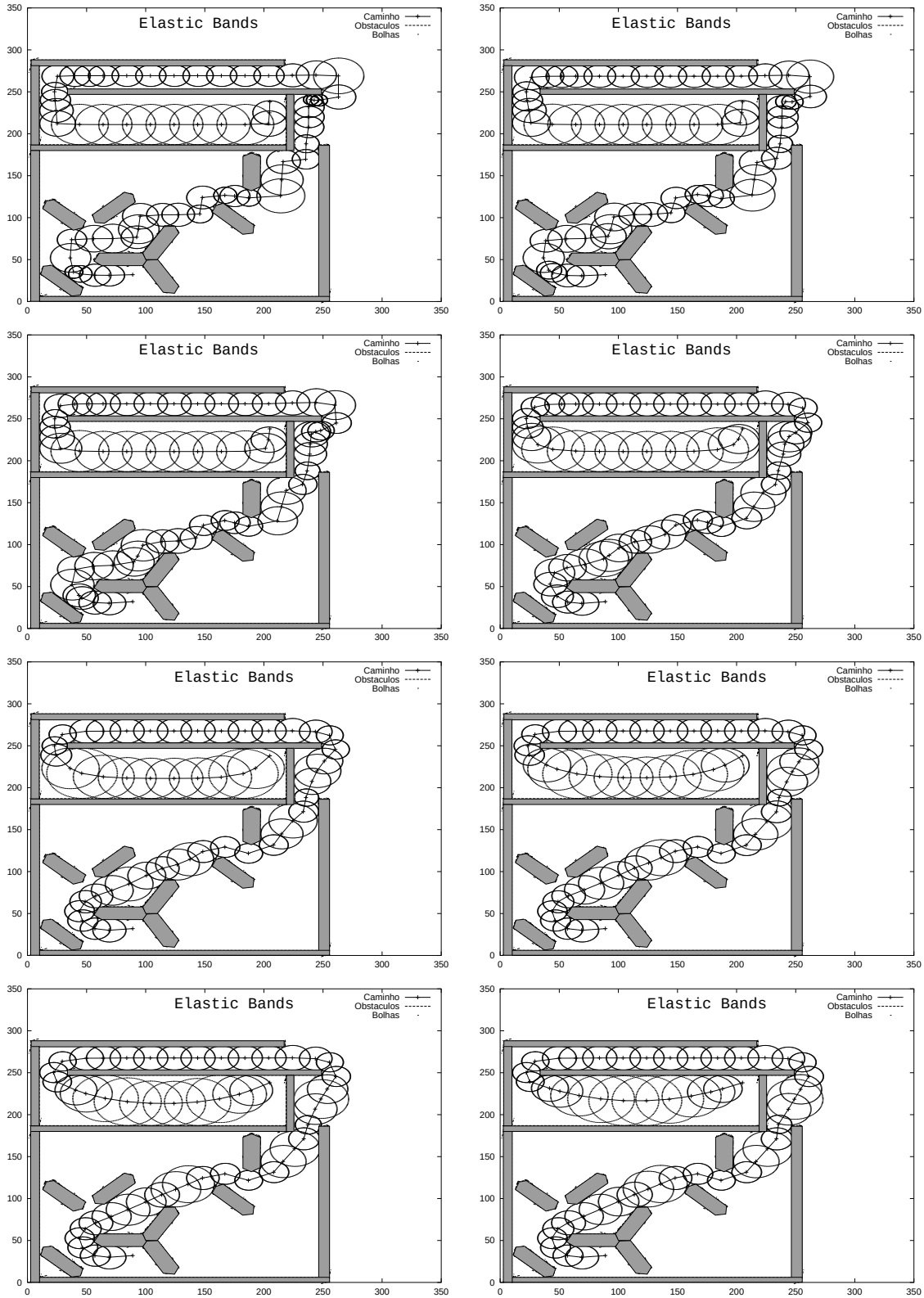

Figura 4.30: Evolução das bolhas de uma banda elástica a partir de um caminho predefinido.

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Considerações finais

AO longo desta dissertação procurou-se descrever o trabalho do autor que começou com o estudo da detecção de obstáculos com vista à aplicação no projecto Po-Robot[33] (Multi-purpose Portuguese Flexible Mobile Robot). Este projecto, que foi patrocinado pela NATO no âmbito do programa Science for Stability, envolveu equipas dos três pólos do ISR (Lisboa, Porto e Coimbra). Foi exactamente na primeira fase deste projecto que se verificou a necessidade de desenvolver um sistema que permitisse as comunicações e controlo entre uma estação de trabalho e a plataforma devido às deficiências apontadas na secção 2.2. Os resultados mostraram a vantagem da utilização de tal sistema, pois o desenvolvimento de aplicações passou a ser uma tarefa mais simples e reduziu-se o próprio ciclo de desenvolvimento uma vez que depois de fazer uma alteração a uma aplicação deixou de existir a fase de carregamento da mesma para o computador de bordo da plataforma, que era de alguns minutos pois era feita através da ligação série a 9600bps. Por outro lado e tal como foi referido na mesma secção, com este ambiente de trabalho deixaram de existir as limitações

de memória e de potência computacional impostas pelo controlador da plataforma, passando estes limites a ser os da estação de trabalho que como se sabe apresentam capacidades cada vez maiores e a preços relativamente baixos. Este ambiente permitiu ainda o teste de aplicações de controlo remoto da plataforma como a que é descrita em [13] e que relata o desenvolvimento de um sistema de tele-operação capaz de ser utilizado em qualquer local visto que se baseia num conjunto de "applets[1]" em JAVA[2] que requerem apenas um "browser[3] " com suporte para esta linguagem.

Ainda numa fase inicial foi feito o estudo dos sensores de ultra-sons tendo-se verificado a escassez de informação contida nas suas leituras. Foi por esta razão que se procurou uma abordagem estocástica [28, 30, 29] e cujos resultados obtidos mostraram que seria útil para a criação de descrições do ambiente e de nomeadamente fornecer descrições de obstáculos não conhecidos "a priori". As descrições obtidas, não sendo de grande precisão possibilitam a navegação da plataforma sem colidir com os obstáculos. Verificou-se na utilização desta abordagem a necessidade da existência de um sistema de localização fiável pois só assim é possível situar correctamente um obstáculo detectado e que o sistema odométrico utilizado pela plataforma, apesar de ser simples, necessita de ser corrigido frequentemente utilizando referências externas que permitam a obtenção de valores deposição absolutos e assim eliminar os erros que a estimação da posição vai acumulando.

A primeira abordagem estudada para a detecção de obstáculos e sugerida por Alberto Elfes, mostrou-se de difícil realização devido ao peso computacional requerido. Por esta razão, foi utilizado um método mais simples e que em vez de fazer a estimação do estado de ocupação das células da grelha com base na probabilidade de estas estarem ocupadas e/ou vazias, baseava essa estimação num histograma de acumulação de leituras. Este método mostrou-se eficaz e possível de ser utilizado numa máquina com uma capacidade de cálculo média (ex. MicroSparc 50MHz).

Numa fase posterior ao projecto Po-Robot, fizeram-se alguns estudos sobre métodos de navegação autónoma que se encontram descritos no capítulo 4. Os métodos baseados na utilização de campos de potencial artificiais mostraram-se simples e eficazes na condução da plataforma móvel. Estes métodos apresentam a limitação comum conhecida como "poços de potencial" e que não é mais do que a existência de pontos

[1] "applets" nome dado pela Sun Microsystems às aplicações de JAVA que necessitam de um "browser" para a sua execução.

[2] JAVA - linguagem de programação orientada a objectos, desenvolvida pela Sun Microsystems e em que as aplicações que resultam da sua aplicação correm numa máquina virtual conhecida como JAVA Machine. Desta forma uma aplicação desenvolvida nesta linguagem pode ser executada em qualquer arquitectura desde que exista a Java Machine para essa arquitectura.

[3] "Browser" é um nome genérico que designa aplicações como Netscape, MS Internet Explorer, Mosaic, entre outros e que permite o acesso a servidores de hipertexto através da rede Internet. Algumas destas aplicações contêm a JAVA Machine o que lhes permite executar applets contidos numa página HTML-Hypertext Markup Language.

no espaço onde o potencial apresenta mínimos locais, o que faz com que a plataforma não chegue ao seu destino uma vez que a força que a conduz se anula nestes pontos. Apesar destas limitações, estes métodos, por apresentarem uma característica reactiva, podem ser muito úteis como módulos de condução entre pontos intermédios de trajectos pré-planeados, desde que estes pontos intermédios sejam escolhidos de forma a não existam os referidos "poços de potencial" entre um ponto intermédio de partida e o ponto intermédio destino. No entanto esta situação nem sempre se verifica, pois apesar de os pontos intermédios serem escolhidos de acordo com a disposição dos obstáculos conhecidos pelo planeador, outros obstáculos que sejam detectados durante a execução da trajectória também podem criar "poços de potencial".

O método das bandas elásticas mostrou ser um método simples de utilizar, com requisitos computacionais moderados o que permite a sua utilização em tempo-real, e ultrapassa o problema dos mínimos locais. Este método é particularmente útil quando os obstáculos são descritos pelas primitivas geométricas da sua fronteira uma vez que isto facilita a determinação da menor distância entre um obstáculo e o robô.

Foram ainda feitas outras experiências que não se encontram descritas nesta dissertação mas que podem ser consultadas nas referências bibliográficas e que foi a utilização se redes neuronais para controlar a navegação da plataforma móvel [6, 4, 1, 3, 5] utilizando como base a estrutura da grelha de ocupação e o cálculo da grelha de potencial descrito na secção 4.4 e ainda um sistema de planeamento de trajectórias com base no conhecimento de características do ambiente conhecidas ("landmarks")[8].

Em termos de trabalho futuro são muitos os caminhos possíveis. Do ponto de vista da construção de mapas é muito importante a utilização de um sistema de localização preciso, pois só desta forma é possível estimar correctamente a localização dos obstáculos detectados. Por outro lado, a utilização de outros tipos de sensores mais precisos, como por exemplo do tipo "laser range finder", iria reduzir a incerteza nas medições.

A nível de navegação um dos caminhos que considero promissores é a da navegação com base em características ambientais. Ou seja as trajectórias são descritas não em termos de primitivas geométricas mas com base nas características que podem ser detectadas pelos sensores durante a execução. Desta forma uma trajectória poderia ser simplesmente definida da seguinte forma: *seguir a parede do lado esquerdo até encontrar a terceira reentrância (que poderia ser uma porta), aqui rodar 90 graus à direita e seguir em frente até encontrar um obstáculo com um circulo vermelho.* Este tipo de navegação é sem dúvida a mais utilizada por nós e por alguns animais. Por exemplo quando há uma falha de energia eléctrica e pretendemos ir buscar outro tipo de iluminação, o que fazemos não é mais do que utilizar o tacto para seguir superfícies e quando detectamos uma determinada característica, que pode ser uma porta ou o canto de uma mesa, alteramos a trajectória de acordo com o plano que fizemos para atingir o objectivo. Sempre que detectamos que existe um obstáculo intransponível,

por exemplo uma porta trancada, temos que re-planear as nossas acções e seguir um trajecto alternativo. A vantagem deste tipo de abordagem tem a ver com o facto de não quase não depender de um sistema de localização e por outro lado a estimativa da posição do robô pode ser corrigida quando encontramos uma característica ambiental ou farol artificial cuja posição é bem conhecida.

5.2 Alterações propostas para a plataforma Robuter

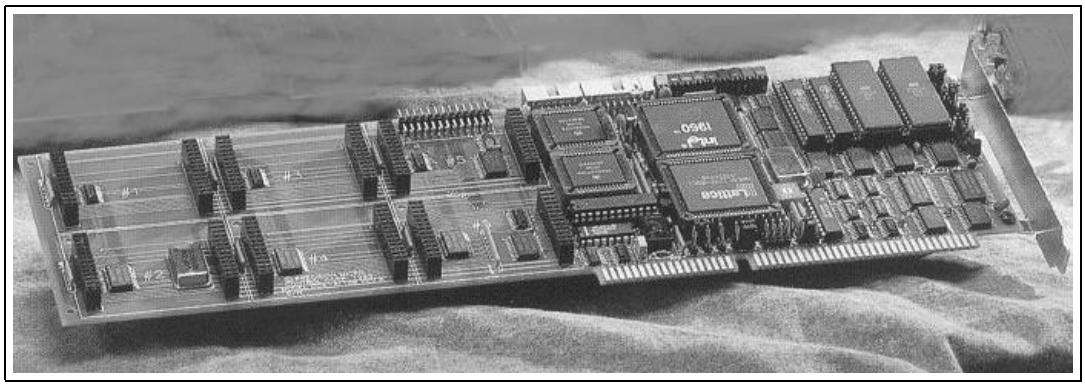

Figura 5.1: Cartão de controlo multi-eixos DCX-200

A plataforma Robuter tem as características de uma plataforma industrial em termos de dimensões, peso e capacidade de carga o que faz com que seja uma boa base experimental para o desenvolvimento de aplicações deste tipo. No entanto e porque a RobotSoft descontinuou a sua produção e ainda porque o sistema de controlo apresenta as limitações já referidas e é de difícil actualização, que considero que o sistema computacional deveria ser substituído.

Existem actualmente no mercado diversos controladores industriais com base na arquitectura dos computadores pessoais com processadores da família i386 da Intel. Estes controladores apresentam a vantagem de existir no mercado inúmeras soluções de interface a preços consideravelmente baixos. Desta forma o controlador da plataforma Robuter poderia ser substituído por aquilo que é vulgarmente referido como um PC industrial podendo os motores ser controlados por um cartão apropriado, por exemplo o DCX-200 da PMC (Precision MicroControl Corp.) com interfaces analógicos DCX-MC500 também da PMC (ver figura 5.1). O computador e placas de expansão poderiam ser albergados num chassis industrial do tipo "rack mount" como o da figura 5.2.

Figura 5.2: Chassis para computador industrial

Uma outra vantagem da utilização deste tipo de computador é a existência de uma vasta gama de software para estes bem como a possibilidade de escolher entre diversos sistemas operativos como WindowsNT, Linux, FreeBSD ou núcleos de tempo-real como o RTKernel por exemplo. Isto abre inúmeras possibilidades de utilização bem como a de efectuar a integração de diversos sistemas sensoriais bastando para isso escolher cartão de adaptação e software apropriados.

5.3 *ALEGRO:* uma plataforma para trabalho futuro

Partindo da experiência ganha ao longo do desenvolvimento do trabalho aqui apresentado,foi iniciado o desenvolvimento de uma plataforma móvel com pernas. Trata-se do projecto Alegro[4], que utiliza como base a estrutura mecânica TITAN VIII, que pode ser vista na figura 5.3. Trata-se de uma estrutura com 4 pernas, onde cada perna é dotada de 3 graus de liberdade. Cada uma das juntas é accionada através de um motor de corrente contínua, sendo a posição obtida através de um potenciómetro multi-voltas. A estrutura inicial já veio equipada com os "drivers" de potência para os motores, não incluindo no entanto qualquer tipo de controlador que permitisse o comando das juntas tendo sido necessário começar por definir uma arquitectura a utilizar no controlo desta estrutura.

[4]Alegro - acrónimo de "A LEGged RObot".

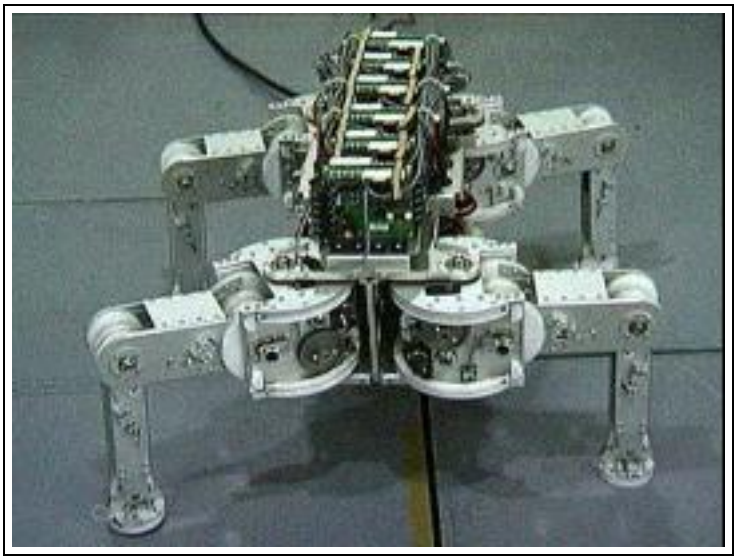

Figura 5.3: Estrutura mecânica TITAN VIII

5.3.1 Arquitectura do Alegro

Dado que se trata de uma plataforma experimental foram estudadas várias hipóteses quanto à arquitectura a escolher para este sistema de forma a facilitar o mais possível o desenvolvimento de aplicações, procurando simultaneamente não introduzir qualquer tipo de restrições mecânicas ou computacionais. A escolha caiu sobre a utilização de um computador industrial com um BUS da norma PC/104 através do qual seriam ligados as placas de interface com os actuadores e sensores existentes na estrutura.

Assim de entre os existentes no mercado, foi seleccionada a placa (motherboard) industrial PCM-5862 da Advantech (figura 5.4). Trata-se de uma placa de dimensões reduzidas (203 mm x 146 mm) que suporta processadores Pentium 200MHz MMX e 128 megabytes de memória RAM e inclui diversos interfaces para dispositivos como: monitor externo SuperVGA ou ecrã LCD, teclado e rato PS/2, disco IDE, rede ethernet 10Mbps, 4 portos série RS232C, entrada/saída audio, interface USB, interface de infravermelhos, expansão PC/104, expansão PCI, e porto paralelo ECP. Como se pode verificar pelas características apontadas, trata-se de um computador com as características dos computadores de secretária mas cujas dimensões permitem a sua colocação no interior da estrutura do robô Alegro. Para o controlo das juntas foram seleccionados dois cartões PC/104 um com 16 saídas analógicas e outro com 16 entradas analógicas. Dado que a estrutura utiliza 12 motores e 12 potenciómetros, ficamos ainda com 4 saídas e 4 entradas analógicas disponíveis para uso futuro. A figura 5.5 mostra a estrutura do robô Alegro depois de integrado o computador de bordo referido.

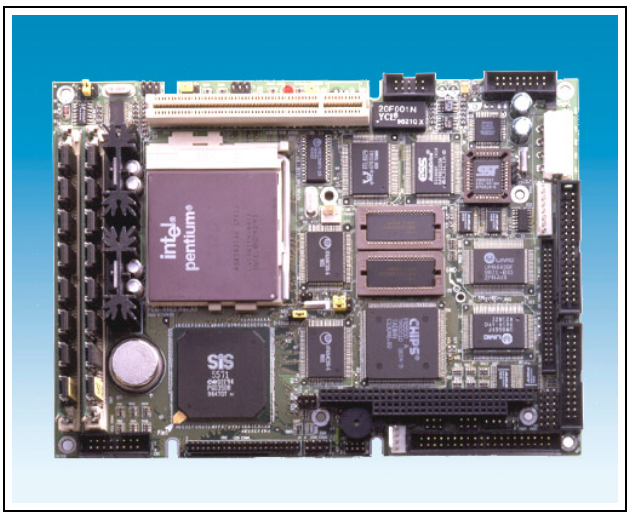

Figura 5.4: Computador industrial PCM-5862

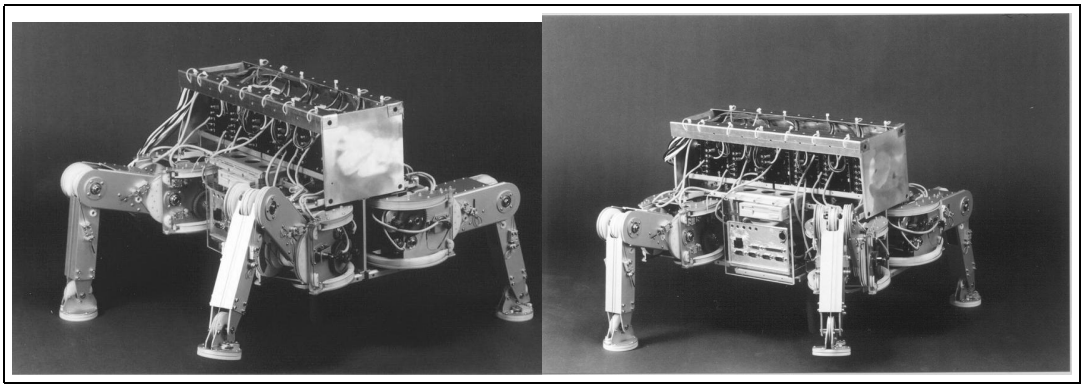

Figura 5.5: Alegro com o controlador embebido instalado

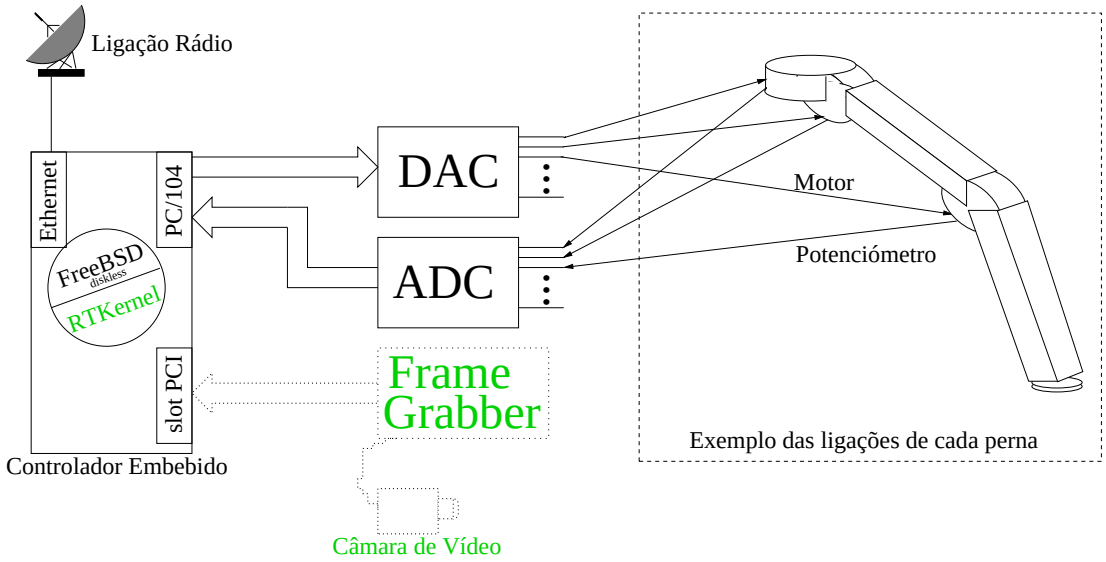

Figura 5.6: Arquitectura de controlo do robô Alegro.

Note-se que esta estrutura permite ainda utilizar outros interfaces com sensores visto que é possível sobrepor outros módulos PC/104 e existe um "slot" PCI disponível.

O resultado da integração encontra-se esquematizado na figura 5.6. As partes do desenho a cinzento referem-se a opções que podem vir a ser utilizadas futuramente. Deve-se notar ainda que além da possibilidade de integrar outros tipos de sensores através do BUS PC/104 é possível ainda utilizar a ligação "ethernet" para operações de tele-operação ou outras que requeiram um controlo distribuído.

5.3.2 Trabalho já realizado

O trabalho nesta área começou com a modelação cinemática e dinâmica de um robô com 6 pernas que se pretendia adquirir inicialmente tendo os resultados sido publicados em [21, 22]. Depois da aquisição da estrutura TITAN VIII e da integração do controlador foi feito o estudo da cinemática da mesma e foram testadas alguns tipos de andamento (gaits) com resultados satisfatórios. O estudo da locomoção começou pela escolha de "gaits" estáticas, ou seja aquelas em que o centro de massa está sempre no interior do polígono definido pelas patas apoiadas. Isto foi feito com base no teste intensivo de todas as combinações de movimentos que conduzem a um deslocamento do robô na direcção desejada, um dos resultados obtidos pode ser visto na figura 5.7. Nesta figura o corpo do robot é representado por um rectângulo e os segmentos de recta mostram a posição relativa das pernas em relação a este. As pernas apoiadas no solo apresentam um círculo fechado na extremidade e a perna levantada um círculo

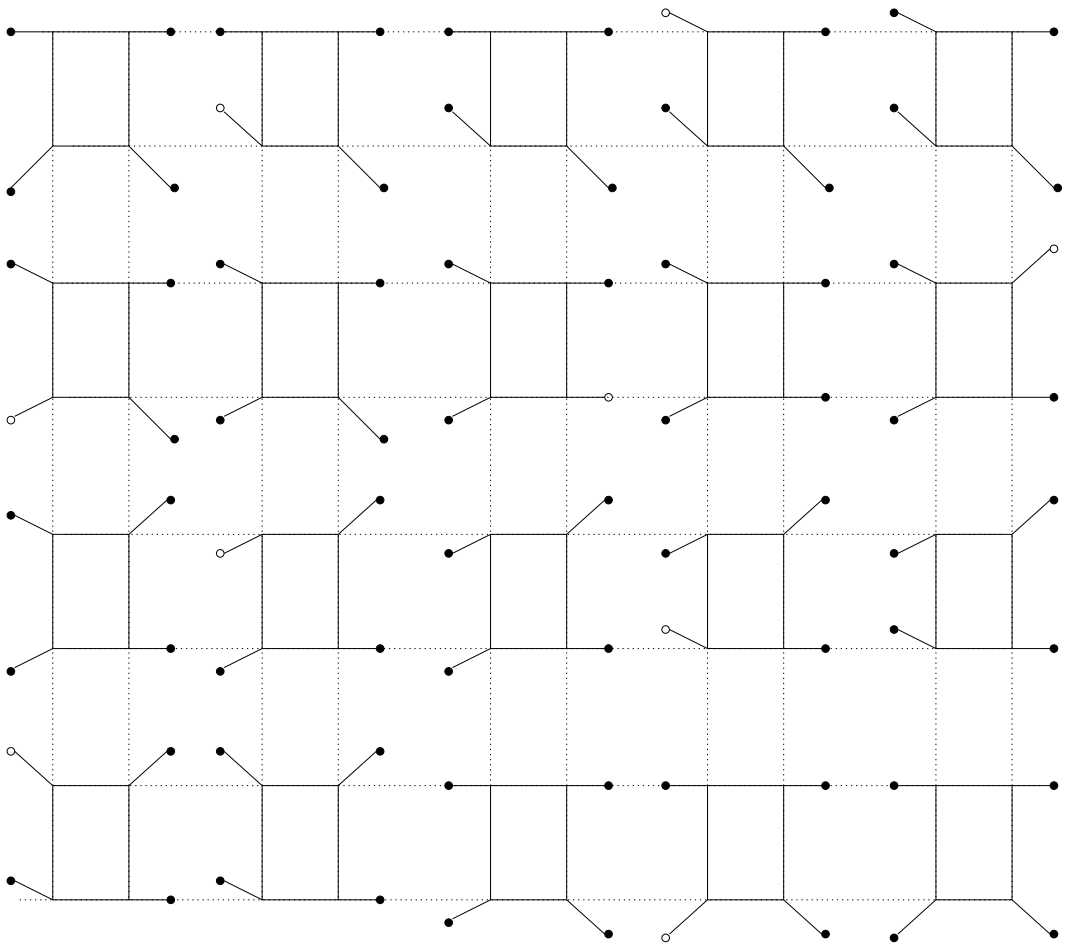

Figura 5.7: Esquema de uma “gait” obtida

aberto. Nesta primeira fase apenas é movida uma perna de cada vez, com excepção da fase de transporte do corpo em que todas as pernas são movidas simultaneamente para trás. Os estudos mais recentes incidiram sobre a utilização de "gaits" estáticas em que todas as pernas se movimentam ao mesmo tempo à semelhança do "passo" de um cavalo.

5.3.3 Trabalho Futuro

O trabalho futuro a desenvolver com base esta plataforma e incidirá numa primeira fase sobre o controlo da locomoção, nomeadamente na obtenção de "gaits" dinâmicas. As "gaits" poderão ter fases instáveis ou marginalmente estáveis visto ser este tipo de andamento que proporciona maiores velocidades de deslocamento, tal como acontece com os mamíferos quando se deslocam a trote ou galope. Para proporcionar um melhor controlo do sistema será efectuada a integração de sensores de força nos pés, com vista à adaptação do andamento ao tipo de terreno: duro, mole, liso ou acidentado.

Com este sistema será possível desenvolver aplicações cujas características de locomoção exigem uma plataforma deste tipo. Esse desenvolvimento será objecto de uma actividade futura.

Apêndice A

Alterações ao Hardware

Para possibilitar a utilização de interrupções aquando da recepção ou do envio de caracteres é necessário efectuar uma pequena modificação nos "jumpers" S4 e S14 de configuração da placa principal[34] cuja localização se pode ver na figura A.

1. No conector S14 ligar SIRQ1 a LRQ3 (ver figura A).
2. No conector S4 ligar "IACK LEVEL 3" a "AVEC" (ver figura A).

A primeira ligação faz com que o módulo de controlo do porto série utilizado gere interrupções locais de nível 3. A segunda ligação indica que essa interrupção é auto-vectorizada ou seja o vector de interrupção está armazenado numa tabela. A outra opção seria a de o próprio dispositivo que gera a interrupção fornecer o endereço da rotina de interrupção.

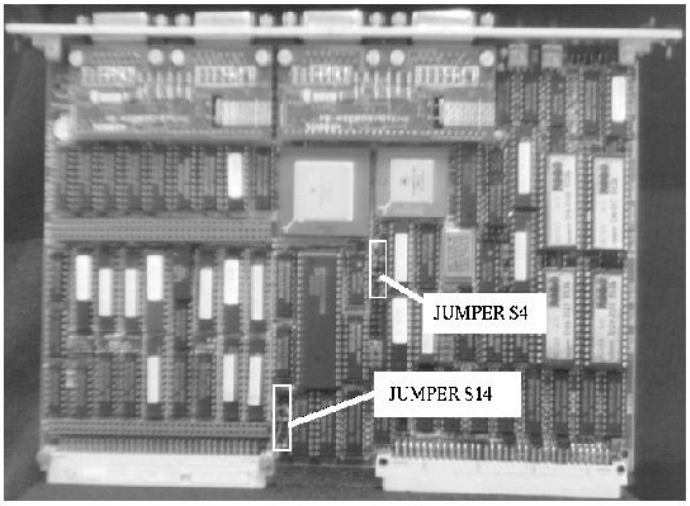

Figura A.1: Localização dos “Jumpers” S4 e S14 na placa principal.

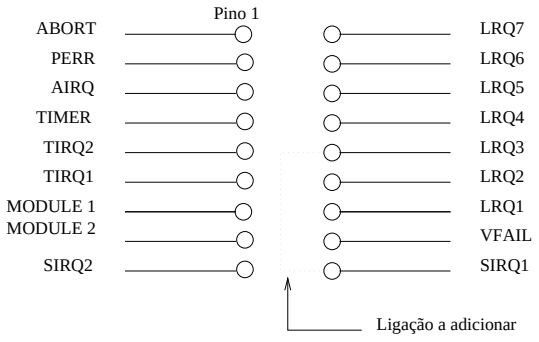

Figura A.2: Selector 14

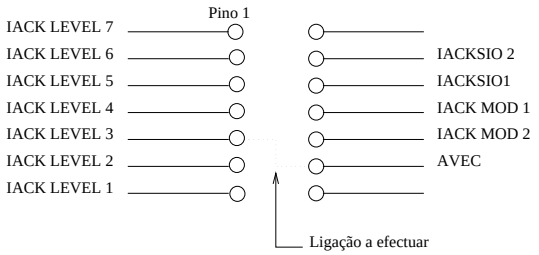

Figura A.3: Selector 4

Apêndice B

Localização dos Sensores de ultrassons

A tabela seguinte apresenta as coordenadas e a orientação de cada um dos 24 sensores de ultrassons em relação ao referêncial interno da plataforma Robuter. Note-se que as coordenadas são dadas em milímetros e a orientação em décimas de grau.

N. do Sensor	${}^{R}X$	${}^{R}Y$	${}^{R}\theta$	N. do Sensor	${}^{R}X$	${}^{R}Y$	${}^{R}\theta$
0	725	0	0	12	-285	0	1800
1	725	-217	0	13	-285	217	1800
2	715	-263	-225	14	-275	263	1575
3	685	-300	-450	15	-245	300	1350
4	650	330	-675	16	-210	330	1125
5	550	-347	-900	17	-120	347	900
6	220	-347	-900	18	220	347	900
7	-120	-347	-900	19	560	347	900
8	-210	-330	-1125	20	650	330	675
9	-245	-300	-1350	21	685	300	450
10	-275	-263	-1575	22	715	263	225
11	-285	-217	-1800	23	725	217	0

Figura B.1: Tabela de localização e orientação dos sensores de ultrassons

Apêndice C

Cálculo do CRC

O algoritmo CRC (cyclic redundancy check) é uma forma de detectar pequenas alterações em blocos de dados. Este algoritmo funciona tratando sequências de bits como sendo polinómios binários. Este tipo de detecção de erros é utilizado em sistemas de transmissão para verificar a consistência dos dados recebidos, mas é também utilizado em sistemas de armazenamento de dados, ex. tapes, discos, etc. Assim, dada uma sequência de bits (quadro), é gerada a "sequência de teste do quadro" ou FCS como é vulgarmente referida. O FCS é gerado de forma que o quadro composto pelo quadro original e pelo FCS, seja divisível por um qualquer polinómio pré-definido. Este polinómio é o chamado divisor, polinómio gerador ou polinómio do CRC.

C.1 Funcionamento do algoritmo CRC

Não se pretende aqui desenvolver a teoria matemática que suporta os códigos polinomiais, mas de uma forma simples, pode-se dizer que este método explora a seguinte propriedade da aritmética binária. Sejam

M - um conjunto k bits a ser transmitido, normalmente designado como quadro.

R - o código FCS que se junta ao quadro a transmitir. Este código tem de $n$bits tal que $k > n$.

G - o polinómio gerador do CRC que terá $(n + 1)$ bits.

Começando por multiplicar a mensagem M por 2^n, o que equivale a deslocar para a esquerda todos os bits da mensagem n vezes, ou, se preferirmos, acrescentar n zeros à direita da mensagem, e de seguida dividir o resultado por G. Temos então

$$\frac{M \times 2^n}{G} = Q + \frac{R}{G} \tag{C.1}$$

em que Q é o quociente resultante e R é o resto.

Então a mensagem a transmitir será

$$T = M \times 2^n + R. \tag{C.2}$$

Ou seja T corresponde à mensagem (ou quadro) original à qual foram acrescentados os bits FCS.

Após a recepção (ou leitura) da mensagem composta T, é necessário verificar a sua integridade e para isso esta é dividida por G.

$$\begin{aligned} T/P &= \frac{M \times 2^n + R}{G} \\ &= \frac{M \times 2^n}{G} + \frac{R}{G} \\ &= Q + \frac{R}{G} + \frac{R}{G} \\ &= Q + \frac{R + R}{G} \end{aligned} \tag{C.3}$$

Como resultado podemos ver que o quociente obtido é o mesmo da expressão C.1, mas o quociente é $R + R$. Ora a soma módulo 2 de qualquer número consigo próprio é zero, logo $R + R = 0$.

C.1.1 Operações módulo 2

Uma parte importante deste algoritmo é o facto de as operações serem efectuadas em módulo 2, ou seja são ignorados os transportes nas adições e subtracções (carry e borrow), cada posição é calculada independentemente das restantes. O resultado é que as operações aritméticas módulo 2 são bastante simples de implementar com funções lógicas, por exemplo tanto a adição como a subtracção são implementados com operações ou-exclusivo. A divisão modulo 2 é semelhante à divisão binária excepto que é efectuado um ou-exclusivo em vez da subtracção. Como se pode então verificar a soma módulo 2 de um número com ele próprio é sempre zero, visto que equivale a fazer a operação ou-exclusivo de cada um dos bits do número com ele próprio.

A vantagem de utilizar este tipo de aritmética reside na simplicidade da lógica necessária para uma implementação em hardware bem como no aumento de velocidade devido a não ser necessário efectuar os transportes (carry).

C.2 Utilização do Algoritmo CRC

O algoritmo CRC foi inicialmente criado para ser implementado em circuitos lógicos, mas existem inúmeras implementações em software, ou simulando o funcionamento

do algoritmo directamente ou utilizando pesquisa em tabelas para acelerar o processo. Em qualquer realização deste algoritmo é importante a escolha apropriada do polinómio gerador, pois este define o tipo de erros a detectar, sendo o polinómio mais comum, o definido pelo CCITT para utilização em linhas telefónicas comutadas: $x^{16} + x^{12} + x^5 + x^0$.

Bibliografia

[1] Arlindo Silva, Paulo Menezes, and Jorge Dias. Grid based navigation for mobile robots - an algorithm based on the integration of vision and sonar data. In *ISIE'97 - IEEE International Symposium on Industrial Electronics, Guimarães, Portugal*, June 7-11 1997.

[2] Adrian Nye and Tim O'Reilly. *X Toolkit Intrinsics Programming Manual - OSF/Motif 1.1 Edition*, volume 4. O'Reilly & Associates, Inc., 1992.

[3] Arlindo Silva, Paulo Menezes, and Jorge Dias. Avoiding obstacles using a connectionist network - experiments based on visual and sonar information. In *IROS'97 - IEEE-RSJ International Conference on Intelligent Robots and Systems, Grenoble, France*, September 7-11 1997.

[4] Arlindo Silva, Paulo Menezes, and Jorge Dias. Avoiding obstacles with mobile robots integrating vision and sonar data in a connectionist approach. In *30th International Symposium on Automotive Technology and Automation, Dedicated Conference on Robotics, Motion & Machine Vision, Florence Italy*, June 16-19th 1997.

[5] Arlindo Silva, Paulo Menezes, and Jorge Dias. Grid-based framework for sensorial data integration in mobile robots. In *SPIE's International Symposium and Education Programa on Intelligent Systems & Advanced Manufacturing - Special Session on Sensor Fusion and Decentralized Control in Autonomous Robotic Systems, Pittsburg, Pennsylvania USA*, October 14-15 1997.

[6] Arlindo Silva, Pedro Silvestre, Paulo Menezes, Jorge Dias, and Aníbal de Almeida. Obstacle detection and avoidance using a connectionist approach. In *27th International Symposium on Industrial Robots, Milan, Italy*, October 6-8 1996.

[7] Billur Barshan and Roman Kuc. Differentiating sonar reflections from corners and planes by employing an intelligent sensor. In *IEEE Trans. on Pattern Analysis and Machine Inteligence*, volume 12. IEEE, 1990.

[8] Catarina Silva, Nuno Rodrigues, Paulo Menezes, and Jorge Dias. Landmark based navigation of a mobile platform. In *European Advanced Robotic Systems Development, IPL Leiria*, September 14-16 1998.

[9] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3), June 1987.

[10] A. Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. In *Proc. Sixth Conf. on Uncertainty in AI - Cambridge*, Cambridge, July 1990.

[11] F. Moita, A. Feijao, and U. Nunes. Ultrasonic and dead reckoning data processing. Technical report, PO-ROBOT - Report of Sub-Task ST1/T2, April 1994.

[12] F. Pedrosa and M. I. Ribeiro. Mobile robot localization using ultrasonic sensors. In *Proceedings of CONTROLO'94, First Portuguese Meting on Automatic Control, Lisbon, Portugal*, September 1994.

[13] Fernando Monteiro, Pedro Rocha, and Paulo Menezes. Teleoperating a mobile robot - a solution based on java language. In *ISIE'97 , - IEEE International Symposium on Industrial Electronics, Guimarães, Portugal*, pages ss263–ss267, 1997.

[14] Open Software Foundation, editor. *OSF/MOTIF - Programmer's Guide*. Prentice Hall, 1991.

[15] Open Software Foundation, editor. *OSF/MOTIF - Programmer's Reference*. Prentice Hall, 1991.

[16] H. Haddad, M. Khatib, S. Lacroix, and R. Chatila. Reactive navigation in outdoor environments using potential fields. In *IEEE International Conference on Robotics and Automation, Leuven,Belgium*, May 1998.

[17] International Organization for Standardization. *SO/IEC 3309:1991(E), Information Technology - Telecommunications and information exchange between systems - High-level Data Link Control (HDLC) - Frame Structure*, fourth edition, January 1992.

[18] J. Barraquad, L. E. Kavraki, J. C. Latombe andT. Y. Li, R. Montani, and P. Raghavan. A random sampling scheme for path planning. In *International Journal of Robotics Research*, volume 16, pages 759–774, 1997.

[19] J. Borenstein and Y. Koren. The vector field histogram - Fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, June 1991.

[20] J. Borenstein and Y. Koren. Real-time Obstacle Avoidance for Fast Mobile Robots. *IEEE Trans. on Systems, Man & Cybernetics*, 19:1179–1187, Sept./Oct. 1992.

[21] João Barreto, António Trigo, Paulo Menezes, Jorge Dias, and Aníbal de Almeida. Fbd - the free body diagram method. In *Proceedings of Controlo 98 - The 3rd Portuguese Conference on Automatic Control, Coimbra, Portugal*, July 20-23 1998.

[22] João Barreto, António Trigo, Paulo Menezes, Jorge Dias, and Aníbal de Almeida. Fbd - the free body diagram method - kinematic and dinamic modelling of a six leg robot. In *Proceedings of AMC'98 - 5th International Workshop on Advanced Motion Control, Coimbra Portugal*, June 29 - July 1 1998.

[23] M. Khatib. *Sensor-based motion control for mobile robots.* PhD thesis, Laboratoire d'Automatique et d'Analyse des Systèmes, 1997.

[24] L. Feng, J. Borenstein, and H.R. Everett. Where am i? - sensors and methods for autonomous mobile robot positioning. Technical Report UM-MEAM-94-21, University of Michigan, December 1994.

[25] Network Working Group. *RFC1662 - PPP in HDLC-like Framing*, July 1994.

[26] Adrian Nye. *Xlib Programming Manual - for Version 11, Release 4*, volume 1. O'Reilly & Associates, Inc., 1992.

[27] O. Khatib. real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 1986.

[28] Paulo Menezes, Helder Araújo, Jorge Dias, and Maria Isabel Ribeiro. Obstacle detection in mobile robots using sonar data. In *Controlo 94, Lisbon,*, September 1994.

[29] Paulo Menezes, Jorge Dias, Helder Araújo, and Aníbal de Almeida. Low cost sensor based obstacle detection and description: Experiments with mobile robots using grid representation. In O. Khatib and J.K. Salisbury, editors, *Lecture Notes in Control and Information Sciences 223, Experimental Robotics IV, The 4th International Symposiom*, pages 231–237. Springer, June 30 - July 2 1995.

[30] Paulo Menezes, Jorge Dias, Helder Araújo, and Aníbal de Almeida. Using sonar sensors to perform obstacle detection. In *1er Colloque International Produtique Robotique du Sud Europe Atlantique, Bourges, France.* Editions Hermés, 1er juin 1995.

[31] Edward S. Plumer. Neural network structure for navigation using potential fields. 1992.

[32] Polaroid. *Ultrasonic Ranging System - Description, operation and use information for conducting tests and experiments with Polaroid's Ultrasonic Ranging System components.*

[33] Isabel Ribeiro. Full plan of the project "po-robot - multi-purpose portuguese flexible mobile robot".

[34] RobotSoft. *THEMIS TSVME 120 68020 CPU Board User's Manual.*

[35] RobotSoft. *Albatros 4.0 Reference Manual*, May 1989.

[36] RobotSoft. *Albatros C Interface*, May 1989.

[37] RobotSoft. *IR Absolute Alocalization System User's Manual V1.0*, March 1992.

[38] RobotSoft. *New Commands Albatros Alpha Release 5.2*, December 1993.

[39] Sean Quinlan. *Real-Time Modification of Collision-Free Paths.* PhD thesis, Dept. Computer Science of Stanford University, 1994.

[40] S. Thrun. Finding landmarks for mobile robot navigation. In *IEEE International Conference on Robotics and Automation, Leuven,Belgium*, May 1998.

[41] S. Thrun. Probabilistic navigation in outdoor environments by a mobile robot. In *IEEE International Conference on Robotics and Automation, Leuven,Belgium*, May 1998.