Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrotécnica

NAVEGAÇÃO ASSISTIDA DE UMA CADEIRA DE RODAS CONTROLADA POR COMPUTADOR E COM INTERFACE DE VOZ

Tese de Mestrado

Gabriel Pereira Pires
Licenciado em Engenharia Electrotécnica

Coimbra
Março de 2001

A todos os meus amigos em especial à minha família e à Charlotte.

Agradecimentos

Em primeiro lugar desejo agradecer ao Prof. Doutor Urbano Nunes pela orientação, apoio e encorajamento prestados ao longo da realização deste trabalho. Quero ainda expressar o meu reconhecimento pelas condições de trabalho que sempre se esforçou por me proporcionar.

Desejo ainda endereçar os meus agradecimentos:

- À Carla Alexandra e ao Nuno Honório pela preciosa ajuda na implementação do servidor de reconhecimento de fala;
- Ao Miguel Aragão pelas ajudas prestadas na montagem e soldadura de componentes de *hardware*;
- Ao João Rodrigues que me iniciou nos programas Tango e Orcad;
- Ao João Coimbra pelos desenhos da cadeira em AutoCad;
- A todos os colegas de laboratório e em particular ao Inácio Sousa, Rui Cortesão e Rodrigo Maia pelas ajudas e sugestões que me foram dando.
- À FCT (Fundação para a Ciência e Tecnologia), projecto Mentor/1998, cujo financiamento proporcionou parcialmente as condições experimentais para o desenvolvimento do trabalho de investigação;
- Ao Instituto de Sistemas e Robótica onde este trabalho foi realizado;
- Ao Instituto Politécnico de Tomar, onde trabalho, por me ter dado a disponibilidade para concluir este trabalho;

Por fim, quero agradecer à minha família e em especial à Charlotte que me deram constante apoio e me ajudaram nos tempos mais difíceis.

Resumo

Nesta dissertação são descritos resultados de investigação em navegação assistida e interfaces homem-máquina em sistemas robóticos orientados a humanos. A plataforma experimental utilizada, uma cadeira de rodas motorizada, foi munida de um conjunto de novas funcionalidades. Estas facilitam a acessibilidade e controlo da cadeira, contribuindo assim para uma maior autonomia dos seus utilizadores.

A dissertação começa por apresentar estratégias de navegação reactiva que permitem implementar um controlo partilhado entre o utilizador e o sistema. São descritos em particular, algoritmos reactivos baseados em comportamentos, nomeadamente, comportamentos de desvio de obstáculos e de seguimento de meta, e também mecanismos de coordenação de comportamentos.

Para a implementação dos algoritmos de navegação, foi necessário desenvolver uma infra-estrutura de *hardware* e *software* de suporte. A cadeira foi assim equipada com sensores de infravermelho, sonares e codificadores nas rodas, necessários para o desenvolvimento dos algoritmos. A arquitectura de *software* foi implementada de forma distribuída, garantindo maior modularidade ao sistema.

Implementou-se uma interface homem-máquina por comandos de voz. Esta nova interface trouxe uma alternativa à condução tradicional através de *joystick*.

Por fim, desenvolveu-se o módulo de navegação reactiva. Este consiste num controlador difuso que coordena a saída de um comportamento de desvio de obstáculos e os comandos fornecidos pelo utilizador. A finalidade do controlador é a de corrigir manobras desajustadas do utilizador. O controlador foi testado e validado experimentalmente, tendo apresentado resultados que mostram claramente a diminuição do esforço do utilizador na condução da cadeira.

Abstract

The current research work applies navigation strategies and new human-machine interfaces on a wheelchair for motor handicapped persons. This work aims to overcome some of the limitations of actual systems. The experimental platform, a powered wheelchair, was provided with a set of augmented functionalities to improve the wheelchair accessibility and control, and thereby to increase users autonomy.

Firstly, the dissertation describes reactive navigation strategies that allow a shared control between the user and the system. This study focused the implementation of behaviour-based reactive algorithms, namely obstacle avoidance and goal following behaviours, as well as behaviour co-ordination.

To implement navigation algorithms, a support hardware and software infra-structure was developed. The wheelchair was equipped with infrared and sonar sensors and wheel encoders needed for algorithms implementation. The software follows a distributed architecture, allowing an increased modularity of the system.

A voice human-machine interface was developed. This new interface brought an alternative to the traditional joystick steering.

At last, a navigation reactive module is presented. This one is implemented by means of a fuzzy logic controller that co-ordinates the output of an obstacle avoidance behaviour and the steering commands provided by the user. The controller aims to correct user's wrong manoeuvres. The controller was experimentally tested and validated, and has presented results that clearly show the user reduction effort to steer the wheelchair.

Conteúdo

Lista de Tabelas

Lista de Figuras

Abreviaturas e Tradução de Termos em Inglês

AKH	"Active Kinematic Histogram" (histograma cinemático activo)
AKW	"Adaptive Kinematic Window" (janela cinemática adaptativa)
API	"Application Program Interface" (interface para programas de aplicação)
AuRA	"Autonomous Robot Architecture"
bps	bit por segundo
EEPROM	"Electrically Erasable Programmable Read Only Memory"
CDC	Combinação Dependente do Contexto ("Context-Dependent Blending" - CDB)
CDG	Centro De Gravidade ("Centre Of Gravity" - COG)
CDMA	Centróide Da Maior Área ("Centroid of Largest Area" - CLA)
CPU	"Central Processing Unit" (unidade de processamento central)
DAMN	Distributed Architecture for Mobile Navigation
Diagramas motores	"Motor schemas"
Direcção do campo de forças	"Steepest descent"
e.g.	exempli gratia (por exemplo)
i.e.	id est (isto é)
IHM	Interface Homem Máquina ("Human Machine Interface" - HMI)
Kernel	Núcleo do sistema operativo
LED	"Light Emitting Diode"

MDM	Media Dos Máximos ("Mean Of Maximum" - MOM)
MVFH	Mínimo VFH
MEF	Máquina de Estados Finitos
MPG	Modus Ponens Generalizado
MTD	Margem de Tempo para Desvio ("reserve avoidance time")
NetBEUI	"NetBIOS Extended User Interface"
PDM	Primeiro Dos Máximos ("First of Maximum" - FOM)
PSD	"Position Sensitive Detector"
PWM	"Pulse Width Modulation" (modulação de largura de pulso)
RAF	Reconhecimento Automático de Fala
Sonar	"SOund NAvigation and Ranging"
Submissão	"Subsumption"
SRM	"Stimulus Response Modeling" (modelação da resposta a estímulos)
TCP/IP	"Transmission Control Protocol/Internet Protocol"
TOF	"Time of Flight" (tempo de vôo)
TPU	"Time Processor Unit" (unidade de processamento temporal)
UART	"Universal Asynchronous Receiver/Transmitter"
UDM	Último dos Máximos ("Last of Maximum" - LOM)
VST	Vencedor Sobrepõe-se a Todos ("winner-take-all")
VFF	"Virtual Force Field" (campo de forças virtuais)
VFH	"Vector Field Histogram" (histograma de campos vectoriais)

Capítulo 1

Introdução

Durante muitos anos, o deficiente foi considerado um elemento à margem da sociedade. Nos dias de hoje, o deficiente já é encarado como um elemento integrante da sociedade, mas ao qual deve ser dada especial atenção por possuir limitações físicas e/ou cognitivas. É precisamente para compensar essas limitações que surgiram alguns programas e associações de apoio tecnológico ao deficiente. São exemplos o programa Europeu TIDE ("Technology Initiatives for Disabled and Elderly People"), a associação Europeia AAATE ("Association for Advancement of Assistive Technology in Europe") e a associação Norte-Americana RESNA ("Rehabilitation Engineering and Assistive Technology Society of North America"). O programa TIDE definiu como objectivo geral a promoção da investigação e desenvolvimento tecnológico, de modo a satisfazer objectivos industriais e sociais [Tid94]. Do ponto de vista industrial, procura-se melhorar a indústria e o mercado europeu em produtos e serviços que vão de encontro às necessidades dos idosos e deficientes. Na perspectiva social, o TIDE tinha por objectivo dar lugar ao desenvolvimento de novas ferramentas e aplicações tecnológicas, de forma a permitir que os deficientes e idosos consigam viver autonomamente e participar mais activamente na vida social e económica. Surge, desta forma, o conceito de *tecnologia de reabilitação* (TR) definida como tecnologia que pode ajudar a compensar limitações funcionais, facilitar uma vida independente e possibilitar que idosos e deficientes tenham uma melhor integração na sociedade e na vida activa. O grupo dos idosos merece especial destaque no programa TIDE, pelo facto de constituir uma fatia crescente da população, em especial nos países mais desenvolvidos, e por representar um grupo que requer elevados recursos a nível social e de serviços de

saúde.

Em Portugal, estima-se em cerca de 10% a população que sofre de limitações físicas, sensoriais ou cognitivas, suficientemente graves para serem potenciais beneficiários de ajudas tecnológicas. Nos países integrantes da União Europeia (UE), essas deficiências parecem atingir mais de 25 milhões de pessoas, as quais se podem reagrupar, em função da origem dos problemas que as atingem, em dois grandes grupos [Tid94]: 1) pessoas portadoras de deficiências congénitas ou resultantes de acidente ou doença; 2) pessoas com capacidades diminuídas em resultado do processo de envelhecimento. Nos países desenvolvidos, são estas últimas as que mais contribuem para o número daqueles que se tornam incapazes de realizar, sem ajudas de terceiros, as tarefas quotidianas. O aumento da longevidade associada à diminuição da natalidade são os principais factores pelo progressivo envelhecimento da população. Estima-se que, em 2020, na UE, um em cada quatro cidadãos terá mais de 60 anos [Tid94].

O Instituto de Sistemas e Robótica (ISR) desenvolve investigação tecnológica nas áreas do controlo (Robótica, Mobilidade e Transporte e Interfaces Homem-Máquina para controlo). Os projectos desenvolvidos nesta área destinam-se essencialmente a deficientes e idosos com capacidades motoras diminuídas (com problemas de mobilidade), mas com capacidade cognitiva normal. Os projectos dividem-se em três grupos: 1) próteses inteligentes e manipuladores; 2) domótica; e 3) robótica móvel.

O presente projecto consiste no desenvolvimento do protótipo de uma cadeira de rodas inteligente, e insere-se no terceiro grupo de desenvolvimento. As cadeiras de rodas representam o único meio de locomoção potencialmente utilizável por deficientes motores. No entanto, este recurso tecnológico não implica total independência dos seus utilizadores. Nomeadamente, as pessoas com limitações físicas muito acentuadas, incapazes de controlar uma cadeira de rodas convencional, dependem de terceiros. Por outro lado, a existência de barreiras arquitectónicas e ambientais intransponíveis por uma cadeira normal representa outro motivo da dependência dos deficientes e idosos nas suas deslocações.

O nosso projecto, designado por RobChair, tem precisamente o objectivo de estudar a aplicação de tecnologias de controlo e robótica em cadeiras de rodas, e consequentemente visa o melhoramento da qualidade de vida de deficientes motores e idosos.

1.1 Projecto RobChair

O ISR adquiriu uma cadeira de rodas motorizada convencional que serviu de plataforma experimental para o desenvolvimento de um projecto denominado **RobChair**. Definiram-se alguns objectivos gerais para o desenvolvimento deste projecto:

1. Garantia da segurança e integridade do utilizador da cadeira de rodas;
2. Diminuição do esforço do utilizador na condução, conferindo crescentes graus de funcionalidade à cadeira através de:
   - Partilha de controlo entre o utilizador e a cadeira;
   - Planeamento local de trajectórias para o desvio de obstáculos;
   - Execução de tarefas consideradas mais complexas. Foram estabelecidas duas tarefas concretas: passagem por uma porta e seguimento de paredes;
3. Garantia de acessibilidade da cadeira de rodas a utilizadores incapazes de controlar uma cadeira, através de um *joystick* convencional;
4. Integração da cadeira numa rede de comunicação, permitindo a comunicação com o exterior;

Traçados os objectivos gerais, definiram-se os módulos de desenvolvimento para alcançar os objectivos propostos:

1. **Sistema sensorial** e **capacidade de processamento** de suporte.
2. **Sistema de Navegação** modular para dotar a cadeira com crescentes níveis de funcionalidade (objectivos 1 e 2);
3. **Interface Homem Máquina (IHM)**, baseada em comandos de voz (objectivo 3);
4. **Sistema de comunicação em rede**, baseado na arquitectura TCP/IP, permitindo troca de mensagens entre utilizador e operadores da rede e ainda a **monitorização e controlo remoto** da cadeira (objectivo 4).

1.2 Sistema de Navegação

O desenvolvimento de técnicas para navegação autónoma em ambientes reais constitui uma das áreas actuais de investigação em robótica. Um dos seus maiores problemas reside em lidar com a incerteza inerente a ambientes naturais. Os dados fornecidos pelos sensores actualmente disponíveis são geralmente pouco informativos, inexactos e muito sensíveis às condições ambientais, dificultando desta forma o desenvolvimento de sistemas de navegação autónomos robustos.

As arquitecturas de controlo utilizadas em robótica móvel dividem-se, quanto à forma de raciocínio, em três grandes grupos: arquitecturas deliberativas - para planeamento global de trajectórias; arquitecturas reactivas - para reagir em tempo real a estímulos; e arquitecturas híbridas - que associam estas duas arquitecturas.

Nas arquitecturas deliberativas, a informação do mundo encontra-se previamente armazenada, existindo um módulo responsável pelo planeamento *off-line* de trajectórias. Eventos inesperados podem ser bloqueantes, na medida em que o robô não é capaz de tomar decisões face a novas situações. As arquitecturas reactivas [Bro86] [Ark87] seguem uma abordagem bastante diferente. Cada acção produzida depende directamente dos estímulos actuais do meio ambiente (sem recurso a estados passados e informação prévia do mundo). O controlo global resulta da interacção entre comportamentos concorrentes do robô. A dificuldade de concepção desta arquitectura reside na conjugação de comportamentos de forma a obter uma resposta coerente. A falta de representação do ambiente constitui ainda uma limitação das arquitecturas reactivas, uma vez que impossibilita o planeamento global de trajectórias. Para ultrapassar as desvantagens de cada uma destas arquitecturas, a arquitectura híbrida propõe a integração de um sistema reactivo no nível mais baixo com um planeamento de acções no nível mais alto.

Nestas estruturas, o papel do homem consiste em definir previamente missões ao robô e, de seguida, em supervisionar a sua execução. Considera-se, nesse caso, que existe uma navegação completamente autónoma. Existem, no entanto, outros sistemas em que há cooperação entre o homem e a máquina. Em sistemas remotos de tele-operação utilizados em ambientes hostis para o ser humano, o controlo é partilhado entre o robô e o operador: o ser humano controla remotamente o sistema, ao mesmo tempo que este corrige os movimentos do operador. Em sistemas de cadeiras de rodas para deficientes [BHH+95] [SL97], foram propostas arquitecturas de controlo em que a condução da cadeira é partilhada

entre o utilizador e a cadeira, resultando num sistema de navegação semi-autónomo.

1.3 Interface Homem Máquina

A aplicação de sistemas robotizados, designadamente na área dos serviços, tem conduzido a uma exigência acrescida na interacção homem-máquina. A voz e os gestos são formas fundamentais de comunicação humana. Os crescentes avanços em reconhecimento e síntese de voz vêm tornando a comunicação homem-máquina por voz uma realidade tecnológica e comercial. O reconhecimento de gestos através de sistemas de visão pode ser considerado um bom suplemento às interfaces de voz. A linguagem natural é, por exemplo, um meio eficiente de controlar robôs de uma forma mais flexível [LLS+94]. Na área da tecnologia de reabilitação, a interface é um factor decisivo para a aceitação de um sistema robótico de ajuda. Alguns trabalhos de investigação nesta área utilizam comandos de voz para controlar um manipulador robotizado de reabilitação [KBB+94] e para controlar uma cadeira de rodas para tetraplégicos [SL97]. A linguagem gestual utiliza-se também para controlar uma cadeira de rodas [BMG+99].

As novas interfaces homem-máquina vêm extender e não substituir as capacidades humanas, tornando mais fácil o controlo dos sistemas e reduzindo ao mínimo o esforço do utilizador.

1.4 Trabalho Desenvolvido

No âmbito desta tese foram realizados os seguintes desenvolvimentos:

1. *Hardware* para interface de sensores optolectrónicos, sensores de ultrassons e codificadores nas rodas.

2. *Software* de aquisição de dados sensoriais; *software* de comunicação entre o subsistema de aquisição de dados e o computador portátil responsável pela execução de algoritmos de navegação.

3. Interface Gráfica para visualização do meio ambiente e controlo remoto da cadeira de rodas.

4. Sistema de comunicação com a rede local, baseado na arquitectura TCP/IP.

5. Sistema de navegação reactivo para desvio local de obstáculos, baseado em lógica difusa. O sistema de navegação constitui o ponto central desta tese, tendo por isso levado a um estudo aprofundado dos principais métodos de navegação reactiva utilizados em robótica móvel.

6. Interface Homem Máquina, baseada em comandos de voz.

1.5 Estrutura da Dissertação

A dissertação é constituída por 9 capítulos, incluindo este primeiro capítulo, de descrição geral, e por dois apêndices.

No **capítulo 2**, faz-se uma introdução à robótica móvel e evidenciam-se as vantagens e desvantagens das diferentes arquitecturas.

No **capítulo 3**, apresentam-se e discutem-se as principais estratégias de navegação local e navegação reactiva utilizadas em robótica móvel.

No **capítulo 4**, evidenciam-se as tendências no desenvolvimento de protótipos de cadeiras de rodas inteligentes.

No **capítulo 5**, descrevem-se as partes mecânica, sensorial e electrónica do sistema RobChair.

No **capítulo 6**, pormenoriza-se o *software* desenvolvido para os vários subsistemas da cadeira de rodas: subsistema de baixo nível para aquisição de dados sensoriais e controlo, comunicação entre sub-sistemas, e comunicações na rede local Ethernet.

O **capítulo 7** descreve a arquitectura de navegação do sistema RobChair, modos de funcionamento e cinemática.

No **capítulo 8** expõe-se o sistema de navegação constituído por um controlador difuso.

O **capítulo 9** discute e apresenta resultados.

No **apêndice A**, introduz-se o conceito de lógica difusa e descrevem-se os vários módulos de um controlador difuso.

No **apêndice B**, apresentam-se os esquemáticos dos módulos de *hardware*.

Capítulo 2

Arquitecturas de Navegação

Para atingir um elevado grau de autonomia e robustez, um robô móvel deve ter a capacidade de percepcionar o meio envolvente, construir ou actualizar mapas, planear e executar acções, e adaptar o seu comportamento a mudanças ambientais. Para implementar estas tarefas, é necessário construir uma arquitectura de controlo em tempo real baseada em informação sensorial que permita uma acção coordenada do sistema. Neste capítulo, descrevem-se as várias arquitecturas de controlo utilizadas em robótica móvel e que serviram como fundamento para a escolha da arquitectura utilizada no sistema RobChair (capítulo 7).

2.1 Robótica Móvel

Um dos principais problemas da navegação autónoma reside na necessidade de fazer frente à incerteza inerente a ambientes reais não estruturados. Primeiro, o conhecimento prévio do ambiente é geralmente incompleto, incerto e aproximado. Os mapas omitem alguns detalhes, as relações espaciais entre objectos podem variar desde a construção dos mapas, e a informação métrica pode ser imprecisa. Segundo, a informação perceptual adquirida no momento não é geralmente fidedigna. A gama limitada de alguns sensores associada à incerteza angular de outros (e.g. sonares), o efeito das características do ambiente, tais como oclusão e características adversas das superfícies dos objectos (e.g. rugosidade, cor), as condições adversas de observação (e.g. baixa luminosidade), e o ruído provocado por diafonia (e.g. sonares), são características que contribuem para dados ruidosos e

imprecisos. Erros na interpretação dos dados recebidos levam implicitamente a resultados incorrectos. Terceiro, ambientes reais possuem uma dinâmica complexa e imprevisível: os objectos mudam de sítio, podem existir outros elementos móveis (e.g. pessoas e robôs móveis), e as condições ambientais variam (e.g. luz, temperatura, humidade). Finalmente, o efeito das acções de controlo nem sempre é fidedigno, o que se deve, por exemplo, ao deslizamento das rodas.

2.2 Arquitecturas de Controlo em Robótica móvel

Neste capítulo, procurar-se-á principalmente definir a terminologia associada às arquitecturas de controlo, assim como discutir questões que surgem aquando da concepção e implementação de uma arquitectura de controlo em robótica móvel. Deverá uma arquitectura ser reactiva ou deliberativa? Hierárquica ou centralizada?

2.2.1 Raciocínio Deliberativo vs. Controlo Reactivo

Num sistema de controlo, é necessário contrapor a necessidade de existir um planeamento óptimo e global de acções que visa alcançar determinados objectivos, e a necessidade de reagir rapidamente a estímulos de um ambiente dinâmico e incerto. O *raciocínio deliberativo* dota o robô da capacidade de planear tarefas e alcançar objectivos de longo alcance, enquanto o *controlo reactivo* fornece ao robô a aptidão para reagir em tempo real garantindo a sua segurança assim como a do ambiente envolvente. A ausência de uma destas formas de agir pode levar a que o robô seja incapaz de satisfazer os seus propósitos.

Raciocínio Deliberativo

Pode-se definir raciocínio deliberativo (ou cognitivo) de um sistema como o conjunto de mecanismos utilizados para planear e executar acções de alto-nível. No domínio da robótica móvel, funciona essencialmente para planear caminhos globais entre diferentes pontos do espaço ambiente. Por isso, depende fortemente de informação adquirida *a priori*, ou seja, necessita de um conhecimento global (ou quase) do mundo para prever o resultado das suas acções e assim planear de forma ideal as suas trajectórias ou tarefas no(s) modelo(s) do mundo que possui. O planeamento deliberativo exige normalmente um elevado esforço computacional, obrigando a paragens para processar a informação.

O raciocínio deliberativo assume que o seu modelo do mundo é baseado em informação consistente, fiável e correcta. Se a informação que o módulo de raciocínio utiliza está incorrecta ou simplesmente foi alterada após a sua aquisição, as suas acções podem fracassar. Assim, em ambientes dinâmicos, torna-se potencialmente perigoso confiar na informação adquirida no passado e que pode já não ser válida. Por este motivo, os modelos do mundo, para além de informação *a priori*, incluem informação actual dos sensores. A capacidade de percepção, no entanto, resulta do processamento da informação integrada dos modelos do mundo, ou seja, a percepção não é ligada directamente à acção.

O raciocínio deliberativo surge geralmente associado às arquitecturas centralizadas e hierárquicas (analisadas na secção 2.2.2)

Controlo Reactivo

As arquitecturas puramente reactivas (baseadas em controlo reactivo) são concebidas para responder directa e instantaneamente a estímulos do meio ambiente, através de mecanismos que mapeiam directamente a percepção à acção. O sistema age sempre de acordo com a percepção actual do mundo. Os métodos de controlo reactivo são métodos desprovidos de memória, ou de memória temporal curta, ou seja, não é guardada informação relativa a estados previamente observados, sendo a acção actual independente dos estados passados. Para além desta característica, evitam-se quaisquer representações explícitas do mundo. Para o bom desempenho desta arquitectura, assume-se que a informação sensorial é essencialmente livre de ruído e que as características do meio envolvente imediato se encontram sempre visíveis/detectáveis pelos sensores do sistema[1]. Contudo, sem representações do ambiente, nem planos, o robô não tem meios para alcançar os seus objectivos, a não ser que estes se encontrem directamente à vista. Caso isso não aconteça, o robô poderá ter de navegar aleatoriamente até que o objectivo seja visualizado. Esta limitação leva a que o robô navegue para becos sem saída ou execute ciclicamente a mesma trajectória sem encontrar uma saída.

O controlo reactivo é normalmente implementado através de comportamentos: unidades básicas de raciocínio que executam tarefas muito específicas, respondendo de imediato a estímulos do meio (ver secção 2.3).

[1] No entanto, estas características quase nunca são satisfeitas

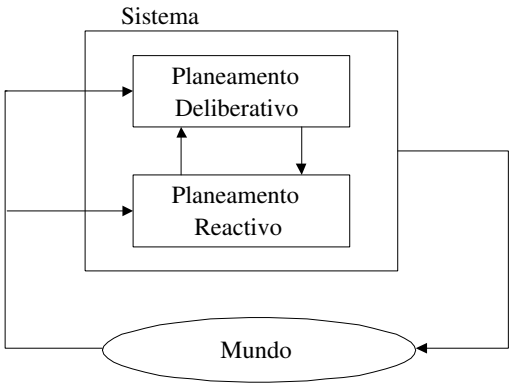


Figura 2.1: Arquitectura híbrida

Arquitecturas Híbridas

As arquitecturas híbridas surgem como uma solução que combina os pontos fortes das duas anteriores arquitecturas. A questão central consiste no desenvolvimento de uma arquitectura unificada que garanta ao sistema a execução de um plano de forma robusta. Uma forma de interligar as duas arquitecturas passa pela integração do planeamento de alto nível e controlo reactivo numa estrutura hierárquica de duas ou mais camadas (figura 2.1). Essa combinação pode ser realizada de duas formas:

- O planeamento deliberativo e a execução reactiva estão envolvidos em actividades diferentes e com tempos de escalonamento diferentes. Planear ou reagir depende da situação actual. Deverá existir um módulo de coordenação que decide qual dos dois níveis de raciocínio deve estar activo.

- A execução das acções do planeador deliberativo ocorre apenas sob a observação do sistema reactivo. Por outras palavras, o sistema deliberativo envia ao módulo reactivo o plano das suas acções e este terá de agir operando simultaneamente com a informação proveniente da camada superior e também com os estímulos do meio ambiente.

2.2.2 Processamento Centralizado vs. Processamento Distribuído

Arquitecturas Centralizadas

Os primeiros robôs que surgiram no domínio da robótica móvel (na década de 80) funcionavam de uma forma clássica: *sentir-planear-agir*. Ou seja, recolhiam todos os dados sensoriais disponíveis criando um modelo completo do ambiente estático, faziam o planeamento das acções a tomar de acordo com o modelo adquirido, e finalmente executava-as (figura 2.2). O robô parava para adquirir nova informação do ambiente e voltava a executar todo o processo. Esta é uma abordagem de *decomposição funcional ou horizontal*. Os módulos funcionais formam uma cadeia pela qual flui a informação desde o módulo de aquisição de sensores até aos actuadores. Este tipo de sistema encontra-se bem adaptado aos ambientes estáticos. Nestas arquitecturas, todos os dados sensoriais recolhidos são fundidos num único modelo monolítico do mundo, e todas as acções são executadas de acordo com o plano de trajectória criado para esse modelo do mundo, independentemente da simplicidade e urgência das acções a tomar. Isto significa que, uma vez formulado o plano de acção, este será executado em malha aberta, isto é, sem ter em conta possíveis mudanças do meio ambiente, e também sem saber se as acções estão a ser executadas como planeadas. Para além desta limitação, este tipo de arquitectura apresenta uma grande fragilidade, no sentido em que basta que um dos seus módulos não esteja a funcionar correctamente para que todo sistema falhe, pois as várias partes estão interligadas e sempre dependentes umas das outras. Para ultrapassar as limitações das arquitecturas centralizadas, surgiram as arquitecturas distribuídas: arquitecturas hierárquicas e arquitecturas baseadas em comportamentos.

Arquitecturas Hierárquicas

Contrariamente às arquitecturas centralizadas, numa arquitectura hierárquica, o sistema encontra-se dividido em vários níveis, cada um deles a funcionar com diferentes escalas de tempo e níveis de abstracção. Os níveis podem ou não ter a mesma decomposição funcional (constituída pelos mesmos módulos que a arquitectura centralizada, sentir-planear-agir), mas com níveis diferentes de raciocínio (figura 2.3). No nível mais alto, formulam-se planos mais globais e menos específicos. Os requisitos de tempo para formular o plano

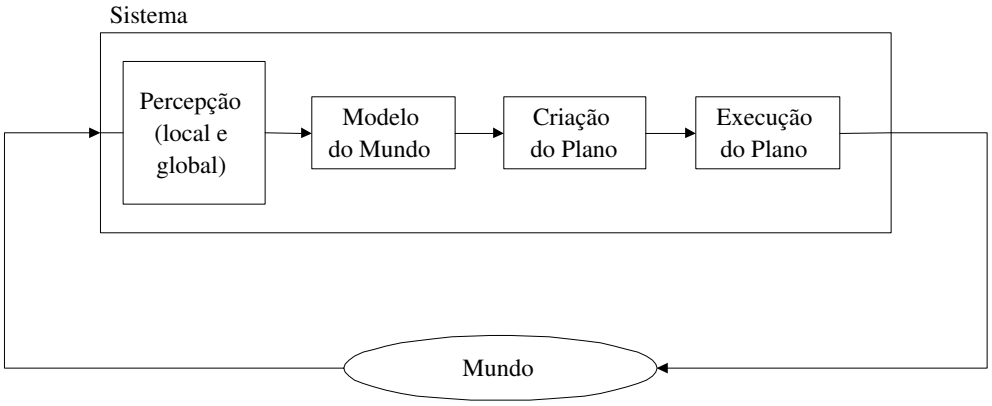


Figura 2.2: Arquitectura funcional ou decomposição horizontal

não são rígidos. À medida que se desce na hierarquia, o planeamento é focado em regiões do mundo mais pequenas, mas exigindo soluções mais rápidas. Nos níveis inferiores, são necessárias respostas em tempo real, mas o planeador apenas utiliza informação do espaço mais próximo. Apesar desta abordagem apresentar aspectos de planeamento deliberativo e reactivo, na estrutura da arquitectura hierárquica, a informação flui de cima para baixo. Isto significa que os níveis superiores passam sucessivamente os seus planos às camadas inferiores assumindo que os seus comandos são executados como esperado. A execução desses planos tem de ser constantemente monitorizada de forma a detectar incumprimento dos planos. Esta estrutura de cima para baixo tende, no entanto, a restringir a capacidade de reacção do sistema.

Arquitectura de Comportamentos

A arquitectura comportamental constitui uma classe que difere substancialmente dos outros sistemas de controlo. O sistema não é composto por módulos funcionais tradicionais (percepção-planeamento-acção) mas sim por comportamentos individuais que implementam tarefas muito específicas (percepção-acção). Este tipo de arquitectura tem sido cada vez mais utilizado no domínio da robótica móvel.

Dado o ênfase das arquitecturas baseadas em comportamentos reactivos neste trabalho, dedica-se a próxima secção a esta arquitectura.

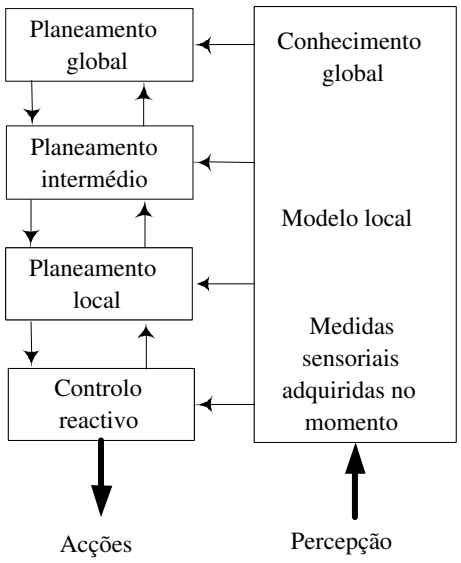


Figura 2.3: Arquitectura hierárquica

2.3 Arquitecturas Baseadas em Comportamentos

Nas arquitecturas baseadas em comportamentos, ou simplesmente *arquitecturas comportamentais*, a acção de controlo do robô resulta da coordenação/interacção de comportamentos (ver secção 3.1). O processamento é do tipo de baixo para cima, ou seja, iniciado pelo estímulo, enquanto o processamento de cima para baixo das outras arquitecturas é iniciado pela intenção. O conceito de comportamento aplicado à robótica nasceu de estudos da neuro-ciência, da psicologia do comportamento, e da etologia[2] [Ark98]. Muitos dos métodos aplicados em robótica móvel, tais como as redes neuronais e o princípio das forças vectoriais tiveram origem nesses estudos. A etologia, por exemplo, fornece uma linguagem apropriada à definição dos comportamentos no domínio da robótica. Segundo a etologia, o comportamento animal pode categorizar-se em 3 classes:

- **Reflexos** - respostas rápidas, automáticas e involuntárias activadas por um certo estímulo do ambiente. A resposta reflexiva persiste apenas enquanto durar o estímulo. Para além disso, a intensidade da resposta está relacionada com a intensidade do estímulo. Os reflexos são utilizados para a locomoção e outras actividades altamente coordenadas.

[2]Ciência que estuda o comportamento animal no seu ambiente natural

- **Contributos** - respostas que orientam o animal em direcção ou em afastamento de um estímulo (atractivo ou adverso). Os contributos ocorrem em resposta a fenómenos visuais, químicos, mecânicos e electromagnéticos (por exemplo, o cheiro de formigas estimula o papa-formigas).

- **Padrões de acções constantes** - respostas activadas por um estímulo, mas que persistem para além deste. A intensidade e duração da resposta não depende da intensidade e duração do estímulo. Este tipo de respostas pode ser motivado, ao contrário dos reflexos, e pode resultar de um conjunto muito mais alargado de estímulos do que os que levam a um simples reflexo. São geralmente comportamentos complexos que fornecem um conjunto ordenado de acções em resposta a um estímulo (por exemplo, uma estratégia para evitar um predador).

O conceito de *diagramas* também surge associado à teoria comportamental animal, sendo definido como a combinação de reflexos, contributos e padrões, activados em resposta a uma combinação apropriada de estímulos.

Aplicado à robótica, Arkin [Ark98] define comportamento: *um comportamento individual corresponde ao par estímulo/resposta para um dado ambiente, modulado pela atenção e determinado pela intenção.* Modulado pela *Atenção*, pois o comportamento é determinado pelo contexto actual do ambiente, e é a *Intenção* que determina quais os comportamentos que devem estar activos baseando-se nos objectivos do agente. Ao longo da tese, comportamento será definido pelo autor da seguinte forma:

> *Comportamento é uma unidade básica de controlo do tipo estímulo/resposta que permite implementar uma tarefa/objectivo particular. Cada comportamento associa directamente estímulos a acções, sendo evitadas representações simbólicas do mundo*

Resumem-se, a seguir, algumas propriedades dos comportamentos aplicados à robótica [Ark98]:

- Comportamentos complexos podem ser construídos a partir de comportamentos mais simples através de combinações ou sequências.

- Estratégias perceptuais devem ser sintonizadas de modo a responder apenas a estímulos específicos relevantes para uma determinada acção.

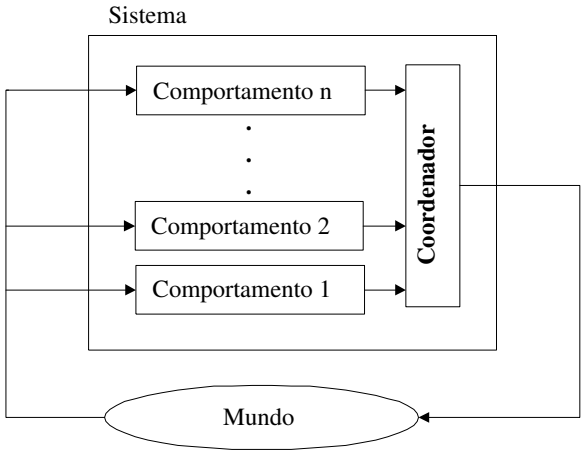


Figura 2.4: Arquitectura baseada em comportamentos

- Comportamentos competitivos devem ser coordenados através de mecanismos de selecção/arbitração.

- Comportamentos puramente reactivos (*comportamentos reflexivos*) não podem utilizar quaisquer representações do mundo (modelos mundo), nem guardar informação de estados passados.

Nas arquitecturas comportamentais, o sistema é composto por comportamentos individuais em que cada um executa uma tarefa específica e limitada. O comportamento encapsula a capacidade de percepção, o planeamento e a acção necessários para a execução de determinada tarefa de controlo. Este tipo de arquitectura encontra-se bem adaptado a ambientes dinâmicos, daí ser utilizado em arquitecturas reactivas. Caracteriza-se pela sua robustez, na medida em que a falha de uma parte do sistema não obriga à falha de todo o sistema. O *calcanhar de Aquiles* do controlo baseado em comportamentos reside no módulo arbitrador que tem por função coordenar todos os comportamentos de forma a alcançar uma resposta coerente do sistema (ver figura 2.4). Os métodos de *coordenação/arbitração* dos comportamentos serão descritos mais à frente na secção 3.1.

2.4 Modelação do Ambiente

A capacidade de navegação de um robô depende da informação/conhecimento que possui do espaço em que se insere. Essa informação pode ser obtida em tempo real ou não, aquando da navegação, ou pode simplesmente ter sido armazenada *a priori*. De acordo com essa classificação, o conhecimento pode também ser caracterizado atendendo à sua durabilidade, distinguindo-se duas formas de conhecimento: *persistente (memória de longo prazo)* e *transitório (memória de curto prazo)*. A primeira refere-se à informação previamente adquirida e que permanece estática ao longo das missões do robô. A segunda, consiste no conhecimento adquirido dinamicamente à medida que o robô se desloca.

Cada tipo de informação encontra-se adequado a um determinado nível de planeamento: local de baixo nível, ou global de alto nível. Os modelos *a priori* (adquiridos e armazenados previamente) contêm informação da estrutura do ambiente (topologia) que permanece estática (e.g. salas, portas, mobília). Dão informação de como ir de uma determinada posição para a outra, podendo incluir informação acerca de pontos de referência úteis para auto-localização e calibração. Os modelos de tempo real são adquiridos pelo equipamento sensorial do robô e retiram informação do meio ambiente não incluída nos modelos adquiridos *a priori*. Devem ainda fornecer ao robô capacidade de reacção aos estímulos do meio envolvente.

A figura 2.5 resume os modelos do mundo e a sua interligação com diferentes níveis de planeamento. Como modelos do mundo tem-se:

- Modelos topológicos - guardam informação topológica da estrutura do meio ambiente, ou seja, contêm a informação de como estão interligadas as várias partes de um determinado espaço. Consiste tipicamente num grafo não direccional em que os nós representam estruturas tais como salas, corredores, cantos, portas, etc., ou ainda, informação mais detalhada como identificadores, marcas no terreno e características geométricas. Esta informação do mundo serve de suporte à camada de mais alto nível de uma arquitectura, ou seja, à camada onde são definidas e planeadas as tarefas que visam atingir os objectivos finais (camada do planeamento global de missões ou caminho), o que pode ser conseguido, por exemplo percorrendo um grafo que une a posição actual à posição destino. Este método não se adequa ao planeamento local, pois não possui informação métrica.

- Modelos geométricos - possuem informação métrica da estrutura estática do espaço. Representam paredes, portas, através de linhas a duas dimensões (2D) ou três dimensões (3D), ou através de modelos completos de CAD (Desenho Assistido por Computador). Como possuem tanto informação topológica como informação métrica, estes modelos são apropriados para o planeamento de trajectórias[3], tanto para navegação global, como para navegação local.

- Modelos locais - são adquiridos pelo sistema sensorial em tempo real. Baseiam-se normalmente em grelhas bi-dimensionais de ocupação, cuja resolução varia normalmente entre os 5 e 20 cm (1 célula da grelha). Nos métodos mais simples, cada célula possui um valor binário indicando a presença ou ausência de obstáculos. Métodos mais rigorosos incluem vários valores por célula, especificando a probabilidade de estar ocupada. Este tipo de representação adequa-se à informação proveniente de sonares. Moravec e Elfes [Elf87] desenvolveram o método de *grelhas de certeza* para mapear medidas imprecisas de sonares através de funções de distribuição de probabilidade. Outras abordagens baseadas em lógica difusa [Sie94], evidência teórica [RP99] e redes neuronais [TBBF98] [CN00] são também utilizados para combater a incerteza das medidas dos sonares.

  Estes modelos são apropriados para a representação do ambiente nas proximidades do robô, sendo por isso utilizados para navegação local (planeamento local de trajectórias).

[3]Note-se que planeamento de trajectórias tem um significado diferente de planeamento de caminhos. O primeiro refere-se a um planeamento de posições x, y, enquanto o segundo refere-se a um planeamento simbólico (e.g. para ir de A para E tem de passar por B, C e D)

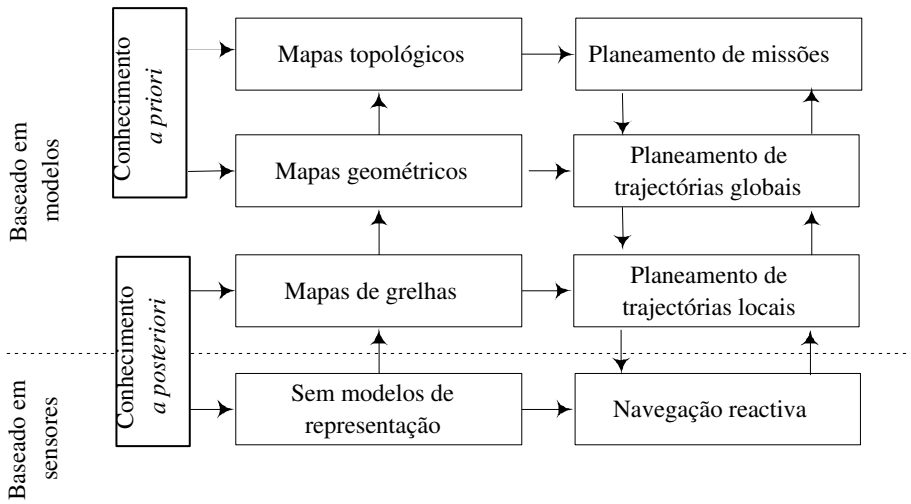


Figura 2.5: Modelos de conhecimento do mundo e sua interligação com os vários tipos de navegação

Capítulo 3

Navegação Reactiva

Este capítulo apresenta as principais estratégias de navegação local baseadas em comportamentos reactivos. Serão essencialmente discutidos e analisados métodos de desvio de obstáculos, não obstante as referências a métodos de planeamento global, tanto para fazer algumas distinções, assim como para referir o papel do planeamento global em estratégias locais. O capítulo começa com uma secção onde se descrevem métodos de coordenação de comportamentos.

3.1 Coordenação de Comportamentos

Após ter-se decidido utilizar uma arquitectura baseada em comportamentos, é necessário definir a forma como estes serão combinados. Da combinação dos comportamentos deverá surgir uma resposta única coerente utilizada para controlar o robô. Os métodos que combinam a saída dos comportamentos classificam-se em duas categorias [Ark98]: métodos competitivos e métodos cooperativos também designados métodos de fusão de comandos.

Métodos Competitivos

Nos métodos competitivos, o coordenador actua como um mecanismo do tipo *o vencedor sobrepõe-se a todos* (VST), ou seja, existe apenas um comportamento que, sobrepondo-se a todos os outros, controla o robô. Os métodos são os seguintes:

- *Arbitração*: o mecanismo de arbitração baseia-se numa estrutura de prioridades

fixas em que existe uma hierarquia de comportamentos. Os comportamentos das camadas superiores podem sobrepor-se aos comportamentos das camadas inferiores (figura 3.1). Este método foi desenvolvido por Rodney Brooks [Bro86] (ver secção 3.2.1).

- *Votos*: neste método cada um dos comportamentos vota a favor ou contra determinadas acções de controlo. A acção mais votada é a que controla o veículo (figura 3.2). Este tipo de coordenação foi implementado na arquitectura DAMN (Arquitectura Distribuída para Navegação Móvel) [Ros95] (ver secção 3.2.6)

Métodos Cooperativos ou Fusão de Comandos

Contrariamente aos métodos competitivos, em que um único comportamento controla o sistema, os métodos cooperativos fornecem ao sistema a capacidade de ser controlado pela combinação da saída de vários comportamentos, oferecendo a vantagem das decisões serem tomadas de acordo com várias considerações simultaneamente. A questão central reside em encontrar uma forma adequada de fusão que garanta uma resposta coerente do sistema. Utilizam-se os seguintes métodos:

- *Soma Vectorial*: este tem sido o método mais utilizado. A saída de cada comportamento é representada por um vector ao qual está associado uma determinada força ou ganho, g_i. Este ganho é multiplicado pelos vectores antes de serem somados, resultando uma saída global que depende do ganho de cada comportamento (figura 3.3). Este processo é baseado nos método dos campos potenciais [Kha86] [KT86] [Ark87] (ver secções 3.2.2, 3.2.3, 3.2.4).

- *Lógica Difusa*: nesta abordagem, cada comportamento possui uma função de intenção (representada por um conjunto difuso) que pode ser combinada através de operadores lógicos difusos. A combinação das acções dos comportamentos faz-se através de um processo de difusão e a acção seleccionada deriva de um processo de desdifusão [YP95] [Saf97a] [RSK95] (ver secção 3.2.7).

- *Selecção de Orientação*: este método consiste na avaliação de zonas livres em todas as direcções à volta do veículo, e consequente escolha da direcção que melhor se adequa à passagem do veículo no caminho da meta. O método mais conhecido

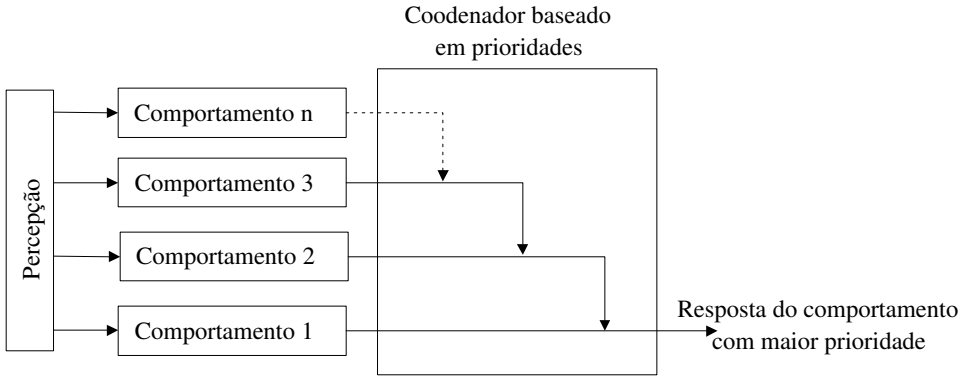


Figura 3.1: Coordenação baseada em prioridades

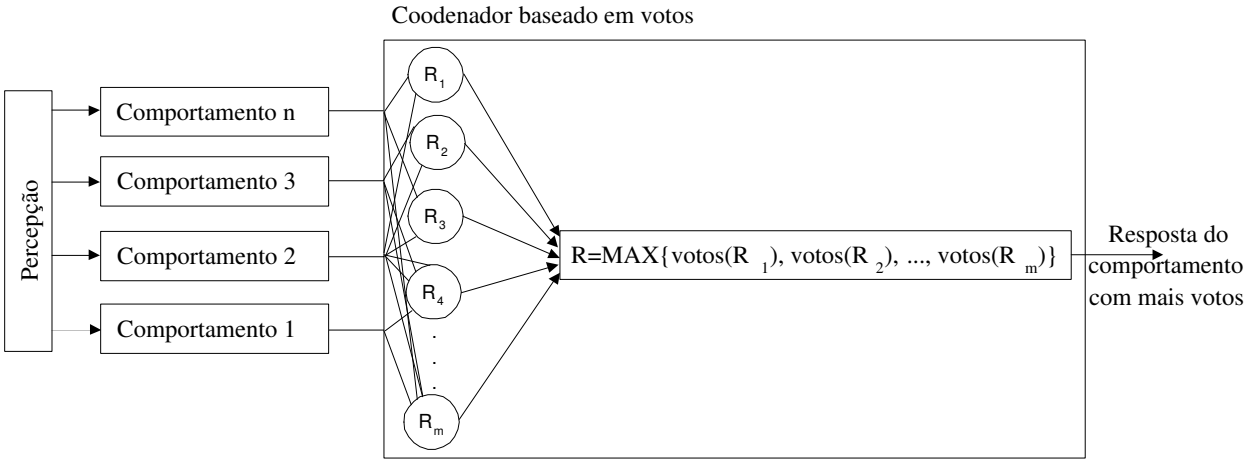


Figura 3.2: Coordenação baseada em votos

desta abordagem é o histograma de campos vectoriais (VFH) [BK91] (ver secção 3.2.5).

3.2 Estratégias de Navegação Baseadas em Comportamentos

Os métodos de navegação baseados em comportamentos são especialmente apropriados à navegação reactiva. O controlo do robô depende da informação actual dos sensores, podendo por vezes depender também da informação adquirida e guardada durante um curto espaço de tempo (conhecimento de curto prazo). A navegação reactiva é utiliza-

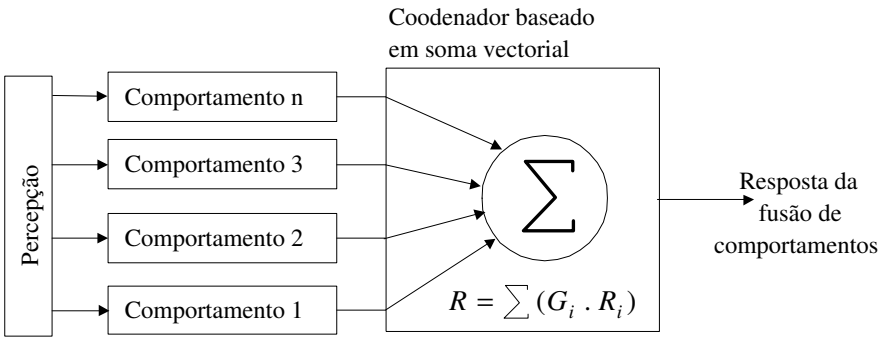


Figura 3.3: Coordenação baseada em soma vectorial

da normalmente para desvio local de obstáculos, evitamento de colisões, seguimento de superfícies, pistas ou alvos móveis, e procura de meta.

3.2.1 Arquitectura de Submissão

A *arquitectura de submissão* desenvolvida por Rodney Brooks [Bro86] é uma arquitectura comportamental amplamente conhecida cuja arbitração é baseada em prioridades. Os comportamentos são puramente reactivos e são representados por camadas separadas que formam níveis de competência. Cada comportamento é representado por uma máquina de estados finitos (MEF) (figura 3.4) que encapsula uma função que define um determinado comportamento. A MEF tem linhas de entrada e de saída interligadas com outros módulos (comportamentos). A designação *submissão* surge do processo de coordenação entre os comportamentos. Comportamentos complexos de camadas superiores subordinam/sobrepõem-se aos comportamentos mais simples das camadas inferiores. A coordenação de comportamentos segue dois mecanismos primários: *inibição* que evita que um sinal seja transmitido para o actuador; e *supressão* que evita que o sinal seja transmitido e substitui-o por uma mensagem de supressão (figura 3.5). Vejamos o caso da inibição através da figura 3.5a): se o comportamento A estiver inactivo, a saída do comportamento B controla o veículo. Se o comportamento A estiver activo, a sua saída inibe a saída do comportamento B e é agora o comportamento A que controla o veículo. No caso do mecanismo de supressão ilustrado na figura 3.5b), se o comportamento B estiver activo e o comportamento A inactivo, a entrada do comportamento B será a entrada B. No entanto, se o comportamento A estiver activo, a sua saída passará a ser a entrada do comportamento B. Em ambos os mecanismos, é o comportamento de nível mais elevado

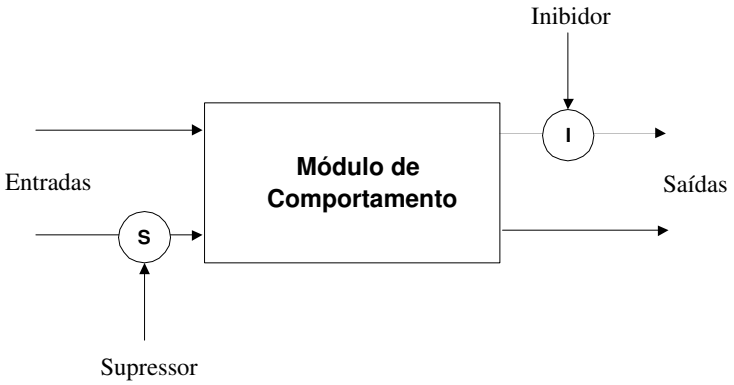


Figura 3.4: MEF utilizada na *arquitectura de submissão*

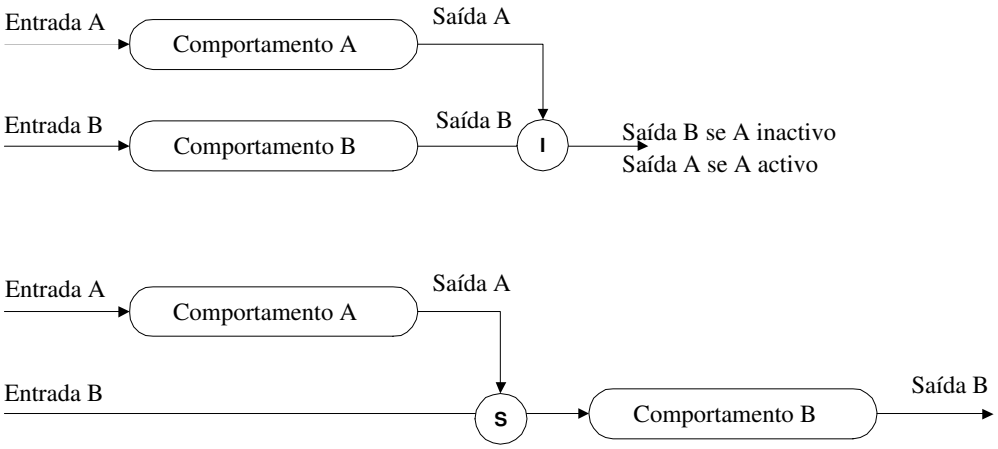


Figura 3.5: Arbitração na *arquitectura de submissão* através de a) Inibição e b) Supressão

que se sobrepõe ao comportamento inferior.

Utilizando este tipo de arbitração, existe apenas um comportamento que, sobrepondo-se a todos os outros, controla o veículo. Desta forma, não é possível satisfazer dois ou mais objectivos simultaneamente.

3.2.2 Campos Potenciais

O método dos *campos potenciais artificiais* foi desenvolvido por Krogh (*campos potenciais generalizado*) [KT86] e Khatib (*campos potenciais clássico*) [Kha86] para planeamento de trajectórias suaves em robótica móvel e manipuladores, respectivamente. Segundo esta abordagem, o robô é tratado como uma partícula que se move sob a influência de um

campo potencial artificial U_{art} [Kha86]. O campo potencial é definido como a soma de um potencial atractivo que puxa o robô em direcção ao objectivo e de um campo potencial repulsivo que repele o robô dos obstáculos (figura 3.6). Seja $X \subset \mathbb{R}^2$ o espaço de configuração do robô e a posição do robô um ponto $\mathtt{x} \in X$ em que $\mathtt{x} = (x, y) \in X$ num determinado referencial, o robô é guiado pela força (vectorial) $f_{art}(\mathtt{x})$ induzida pelo campo potencial (escalar) $U_{art}(\mathtt{x})$ que lhe dá a melhor direcção e velocidade de movimento a seguir:

$$f_{art}(\mathtt{x}) = -\nabla U_{art}(\mathtt{x}) \tag{3.1}$$

em que ∇ representa o sinal de gradiente. O campo potencial artificial U_{art} é dado pela soma do campo potencial atractivo U_a e do campo potencial repulsivo U_r associado aos obstáculos:

$$U_{art}(\mathtt{x}) = U_a(\mathtt{x}) + U_r(\mathtt{x}) \tag{3.2}$$

U_a é independente dos obstáculos assim como U_r é independente da posição do objectivo. A força artificial resultante que guia o robô é dada por:

$$f_{art}(\mathtt{x}) = -\nabla U_a(\mathtt{x}) - \nabla U_r(\mathtt{x}) = f_a(\mathtt{x}) + f_r(\mathtt{x}) \tag{3.3}$$

A força atractiva f_a é normalmente proporcional à distância entre o robô e a posição do objectivo:

$$f_a(\mathtt{x}) = -K_a(\mathtt{x} - \mathtt{x}_{obj}) \tag{3.4}$$

em que K_a é uma constante de proporcionalidade positiva e $\mathtt{x}_{obj}$ representa a posição do objectivo no espaço X. Por forma a conseguir a estabilidade do sistema, Kathib introduz ainda uma força $F_v = -K_v\dot{\mathtt{x}}$ função da velocidade, resultando uma força atractiva:

$$f_a(\mathtt{x}) = -K_a(\mathtt{x} - \mathtt{x}_{obj}) - K_v\dot{\mathtt{x}} \tag{3.5}$$

A força repulsiva f_r resulta da soma das forças repulsivas associadas a cada um dos obstáculos f_{ri}:

$$f_r(\mathtt{x}) = \sum_i f_{r_i}(\mathtt{x}) \tag{3.6}$$

No caso de desvio local de obstáculos, os campos são calculados localmente em tempo real, com base em leituras actuais dos sensores do robô. A força repulsiva, é por isso mais correctamente expressa por:

$$f_r(\mathtt{x}) = \sum_i f_{sensor_i}(\mathtt{x}) \tag{3.7}$$

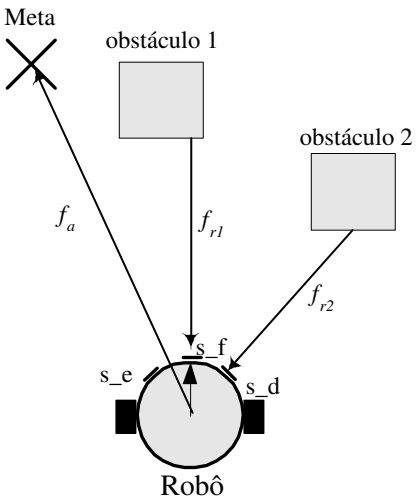


Figura 3.6: Exemplo de um robô equipado com três sensores sujeito a forças repulsivas (f_r) e a uma força atractiva (f_a) em direcção à meta. As forças repulsivas e atractiva do desenho não representam a intensidade das forças, mas apenas a sua direcção. Para o cálculo da intensidade deverá seguir as expressões 3.8 e 3.5 respectivamente

As forças repulsivas não devem afectar o movimento do robô quando este se encontra afastado dos obstáculos. Por este motivo, a força repulsiva é geralmente dividida em duas regiões de influência. A força repulsiva é dada simplificadamente por:

$$f_{ri}(\mathtt{x}) = \begin{cases} \frac{K_r}{d^2} & \text{se } d \geq \mathtt{x}_0 \\ 0 & \text{se } d < \mathtt{x}_0 \end{cases} \tag{3.8}$$

em que d corresponde à menor distância[1] entre o robô e um obstáculo O e $\mathtt{x}_o$ representa a distância limite de influência do campo potencial, ou seja, para uma distância superior a $\mathtt{x}_o$, o robô não conta com a influência do obstáculo. Quando se encontra a uma distância inferior a $\mathtt{x}_o$, o robô fica sujeito a forças repulsivas que seguem o princípio da gravidade, ou seja, inversamente proporcionais ao quadrado da distância entre o veículo e os objectos circundantes. Existem, no entanto outras abordagens [Ark90].

O método dos campos potenciais, foi inicialmente desenvolvido para desvio local de obstáculos em tempo real, ou seja, para situações em que não há informação *a priori* do meio ambiente. O cálculo da força artificial baseia-se apenas em informação actual dos

[1] É importante referir que o potencial é difícil de utilizar para objectos assimétricos, pois a separação entre a superfície do obstáculo e as superfícies equipotenciais varia muito. Kathib propôs por isso o campo potencial função da distância mínima a um determinado obstáculo

sensores para guiar o robô, sendo por isso considerado um método reactivo. O método dos campos potenciais pode, no entanto, ser utilizado como método global, isto é, para planeamento global de trajectórias. Utilizando um modelo pré-definido do mundo, podem ser utilizados métodos sistematizados de planeamento de trajectória [Lat91], eliminando algumas das desvantagens inerentes aos campos potenciais, nomeadamente a susceptibilidade de o robô ficar preso devido a mínimos locais (figura 3.7) e evidenciar comportamento cíclico (figura 3.8). Um mínimo local surge quando o somatório das forças repulsivas com a força atractiva é zero, levando à paragem do robô. Este problema e outros tais como a incapacidade de passar entre obstáculos pouco espaçados, a ocorrência de oscilações na presença de obstáculos e oscilações em passagens estreitas foram discutidos em [KB91] e apresentados como limitações inerentes aos campos potenciais. Conhecendo todo o espaço de configuração (com descrição dos obstáculos), é possível especificar uma função potencial com um único mínimo local localizado na posição objectivo. Note-se, no entanto, que quando utilizado para planeamento global de trajectórias, o campo potencial total tem de ser calculado previamente, e de uma só vez, o que exige um elevado esforço computacional. Na prática, na presença de ambientes dinâmicos, torna-se sempre necessário uma reactualização (em tempo real) dos campos potenciais nas proximidades do robô. A figura 3.9a) mostra as linhas equipotenciais de todo o campo potencial e uma trajectória que segue o gradiente da função potencial. A figura 3.9b) mostra as forças vectoriais que derivam do campo potencial total e que são utilizadas como comandos para guiar o robô.

Para além da classificação de método local vs. global, os campos potenciais são também classificados de acordo com o tipo de potencial: *clássico* ou *generalizado*. O conceito de *campos potenciais generalizado* foi introduzido por Krogh [KT86]. Segundo esta abordagem, a intensidade do campo potencial não depende apenas da posição `x` do robô (distância aos obstáculos) como na abordagem clássica, mas também da velocidade `v` do robô nesse instante. O potencial é calculado em função da proximidade aos obstáculos e também do tempo até um potencial impacto (tempo de colisão). Krogh define o potencial como o inverso da *margem de tempo para desvio* (MTD), que é a diferença entre o tempo que o robô levaria até que a velocidade fosse zero se desacelerasse à taxa mínima (tempo máximo, t_{max} (tempo de colisão)) e o tempo que levaria até que a velocidade fosse zero se desacelerasse à taxa máxima (tempo mínimo, t_{min}). O potencial repulsivo é desta

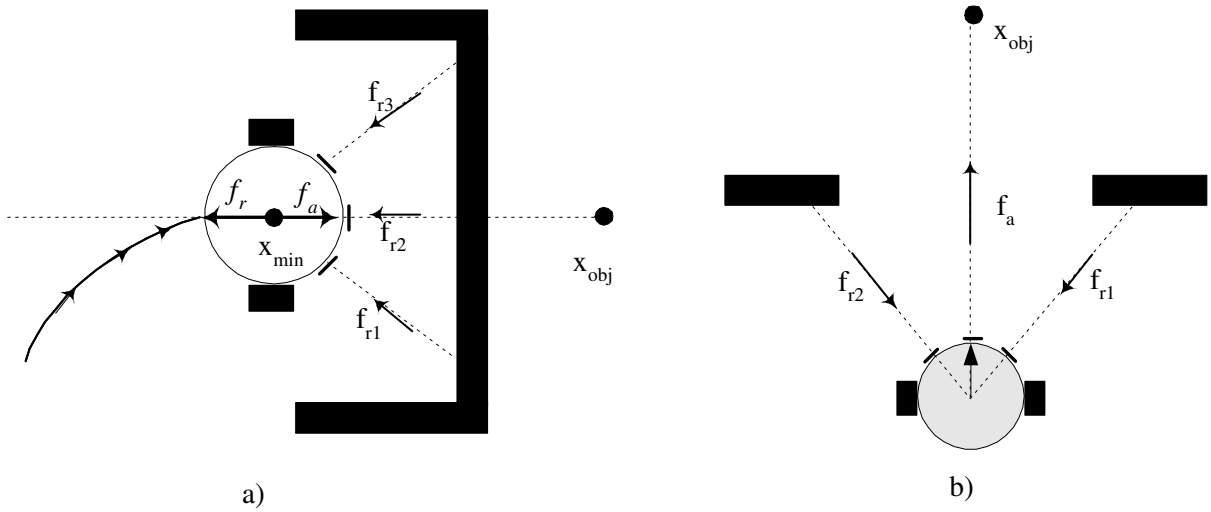


Figura 3.7: Dois cenários possíveis para a ocorrência de mínimos locais: a) Na posição x_{min} a força repulsiva iguala a força atractiva resultando uma força total nula. Esta situação pode levar a paragem do robô; b) Apesar do espaço à frente d o robô se encontrar desimpedido, as forças repulsivas induzidas pelas leituras dos sensores laterais podem levar a uma situação de mínimo local

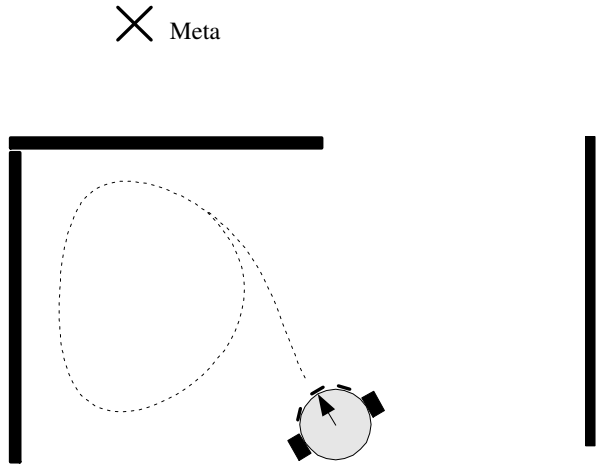
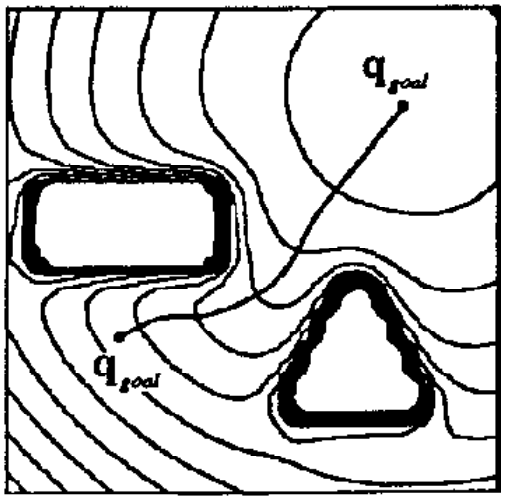


Figura 3.8: Exemplo de um comportamento cíclico do robô: a falta de conhecimento do mundo e de memória levam a que o robô execute a mesma trajectória repetidamente sem que disso se dê conta e sem capacidade para sair desta situação

a)



b)

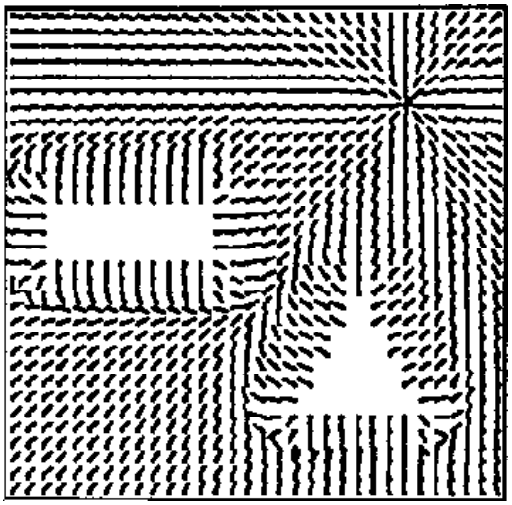


Figura 3.9: a) Linhas equipotenciais (figura reproduzida de [Lat91]); b) Forças vectoriais (figura reproduzida de [Lat91])

forma dado por:

$$U_r(\mathtt{x}, \mathtt{v}) = \frac{\mathtt{v}_0}{\mathtt{v}_0^2 - 2a_{max}\mathtt{x}_0} \tag{3.9}$$

em que $\mathtt{v}_0$ é a velocidade em direcção ao obstáculo, $\mathtt{x}_0$ é a distância ao obstáculo e a_{max} é a aceleração máxima do robô. Esta abordagem traz principalmente duas vantagens em relação à abordagem clássica:

- Obstáculos já ultrapassados ou paralelos ao robô deixam de influenciar a trajectória do veículo, permitindo que este siga uma trajectória mais directa para o alvo.
- Se o veículo se encontrar parado (velocidade nula) junto de um obstáculo, o potencial criado pelo obstáculo não exerce forças repulsivas contra o robô.

Estas vantagens levam implicitamente a que sejam definidas trajectórias mais suaves.

Algoritmos de navegação local

A capacidade de desvio de obstáculos do robô, depende obviamente da forma como é utilizado o campo potencial na navegação. Existem dois métodos popularmente conhecidos: *direcção do campo de forças* e o *controlo de forças* [Til90].

O método de *direcção do campo de forças* ignora a velocidade e aceleração do veículo e utiliza o potencial para gerar uma sequência de pontos que define a trajectória. Este método, tal como o nome indica, resume-se a escolher em cada avanço do robô a direcção do campo de forças. Em termos algorítmicos pode ser apresentado da seguinte forma:

```
x_actual = posição inicial
repete
     calcula f_artificial = f_atr + f_rep
     dir = vector unitário na direcção do campo de força
     x_actual ← x_actual + avanço×dir
até (distância à meta) < limiar
```

No método de *controlo de força*, abordagem seguida por Krogh e Kathib, o campo

define a direcção de aceleração em vez de escolher simplesmente a nova direcção do robô. O algoritmo ilustra de forma simplificada este método (são impostos limites à aceleração e velocidade máxima):

```
// a_max = aceleração máxima
// v_actual = velocidade actual
// v_max = velocidade máxima
v_actual = velocidade mínima
d_actual = direcção à meta
repete
     calcula f_artificial = f_atr + f_rep
     f_artificial ← min(f_artificial,a_max)
     v_actual ← v_actual + f_artificial×T
     v_actual ← min(v_actual,v_max)
     x_actual ← x_actual + v_actual×T
até (distancia à meta) < limiar
```

A variável T define um determinado período de tempo. O método de *controlo de forças* pode não garantir a ausência de colisões, uma vez que devido à inércia, o robô pode não conseguir parar ou virar a tempo. O método de *direcção de campo de forças* garante o desvio de obstáculos a tempo, no entanto, não são conseguidas trajectórias tão suaves como no método de controlo de força.

Existem muitos métodos de navegação local baseados no princípio dos campos potenciais, i.e., em que o robô é guiado pelo efeito de forças atractivas e repulsivas. Dois dos métodos mais conhecidos, o método *diagramas motores* e o método *VFF* (campo de forças virtuais) serão abordados nas secções seguintes.

3.2.3 Diagramas Motores

A teoria dos diagramas motores foi desenvolvida no seio da psicologia cognitiva e comportamento animal, como um meio para codificar e coordenar as acções motoras e a actividade perceptual. Estes conceitos foram aplicados no domínio da robótica. Nes-

te contexto, *diagramas ou comportamentos* representam unidades básicas de actividade motora e percepção.

O método dos *diagramas motores* desenvolvido por Arkin [Ark87] é um método de navegação local reactivo baseado nos campos potenciais. Os diagramas motores representam comportamentos primitivos, nos quais estão embebidos *diagramas perceptuais*. Estes fornecem a informação perceptual específica apenas para um determinado comportamento. A saída de cada comportamento motor é representada por um vector que indica uma direcção e velocidade a seguir. A coordenação entre os vários comportamentos obtém-se através da soma vectorial da saída de todos os comportamentos motores, resultando num único vector utilizado para guiar o veículo em tempo real. As diferenças entre este método e o método clássico dos campos potenciais são descritas a seguir:

1. Implementa tarefas mais complexas que resultam essencialmente da incorporação de novos comportamentos, como o comportamento de seguimento de superfícies e de seguimento de objectos móveis, entre outros.

2. Solicita um comportamento para cada um dos obstáculos. Sempre que um obstáculo é detectado, o comportamento motor associado depende também do grau de certeza de este existir. Por exemplo, suponhamos que o robô está a ser controlado pelo diagrama motor *segue-em-frente*. Se o diagrama perceptual *encontra-obstáculos* detectar a presença de um obstáculo estático, o diagrama motor *evita-obstáculo-estático* é activado. O diagrama motor fica num estado de hibernação até que o diagrama perceptual confie suficientemente na existência dos obstáculos detectados. Se o diagrama perceptual provar que foi falso alarme, o diagrama motor é desactivado; se pelo contrário for garantida a presença do obstáculo (acima de um determinado limiar), o diagrama motor começa a produzir uma esfera de influência à volta do obstáculo. A esfera de influência e a intensidade da repulsão do obstáculo dependem da distância ao robô e também do grau de certeza da existência desse obstáculo.

3. Lida com o problema dos mínimos locais através da troca de mensagens entre comportamentos. Por exemplo, se dois comportamentos estiverem activos de forma conflituosa, resultando numa velocidade de controlo zero, um dos comportamentos poderá desactivar-se temporariamente ou então simplesmente reduzir o campo de

forças.

O método dos diagramas motores foi aplicado na arquitectura AuRA ("Autonomous Robot Architecture") desenvolvida por Arkin [Ark90]. Trata-se de uma arquitectura híbrida na qual existe informação *a priori* do espaço, assim como a capacidade de construir modelos dos obstáculos circundantes (informação armazenada a curto prazo). Estes módulos não pretendem substituir o papel dos comportamentos motores, mas sim fornecer informação útil que permita ultrapassar as limitações já referidas dos campos potenciais. Alguns dos comportamentos motores implementados na arquitectura AuRA são: 1) *segue-em-frente*, 2) *segue-para-objectivo*; 3) *evita-obstáculo-estático*; 4) *manter-no-trajecto*. Existe um vector associado a cada um dos comportamentos que indica a intensidade de velocidade, V_i, e a direcção de velocidade, $\hat{v}_d$, que o veículo deve tomar. V_i e $\hat{v}_d$ são definidos para cada um dos comportamentos da seguinte forma:

- segue-para-objectivo (figura 3.10a)):
  V_i = ganho constante
  $\hat{v}_d$ = direcção à meta detectada

- evita-obstáculo-estático[2](figura 3.10b)):
  $$V_i = \begin{cases} 0 & \text{se } d > S \\ \frac{S-d}{S-R}G & \text{se } S \leq d < R \\ \infty & \text{se } d \geq R \end{cases}$$
  $\hat{v}_d$ = radialmente ao longo de uma linha, no sentido que vai do centro do obstáculo ao centro do robô
  onde
  - S = esfera de influência (com origem no centro do obstáculo)
  - R = raio do obstáculo
  - G = ganho ajustável
  - d = distância do robô ao centro do obstáculo

- manter-no-trajecto (figura 3.10c)):
  $$V_i = \begin{cases} P & \text{se } d > W/2 \\ \frac{d}{W/2}G & \text{se } d \geq W/2 \end{cases}$$

[2]Observe-se que neste caso a força repulsiva não é inversamente proporcional ao quadrado da distância como no método clássico dos campos potenciais, mas sim inversamente proporcional à distância

$\hat{v}_d$ = ao longo de uma linha em direcção ao centro do trajecto
onde

- W = largura do caminho
- P = ganho quando está fora do trajecto
- G = ganho quando está no trajecto
- d = distância do robô ao centro do trajecto

A figura 3.10d) ilustra o efeito combinado de três diagramas motores e o trajecto percorrido pelo veículo até atingir a meta[3]

3.2.4 VFF - Campo de Forças Virtuais

O método VFF (campo de forças virtuais) foi proposto por Borenstein e Koren [BK89] como um método de desvio de obstáculos em tempo real para robôs móveis. Esta abordagem aplica o método dos campos potenciais a uma grelha de certeza [Elf87]. É considerado um método não reactivo, ou pelos menos, não puramente reactivo, na medida em que a navegação baseia-se na construção de modelos do espaço na forma de grelha (histogrâmica bidimensional). O centro da grelha coincide com o centro do robô, acompanhando os seus movimentos (figura 3.11). Este método possui memória de curto prazo, ou seja, a informação associada à grelha num determinado instante é guardada desde que esteja nos limites da grelha. A informação armazenada permite actualizar cada célula (i, j) da grelha, aumentando assim o grau de certeza da existência de um determinado obstáculo nessa posição. O processo de actualização utiliza uma abordagem simplificada por forma a garantir a sua execução em tempo real: apenas é actualizada a célula que corresponde ao centro do cone de radiação do sonar e à respectiva distância fornecida pela leitura. É assim construída uma grelha de células às quais está associada um determinado valor de certeza $C(i, j)$. Esta característica torna este método mais vantajoso em relação ao método clássico dos campos potenciais. Cada célula da grelha exerce uma força repulsiva $f_r(i, j)$ proporcional ao valor de certeza da célula e inversamente proporcional ao quadrado

[3] As figuras apresentam as forças em toda a região do espaço apenas para facilidade de compreensão. Estas são calculadas em tempo real apenas no espaço envolvente ao robô

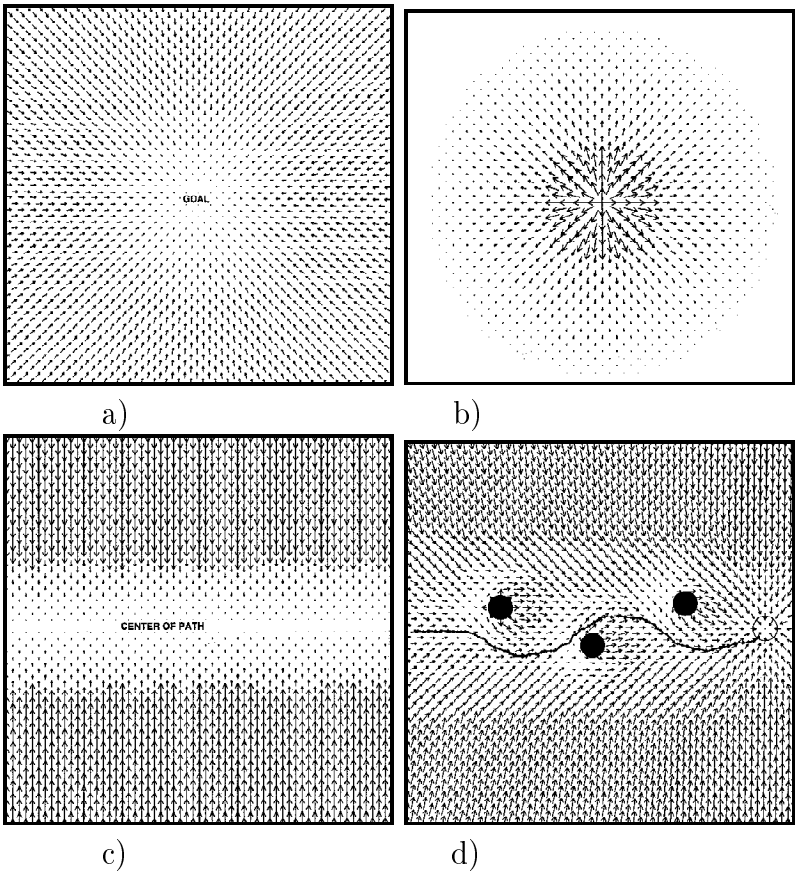


Figura 3.10: Diagramas motores (figuras reproduzidas de [Ark98]). a) Diagrama motor para seguimento de meta; b) Diagrama motor para desvio de obstáculos; c) Diagrama motor para se manter no trajecto; d) Combinação dos três diagramas motores

da distância $d(i,j)$ entre a célula e o robô:

$$f_r(i,j) = \frac{K_r.C(i,j)}{d^2(i,j)} \tag{3.10}$$

em que K_r é uma constante de ganho repulsivo. A força total repulsiva calcula-se a partir da soma das forças repulsivas associadas a cada uma das células:

$$f_r = \sum_{i,j} f_r(i,j) \tag{3.11}$$

A força atractiva f_a que puxa o robô em direcção ao alvo assume essa direcção mas a sua intensidade é constante, K_a , isto é, não depende da distância ao alvo:

$$f_a = K_a \tag{3.12}$$

Este método, apesar de trazer algumas melhorias ao método dos campos potenciais, continua sujeito às limitações inerentes dos campos potenciais (problema dos mínimos locais e passagem em espaços estreitos). Por outro lado, surge outra limitação associada à natureza discreta da grelha, que leva a que pequenos movimentos do robô possam resultar em mudanças muito acentuadas no valor das forças repulsivas e consequentemente a movimentos bruscos do robô.

De forma a ultrapassar as limitações deste método, Borenstein e Koren propuseram um novo método designado VFH (Histograma de Campos Vectoriais).

3.2.5 VFH - Histograma de Campos Vectoriais

À semelhança do método VFF, o método VFH [BK91] foi proposto como um método de desvio de obstáculos de execução em tempo real. Apesar deste método ter sido criado a partir do método VFF anteriormente proposto, não segue o princípio dos campos potenciais para guiar o robô. O controlo do robô realiza-se agora seleccionando a direcção mais adequada para atingir uma determinada meta. Este método designa-se por esse motivo de *selecção de orientação*, sendo precisamente este o mecanismo de arbitração para comutação entre o comportamento de desvio de obstáculos e de procura de meta.

A primeira fase deste método consiste na actualização da grelha histogrâmica bidimensional da mesma forma que era feito no método VFF. Na segunda fase, a grelha histogrâmica é convertida numa grelha polar unidimensional de 360º (ver figura 3.12). Esta

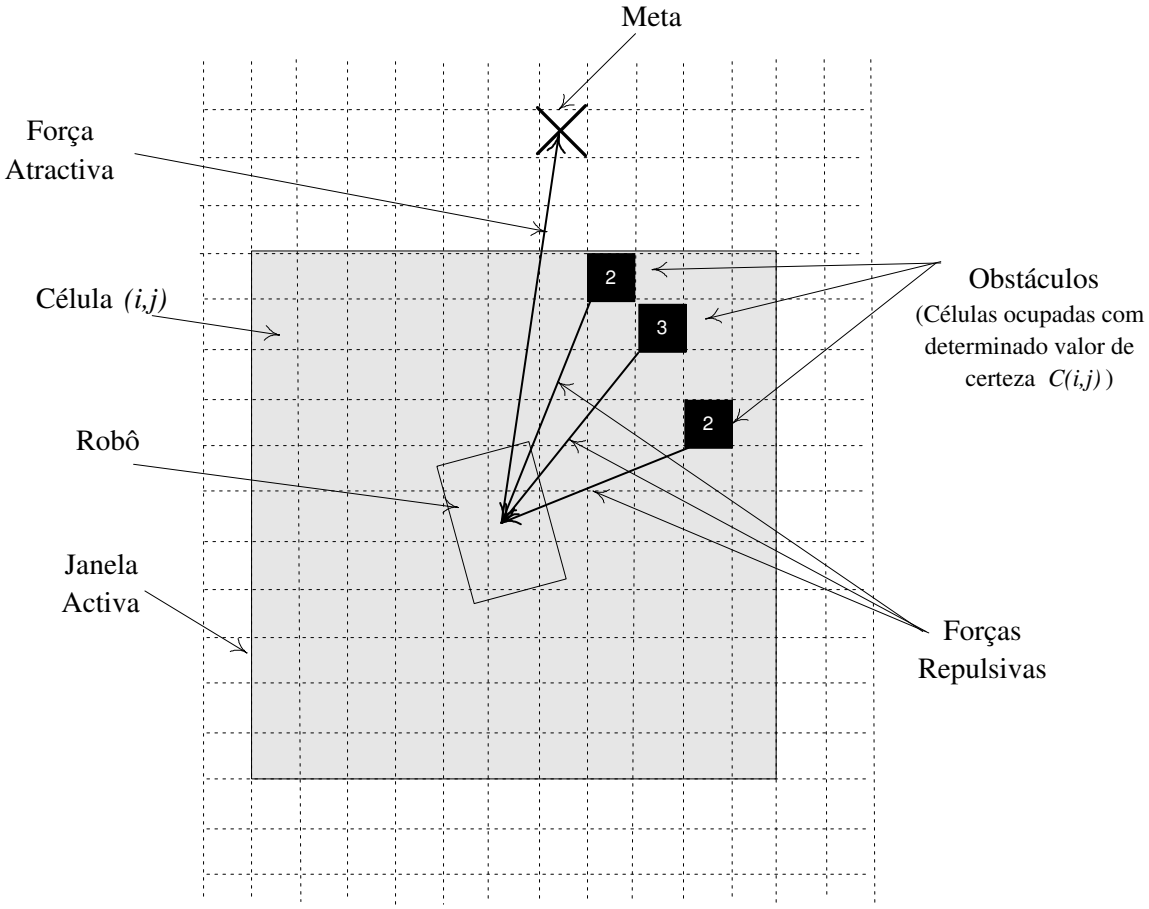


Figura 3.11: Campo de forças virtuais (VFF)

conversão consegue-se através da divisão em sectores angulares da grelha histogrâmica. A cada sector da grelha, fica associado um ângulo e uma intensidade que dá a densidade de obstáculos nessa direcção. Essa intensidade, para um dado sector k, é dada por:

$$h_k = \sum_{i,j} m_{i,j} \tag{3.13}$$

em que $m_{i,j}$ representa a intensidade dos vectores associados a cada célula i, j. $m_{i,j}$ é dado por:

$$m_{i,j} = C(i,j)^2(a - b \times d_{i,j}) \tag{3.14}$$

em que $C(i,j)$ e $d_{i,j}$ têm o mesmo significado que no método VFF, e a e b são constantes positivas.

O histograma polar apresenta a densidade de obstáculos de acordo com um referencial polar. A direcção que o robô deve seguir passa pela escolha das regiões com menor densidade de probabilidade obstáculos (vales) e que melhor se aproximam da direcção da meta. Existe um valor limiar ajustável que regula a largura dos vales: subindo demasiado o limiar, a largura dos vales aumenta levando a que o veículo perca a percepção da presença de obstáculos; diminuindo demasiado o limiar, os vales estreitam fazendo com que o veículo não passe em passagens estreitas.

O método VFH permite eliminar as oscilações bruscas no movimento do robô, melhora a capacidade de atravessar passagens estreitas (dependente do valor limiar) e elimina o problema dos mínimos locais.

3.2.6 Arquitectura DAMN

A arquitectura DAMN [Ros95] consiste num conjunto de módulos (comportamentos) que partilham o controlo do robô. Esta arquitectura possui um mecanismo de coordenação de comportamentos único no domínio da robótica: cada comportamento, tais como *seguimento de estrada*[4] ou *desvio de obstáculos* envia ao módulo de arbitração *votos a favor ou contra* uma determinada acção de controlo. O arbitrador[5] faz a fusão dos comandos e envia o sinal de comando ao controlador do veículo. A cada comportamento está associado um determinado peso que reflecte a sua prioridade relativa para controlar o veículo,

[4] Esta arquitectura foi aplicada num veículo automóvel

[5] São realmente 2 arbitradores: um arbitrador de direcção e um arbitrador de velocidade

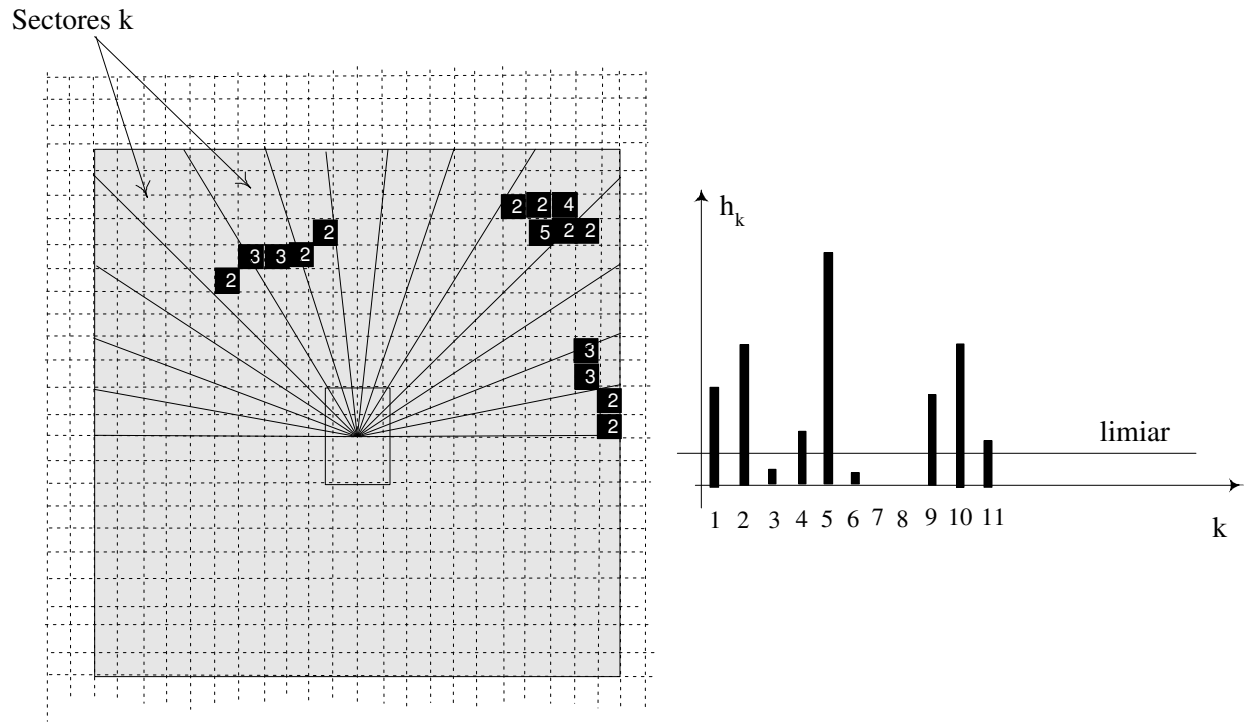


Figura 3.12: Construção do diagrama polar uni-dimensional a partir da grelha histogrâmica bi-dimensional (método VFH)

podendo os pesos de cada comportamento ser alterados dinamicamente. O autor não explica, no entanto, quem altera os pesos nem como estes são alterados.

Existem comportamentos que utilizam toda a informação disponível, incluindo informação adquirida previa e posteriormente. Enquanto os comportamentos de baixo nível (e.g. desvio de obstáculos) funcionam a elevadas taxas para garantir reactividade, os comportamentos de mais alto nível podem processar mapas ou informação simbólica, a taxas inferiores, e periodicamente enviar votos para o arbitrador. O comportamento de *procura de meta* é um exemplo de um comportamento de alto nível. Tem por objectivo direccionar o robô para um conjunto de pontos objectivo definidos pelas coordenadas globais introduzidas pelo utilizador, ou por um planeador global. Técnicas baseadas em algoritmos de procura como o A* são aplicadas a grelhas de certeza para planeamento de trajectórias.

Vejamos o exemplo ilustrado na figura 3.13 de combinação de dois comportamentos: desvio de obstáculos e procura de meta.

1. Cada comportamento vota em cada uma das acções de controlo do arbitrador de direcção, aqui definidas por 5 opções: esquerda acentuada, esquerda suave, em frente, direita suave e direita acentuada.

2. A amplitude de cada voto varia entre -1 e +1 (positivo a favor, e negativo contra).

3. Sendo mais importante o desvio de obstáculos do que tomar o caminho mais curto para a meta, o comportamento de desvio de obstáculo tem um maior peso relativamente ao comportamento de procura de meta.

4. O arbitrador calcula a soma pesada dos votos recebidos dos comportamentos, sendo escolhida a acção de controlo com mais votos (figura 3.14).

3.2.7 Lógica Difusa

A lógica difusa[6] tem sido uma das abordagens intensamente utilizadas para controlo em robótica móvel, essencialmente na implementação de comportamentos reactivos. Seguimento de características ambientais tais como paredes, contornos de estradas, linhas

[6] O apêndice A apresenta o formalismo matemático necessário para a construção de um controlador difuso

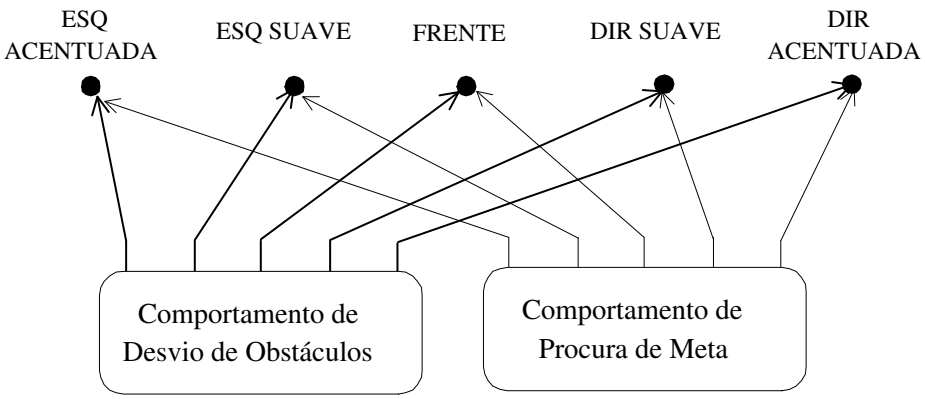


Figura 3.13: Cada comportamento vota a favor ou contra uma acção de controlo. O comportamento de desvio de obstáculos tem maior prioridade do que o comportamento de procura de meta, pelo que os seus votos têm um maior peso

brancas no chão ou seguimento de trajectórias, e desvio local de obstáculos representam a maioria dos comportamentos implementados através de lógica difusa. As técnicas de lógica difusa utilizam-se também na construção e actualização de mapas de grelha. A sua vasta aplicação em robótica móvel resulta da sua relativa facilidade de implementação, assim como da sua capacidade de adaptação às exigências da robótica móvel. Os controladores difusos apresentam algumas propriedades vantajosas à robótica móvel, nomeadamente:

- Incorporam conhecimento heurístico expresso na forma de regras `SE - ENTÃO` e variáveis linguísticas, facilmente escritas/deduzidas para implementação de comportamentos simples. Esta forma de conhecimento é particularmente importante quando um modelo matemático linear e preciso do robô não pode ser facilmente obtido.

- Transferem-se mais facilmente de uma plataforma para outra sem sofrer grandes alterações.

- Considera-se uma metodologia adequada e com bom desempenho, face a dados com incerteza associada [Saf97b]. Isto deve-se principalmente ao mecanismo interpolador do controlo difuso, em que entradas semelhantes dão origem a acções semelhantes. Esta característica resulta em movimentos suaves do robô, face a erros e flutuações nos dados sensoriais.

- Exigem normalmente pouco peso computacional.

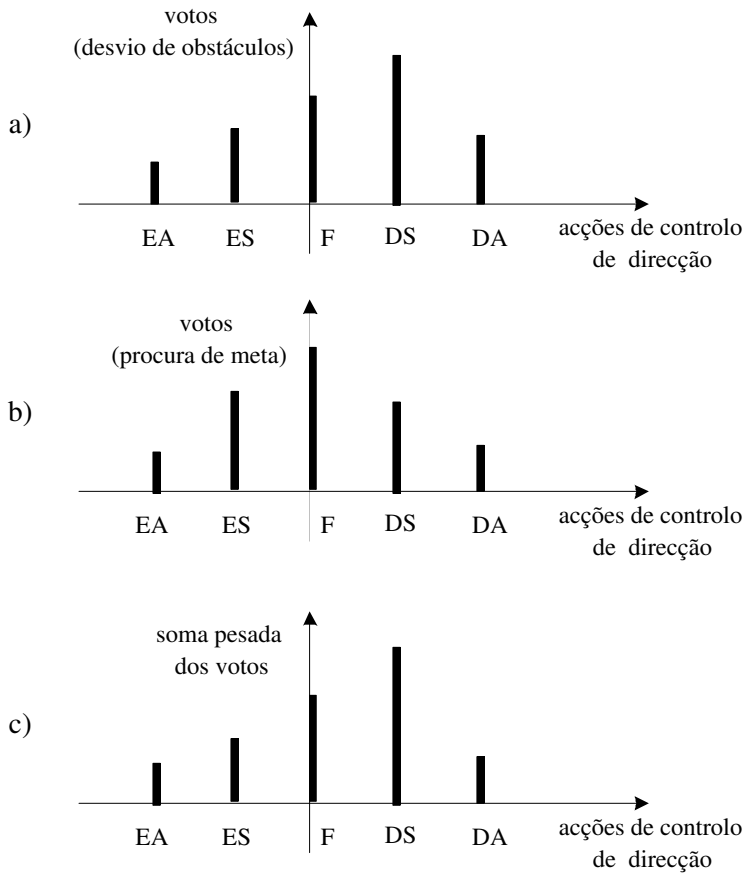


Figura 3.14: a) Votos do comportamento de desvio de obstáculos (peso 0.75), a acção de contolo é DIREITA SUAVE; b) Votos do comportamento de procura de meta (peso 0.25), a acção de controlo é FRENTE; c) Soma pesada dos votos dos dois comportamentos, a acção de controlo seria sensivelmente DIREITA SUAVE

Estas propriedades adequam-se bem às necessidades da robótica móvel, onde modelos matemáticos do ambiente não se encontram normalmente disponíveis, os dados sensoriais são quase sempre incertos e imprecisos, e o funcionamento em tempo real é essencial.

Contudo, o facto de não existir um modelo matemático é também uma desvantagem, na medida em que não se pode garantir[7] que o controlador difuso tenha o comportamento desejado. A concepção do controlador (base de regras e funções de pertença) depende do conhecimento de utilizadores (peritos) que sabem como o sistema a controlar funciona. A eliminação de erros e a afinação passam por ensaios experimentais. Existem, no entanto, outras técnicas para obter dados para a construção do controlador, por exemplo, baseados na extracção de informação a partir da observação das acções de operadores humanos, ou ainda, na utilização de técnicas de aprendizagem.

Comportamentos

Os controladores baseados em lógica difusa utilizam-se na implementação de comportamentos simples ou comportamentos mais complexos que visam alcançar vários objectivos simultaneamente. Neste último caso, a implementação dos comportamentos pode ser realizada de duas formas de acordo com a complexidade das regras:

1. Escrever regras complexas em que os antecedentes consideram vários objectivos simultaneamente (figura 3.15). As regras tomam a seguinte forma:
   ```
   SE condição_trajectória i E condição_obstáculo j ENTÃO comando l
   SE condição_trajectória m E condição_obstáculo n ENTÃO comando p
   ⋮
   ```

2. Escrever regras simples, cada uma especificando um único objectivo, e combinar a sua saída através de operadores de lógica difusa (figura 3.16). As regras tomam a seguinte forma:
   ```
   SE condicao_trajectoria i ENTÃO comando j
   SE condicao_trajectoria l ENTÃO comando m
   SE condicao_obstaculo n ENTÃO comando p
   SE condicao_obstaculo q ENTÃO comando r
   ⋮
   ```

[7] Apenas garantias baseadas em testes empíricos

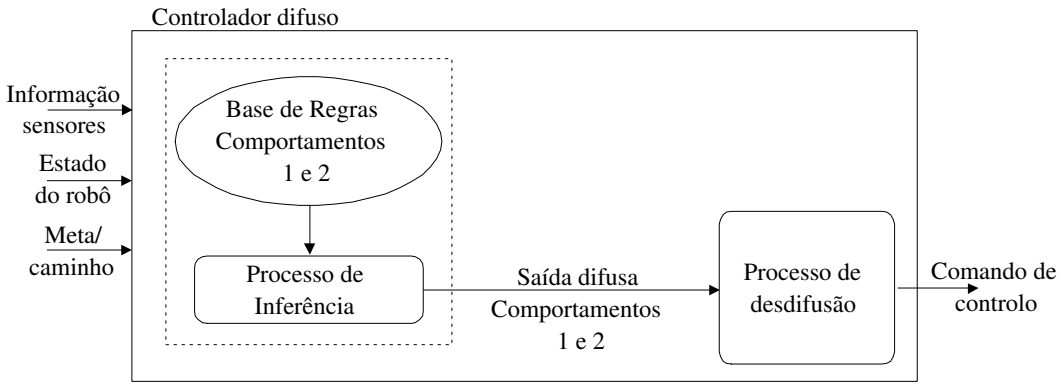


Figura 3.15: As regras são escritas considerando mais do que um objectivo simultaneamente (vários comportamentos)

A primeira abordagem adequa-se às situações em as interacções entre objectivos são muito importantes, pois assegura-se que estes vão ser combinados de forma coerente obtendo a resposta desejada. No entanto, se o número de variáveis de entrada aumentar, o número de regras aumenta significativamente. Quanto à segunda abordagem, as regras são mais fáceis de escrever e não possui a limitação da primeira. No entanto, surge o problema de combinar a saída de cada um dos comportamentos, de forma a obter uma resposta coerente. Esta abordagem é por isso mais sensível e dependente das técnicas de desdifusão.

A partir do exemplo da figura 3.17, comparam-se algumas formas de implementar um controlador difuso para navegação reactiva. A posição da meta sugere que o robô rode à esquerda, no entanto, a presença de um obstáculo obstruindo o caminho para a meta, faz com que o robô siga em frente ou ligeiramente à direita.

Yen e Pfluger [YP95] utilizam um controlador difuso baseado na segunda abordagem. O método aplica uma nova técnica de desdifusão designada por CDMA (centróide da maior área) para enfrentar situações em que as saídas dos comportamentos são contraditórias. As acções de controlo do robô resultam da fusão das saídas de dois comportamentos: seguimento de meta e desvio de obstáculos.

A - Comportamento de seguimento de meta

A meta encontra-se sensivelmente a $\theta = -22.5^{o}$ em relação à direcção de deslocamento do robô. Este valor de θ activa as 2 regras (R1 e R2) que se observam na figura 3.18a).

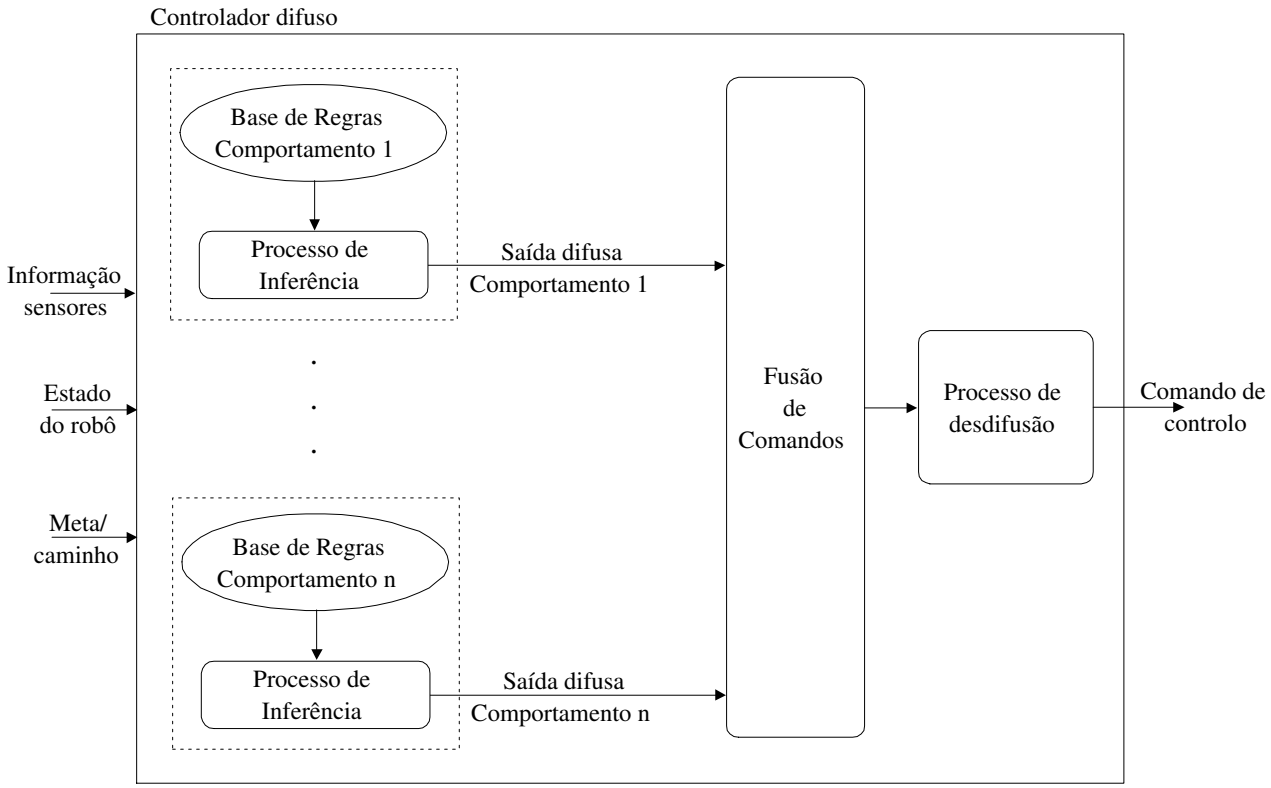


Figura 3.16: As regras são escritas considerando apenas um objectivo de cada vez (apenas um comportamento)

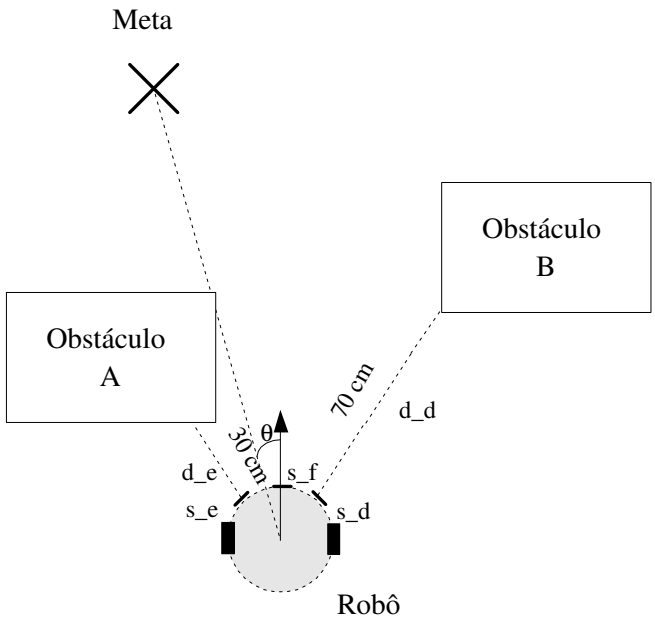


Figura 3.17: Exemplo de um cenário típico. O controlador utiliza apenas informação de 3 sensores para o comportamento de desvio de obstáculos

O processo de inferência utilizado para combinar as regras é o soma-produto, isto é, o consequente (B_i) de cada regra é multiplicado pelo grau de verdade do antecedente (A_i):

$$\mu_{Bi}(z) = \mu_{Ai}(x).\mu_{Bi}(z) \tag{3.15}$$

A combinação das regras resulta da soma do consequente de cada uma das regras (B_i):

$$\mu_B(z) = \mu_{B1}(z) + \mu_{B2}(z) \tag{3.16}$$

A interpretação gráfica do processo de inferência pode ver-se na figura 3.18b). A escolha deste método de inferência e não de outro (por exemplo *max-min*) justifica-se por permitir que o comportamento faça uma interpolação linear entre as regras cujas funções de pertença antecedentes são adjacentes.

B - Comportamento de desvio de obstáculos

O comportamento de desvio de obstáculos baseia-se na informação de 3 sensores dianteiros (s_e, s_f e s_d). Existem duas regras (R3 e R4) activas (figura 3.19a)). Os conjuntos difusos dos consequentes das regras representam direcções a evitar. O mecanismo de inferência

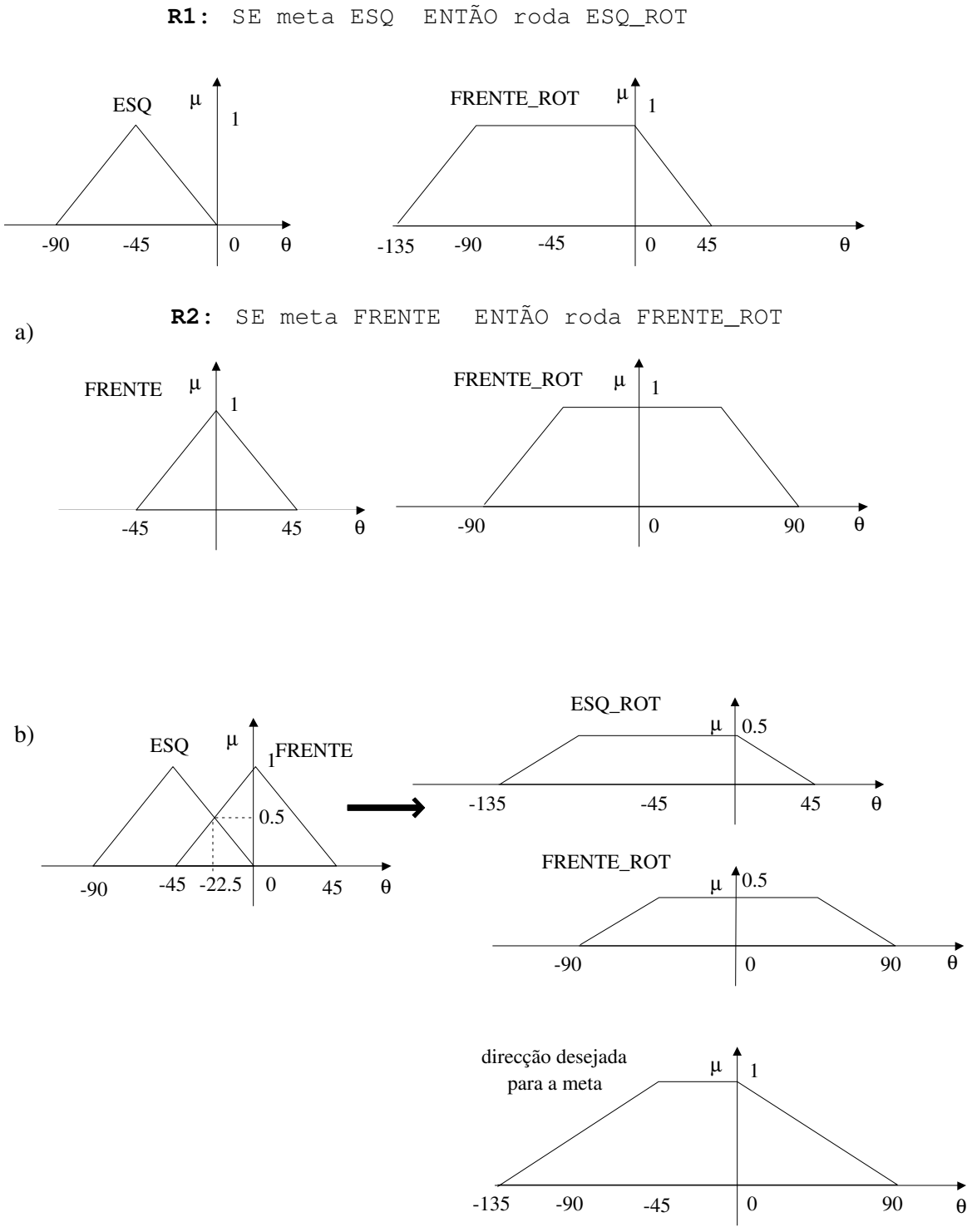


Figura 3.18: Comportamento de seguimento de meta: a) Regras utilizadas; b) Método de inferência utilizado para encontrar a direcção para a meta (soma-produto)

é o *max-min* (figura 3.19a)). A escolha justifica-se seguindo a intuição de que o grau da direcção a evitar deve ser determinado essencialmente pelo sensor que tem a opinião mais forte acerca dessa direcção.

C - Fusão de comandos

O comando de direcção final obtém-se através da combinação dos dois comportamentos, representados respectivamente pela direcção desejada e pela direcção a evitar. Uma vez que a acção final de controlo deverá satisfazer ambos os comportamentos, o módulo de fusão de comandos utiliza o operador *min*, ou seja, a conjunção das saídas dos dois comportamentos:

$$\mu_{direcao\_rotacao}(x) =$$
$$\mu_{direcao\_desejada} E NAO direcao\_evitar =$$
$$min\{\mu_{direcao\_desejada}, \mu_{NAO direcao\_evitar}\} =$$
$$min\{\mu_{direcao\_desejada}, 1 - \mu_{direcao\_evitar}\}$$

Esta operação está representada graficamente na figura 3.20. A acção de controlo final consegue-se a partir do processo de desdifusão do conjunto resultante. É neste processo que surge a novidade deste controlador. Os dois métodos de desdifusão mais utilizados são normalmente o CDG (centro de gravidade) e o MDM (média dos máximos). Este último calcula a média dos máximos, ou seja, não usa toda a informação contida no conjunto difuso, o que pode levar a comandos menos suaves e pode dar lugar a resultados contraditórios com a intuição. Este processo de desdifusão é comparável ao mecanismo VST. Aplicando o desdifusor CDG a conjuntos difusos caracterizados por dois ou mais picos (no exemplo ilustrado temos dois picos), o comando resultante da desdifusão pode ser oposto à acção que deveria ser tomada. Este problema surge normalmente quando os comportamentos têm acções contraditórias, sendo típico o cenário em que o robô encontra uma parede em frente na direcção da meta. Yen e Pfluger [YP95] desenvolveram um novo método de desdifusão para compensar este problema ao qual chamaram CDMA (centróide da maior área). Este método consiste em dividir o conjunto difuso em vários sub-conjuntos disjuntos, cada um deles correspondendo a um possível comando difuso. O sub-conjunto com maior área é seleccionado e desdifusado utilizando o método CDG.

Aplicando o método de desdifusão CDG ao exemplo ilustrado, o comando de direcção

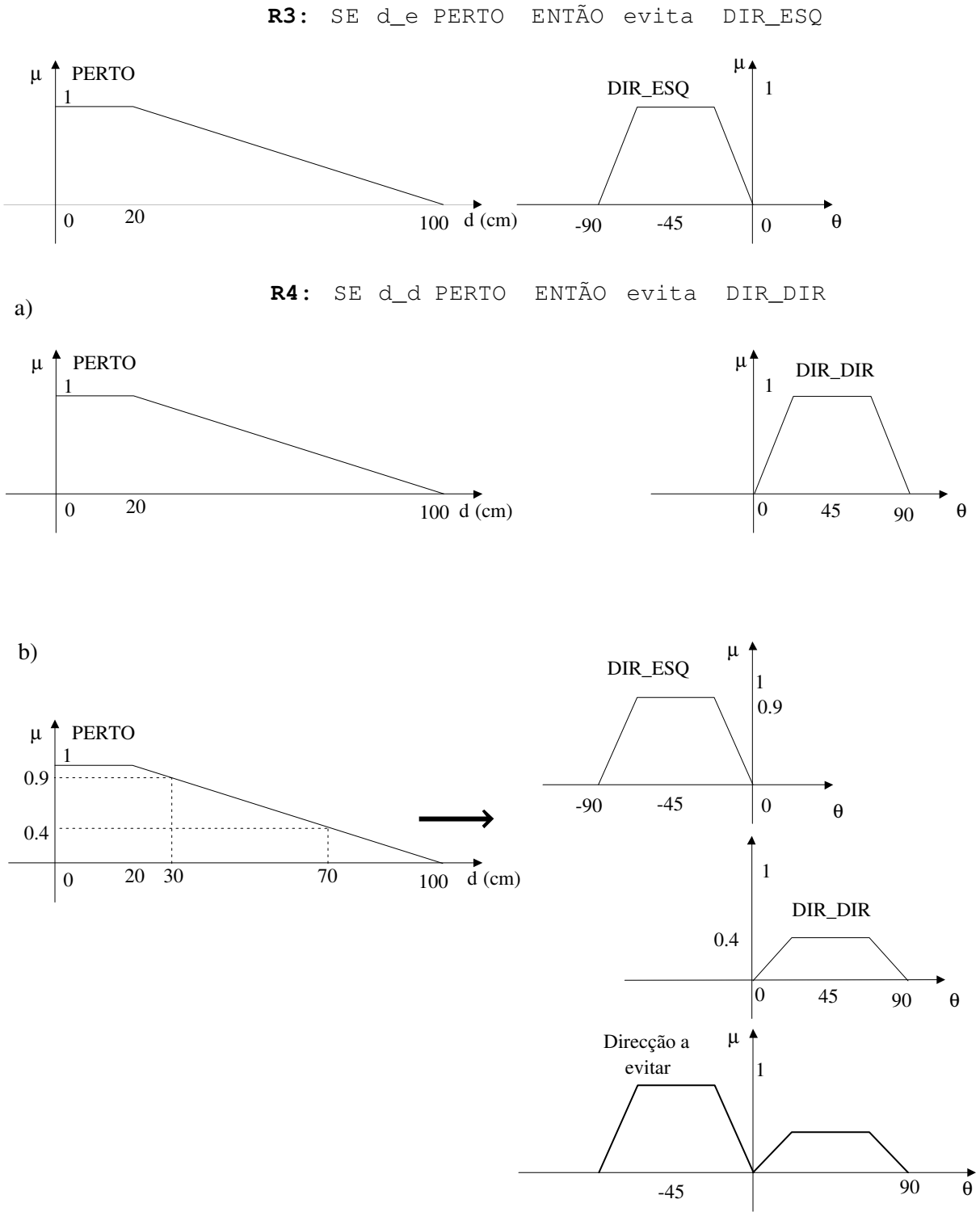


Figura 3.19: Comportamento de desvio de obstáculos. a) Regras utilizadas; b) Método de inferência utilizado para determinar a direcção a evitar (método *max-min*)

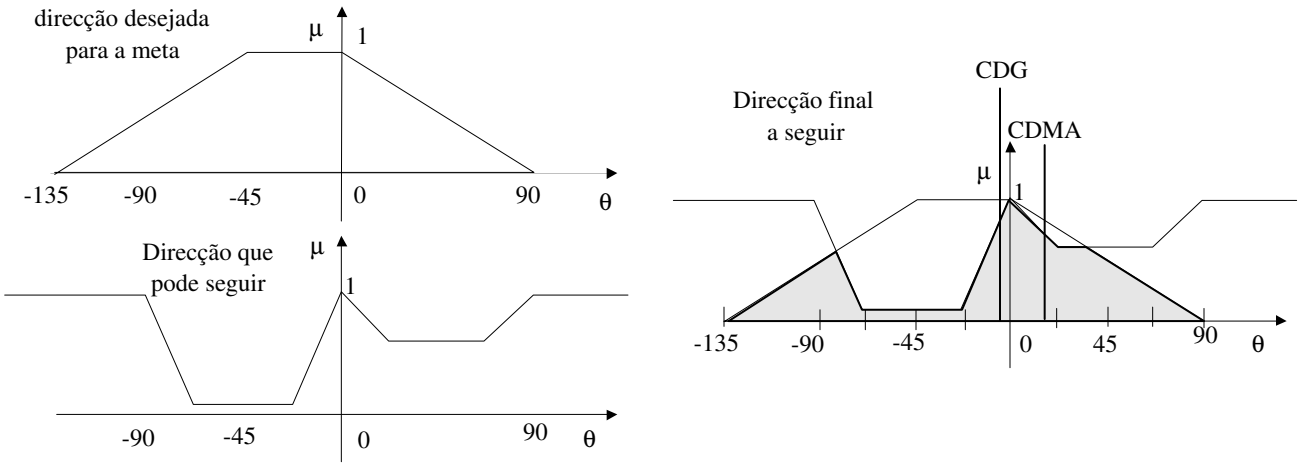


Figura 3.20: Fusão da saída dos dois comportamentos (utilização do operador *min*)

obtido é de -5^o (utilizando valores discretos de $\theta = 10^o$). Esta resposta é pouco adequada pois existe um obstáculo à esquerda bastante perto. Aplicando o método CDMA, obtém-se uma orientação de saída mais ajustada de 19^o.

Funções de preferência

Em [Saf97a] [SRK93] [RSK95] [KMRS97], a saída de cada comportamento é vista como uma função que indica a preferência do comando a aplicar. O grau de preferência representa-se por um conjunto difuso no espaço de comandos. O controlador difuso proposto é um controlador hierárquico designado por CDC (combinação dependente do contexto) representado na figura 3.21. A combinação passa por várias etapas: 1) cada comportamento gera uma preferência de acção de acordo com o seu objectivo; 2) cada comportamento tem um contexto de activação que indica em que situação o comportamento deve ser utilizado e qual o seu peso de activação; 3) as preferências de todos os comportamentos, pesadas pelo grau de verdade do respectivo contexto de activação, são fundidas formando uma preferência global; e 4) o comando de acção é escolhido através do processo de desdifusão.

Cada comportamento é definido por um conjunto de regras difusas do tipo `SE A ENTÃO c`, em que A é uma expressão difusa composta por predicados e conectivas ('E', 'OU' e 'NÃO'), e c é um vector de valores das variáveis de controlo. A saída de cada comportamento resulta da união (operador *max*) da saída de cada uma das regras, e designa-se

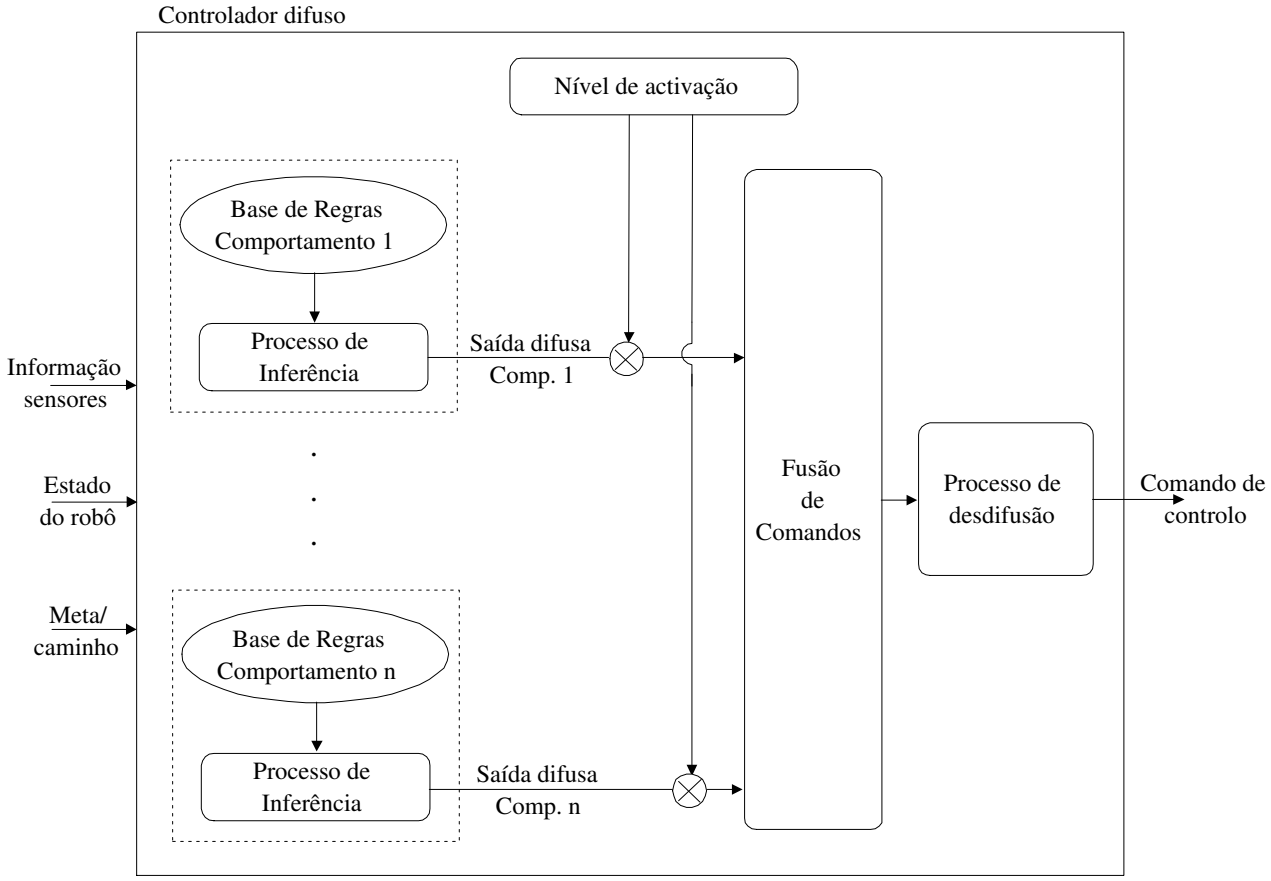


Figura 3.21: CDC: Controlador baseado em níveis de activação de comportamentos

por *função de preferência* do comportamento B ($Des_B(s,c)$):

$$Des_B(s,c) = (A_1(s) \wedge C_1(c)) \vee ... \vee (A_n(s) \wedge C_n(c)) \tag{3.17}$$

sendo s um estado do espaço e c um vector de controlo. Os operadores $\wedge$ e $\vee$ denotam a operação *min* e *max* respectivamente.

Quando vários comportamentos são simultaneamente activos, as funções de preferência de cada comportamento combinam-se através do operador *min*. O conjunto difuso resultante fornece a preferência de controlo de ambos os comportamentos à qual se chama preferência global. O conjunto é por fim desdifusado através do método CDG. Se, no entanto, existirem comportamentos cujas saídas são conflituosas (objectivos incompatíveis) a combinação dos comportamentos através do operador *min* não pode ser utilizada. Uma forma de ultrapassar esta situação consiste em definir um contexto de aplicabilidade, Cxt, para cada comportamento, sendo a função de preferência considerada apenas quando apropriada. Cxt fornece um peso (ou grau de verdade) que define o grau de activação de determinado comportamento [SRK93]. Dado um conjunto de comportamentos $B_1, ..., B_k$, a função de preferência global que define a combinação entre os vários comportamentos é dada por:

$$Des(s,c) = (Cxt_1(s) \wedge Des_1(s,c)) \vee ... \vee (Cxt_k(s) \wedge Des_k(s,c)) \tag{3.18}$$

em que Des_i é a função de preferência do comportamento B_i, e Cxt_i o seu contexto. Na prática, o mecanismo de combinação dos comportamentos implementa-se através de regras designadas por meta-regras ou regras de contexto do tipo:

`SE A' ENTÃO activa comportamento B`

A aplicação deste contexto de activação consiste em transformar cada uma das regras do tipo `SE A ENTÃO c` em regras do tipo `SE A' E A ENTÃO c`. Vejamos o exemplo da figura 3.22a) que ilustra o mecanismo CDC. A meta encontra-se em frente e os sensores esquerdo e dianteiro detectam a presença de obstáculos. É activada a regra R1 do comportamento de seguimento de meta (como $\theta = 0^o$ a saída de controlo é activada com grau de pertença 1) (figura 3.23). No comportamento de desvio de obstáculos, a regra R2 encontra-se activa (a saída de controlo é activada com um grau de pertença de aproximadamente 0.8). Combinando estes dois comportamentos obtemos o saída difusa representada na figura 3.23. Utilizando o método de desdifusão CDG obtemos um valor de $\approx 21^o$. Vejamos agora o mesmo exemplo, mas aplicando regras de contexto. Como o robô se encontra

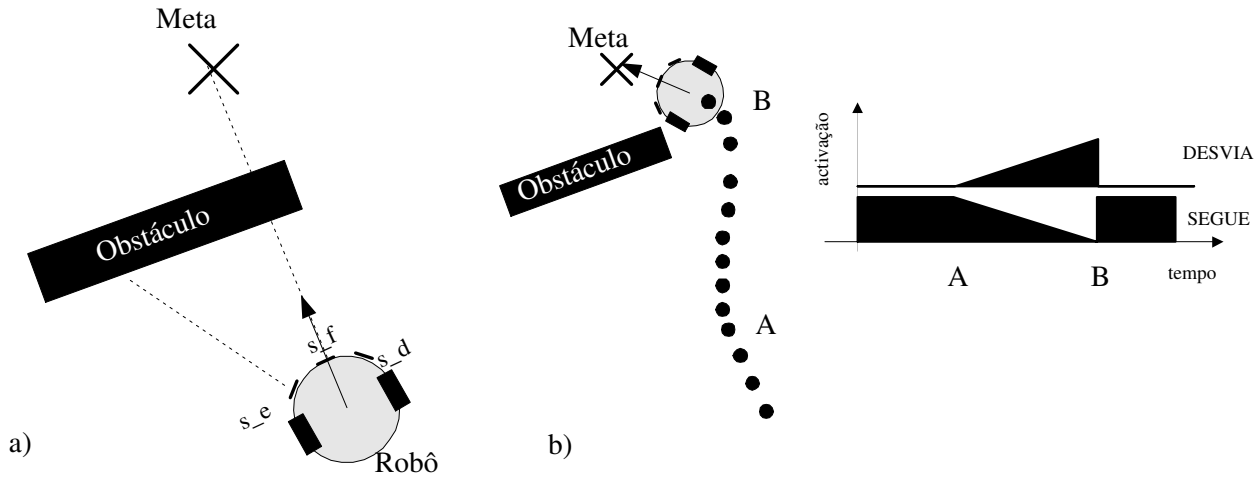


Figura 3.22: a) Exemplo de um cenário; b) Contextos de activação do comportamento de desvio de obstáculos e do comportamento seguimento de meta de acordo com a posição do robô

perto do obstáculo, o comportamento de desvio é mais importante que o comportamento de seguimento, pelo que deverá ter um peso superior. As regras de contexto (figura 3.23b)) dão um peso 0.8 ao comportamento de desvio e 0.2 ao comportamento de seguimento. Desta forma, o comportamento de desvio domina o comportamento global do robô, sendo a saída de controlo de $\approx 32^o$. A figura 3.22b) ilustra o efeito de activação para estes dois comportamentos. Quando o robô se encontra distante do objecto, o seguimento de meta domina, mas à medida que se aproxima do obstáculo o comportamento de desvio é dominante até ultrapassar o obstáculo.

O contexto de activação de cada comportamento equivale a atribuir prioridades dinâmicas aos comportamentos.

Combinação de comportamentos na base de regras

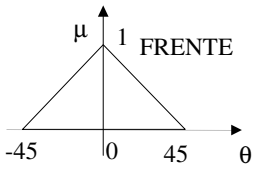
Uma outra forma de ultrapassar a combinação de comportamentos cujas saídas são conflituosas consiste em combinar os comportamentos aquando da criação das regras, ou seja, construir a base de regras atendendo a vários objectivos. Neste caso, utilizar-se-á um controlador do tipo da figura 3.15. Vejamos a aplicação desta abordagem para o exemplo da figura 3.22a). Uma possível regra activada seria:

```
SE d_e PERTO E d_f PERTO E meta FRENTE ENTÃO roda DIREITA
```

a)

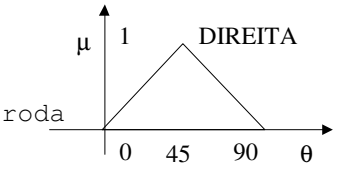
Comportamento de seguimento de meta

R1: SE meta FRENTE ENTÃO roda

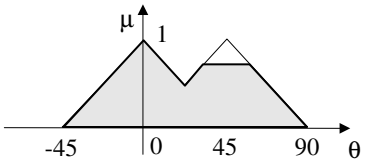


Comportamento de desvio de obstáculos

R2: SE d_e PERTO E d_f PERTO ENTÃO roda



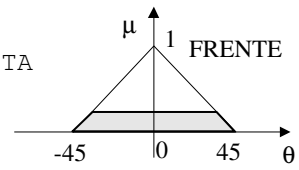
Combinação dos dois comportamentos



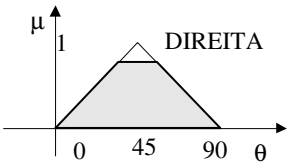
b)

Regras de Contexto

R3: SE NÃO (obstáculo PERTO) ENTÃO activa SEGUE_META



R4: SE obstáculo PERTO ENTÃO activa DESVIA



Combinação dos dois comportamentos utilizando o contexto de activação

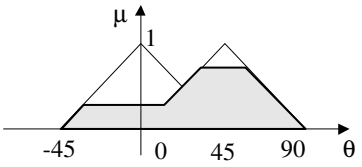


Figura 3.23: a) Regras para cada um dos comportamentos; b) Aplicação de contextos de activação

Esta abordagem oferece uma boa solução para enfrentar os problemas de interacção entre objectivos, pois a acção de controlo recai obrigatoriamente na acção desejada. Elimina-se os problemas relativos à desdifusão de conjuntos multi-modais resultantes da combinação de comportamentos conflituosos.

3.3 Conclusões

Este capítulo apresentou várias formas de combinar e implementar comportamentos de navegação reactiva. Após uma análise detalhada dos principais métodos, conclui-se que os controladores de lógica difusa apresentam em relação aos outros métodos, uma grande versatilidade.

Relembremos algumas das principais características destes métodos. Na arquitectura de submissão os comportamentos possuem prioridades fixas e apenas pode existir um comportamento activo de cada vez. Os métodos de campos potenciais são limitados pelos mínimos locais. O método VFH surge como uma boa alternativa, no entanto, requer a construção de um mapa da zona envolvente ao robô (o esforço computacional é também superior aos outros métodos). Finalmente, o controlo por lógica difusa pode ser modelado como qualquer um dos outros métodos (excepto os métodos VFF e VFH). A sua versatilidade permite construir a base de regras de forma a implementar, por exemplo, o método dos campos potenciais (eliminado o problema dos mínimos locais). Também facilita a implementação dos mecanismos de coordenação de comportamentos por prioridades ou fusão, para além de poder ser implementado em qualquer sistema, independentemente de existirem ou não modelos matemáticos. Finalmente, não exige elevado esforço computacional. A sua principal desvantagem reside na dificuldade de afinação do controlador.

Capítulo 4

Cadeiras de Rodas Inteligentes

Este capítulo descreve alguns exemplos que ilustram as tendências actuais no desenvolvimento de protótipos de cadeiras de rodas inteligentes. Mostra também como melhorias tecnológicas em cadeiras convencionais podem facilitar e aperfeiçoar a sua utilização por pessoas com deficiências motoras acentuadas, limitações cognitivas e idosos.

4.1 Cadeiras de Rodas

A cadeira de rodas é um sistema de reabilitação que se enquadra no domínio da robótica móvel e controlo. Possui, contudo, particularidades que a tornam substancialmente diferente da maioria das aplicações de robótica móvel, por se tratar de um sistema que integra o ser humano.

Antes de mais, importa definir a quem se destina este tipo de sistema e quais as limitações de uma cadeira de rodas convencional. Estas duas questões estão obviamente interligadas. As cadeiras de rodas, de uma forma geral, destinam-se a pessoas com deficiências motoras nos membros inferiores, isto é, pessoas com limitações ou mesmo sem capacidades de locomoção. Podem-se referir os portadores de: 1) paralisias múltiplas (hemiplégicos, paraplégicos, tetraplégicos); 2) tremura acentuada; 3) degeneração generalizada dos músculos por doença ou por processo de envelhecimento (idosos); 4) malformações congénitas; e 5) amputações; [Car95]. Os utilizadores cuja limitação física se circunscreve aos membros inferiores (e.g. paraplégicos), mas com plena capacidade nos membros superiores, estão aptos para controlar uma cadeira de rodas através de um dispositivo normal

tal como um *joystick*. As pessoas que, para além de deficiências motoras nos membros inferiores, possuem também limitações motoras nos membros superiores encontram extrema dificuldade, quando não mesmo a impossibilidade, em utilizar dispositivos normais para comandar uma cadeira. Por exemplo, um tetraplégico (paralisia de tronco e membros) apenas pode utilizar a cabeça para comandar a cadeira, ou seja, não poderá utilizar um *joystick* convencional. Muitos destes utilizadores são forçados a controlar a cadeira com o queixo ou língua através de manípulos especiais, o que constitui uma tarefa muito difícil e morosa para o utilizador. O aparecimento de novas interfaces homem-máquina vem melhorar a acessibilidade das cadeiras de rodas. Por exemplo, o controlo por comandos de voz vem garantir o acesso da cadeira aos tetraplégicos. No entanto, a interface por comandos de voz, não resolve por si só o problema do controlo da cadeira. Em ambientes congestionados, o utilizador precisaria de efectuar dezenas de comandos por minuto por forma a não colidir com nenhum obstáculo. Por este motivo, é imprescindível que o sistema possua também um módulo de navegação, cuja função seja assistir as manobras do utilizador, permitindo reduzir a um mínimo o número de comandos. Para pessoas com tremura acentuada nos membros superiores, ou seja, que têm dificuldade em utilizar com precisão um *joystick*, é também vantajoso que um módulo de navegação corrija comandos errados do *joystick*, calculando as melhores trajectórias entre os obstáculos. Em ambientes domésticos, normalmente caracterizados por espaços reduzidos, o controlo de uma cadeira exige manobras complexas difíceis de executar mesmo por pessoas sem limitações físicas nos membros superiores.

A tecnologia utilizada em robótica móvel tem sido aproveitada para as cadeiras de rodas motorizadas, conferindo-lhes um grau de funcionalidade semelhante à dos robôs móveis. Contudo, o sistema "cadeira de rodas" é específico, pois é um sistema que interrelaciona a máquina e o ser humano. Requisitos de segurança, de partilha de comando entre o utilizador e cadeira, e estratégias de navegação devem ser avaliados com mais atenção. Por exemplo, a trajectória resultante de um algoritmo de navegação deve ser suave e coerente, de forma a inspirar confiança ao utilizador. Assim, estes aspectos, que por vezes são negligenciados em algumas aplicações da robótica móvel, não podem aqui ser descurados. Por outro lado, o ser humano vem permitir diferentes graus de autonomia ao sistema, isto é, o sistema poderá não ser completamente autónomo, na medida em que o utilizador pode intervir a vários níveis, executando algumas tarefas. A partilha

de comando entre o utilizador e a máquina é um dos assuntos de investigação de maior interesse, mas também de maior complexidade: como conjugar os comandos do utilizador com as acções dos algoritmos de planeamento de trajectória.

Salientam-se ainda outras áreas de investigação que visam melhorias de carácter mecânico para aperfeiçoar a manobrabilidade da cadeira e a sua capacidade em ultrapassar barreiras arquitectónicas.

4.2 Áreas de Investigação e Desenvolvimento

De acordo com o que já foi dito atrás, poderemos dividir as áreas de investigação e desenvolvimento em três grupos: 1) módulos de navegação e segurança; 2) interfaces homem-máquina; e 3) adaptações mecânicas e manipuladores. No âmbito desta tese, apresentam-se alguns trabalhos referentes às duas primeiras áreas.

4.2.1 Navegação e Segurança

[Projecto NavChair] - O projecto NavChair tem sido desenvolvido por uma equipa de investigadores da Universidade de Michigan. Este sistema de navegação baseia-se no método VFH descrito na secção 3.2.5. O algoritmo foi adaptado para ser usado em sistemas de controlo homem-máquina nomeadamente em cadeiras de rodas [BBL+94].

A figura 4.1 ilustra a aplicação do método VFH numa cadeira de rodas. O histograma polar representa a densidade de obstáculos, ou seja, valores elevados (picos) representam a presença de obstáculos, enquanto valores pequenos (vales) indicam possíveis zonas de passagem. A direcção que o utilizador pretende seguir é dada pelo *joystick* (seta a cheio). O VFH escolhe o vale (direcção representada pela seta a tracejado) que mais se aproxima da direcção fornecida pelo *joystick*. O comando que chega aos motores é desta forma um comando modificado. Depois de escolhida a direcção, esta vai sendo alterada pelo efeito de forças repulsivas exercidas pelos obstáculos mapeados na grelha cartesiana. As forças repulsivas são calculadas tendo em consideração os limites da cadeira e não como se esta fosse um ponto. O efeito dos erros de localização é minimizado graças à rápida actualização dos mapas, pois reduz o efeito das leituras anteriores. Segundo os autores [BBL+94], uma das principais vantagens do método VFH resulta da sua capacidade em garantir elevada rapidez de navegação em ambientes congestionados, sem que para isso tenha de reduzir

muito a sua velocidade. Esta característica torna-se, no entanto, desconfortável quando se encontra uma pessoa no veículo, pois os movimentos são muito repentinos. A solução para este problema consistiu em diminuir a velocidade da cadeira proporcionalmente à diferença entre a direcção fornecida pelo *joystick* e a direcção calculada pelo VFH. Se a direcção escolhida pelo VFH for muito diferente da fornecida pelo *joystick*, a cadeira pára indicando que o caminho escolhido pelo utilizador se encontra impedido. O VFH foi testado para navegação em corredores e para passagem de portas. No primeiro caso, os resultados foram aceitáveis apresentando movimentos suaves e coerentes. As experiências para passagem de porta provaram que o sucesso da passagem dependia da largura da porta. Para portas de largura normal, a percentagem de sucesso caiu para 20%.

Para melhorar o desempenho do método VFH, nomeadamente na passagem de portas criou-se um novo método adaptativo o qual foi designado por MVFH (mínimo VFH) [BLK+94]. Este método cria vários níveis de autonomia que tornam o VFH mais ou menos dependente dos desejos do utilizador. O histograma polar é calculado da mesma forma que no VFH, no entanto, a direcção é escolhida de forma diferente. Ao histograma polar soma-se uma função pesada $f(x)$, sendo a direcção escolhida a correspondente ao mínimo do histograma polar resultante (figura 4.2). A função $f(x)$ é uma parábola com o mínimo na direcção escolhida pelo utilizador. A soma desta parábola faz com que a direcção escolhida seja a indicada pelo *joystick*, pois todos os valores do histograma que se afastam da direcção do *joystick* ficam com valores elevados. A forma da parábola cria vários graus de autonomia: se se tratar de uma parábola acentuada comparativamente ao histograma polar, é dada importância à vontade do utilizador, tendo o desvio de obstáculos praticamente nenhum efeito; se a função for plana, o comportamento da cadeira apenas depende do VFH (maior autonomia). Analisando o cenário da figura 4.2, verifica-se que utilizando o método VFH, a cadeira não passaria a porta, pois existe um vale abaixo do nível limiar na direcção **c**. Somando a função $f(x)$, o único vale candidato é agora o que se aproxima da direcção dada pelo utilizador. A direcção escolhida é dada por **c** correspondendo ao centro do novo vale. Alterando em tempo real a forma da função pesada, a autonomia do sistema varia satisfazendo as necessidades do utilizador. A variação do nível de autonomia é utilizada para a comutação entre dois modos de operação do sistema: passagem de porta e desvio de obstáculos. Foi desenvolvido um método para selecção automática entre estes dois modos, designado por *modelação da*

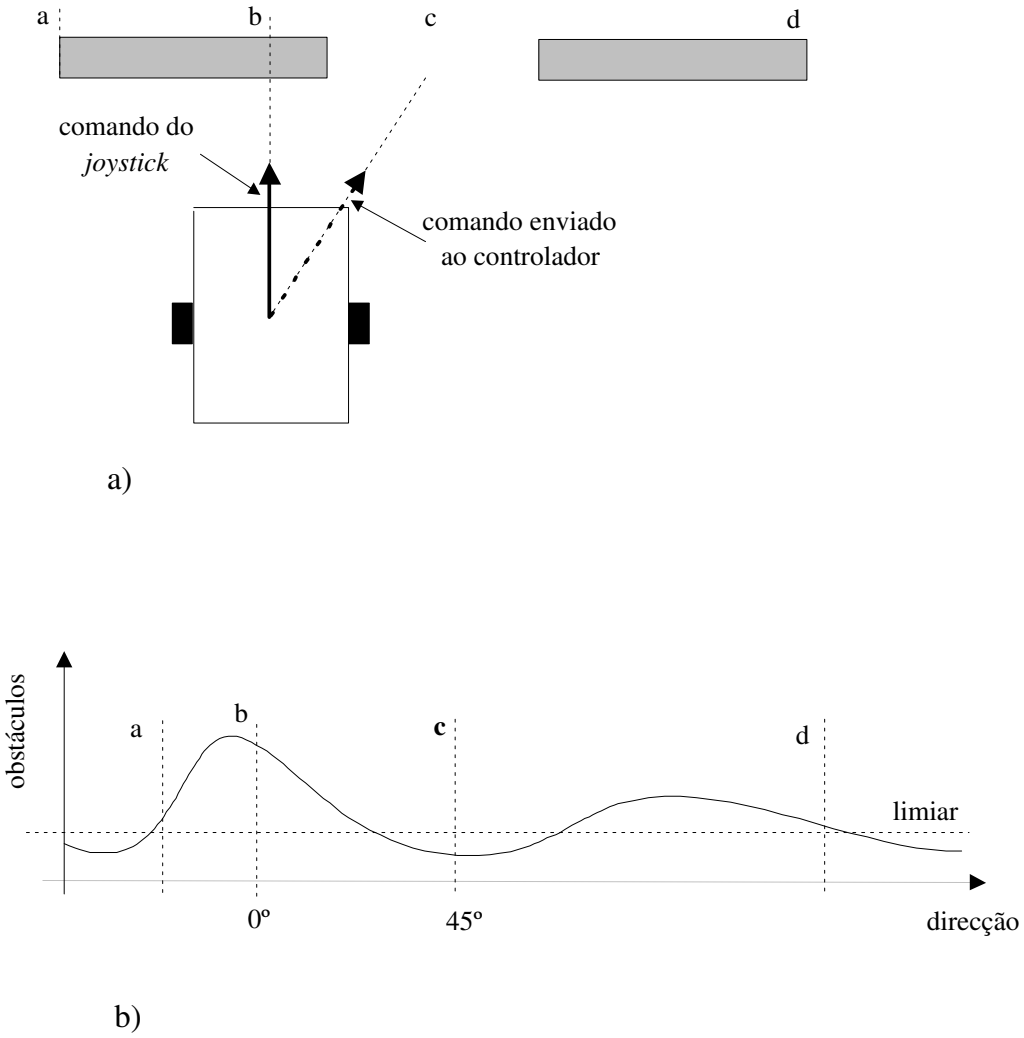


Figura 4.1: Desvio de obstáculos utilizando o método VFH

resposta a estímulos (SRM). Suponhamos novamente o cenário da figura 4.2: o sistema deverá decidir-se entre passar a porta ou desviar-se. A detecção da porta não implica que a opção seja atravessar a porta. A selecção de uma das tarefas depende do comportamento do utilizador. O SRM[1] observa as respostas do utilizador e compara-as com respostas a estímulos conhecidos. O sistema assume à partida que o utilizador pretende continuar no modo de desvio de obstáculos, deslocando-se por isso ligeiramente para a direita. Se o utilizador responder de forma a tentar corrigir esta manobra (de uma forma semelhante

[1] O autor não esclarece quanto à implementação do SRM

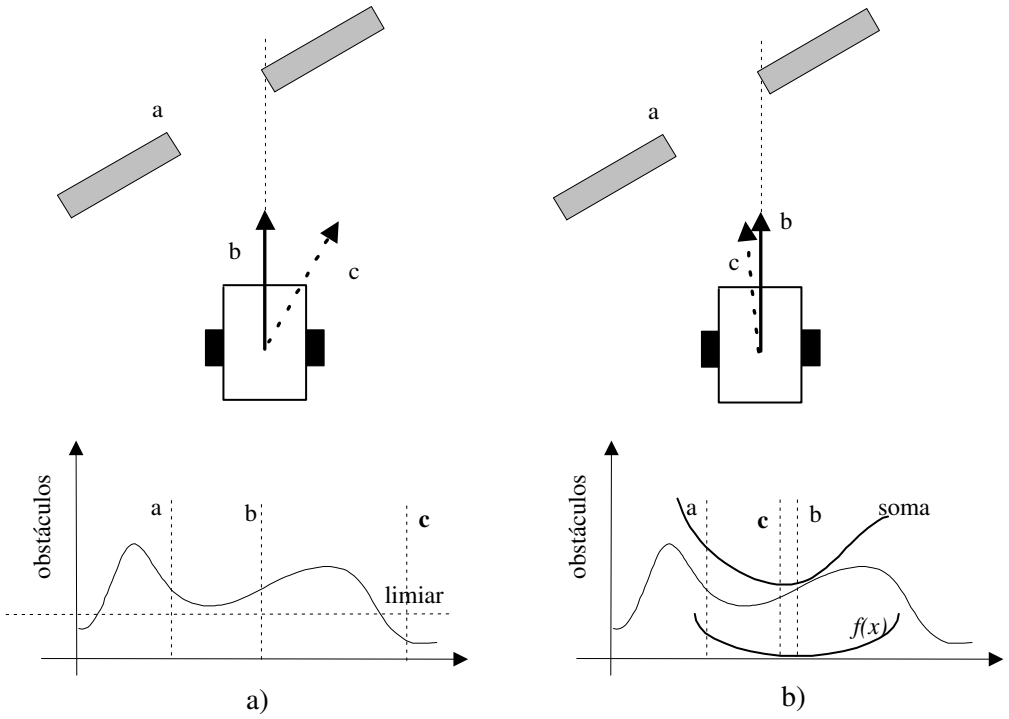


Figura 4.2: Comparação entre o método VFH (a) e MVFH (b)

a um comportamento de passagem de porta já memorizado), então o sistema selecciona o modo de passagem de porta alterando o nível de autonomia do MVFH, ou seja, alterando a forma da parábola.

[Projecto SENARIO] - Do projecto Europeu SENARIO [KTP+97], resultou um sistema de navegação com dois modos de funcionamento a saber, um modo *semi-autónomo* e modo *completamente autónomo*. No primeiro, a responsabilidade das acções de controlo é partilhada entre o utilizador e a cadeira. O sistema aceita comandos típicos de movimento, através de um *joystick* ou voz, e executa manobras de correcção, quando necessárias, para desvio de obstáculos. O segundo inclui todas as capacidades do modo semi-autónomo, acrescidas da capacidade em executar autonomamente tarefas de alto nível (e.g. navegação entre pontos previamente definidos sem intervenção do utilizador).

O planeamento global de trajectórias baseia-se num mapa topológico do ambiente e num mapa de grelhas com informação qualitativa adquirida previamente. O mapa qualitativo (figura 4.3) é constituído por uma grelha uniforme de células rectangulares. Cada célula do mapa contém dois tipos de informação: 1) indicação de quais os sensores

da cadeira activos nessa posição do mapa (distância medida inferior a um determinado limiar); 2) informação qualitativa das diferenças das leituras dos sensores entre células adjacentes. As diferenças expressam-se através de 3 termos: aumenta (A), diminui (D) e estável (E). Como se pode observar na figura, cada célula contém um vector de 1s e 0s indicando quais os sensores activos ou não, e possui um conjunto de 8 vectores com a informação que representa a variação qualitativa do comportamento dos sensores. Cada sensor é numerado de 0 a 7 (0-N, 1-NE, ..., 7-NW). No exemplo da figura, os sensores relativos às orientações N, NE e E estão inactivos (sem obstáculos) e os restantes estão activos. Por exemplo, a célula adjacente a NE indica que a cadeira nessa posição teria os sensores N, NE e E a medir as mesmas distâncias que esses sensores na posição actual, e que as leituras dos sensores SE a NW aumentariam. O algoritmo de planeamento global calcula (sem ser em linha) a relação topológica entre o ponto actual do veículo e o ponto destino, armazenando um conjunto de direcções para atingir o destino de forma a minimizar o percurso global. O planeador estima em tempo real a posição do veículo e, para cada ponto do mapa, encontra e segue as direcções preferidas para chegar ao destino. Se encontrar direcções alternativas, escolhe a que se aproxima mais da orientação do veículo. O planeador possui ainda memória de curto prazo utilizada para evitar que o veículo se desloque para posições anteriormente identificadas como obstruídas (importante em ambientes dinâmicos) e actualiza o mapa.

O sistema de navegação possui também um módulo de desvio de obstáculos que pode ser utilizado simultaneamente com o planeamento global de trajectórias ou apenas conjugado com comandos do utilizador através de um *joystick* ou voz, resultando daí um modo de funcionamento semi-autónomo. Este módulo visa proteger a cadeira de colisões e evitar obstáculos retomando o percurso ou movimento desejado pelo utilizador. O método de desvio de obstáculos baseia-se no método VFH. Foi designado *histograma cinemático activo* (AKH) porque as modificações ao método VFH satisfazem as restrições cinemáticas da cadeira. Em particular, este método pretende minimizar os efeitos prejudiciais das rodas dianteiras da cadeira de rodas. Por exemplo, quando a cadeira executa movimentos de rotação acentuados e depois pretende executar movimentos rectilíneos, as rodas dianteiras impedem que a cadeira siga um movimento rectilíneo, por ficarem enviesadas e desviarem o movimento inicial da cadeira[2]. O AKH pretende precisamente minimizar o efeito de

[2]Este problema foi também identificado no nosso sistema RobChair

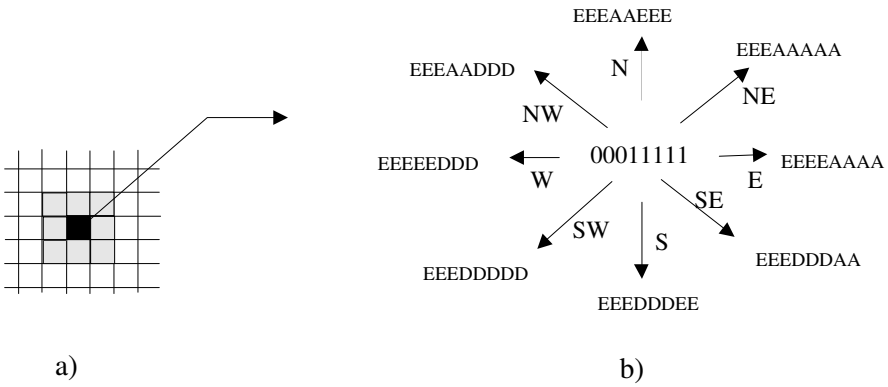


Figura 4.3: a) Mapa em forma de grelha; b) Exemplo de informação qualitativa de uma célula da grelha

rotações muito acentuadas da cadeira. Para isso, o AKH utiliza uma janela adaptativa (AKW) conjuntamente com um nível limiar adaptativo. A figura 4.4 representa um cenário possível e o correspondente histograma polar. A janela adaptativa, centrada no eixo do veículo, reduz o conjunto de direcções a seguir à zona definida pela janela, ou seja, haverá preferência para seguir direcções que não envolvam rotações acentuadas. Note-se que, se não houver direcções livres na janela, o veículo terá de rodar. No exemplo da figura 4.4, a direcção escolhida pelo método VFH para atingir o ponto G seria a zona livre localizada a 90º. Utilizando o AKH, o veículo segue uma trajectória frontal evitando uma rotação acentuada. A diminuição do nível limiar faz com que o veículo comece a escolher o caminho mais cedo tornando assim as manobras mais suaves.

[Projecto da cadeira de Bremen] - Este projecto [RL99] dá ênfase às questões de segurança. Possui um módulo de evitamento de colisões cuja única função é a de parar antes da ocorrência de uma colisão. Para determinar quando o sistema deve parar, o módulo de segurança leva em atenção um conjunto de parâmetros tais como a velocidade e desaceleração da cadeira. Estes parâmetros permitem calcular a distância de paragem e a distância que o veículo pode percorrer até colidir com um obstáculo. A implementação deste módulo divide-se em duas partes: geração de um mapa local e cálculo da distância ao obstáculo mais perto. O mapa utiliza uma grelha quadrática de células [Elf87] que indica a presença (com vários graus de certeza) de um obstáculo numa determinada posição. Para calcular a distância ao obstáculo mais próximo, utiliza-se um conceito de sensores virtuais.

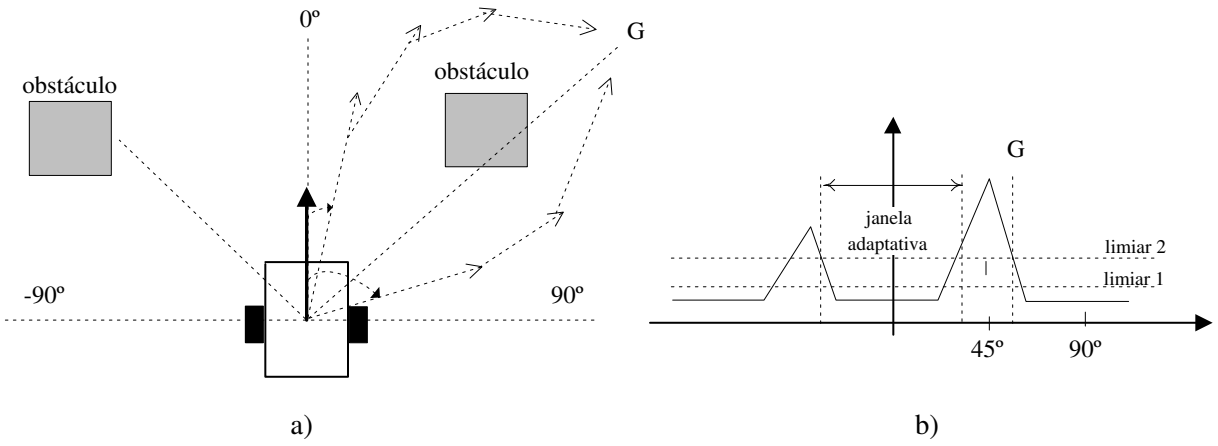


Figura 4.4: a) Cenário; b) AKH correspondente ao cenário

Este método consiste na pré-determinação da trajectória da cadeira, o que permite saber exactamente quais as distâncias aos obstáculos que o veículo vai encontrar nesse percurso.

Para além do módulo de segurança, a cadeira de Bremen possui um módulo de controlo partilhado composto por vários modos de funcionamento: controlador de velocidade, desvio de obstáculos e comportamento automático. O controlador de velocidade pretende reduzir a velocidade de acordo com a distância entre o obstáculo e o veículo. No entanto, os comandos de direcção via *joystick* fornecidos pelo utilizador permanecem inalterados. No modo de desvio de obstáculos, o utilizador dá a velocidade máxima e dá uma indicação grosseira da direcção, enquanto o sistema determina a velocidade e direcção do veículo. No modo de comportamento automático, utilizado para tarefas tais como seguimento de paredes e entrada em portas, o sistema de navegação tem total controlo. O utilizador apenas pode intervir para parar a execução do comportamento.

4.2.2 Interfaces Homem-Máquina

[*Joystick* com efeito reflectivo] - A cadeira de rodas *Luoson* [LHCL99] utiliza um *joystick* que simula o efeito da presença de paredes ou obstáculos, exercendo forças reflexivas de volta ao utilizador. Quando o utilizador opera o *joystick* sente uma oposição (resistência) repulsiva no caso de uma parede ou um efeito de tremura que assinala a presença de obstáculos. A cadeira encontra-se equipada com um anel de sensores de ultrassons cuja informação permite criar uma distribuição de obstáculos à volta da cadeira.

A partir da intenção do utilizador fornecida pelo *joystick*, atribui-se a cada objecto um peso que depende de uma distribuição Gaussiana centrada na intenção do utilizador (direcção). Os pesos são somados para escolher o tipo de efeitos da força de reflexão. A amplitude da força é então calculada através de um controlador difuso que aceita à entrada o peso dos objectos e o comando do utilizador.

[Controlo por detecção de movimentos da cabeça] - Um sistema de visão para detecção de movimentos da cabeça foi concebido em [BMG+99] para controlar uma cadeira. É particularmente útil para pessoas com deficiências motoras severas e/ou, por exemplo, com problemas de fala. O sistema consegue detectar um conjunto de movimentos da cabeça e expressões faciais aos quais correspondem comandos de movimento, tais como *roda, aumenta velocidade, liga/desliga.*

[Controlo por comandos de voz] - Em [SL97] é analisado o uso de uma interface por comandos de voz para controlar a cadeira NavChair. O utilizador tem à sua disposição um pequeno conjunto de comandos de direcção que utiliza para conduzir a cadeira. O sistema de navegação assiste o utilizador executando, quando necessárias, manobras de correcção para ajustar a trajectória do veículo entre os obstáculos. Se ocorrerem comandos mal interpretados ou simplesmente ignorados, o módulo de navegação garante que estes comandos não provoquem colisões. Alguns dos comandos utilizados estão representados na tabela seguinte:

Comando	
`parar`	a cadeira pára imediatamente
`frente/trás`	a cadeira move-se para a frente/trás
`roda direita/esquerda`	a cadeira faz uma rotação pura
`direita/esquerda forte`	a cadeira vira 20º e depois continua o movimento para a frente/trás
`direita esquerda suave`	a cadeira vira 20º e depois continua o movimento para a frente/trás

4.2.3 Adaptações Mecânicas

O projecto europeu OMNI [BHH+95] desenvolveu uma cadeira de rodas cuja principal inovação é a sua elevada manobrabilidade. Possui 4 rodas especiais designadas por rodas *Mecanum* que permitem à cadeira movimentar-se com três graus de liberdade num plano, tornando possível qualquer combinação de movimentos: frente, lateral e rotação. Esta capacidade elimina muitas das limitações cinemáticas das cadeiras normais, que em ambientes pouco espaçosos e congestionados se tornam difíceis de controlar. A cadeira possui também um assento especial que permite aos utilizadores elevarem-se cerca de 90 cm. O assento ajusta-se às necessidades do utilizador e pode ser inclinado até 30º.

Em [WKK94] descreve-se um protótipo de uma cadeira de rodas equipada com rodas e pernas. Esta adaptação visa fornecer ao utilizador a capacidade de vencer grande parte das barreiras arquitectónicas existentes em ambientes exteriores. Este sistema pode circular em terrenos irregulares, sendo capaz de ultrapassar obstáculos como por exemplo degraus e transposição de passeios.

Capítulo 5

Cadeira de Rodas RobChair: Hardware

Descreve-se, neste capítulo, a plataforma que serviu de base experimental ao trabalho desenvolvido.

5.1 Cadeira de Rodas

A figura 5.1a) ilustra a cadeira de rodas fabricada pela empresa Vector Mobility e adquirida ao instituto americano KIPR (Kiss Institute for Practical Robots) [fPR]. É uma cadeira de rodas motorizada convencional controlada por meio de um *joystick* analógico e preparada para uso interior e exterior. Possui cinco rodas: duas rodas motorizadas, duas pequenas rodas dianteiras livres e ainda uma última roda livre na parte traseira adaptada através de um amortecedor. Esta roda confere maior estabilidade e segurança à cadeira. A sua autonomia, de algumas horas, é garantida por duas baterias de 12 Volts (V). A cadeira, baptizada TinMan, foi modificada pela equipa de trabalho do KIPR, que a equipou com um microcontrolador da Motorola M68332 (Onset TT8 - "Onset TattleTale model 8") [Ons95]), 12 sensores opto-electrónicos digitais reflectivos, um pára-choques de contacto e codificadores ópticos rudimentares nas rodas.

a)

b)

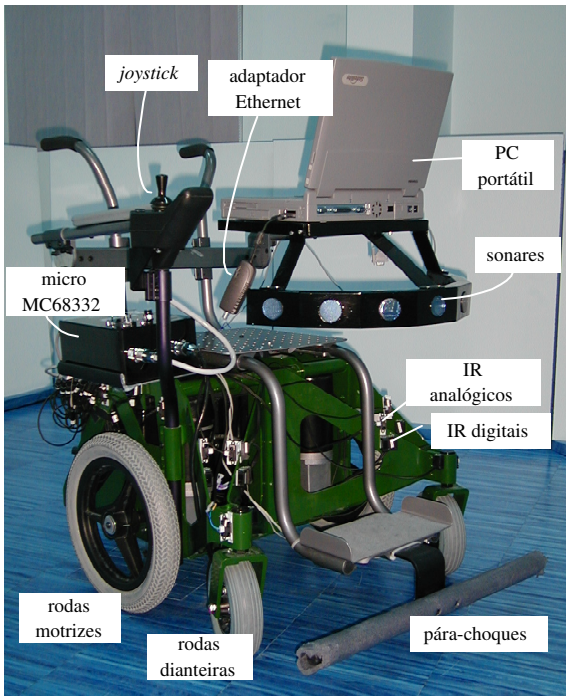


Figura 5.1: a) Cadeira TinMan; b) Cadeira RobChair

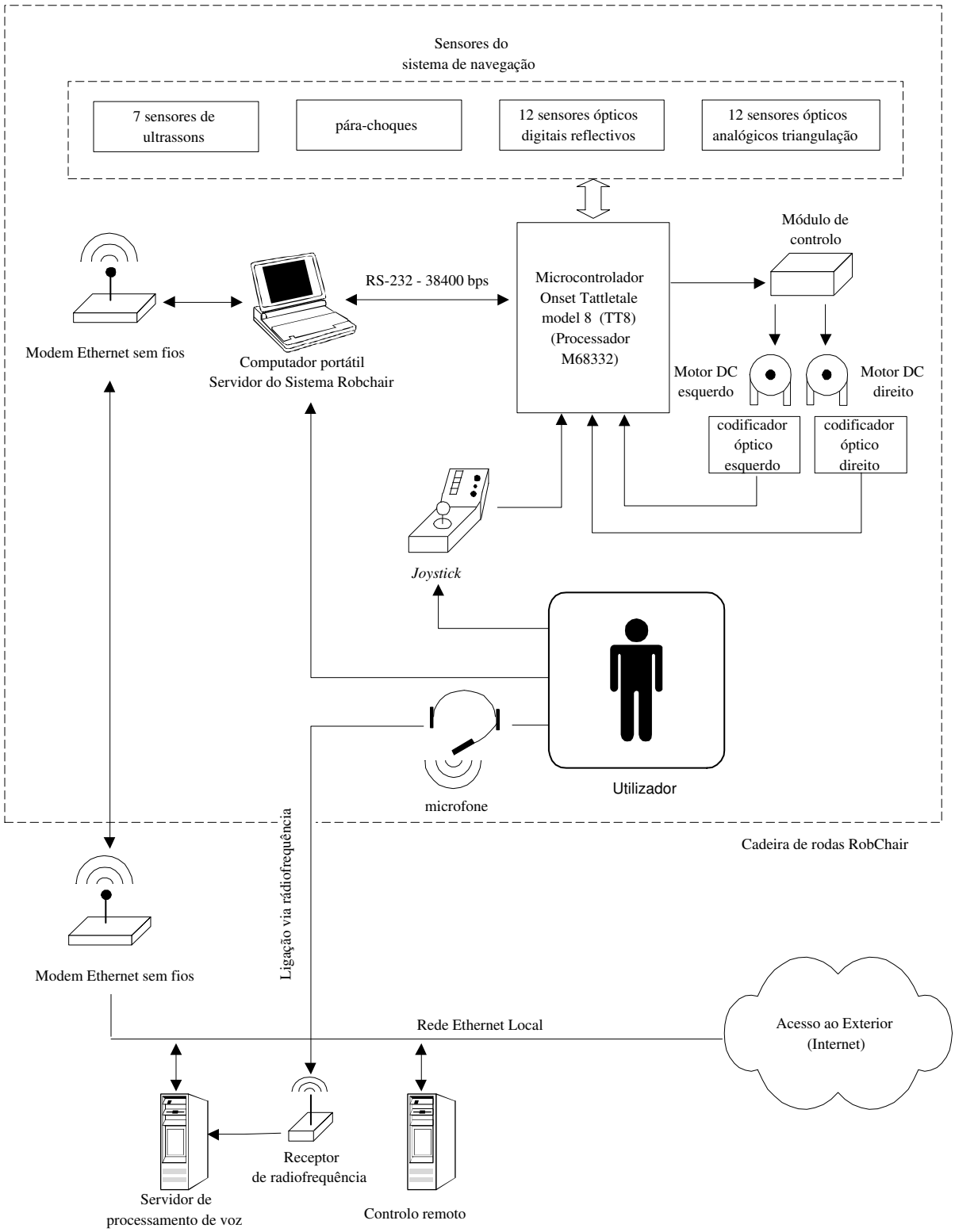


Figura 5.2: Sistema RobChair

5.2 Descrição Geral

Numa primeira fase do trabalho estudou-se o funcionamento do *hardware* que equipava a cadeira TinMan, nomeadamente, o controlo dos motores e as interfaces com o *joystick* e os sensores. Seguidamente, melhorou-se o sistema de navegação. Integraram-se novos sensores opto-electrónicos analógicos por triangulação, sensores de ultrassons e um sistema de odometria mais fiável, o que passou pelo desenvolvimento de *hardware*. Incorporou-se a bordo, um computador portátil (Pentium 200Mhz). Fora da cadeira, existe ainda uma estação de trabalho responsável pelo reconhecimento de voz e uma estação de trabalho para monitorização e controlo remoto da cadeira. O diagrama de blocos da figura 5.2 apresenta o sistema RobChair. O microcontrolador realiza as tarefas de baixo nível enquanto o computador portátil executa os algoritmos de navegação. A ligação entre estes dois sub-sistemas faz-se através de uma ligação série. O computador encontra-se ligado à rede Ethernet do ISR, permitindo comunicação TCP/IP com as estações de reconhecimento de voz e de controlo remoto, ou ainda com qualquer outro computador ligado à Internet.

5.3 Electrónica

5.3.1 Microcontrolador

O microcontrolador é responsável pelas tarefas de baixo nível: aquisição sensorial, geração de sinais de velocidade e direcção, e interface com *joystick*. Encontra-se ligado ao computador portátil através de uma ligação série RS-232 bi-direccional a um débito de 38400 bits por segundo (bps).

O microcontrolador TT8 baseia-se no microprocessador da Motorola MC68332 [Mot95]. Este processador de 32 bits funciona a frequências de relógio dinamicamente ajustáveis entre 160Khz e 16Mhz. Possui uma unidade de processamento temporal (TPU) que controla 16 linhas de entrada/saída (I/O), cada uma delas individualmente programável, e dois temporizadores.

Resumem-se, em seguida, as componentes mais relevantes do microcontrolador (especificações técnicas detalhadas encontram-se em [Mot95] [Ons95]):

- Processador MC68332 da Motorola;

- Conversor A-D (Analógico-Digital) de 12 bits com 8 canais;
- 8 entradas/saídas (I/O) digitais;
- Unidade de Processamento Temporal (TPU) - trata-se de um microcontrolador inteligente concebido para o controlo de eventos e variáveis temporizadas. Funciona independentemente do CPU (Unidade de processamento Central) e foi especialmente concebido para executar operações de captura. A ocorrência destas operações denomina-se um evento. A programação de uma série de acções resulta numa função. As funções TPU substituem funções de *software* que se fossem utilizadas obrigariam a uma interrupção do CPU. Desta forma, a TPU liberta o CPU para execução de outras tarefas. Cada um dos seus 16 canais pode ser programado para as seguintes funções:

  1. Entrada/saída discreta;
  2. Captura de bordo ascendente/descendente de sinal de entrada;
  3. Contador de transições de uma entrada;
  4. Medição de período;
  5. Modulação de largura de pulso (PWM): permite gerar uma saída com frequência e ciclo de trabalho programáveis;

  A TPU possui 2 contadores de 16 bits que fornecem uma base de tempo para cada um dos eventos. As bases de tempo dos contadores são controladas através de dois pré-escaladores: TCR1 e TCR2. No primeiro, a entrada corresponde à frequência do relógio do sistema (16 MHz) dividida por 4 ou 32. O pré-escalador divide esta entrada por 1, 2, 4 ou 8. Consegue-se assim uma base de tempo mínima de 250 ns. O pré escalador TCR2 tem um funcionamento semelhante, mas permite ter como entrada um relógio externo.

- Interface de comunicação série RS-232 UART, para comunicação com dispositivos externos. Funciona em modo bi-direccional simultâneo;
- 256 Kb de RAM e 256 Kb de Flash EEPROM;

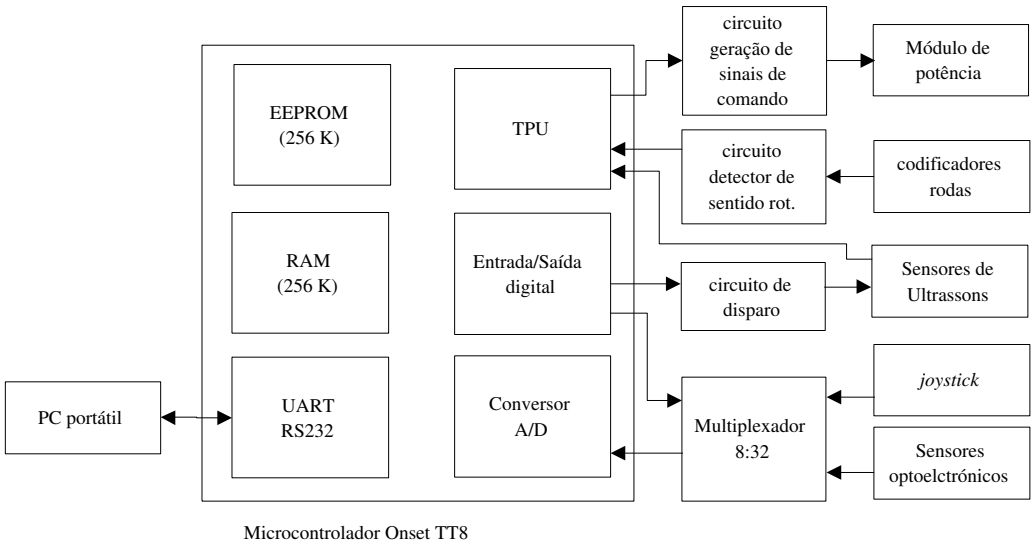


Figura 5.3: Ligação entre o microcontrolador TT8 e os dispositivos da cadeira

- Programação em C ou BASIC. No nosso caso, utilizámos o ambiente ARC C desenvolvido para programar o processador MC68332. O ARC [WSW96] foi desenvolvido pelo *Newton Research Labs* [Lab] para programar sistemas embebidos de 32 bits. Consiste num conjunto de ferramentas de compilação, que incluem o compilador de C, `arcc`, e um programa de interacção série, `arc`, utilizado para carregar os programas para o microcontrolador e também para fazer a depuração de programas.

O diagrama de blocos da figura 5.3 representa a ligação entre os vários dispositivos do sistema e os módulos do microcontrolador.

5.3.2 Módulo de Controlo dos Motores e *Joystick*

O módulo de controlo dos motores [Gil], do fabricante "Penny & Giles" é fornecido com um *joystick* analógico para o qual se encontra adaptado. O módulo de controlo inclui um módulo de cinemática e um módulo de potência (figura 5.4). O módulo de cinemática recebe do *joystick* dois sinais analógicos, intensidade do *joystick*, $\bar{r}$, e ângulo do *joystick*, $\bar{\varphi}$ que servem para determinar a velocidade linear, v, e a velocidade angular, w da cadeira. Com base nestes sinais o módulo de cinemática determina as velocidades para cada roda (w_e e w_d). Veremos mais à frente como se relacionam os sinais $\bar{r}$ e $\bar{\varphi}$ com a velocidade linear, v, e a velocidade angular w. Os sinais de velocidade w_e e w_d são enviados ao módulo

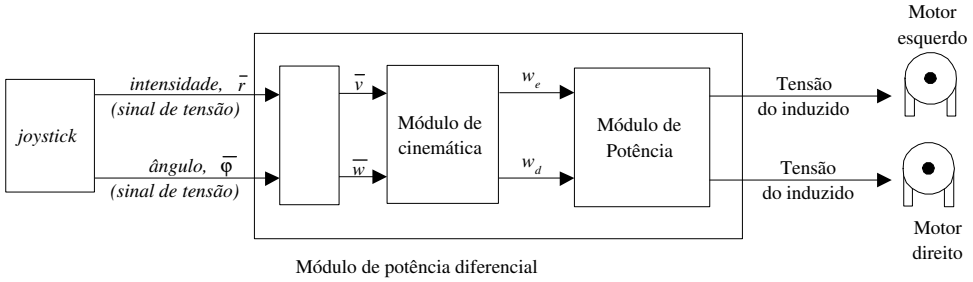


Figura 5.4: Módulo de potência diferencial: módulo de cinemática e módulo de potência

de potência que gera os sinais de comando (tensão do induzido - sinais PWM de 24 V) para os motores DC. É importante referir que as velocidades são determinadas pelo módulo de cinemática e este só pode ser programado pelo fabricante. Dada a indisponibilidade para programarmos internamente o módulo de controlo, este é visto como uma "caixa negra" em que apenas se conhecem as entradas e as saídas.

Joystick

A ligação entre o *joystick* e o módulo de controlo é composta por 16 sinais (ver apêndice B). Para o controlo de movimento, apenas interessam 3 destes sinais: a intensidade do *joystick* , r, o ângulo do *joystick*, φ, e a referência do *joystick*. De acordo com a posição do *joystick*, são gerados níveis de tensão independentes para o sinal de intensidade e para o sinal de ângulo. Estes níveis de tensão variam em torno da tensão de referência do *joystick* (tabela 5.1). A figura 5.5 faz a correspondência entre a posição do *joystick* e os níveis de tensão associados. A intensidade corresponde à amplitude do deslocamento do *joystick* e é independente da sua posição angular, ou seja, o valor de tensão é idêntico para qualquer posição do *joystick* pertencente à mesma linha circular superior para o sentido positivo, e à mesma linha circular inferior para o sentido negativo (ver figura 5.5a); o ângulo corresponde à posição angular do *joystick* e é independente da amplitude de deslocamento do *joystick*. A tensão varia entre o seu máximo e mínimo por quadrante. Para valores angulares positivos (1º e 4º quadrante), a tensão varia entre os 5.8 e 6.8 V, e para valores angulares negativos (2º e 3º quadrante) a tensão varia entre os 4.8 e 5.8 V (ver figura 5.5b). Os sinais de intensidade e ângulo do *joystick* (r, φ) podem ser vistos como comandos de velocidade linear e angular (v, w), sendo v o sinal proporcional à velocidade linear desejada, e w o sinal proporcional à velocidade angular desejada. Observando a

	Tensão (V)
Intensidade, $\bar{r}$	4.8 - 6.8
Ângulo, $\bar{\varphi}$	4.8 - 6.8
Referência *joystick*	5.8

Tabela 5.1: Valores de tensão do *joystick*

figura 5.6 verificamos que (v, w) pode ser obtido a partir de (r, φ) da seguinte forma:

$$\begin{cases} v = r\cos(\varphi) \\ w = r\sin(\varphi) \end{cases} \tag{5.1}$$

Quando o *joystick* se encontra alinhado segundo o eixo x, ou seja, $\varphi = 0$, então:

$$\begin{cases} v = r\cos(0) = r \\ w = r\sin(0) = 0 \end{cases}$$

ou seja, a cadeira tem um deslocamento linear a uma dada velocidade linear proporcional a v. Se o *joystick* estiver alinhado segundo o eixo y, isto é $\varphi = \pi/2$, então:

$$\begin{cases} v = r\cos(\pi/2) = 0 \\ w = r\sin(\pi/2) = r \end{cases}$$

ou seja, a cadeira sofrerá uma rotação pura (rotação sobre si própria) com uma determinada intensidade proporcional a w.

Modos de Funcionamento da Cadeira RobChair

A cadeira RobChair possui dois modos de funcionamento: o modo manual e o modo automático (figura 5.7). Estes modos são seleccionados através de um interruptor (manual/auto). No modo manual, a cadeira funciona como uma cadeira de rodas convencional controlada por *joystick*. O módulo de controlo recebe directamente do *joystick* as acções de comando (velocidade linear e angular); no modo automático, existem duas formas de controlar a cadeira: através de comandos de voz e através do *joystick*. No primeiro caso, o comando de voz é processado pelo software de reconhecimento, convertido num comando numérico, filtrado pelo algoritmo de navegação (que decide as acções de controlo em função de estímulos do meio ambiente ou do planeador de trajectória) e enviado ao microcontrolador. O comando numérico é então convertido nos sinais de velocidade linear

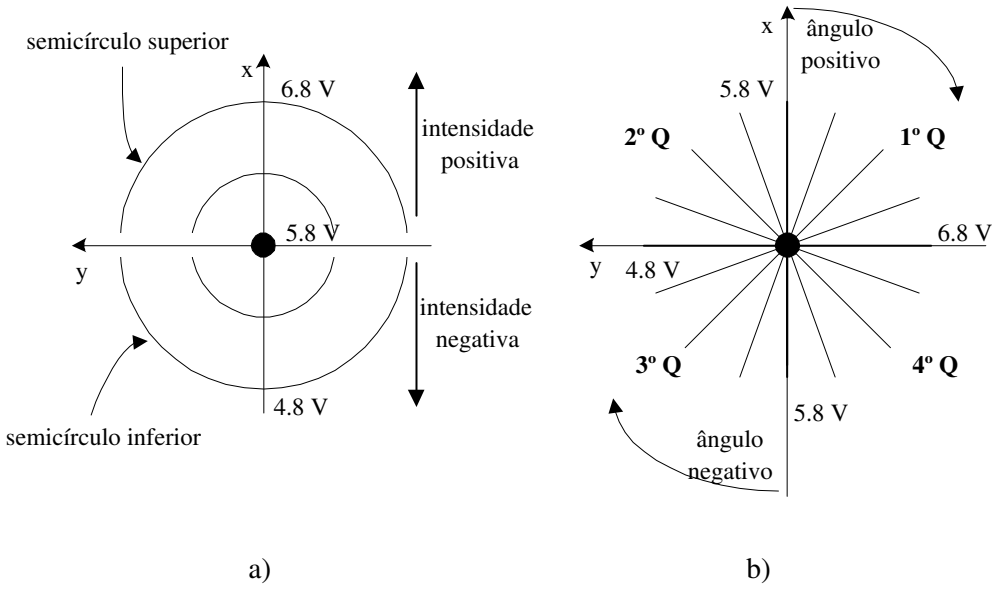


Figura 5.5: Níveis de tensão associados à posição do *joystick*. a) Intensidade, $\bar{r}$; b) Ângulo, $\bar{\varphi}$

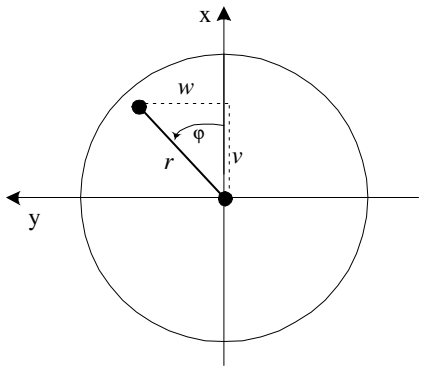


Figura 5.6: Correspondência entre a posição do *joystick* (r, φ) e os comandos de velocidade linear e angular (v, w)

	Entrada		Saída do multiplexador (V)	Saída do filtro (V)
	Sentido B	Intensidade A	COM	
Sentido positivo	1	0	6.8 (ch2)	5.8 - 6.8
	1	1	5.8 (ch3)	
Sentido negativo	0	0	4.8 (ch0)	4.8 - 5.8
	0	1	5.8 (ch1)	

Tabela 5.2: Sinal à saída do multiplexador

e angular e enviado para o módulo de controlo. No segundo caso, a posição do *joystick* é lida pelo microcontrolador através do conversor Analógico para Digital, convertida num comando numérico, seguindo-se um processo idêntico ao dos comandos de voz. Os sinais de velocidade linear e angular enviados ao módulo de controlo são gerados pelo microcontrolador, devendo ser iguais aos sinais enviados directamente pelo *joystick* em modo manual.

Níveis de Tensão Gerados pelo Microcontrolador

No modo automático é necessário gerar os sinais de tensão de intensidade e ângulo a enviar ao módulo de controlo dos motores. O circuito que gera o sinal de tensão relativo à intensidade está ilustrado no diagrama de blocos simplificado da figura 5.8. O circuito tem por função gerar um sinal contínuo, variável entre 4.8 e 6.8 V, através da variação do ciclo de trabalho de um sinal PWM. As linhas de endereçamento do multiplexador, A e B, têm à entrada, respectivamente, um sinal PWM cujo ciclo de trabalho é proporcional ao módulo do sinal de intensidade, r, e um sinal digital (0/1) que indica o sentido do *joystick* (negativo ou positivo). Cada canal do multiplexador (ch0 a ch3) tem um nível de tensão associado (4.8, 5.8, 6.8 e 5.8 V, respectivamente). Assim, obtém-se, à saída do multiplexador, um sinal PWM cuja tensão está compreendida entre os 5.8 e 6.8 V para o sentido positivo e entre 4.8 e 5.8 V para o sentido negativo (ver tabela 5.2). O sinal PWM resultante passa de seguida por um filtro RC passa-baixo que apenas retém a componente média do sinal. Variando o ciclo de trabalho do sinal PWM, consegue-se variar a tensão média entre 4.8 e 6.8 V. Aumentando a frequência do sinal PWM, obtém-se uma melhor resolução na tensão de saída.

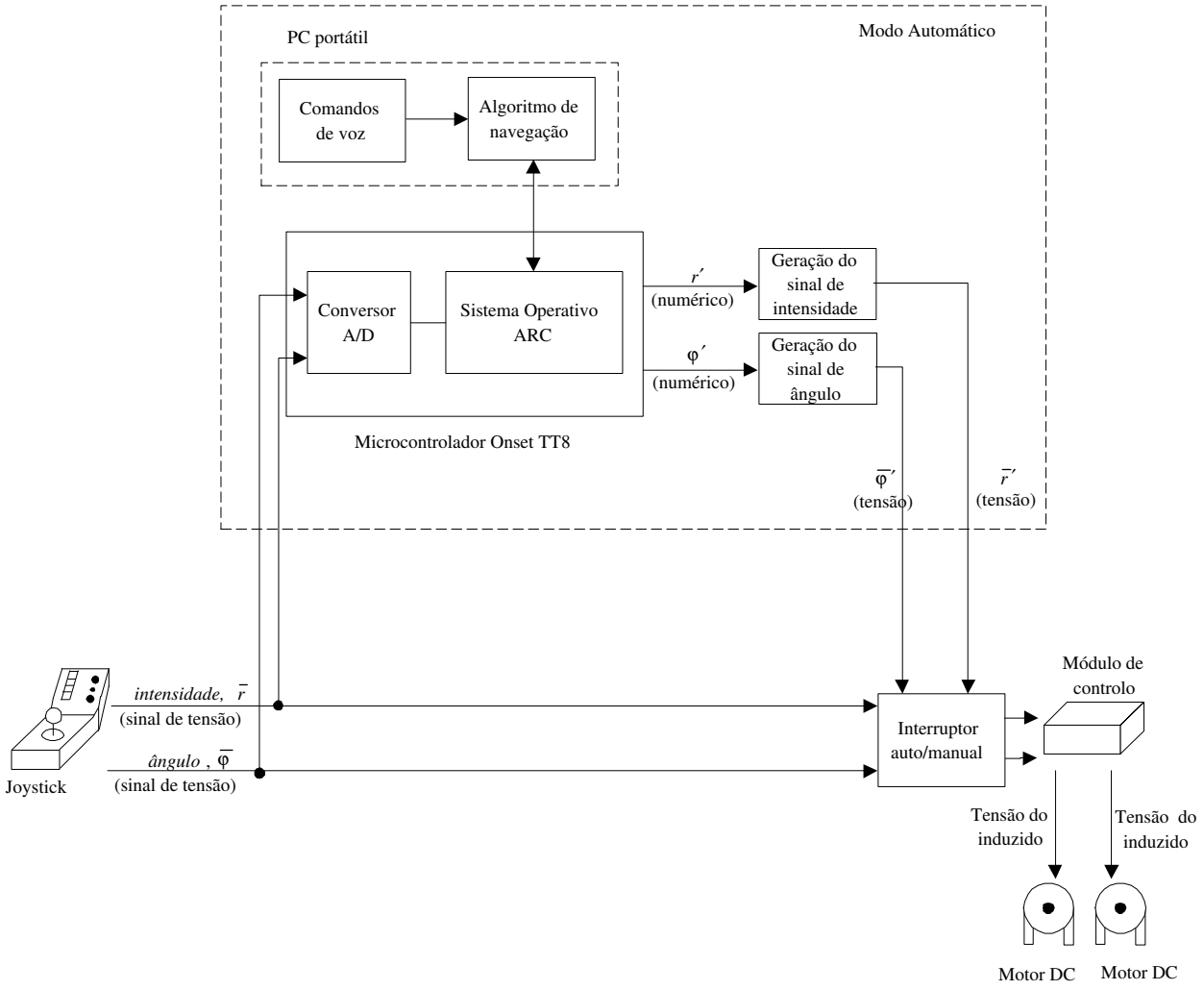


Figura 5.7: Modos de funcionamento: a) Manual - o módulo de controlo recebe comandos directamente do *joystick*; b) Automático - os comandos do *joystick* são lidos pelo microcontrolador que os processa e envia para o módulo de controlo

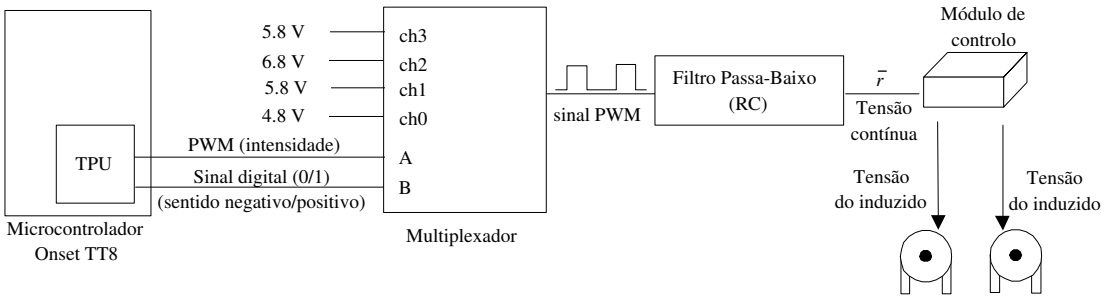


Figura 5.8: Exemplo para a geração do sinal de velocidade pelo microcontrolador. O sinal de direcção é obtido de forma idêntica

5.3.3 Sistema Sensorial

Em robótica móvel são vulgarmente utilizados sensores opto-electrónicos e sensores de ultrassons. A percepção do meio ambiente fornecida por estes tipos de sensores, ainda que bastante pobre quando comparada com sistemas de visão, adequa-se a muitas aplicações de robótica. Aproveitando a elevada direccionalidade dos sensores opto-electrónicos e a capacidade dos sensores de ultrassons em medir distâncias grandes com elevada resolução, diminui-se o grau de incerteza dos dados adquiridos.

No sistema de navegação RobChair, utilizaram-se 12 sensores opto-electrónicos digitais reflectivos, 12 sensores opto-electrónicos analógicos por triangulação, 7 sensores de ultrassons, um pára-choques dianteiro e codificadores ópticos para odometria. Nas próximas secções, descrevem-se os modos de funcionamento destes sensores.

Sensores Opto-electrónicos

Os sensores opto-electrónicos digitais reflectivos são constituídos por uma fonte de luz e um foto-detector colocados lado a lado. O seu princípio de funcionamento consiste na medição da intensidade de luz reflectida pela superfície reflectora que varia de acordo com a distância. Estes sensores são bastante sensíveis às condições de luminosidade e propriedades ópticas (cor e textura) dos objectos reflectores, e à inclinação entre sensor e superfície [MNdA99]. O princípio de funcionamento dos sensores analógicos GP2D12 baseia-se no método de triangulação (figura 5.9) [HS90]. Um feixe de luz é emitido por um emissor de luz (LED), focado por uma lente de transmissão. A luz reflectida, segundo um determinado ângulo de incidência, é focada através de uma lente de recepção, incidindo num sensor PSD (detector sensível à posição), criando um triângulo entre o ponto de reflexão, o emissor e o detector. A posição detectada pelo PSD permite determinar o ângulo de reflexão e consequentemente a distância ao obstáculo. O método de triangulação oferece como principais vantagens maior imunidade à interferência da luz ambiente e insensibilidade às cores dos objectos reflectores.

Os sensores digitais (SUNX CX-22) aceitam um ajuste de limiar de detecção entre 10 e 100 cm (figura 5.10). O sensor coloca à saída 12V se detectar um obstáculo e 0 V caso não haja detecção. Os sensores analógicos (Sharp GP2D12) têm um resposta não linear (figura 5.11). No entanto, pode-se aproximar a curva por três zonas lineares, entre 10 e 20 cm, 20 e 40 cm, e entre 40 e 80 cm (valores acima dos 80 cm são pouco fiáveis). Os

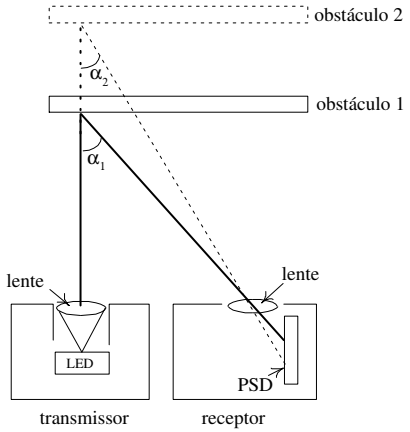


Figura 5.9: Princípio de funcionamento por triangulação utilizado pelos sensores opto-electrónicos Sharp GP2D12

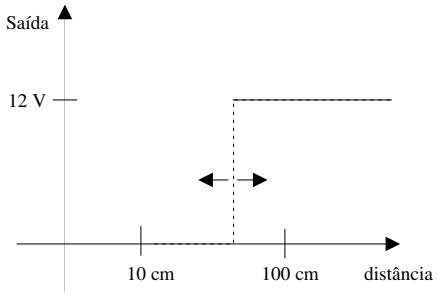


Figura 5.10: Saída do sensor opto-electrónico digital SUNX CX-22. O sensor possui um ajuste de detecção

valores de tensão vs. distância são armazenados em tabelas, permitindo posteriormente obter as distâncias a partir de interpolação linear dos valores de tensão armazenados.

A interface destes sensores é muito simples, uma vez que os valores de tensão dos sensores são lidos directamente pelo conversor A-D do microcontrolador. Dado só existirem 8 canais A-D, foi necessário desenvolver uma placa de multiplexagem que permite a ligação de 32 entradas analógicas. O conversor A-D de 12 bits permite uma resolução de $5/4096 = 1.22mV$

Sensores de Ultrassons (sonares)

Os sensores de ultrassons são os sensores mais utilizados em robótica móvel. Utilizam-se para desvio de obstáculos em ambientes dinâmicos não estruturados, assim como pa-

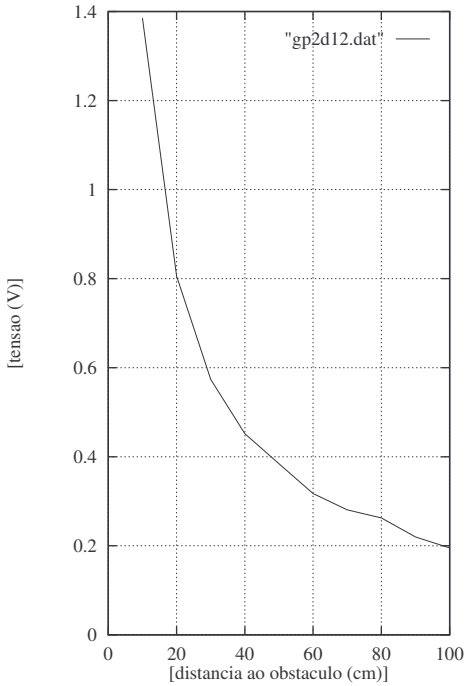


Figura 5.11: Curva de saída de tensão vs. distância do sensor opto-electrónico analógico Sharp GP2D12

ra construção de mapas. A sua vasta utilização deve-se essencialmente a três factores: adequam-se às aplicações de tempo real; permitem medir distâncias grandes (até 10 m) com elevada resolução; e são de baixo custo. Os transdutores de ultrassons apresentam um diagrama de radiação geralmente modelado por um cone com determinada abertura angular θ (figura 5.12a)). Devido à sua fraca direccionalidade, a localização espacial do obstáculo é imprecisa (figura 5.12a)). No entanto, esta característica também pode ser proveitosa, na medida em que o sensor cobre uma maior área de espaço. Estes sensores apresentam ainda duas outras desvantagens: a não detecção do eco quando a superfície reflectora e o transdutor perfazem um ângulo superior à abertura do cone; e reflexões múltiplas (figura 5.12b)).

Utilizaram-se os sensores de ultrassons da Polaroid [Pol]. Estes sensores funcionam simultaneamente como transmissor/receptor e possuem dois modos de operação: mono-eco e multi-eco. No primeiro modo, por cada transmissão é detectado apenas um eco, enquanto no segundo modo, são detectados ecos múltiplos provenientes de vários alvos. A transmissão do sensor corresponde ao envio de 16 pulsos a uma frequência de 49.4 Khz. O diagrama de radiação do transdutor aproxima-se geralmente de um cone de abertura

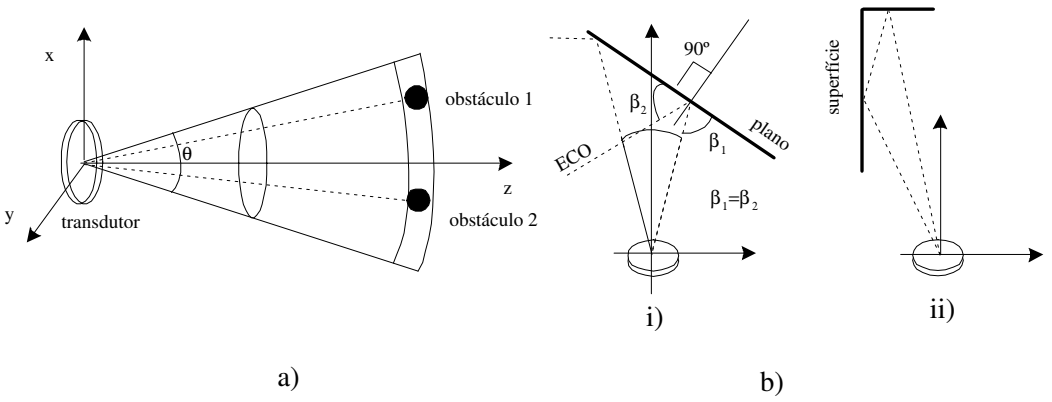


Figura 5.12: a) Modelo de radiação do sonar; b)i) Exemplo de uma situação em que o obstáculo não é detectado. Esta situação acontece quando o plano se encontra com uma orientação superior a $\theta/2$; ii) Exemplo de reflexões múltiplas que levam a medidas incorrectas

angular de 20º. É possível controlar o ganho do módulo receptor, o que permite minimizar o ruído e as detecções dos lobos laterais.

A activação da transmissão é dada pelo sinal INIT (figura 5.13). Para evitar que a transmissão seja detectada como um eco, existe uma paragem interna de 2.38 ms após a transmissão. Este valor corresponde a uma distância aproximada de 40 cm, o que significa que essa será a distância mínima detectada. No entanto, seria possível diminuir a paragem interna colocando o sinal BINH (figura 5.13) a um (5 V) (não utilizado no nosso caso).

O modo de funcionamento utilizado, mono-eco, encontra-se descrito na figura 5.13. O sinal é transmitido colocando o sinal INIT a um. O sinal reflectido (eco) é detectado e amplificado, aparecendo como um nível de tensão a um (ECHO). O tempo decorrido entre o sinal INIT a um e o sinal ECHO a um, tempo de vôo (TOF), é directamente proporcional à distância entre o transdutor e o alvo, $d = \frac{c \times t_{OF}}{2}$ (m). $c = v_0 \times \sqrt{\frac{T}{T_0}}$ (m/s) é a velocidade do som, sendo T a temperatura (ºK), T_0 a temperatura absoluta (273 ºK) e v_0 a velocidade do som para 273 ºK.

A medição do tempo decorrente entre a transmissão do pulso e a detecção do eco é feita através de uma função TPU do microcontrolador. Esta permite ler o valor de um contador de 16 bits quando ocorre um evento (bordo ascendente do eco). O valor do contador é lido quando o pulso é transmitido e lido novamente quando o sinal ECHO passa a um (eco recebido). A frequência do relógio do contador foi programada para 0.5 Mhz permitindo uma resolução 2 μs. A diferença entre o valor final e inicial do contador

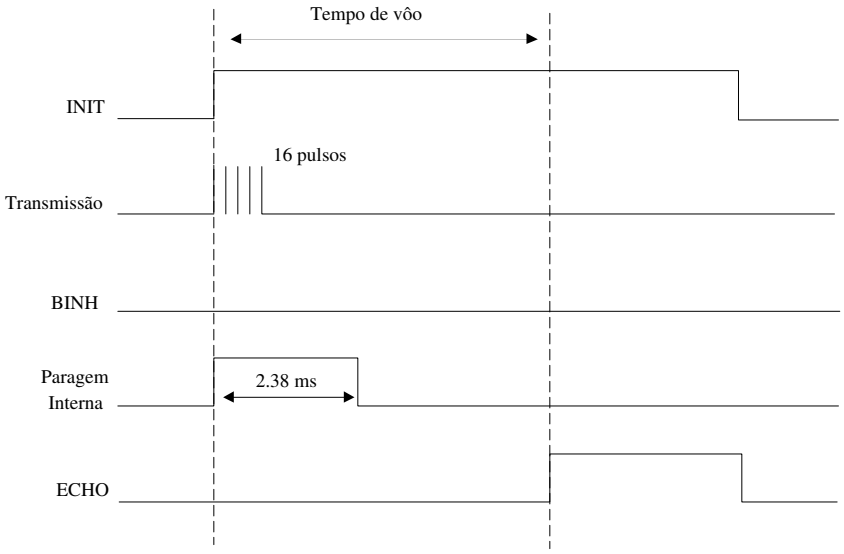


Figura 5.13: Exemplo de um ciclo em modo mono-eco

é então traduzido em tempo.

O diagrama de blocos da figura 5.14 apresenta o sistema desenvolvido para aquisição de informação dos sonares. Os sinais de INIT são controlados por *software* utilizando as linhas de I/O. O sinal de ECHO de cada sensor está ligado a um canal TPU.

Codificadores Ópticos das Rodas

A capacidade de auto-localização é essencial para o planeamento de trajectórias locais ou globais. Alguns métodos utilizados em robótica móvel passam pelo posicionamento de marcas no terreno em posições pré-definidas e que servem para posterior re-localização. Outra forma de auto-localização consiste em colocar nas rodas do veículo sistemas odométricos que fornecem informação sobre os movimentos de cada uma das rodas.

Inicialmente, a cadeira possuía um codificador óptico artesanal em cada umas das rodas motrizes (figura 5.15). O codificador consistia num disco rotativo de cor opaca com 40 buracos distribuídos na sua extremidade, acoplado ao motor, e num par emissor óptico/foto-detector localizado na extremidade do disco 5.15. À passagem de cada buraco, o foto-detector recebia a luz proveniente do emissor, gerando um pulso. O factor de desmultiplicação da engrenagem do motor é de 10, o que permitia obter 400 pulsos por cada revolução da roda. A baixa resolução destes codificadores, a falta de fiabilidade (dado

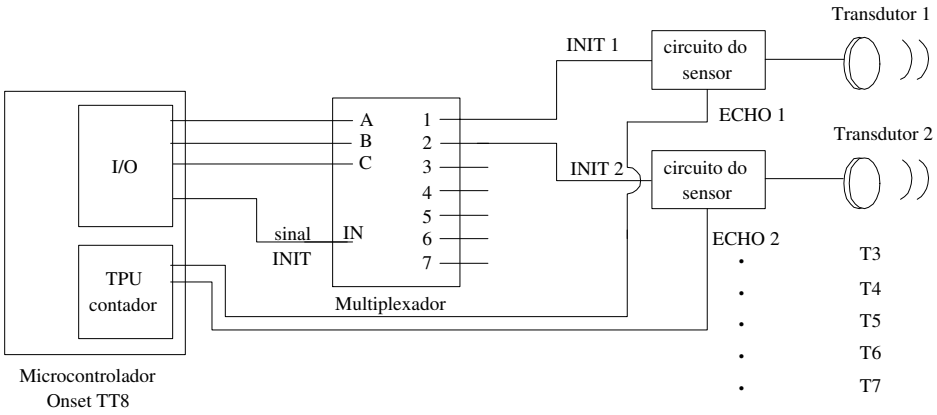


Figura 5.14: Sistema de controlo e aquisição de dados dos sonares

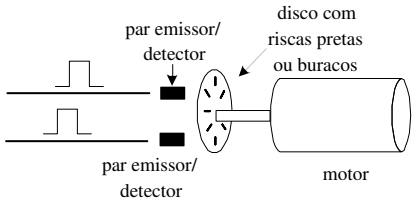


Figura 5.15: Esquema simplificado de um codificador óptico acoplado ao motor

o seu fabrico artesanal) e principalmente a incapacidade de detectar a direcção de rotação levaram-nos a substituí-los por um sistema mais eficiente. Adquiriram-se codificadores de quadratura accu-coder modelo 260 [Co.]. Estes codificadores têm uma resolução de 500 pulsos, o que corresponde a 5000 pulsos por cada revolução da roda. O princípio de funcionamento é semelhante ao codificador artesanal, no entanto, como possui dois pares emissor/foto-detector, devolve à saída dois sinais em quadratura (figura 5.16). Quando o sinal A está adiantado em relação a B, a rotação é no sentido directo, caso contrário a rotação é no sentido inverso. Para determinar a direcção de rotação pelo microcontrolador, desenvolveu-se um circuito de interface baseado num *flip-flop* (figura 5.17). No sentido directo, a saída Q está a 1, e no sentido inverso a saída está a zero.

O número de pulsos é directamente proporcional ao deslocamento de cada uma das rodas. Para determinar a relação entre o deslocamento linear da roda e o número de pulsos, é necessário saber o raio da roda (R_w) e o número de pulsos por cada revolução da roda (N_{rev}). Um pulso corresponde a:

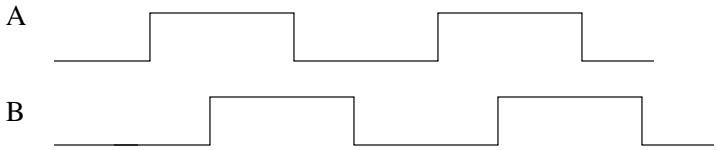


Figura 5.16: Sinais em quadratura do codificador óptico

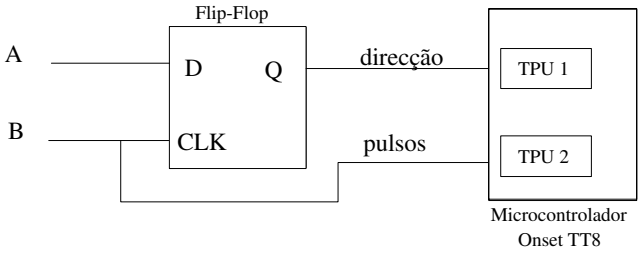


Figura 5.17: Circuito de interface do codificador utilizado para detecção do sentido de rotação

$$1pulso = \frac{2\pi R_w}{N_{rev}} \tag{5.2}$$

No nosso sistema, o raio da roda é $R_w = 177mm$ e o número de pulsos por revolução é $N_{rev} = 5000$. Substituindo na fórmula 5.2, conclui-se que cada pulso corresponde a um deslocamento linear de 0.222 mm. Esta expressão é válida para condições ideais. As rodas da cadeira, do tipo pneumático, sofrem variações significativas no seu raio, que vão depender da pressão do pneu, do peso da cadeira e do utilizador, e das irregularidades do piso. É, por isso, mais correcto utilizar na equação 5.2 o raio efectivo da roda, R_e ($\approx 170mm$), representado na figura 5.18.

Pára-Choques

A cadeira possui um pára-choques dianteiro, representado na figura 5.19. Este consiste numa barra metálica que possui um certo grau de liberdade e à qual estão fixos dois ímanes (esquerdo e direito). Na estrutura fixa existem, por sua vez, dois sensores de *efeito de Hall* localizados perto dos ímanes. Quando ocorre uma colisão, os ímanes aproximam-se destes sensores, activando-se. O sistema consegue assim detectar uma colisão frontal, esquerda ou direita consoante os sensores activados.

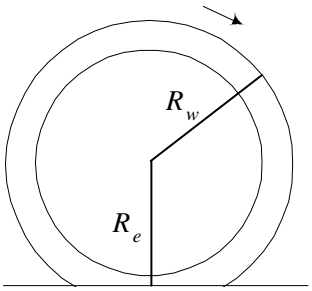


Figura 5.18: Raio efectivo da roda

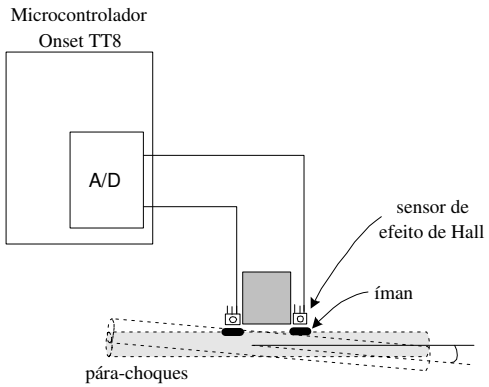


Figura 5.19: Sistema utilizado para detectar colisões frontais

5.4 Rede Local Ethernet

O computador portátil a bordo da cadeira encontra-se ligado à rede local Ethernet 10BaseT do ISR. A ligação pode ser feita através de um cabo UTP ou através de uma ligação sem fio (figura 5.20). Neste último caso, a ligação é estabelecida utilizando um modem (Modulador/Desmodulador) Ethernet ligado à rede fixa e outro modem localizado na cadeira. Nesta configuração básica de modems, existe apenas um ponto de acesso (ponte) [Com98] ligado à rede fixa 10BaseT. Este ponto de acesso faz a ponte entre a rede fixa e os modems que se encontram dentro do seu alcance. A ponte adapta-se à rede, ficando a conhecer os endereços dos computadores de cada um dos lados da fronteira. Esta capacidade permite-lhe segmentar a rede em dois domínios de colisão. A comunicação entre modems não será afectada pelo tráfego da rede fixa, uma vez que o ponto de acesso apenas deixa passar os pacotes da rede destinados aos modems Ethernet.

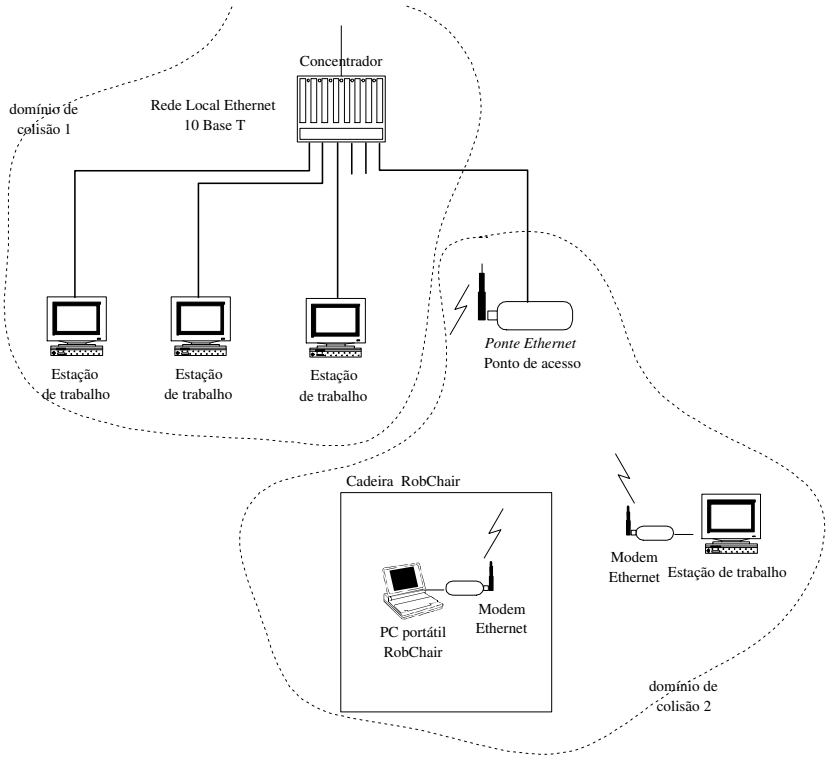


Figura 5.20: Configuração básica para ligação dos modem Ethernet

Capítulo 6

Cadeira de Rodas RobChair: Software

O bom funcionamento de sistemas que exigem execução em tempo real depende fortemente da implementação da arquitectura de *software* e da estrutura de comunicação entre as suas diferentes partes. Este capítulo descreve todo o ambiente de *software* desenvolvido no sistema RobChair.

6.1 Sistema Global

Desenvolveu-se um sistema distribuído constituído por três partes: 1) o programa servidor a correr no microprocessador MC68332 designado por *ARC_Server*; o programa[1] servidor a correr no portátil com sistema operativo Linux designado por *RobChair_Server*; e 3) o programa de reconhecimento de fala *Rec_Voz* a correr num computador pessoal com sistema operativo Windows 95 .

6.2 ARC

Para programar o microprocessador MC68332 utilizou-se o ambiente de desenvolvimento ARC [WSW96]. Foi criado pelos laboratórios Newton Research para programar sistemas embebidos de 32 bits. É constituído pelo compilador de C, arcc, por um programa de inte-

[1] Na verdade são três programas como veremos mais à frente

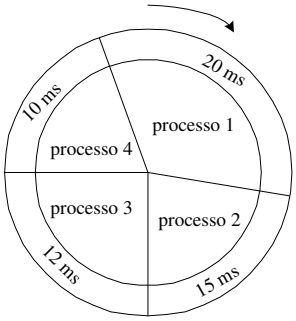


Figura 6.1: Exemplo de funcionamento do sistema multi-tarefa *round-robin* com 4 processos

racção série, arc, que serve para carregar programas para o micro e fazer o sua depuração, e finalmente pelo *Kernel* (sistema operativo do microprocessador). Pode instalar-se em plataformas Linux, SunOS, Solaris, etc. No nosso caso está instalado no sistema operativo Linux.

6.2.1 Sistema de Tempo Real

O *Kernel* é a parte do sistema ARC que corre no processador MC68332. Este contém o sistema operativo e livrarias partilhadas pelos programas do utilizador (programador). Quando se compila um programa, este é ligado a uma imagem do *Kernel*, tendo o programa acesso directo às funções do *Kernel* e ao *hardware*. Para garantir aplicações de tempo real, possui um sistema multi-tarefa que permite gerir 32 processos simultaneamente. Os processos são escalonados tipo *round-robin*, isto é, o sistema executa-os sequencialmente segundo a tabela de processos. A comutação entre processos ocorre quando o tempo atribuído ao processo expira (figura 6.1).

Um processo define-se aqui de uma forma diferente da normalmente utilizada em sistemas operativos Unix. Enquanto no sistema Unix, os processos correspondem a programas separados, no sistema ARC, cada programa pode ter um número arbitrário de processos com espaço de memória partilhada. Um processo é criado através da função `start_process()` cuja sintaxe é:

```
pid start_process(const char *nome_processo, void funcao,
                  int tamanho pilha, int fatia_tempo)
```

em que `pid` é a identificação do processo criado, *nome_processo* é o nome do processo, `funcao` é a função chamada pelo processo, `tamanho_pilha` é o tamanho da pilha em *bytes*, e `fatia_tempo` é o tempo em unidades de base de tempo atribuído ao processo, ou seja, o tempo que decorre entre o instante em que se comuta para o processo e o instante em que este é substituído. Uma unidade de base de tempo corresponde ao período de interrupção e equivale por defeito a 976000 ns ($\simeq 1ms$). O tempo de cada processo pode ser modificado dinamicamente, podendo também criar-se ou anular-se processos durante a execução do programa. Existem duas funções particularmente importantes neste sistema multi-tarefa: `defer()` e `hog_processor()`. A função `defer()` permite que o processo desista do resto do seu tempo e passe para o processo seguinte. A função `hog_processor()` dá ao processo mais 256 unidades de base de tempo. Esta função tem utilidade quando se quer ter a certeza que a tarefa é completada até ao fim. No entanto, há que ter algum cuidado na sua utilização, pois a largura de banda diminui drasticamente levando a limitações no controlo da cadeira em tempo real.

O sistema operativo ARC fornece ainda outras funções, tais como, filas de espera e semáforos. A primeira é útil para comunicação entre processos, e a segunda para garantir exclusão mútua no acesso de alguns recursos (e.g. zonas de memória partilhada e porto série).

6.3 Arquitectura: Servidor ARC_Server

Desenvolveu-se um servidor a correr no micro MC68332, cujas principais funções são a aquisição sensorial, controlo dos motores e comunicação com o servidor *RobChair_Server*. A designação de servidor é talvez um pouco abusiva quando comparada com os sistemas cliente-servidor de outros contextos de arquitecturas distribuídas. Pretende, no entanto, dar a ideia do acesso a vários recursos, mas apenas a um cliente. No sistema Unix, este conceito estende-se a vários clientes simultâneos.

O servidor *ARC_Server* é constituído por seis processos distintos descritos na tabela 6.1.

Processo	Descrição
`le_sensores`	Actualiza o valor das leituras dos sensores de infravermelhos analógicos e digitais, pára-choques e posição do *joystick*
`le_sonares`	Actualiza o valor das leituras dos sensores de ultrassons
`le_codificadores`	Actualiza o valor dos codificadores das rodas esquerda e direita
`envia_pacotes`	Envia pela porta série pacotes com informação sensorial
`le_comando`	Recebe e processa os comandos recebidos pela porta série
`actuador/reactivo`	Executa os comandos de movimento / desvio reactivo de obstáculos

Tabela 6.1: Descrição dos processos do servidor ARC

6.3.1 Comunicação Série

O processo `envia_pacotes` do servidor *ARC_Server* envia os pacotes de dados a intervalos fixos (sincronamente). Esse intervalo corresponde ao ciclo completo de controlo e é a partir dele que se calcula a largura de banda do sistema. A comunicação entre este processo e os processos de actualização dos sensores faz-se através de variáveis globais. A figura 6.2 ilustra a comunicação entre processos e a ligação série com o servidor *RobChair_Server*. O processo `le_comandos` recebe sincronamente comandos provenientes do servidor *RobChair_Server*, os quais podem ser de condução ou pedidos de informação. A resposta aos comandos que exijam confirmação e aos pedidos de informação é também enviada sincronamente. A seguir apresenta-se em pseudo-código este processo:

```
// processo le_comando

/* Variáveis globais */
int vel_linear, vel_rot;

le_comando:

    comando = processa_comando(ligacao serie);
```

```
    switch (comando){
        case VEL_LINEAR_ACK:
                valor = processa_comando(ligacao serie);
                vel_linear=valor;
                envia(ACK);
                break;

        case VEL_ROT:
                valor = processa_comando(ligacao serie);
                vel_rot=valor;
                break;

            ...

        case PEDE_VELMAX:
                envia(vel_max);
                break;

        case DESLIGA:
                kill_process(pid);
                break;
    }
fim le_comando;
```

O formato dos comandos e respostas está representado na tabela 6.2. O primeiro *byte* é um caracter de sincronismo, o segundo *byte* corresponde ao tipo de comando/resposta, e o terceiro *byte* ao valor associado ao comando/resposta.

Depois de processado o comando, a variável correspondente é actualizada e utilizada pelo processo `actuador` ou pelo processo `reactivo`. Estes processos são utilizados consoante o modo de operação do servidor *ARC_Server*. Os modos são os seguintes:

- **Isolado -** Neste modo, o algoritmo de navegação corre no servidor *ARC_Server*, ou seja, as acções de controlo não dependem do servidor *RobChair_Server*. Este

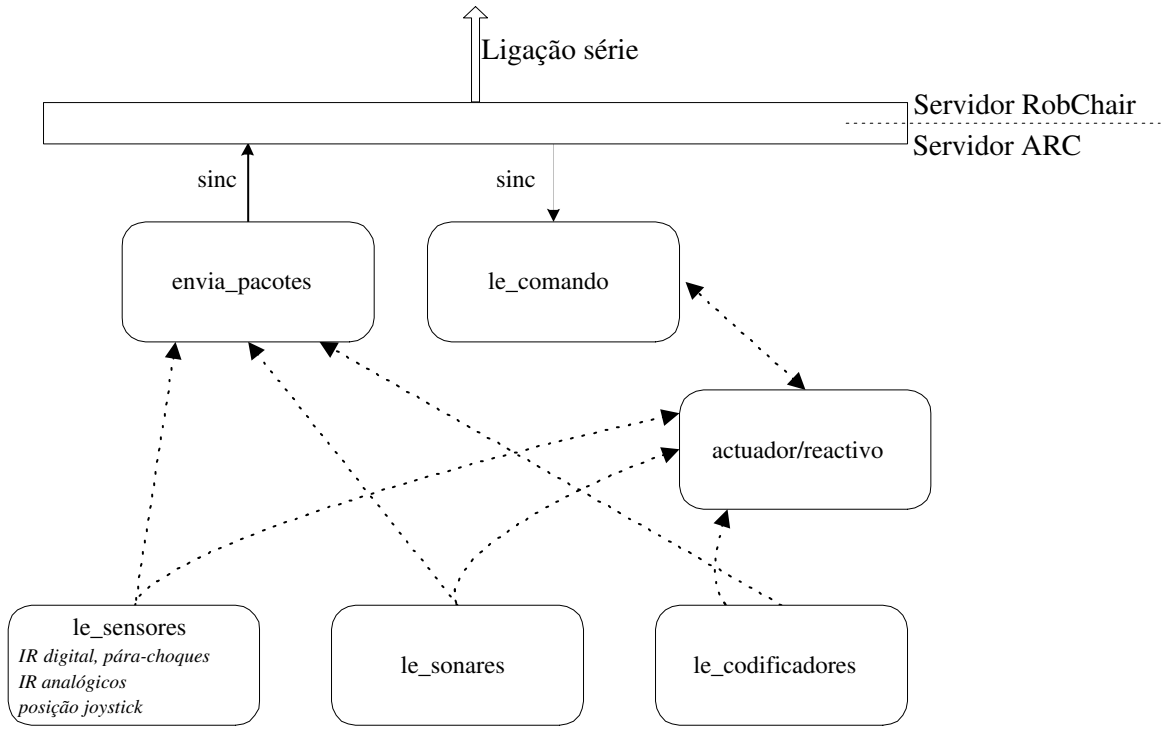


Figura 6.2: Comunicação entre processos e ligação série

modo foi utilizado para implementar algoritmos reactivos simples para desvio de obstáculos. A sua utilização funciona com o processo reactivo.

- **Actuador -** As acções de controlo são calculadas pelo algoritmo de navegação executado no servidor *RobChair_Server* (descrito no capítulo 8). Utiliza-se o processo `actuador` para executar as acções de controlo recebidas. Este processo é também responsável pela detecção de colisões, através da verificação do estado dos pára-choques. Este é o modo actualmente utilizado.

6.3.2 Aquisição Sensorial

As leituras dos sensores são actualizadas por três processos: `le_sensores`, `le_sonares` e `le_codificadores`. O primeiro processo actualiza as leituras dos sensores de infravermelhos digitais e analógicos, do pára-choques e da posição do *joystick*. O segundo actualiza as leituras dos sensores de ultrassons e o terceiro actualiza a informação dos codificadores das rodas. Esta informação é empacotada e enviada ao servidor *RobChair_Server* pelo processo `envia_pacotes` a cada intervalo de tempo definido pela variável `intervalo_pacote`.

SINC	tipo	valor

Tabela 6.2: Formato dos comandos e respostas

O formato dos pacotes pode observar-se na tabela 6.3. A seguir apresenta-se em pseudo-código o processo `envia_pacotes`:

```
/* Variáveis globais */
int tempo_anterior;

const intervalo_pacote;

envia_pacote:

    espera_intervalo:
        while (tempo=tempo_actual - tempo_anterior  < INTERVALO_PACOTE){
            nada;
        }

    envia(SINC);
    envia(TIPO);
    envia(tempo);
    envia(sensores);
    envia(sonares);
    envia(codificadores);

fim envia_pacote;
```

6.4 Arquitectura: Servidor RobChair

O servidor *RobChair_Server* inicia através de três programas: *Serie*, *Difuso* e *Gui_Chair* (figura 6.3). O programa *Serie* é responsável pela gestão do fluxo de comunicação de entra-

nº *bytes*	1	1	2	2	24
	SINC	tipo	tempo	IR digital e pára-choques	IR analógicos

12	4
sonares	codificadores

Tabela 6.3: Formato do pacote com sensores

da/saída com o servidor *ARC_Server*. Este recebe sincronamente do servidor *ARC_Server*, os pacotes de dados com a informação sensorial e transmite também sincronamente os comandos de controlo. O módulo de navegação corre no programa *Difuso*. No nosso caso, consiste num controlador de navegação reactiva baseado em lógica difusa (ver capítulo 8). O programa *Gui_Chair* é uma interface gráfica utilizada para visualização e controlo remoto da cadeira. Encontra-se também ligado ao programa *Rec_Voz* executado numa estação com sistema operativo Windows 95, sendo responsável pela captação e reconhecimento de voz do utilizador.

6.4.1 Comunicação entre Processos

A comunicação entre os três processos do servidor faz-se através de memória partilhada (figura 6.3). Esta corresponde ao mapeamento de uma área (segmento) de memória podendo ser mapeada e partilhada por mais do que um processo. Este é, sem dúvida, o método mais rápido de comunicação inter-processo, pois faz-se de forma directa sem serem necessários canais de comunicação. O segmento pode ser criado por um processo, e de seguida escrito e lido por qualquer número de processos.

No servidor *RobChair_Server* criaram-se 2 segmentos de memória (figura 6.4): `sens_mem` e `comando_mem`. O processo *Serie* escreve no segmento de memória, `sens_mem`, os dados contidos nos pacotes de informação recebida pela ligação série. Esta informação é lida pelos processos *Difuso* e *Gui_Chair* a intervalos regulares. O processo *Gui_Chair* lê a informação sensorial cada 500 ms através de um temporizador (alarme accionado periodicamente), enquanto o processo *Difuso* lê a informação cada ciclo de controlo (sensivelmente 40 ms sem utilização dos sonares). Quando o algoritmo difuso calcula uma acção de controlo, esta é escrita no segmento de memória `comando_mem` e uma bandeira marca o comando como novo (a enviar). O programa *Gui_Chair* actua da mesma forma quando é solicitada uma acção de controlo pelo operador remoto. Em cada ciclo de controlo, o

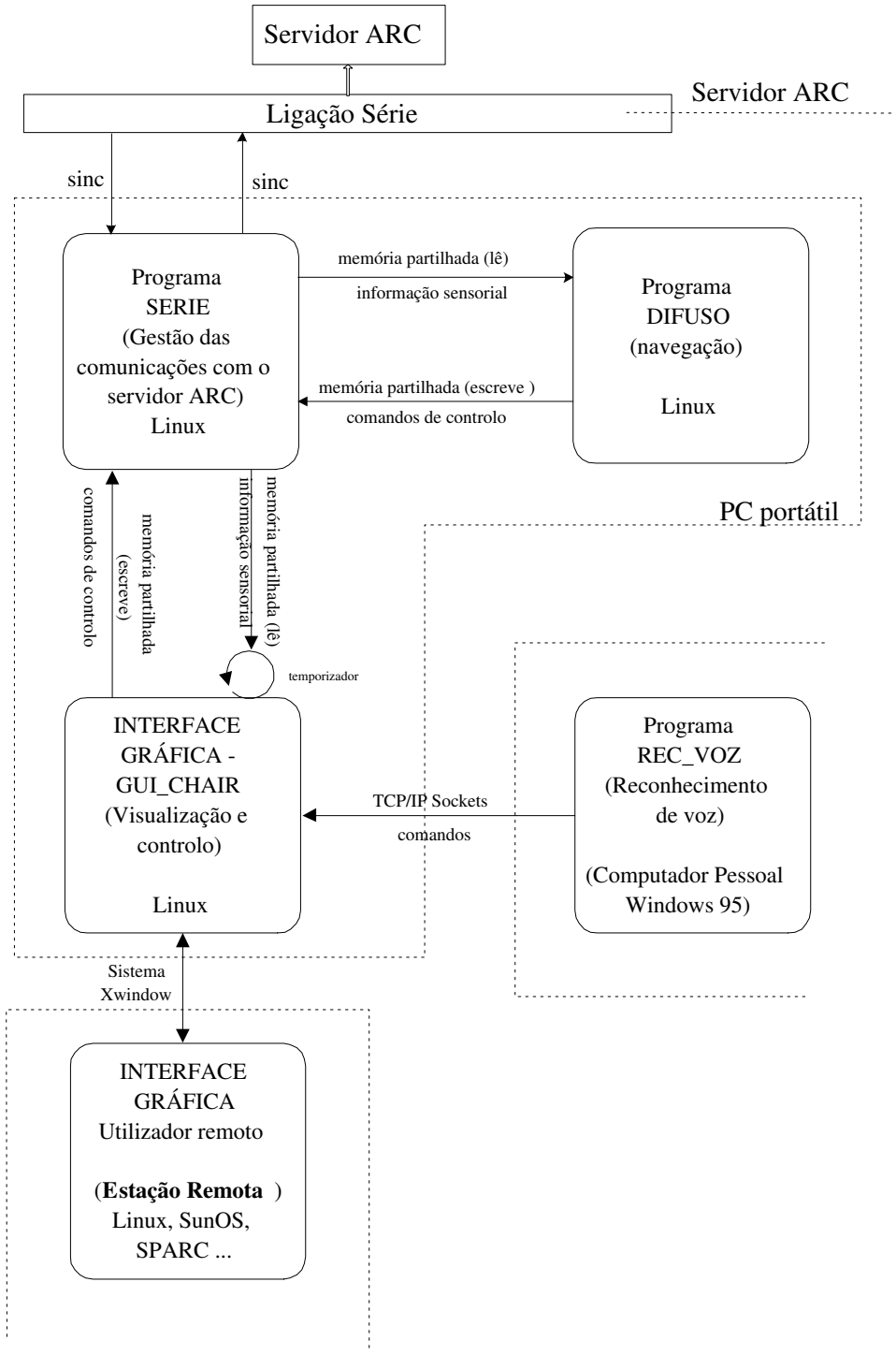


Figura 6.3: Comunicações entre os diferentes processos do sistema RobChair

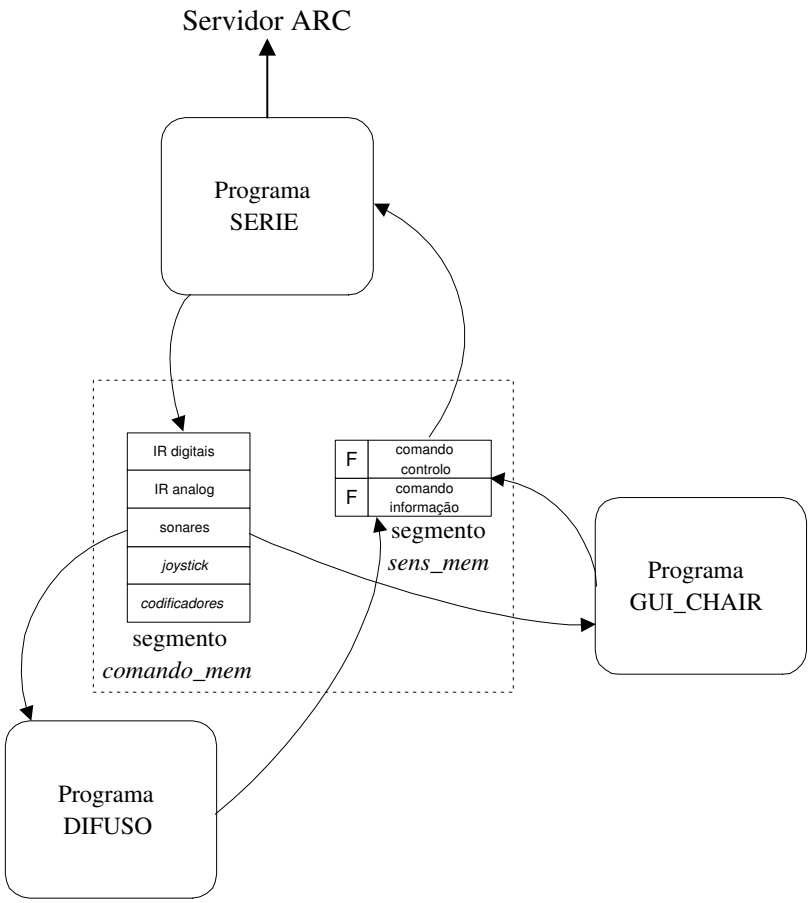


Figura 6.4: Segmentos de memória criados para inter-comunicação de processos. O segmento de memória *sens_mem* é constituído por duas regiões: região onde é escrito o comando e região onde é escrita a bandeira (F)

programa *Serie* verifica se existem comandos novos para enviar. Caso existam, lê-os e transmite-os pela porta série, alterando o estado da bandeira do segmento de memória.

6.4.2 Interface Gráfica (Gui_Chair)

A interface gráfica foi inicialmente criada para fornecer ao programador uma forma transparente de receber e visualizar dados sensoriais, e enviar comandos de controlo à cadeira. Tornou-se uma ferramenta muito útil para testar e fazer a depuração dos algoritmos de controlo. É constituída essencialmente por dois blocos:

1. Tela de desenho que permite a visualização gráfica a duas dimensões (2D) do ambi-

ente de operação da cadeira. Os movimentos da cadeira são visualizados em tempo real, ao mesmo tempo que as leituras dos sensores são visualizadas. Podem-se construir mapas que representam ambientes domésticos reais utilizando estruturas geométricas, tais como rectas e círculos (figura 6.5).

2. *Joystick* gráfico manipulado através do rato (figura 6.6) que permite enviar comandos de direcção à cadeira.

A interface gráfica oferece um conjunto de vantagens, designadamente as seguintes:

1. **Teste aos algoritmos de navegação -** a visualização em tempo real dos movimentos da cadeira, acompanhada da informação da leitura dos sensores fornece ao programador um mecanismo para avaliar o comportamento e a eficiência dos algoritmos de navegação, facilitando a sua afinação.

2. **Modelos do mundo -** a recriação de ambientes reais, com informação geométrica, constitui uma forma de conhecimento que poderá ser utilizada para planeamento local e global de trajectórias.

3. **Efectuar tarefas de tele-operação -** pode ser um módulo de base para implementação de um sistema de tele-operação (com inclusão de câmaras).

4. **Ambiente de simulação -** esta capacidade não faz ainda parte do sistema RobChair actual. No entanto, se incluirmos o modelo dinâmico da cadeira, é possível utilizar a interface gráfica para simular os algoritmos de navegação. A simulação é uma ferramenta extremamente útil pois permite executar e testar os algoritmos de forma repetida e rápida, sem consequências desastrosas no caso de erros algorítmicos ou de programação.

Implementação da Interface Gráfica - Gui_Chair

Para implementar a interface gráfica, usou-se o *software* XFORMS [ZO96]. Trata-se de um pacote de ferramentas utilizado para construir aplicações gráficas a 2D no sistema X Window. É constituído pela livraria FORMS e por um construtor de FORMS (uma interface gráfica que permite desenhar o ambiente gráfico do utilizador). A livraria FORMS é uma livraria de rotinas C, de alto nível, que apenas utiliza os serviços fornecidos pela

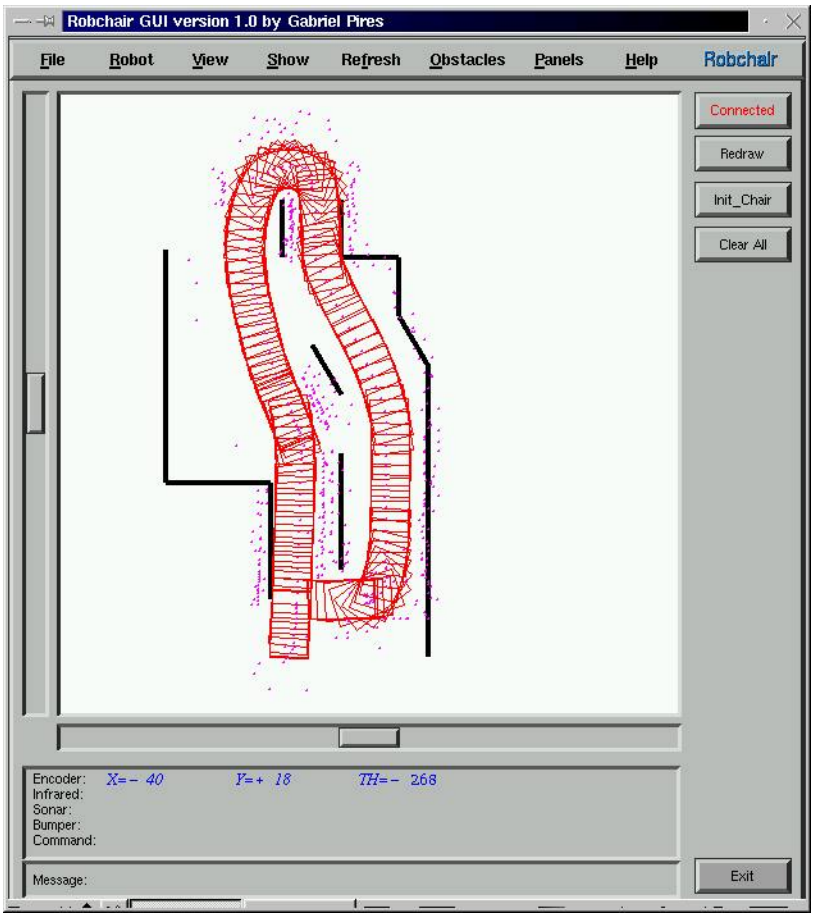


Figura 6.5: Interface gráfica utilizada para visualização dos movimentos da cadeira e de dados sensoriais

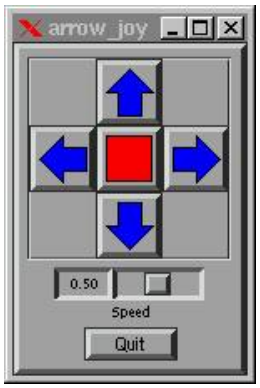


Figura 6.6: *Joystick* gráfico utilizado para comando da cadeira

livraria Xlib. Esta livraria corresponde ao nível mais baixo de linguagem C de interface como o sistema X. A sua principal função consiste em traduzir as estruturas de dados C e procedimentos em eventos X. O sistema X Window é introduzido na sub-secção seguinte.

O pacote XFORMS permitiu construir, com relativa facilidade, botões, painéis, movimentos da janela, menus, e uma tela para desenho 2D. A tela de desenho é a parte central da interface gráfica. É nela que se visualizam os movimentos da cadeira e as leituras dos sensores e também onde são criados os mapas do ambiente.

Sistema X Window: comunicações

O sistema X Window, ou simplesmente X, permite controlar cada píxel do ecrã, aceitando eventos do rato ou do teclado. Divide o ecrã em janelas que funcionam como terminais virtuais. Contrariamente a outros sistemas de janelas, baseia-se num protocolo de rede assíncrono (protocolo X), em vez de chamadas de procedimentos [Nye90]. Esta característica traz-lhe as seguintes vantagens:

- Tanto as ligações locais como as ligações em rede são efectuadas da mesma forma através do protocolo X. Logo, as ligações em rede são transparentes para o utilizador e programador.

- Pode ser implementado em arquitecturas e sistemas operativos diferentes tornando-o independente dos dispositivos.

O protocolo X funciona segundo uma arquitectura cliente/servidor cujo significado é diferente de outros contextos de computação. Para o sistema X, o servidor é o *software* que gere a visualização, o teclado e o rato. O cliente é o programa visualizado no ecrã e que aceita comandos do rato e teclado (figura 6.7). O cliente e o servidor podem estar na mesma máquina ou em máquinas separadas da rede. Os clientes implementam o protocolo X através de livrarias de programação tais como a livraria Xlib (de baixo nível) ou por exemplo a livraria Xforms (de mais alto nível). A ligação cliente/servidor utiliza os canais de comunicação inter-processo, tais como memória partilhada e os sockets da Berkeley. A interface gráfica do servidor RobChair serve-se dos mecanismos disponibilizados pelo sistema X Window. O programa *Gui_Chair* (cliente) arranca no computador portátil, mas é visualizado numa estação remota (servidor) com o sistema X Window. Para que a

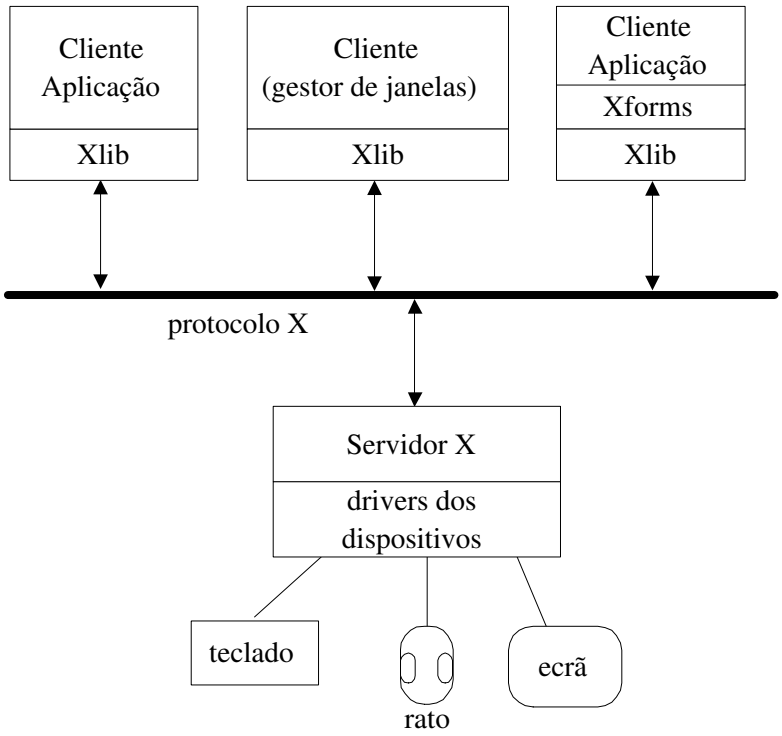


Figura 6.7: Sistema X Window (protocolo X)

estação remota aceite visualizar a interface gráfica dos programas cliente basta digitar na *shell*:

xhost +máquina

No computador portátil é necessário indicar qual a estação que irá receber a interface gráfica:

setenv DISPLAY [hospedeiro]:ecran

Comunicação entre os Processos Rec_Voz e Gui_Chair

A comunicação entre o programa *Gui_Chair* e o programa *Rec_Voz* realiza-se através de *sockets*. Os *sockets* são uma interface entre os programas de aplicação (API) e os protocolos de comunicação. Permitem desenvolver aplicações de rede baseadas no protocolo TCP/IP. Do lado Unix (Linux), utilizaram-se os *sockets* da Berkeley, e do lado Windows 95, utilizaram-se os *Winsock* (criados também a partir dos Sockets da Berkeley) (a figu-

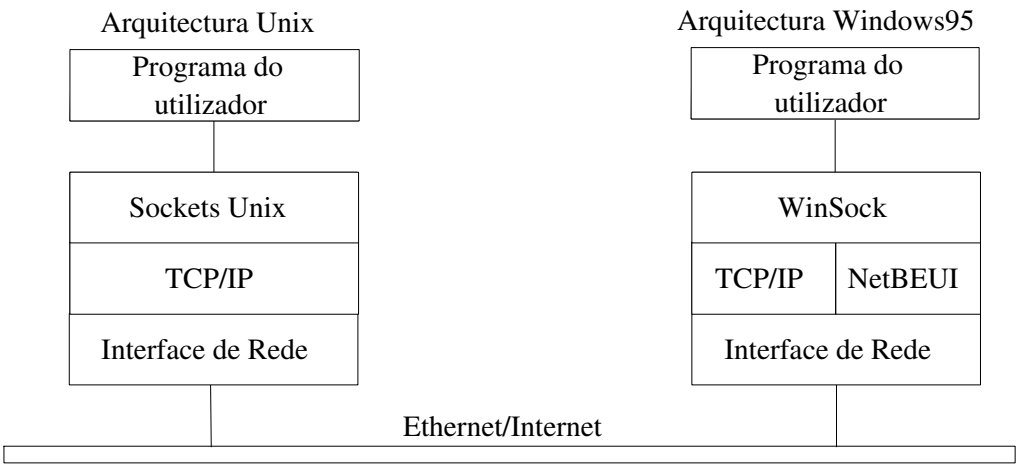


Figura 6.8: Os sockets na arquitectura Unix e Windows 95

ra 6.8 ilustra de forma simplificada as duas arquitecturas). Inicialmente, a arquitectura Windows 95 possuia apenas o protocolo de rede NetBEUI. No entanto, como este protocolo é não encaminhável (limita-se a um único segmento da rede), a Microsoft decidiu adicionar o protocolo TCP/IP recorrendo ao API *Winsock*. Neste momento, todas as aplicações de rede da arquitectura Windows 95 recorrem aos WinSock, ou directamente, em aplicações como `telnet`, ou indirectamente, através de NetBIOS em cima de TCP/IP para aplicações como "vizinhança de rede".

No sistema RobChair, implementaram-se sockets do tipo *stream* que utilizam o protocolo TCP na camada de transporte. Este protocolo bastante fiável fornece um fluxo de dados bi-direccional, sequencial e sem duplicação.

O desenvolvimento de aplicações de rede passa pela chamada de um conjunto de procedimentos entre os quais se destaca o `socket, bind, close, send, recv`. Para iniciar a rede, o comando `socket` deve ser chamado. Este comando define a família protocolar, o tipo de comunicação e o protocolo específico que no nosso caso são respectivamente Internet, orientado à ligação, e TCP. De seguida, o comando `bind` é utilizado para definir o endereço IP e o porto de comunicação (TCP). Os comandos `send` e `recv` servem para transmitir e receber dados, respectivamente. Para desligar a comunicação chama-se o comando `close`.

6.5 Servidor de Reconhecimento de Voz

O reconhecimento automático de fala (RAF) é o processo pelo qual a fala é traduzida em texto ou símbolos. O primeiro sistema RAF data dos anos 50. Foi desenvolvido pelos laboratórios AT&T Bell e conseguia reconhecer dígitos ingleses de 0 a 9. Os sistemas RAF têm uma vasta gama de potenciais aplicações, designadamente em sistemas de segurança, acesso a serviços telefónicos, interfaces multi-modais utilizador-computador e novas interfaces para deficientes. Se atendermos à quantidade de padrões de voz existente para pronunciar uma mesma palavra e à forma como articulamos as palavras nas frases, rapidamente nos apercebemos da dificuldade em criar algoritmos de reconhecimento de fala. Os sistemas são geralmente classificados em dependentes/independentes do locutor, e de discurso discreto/contínuo [Lee89]. Segue-se a distinção entre um sistema dependente e independente do locutor:

Independente do locutor - é capaz de reconhecer voz de qualquer locutor independentemente da sua idade e sexo ou tom de voz. São sistemas muito difíceis de desenvolver na medida em que as representações da voz são muito dependentes do locutor e um conjunto de padrões adequados a um locutor têm fraco desempenho para outros. Por isso, os modelos de voz são criados com base em amostras de centenas de locutores. No entanto, se o vocabulário for muito extenso (muitas palavras), o número de modelos é extremamente grande. Este sistema é adequado para aplicações em que existem muitos utilizadores simultâneos, como é o caso do acesso a serviços telefónicos.

Dependente do locutor - dada a dificuldade de implementação do sistema anterior, a maioria dos sistemas são dependentes do locutor, ou seja, requerem que o utilizador (locutor) treine o sistema. O treino consiste na repetição por parte do locutor de um conjunto de palavras do vocabulário disponível. Desta forma, a eficácia do sistema RAF aumenta drasticamente, mas apenas poderá ser utilizado por um utilizador de cada vez. Para além desta desvantagem, a sessão de treino pode revelar-se incómoda e muito morosa, havendo também o risco da fala do utilizador variar devido a "stress", fadiga ou doença.

Quanto ao tipo de discurso aceite pelo sistema, teremos:

Discurso contínuo - o reconhecimento de fala contínua é significativamente mais difícil do que de palavras isoladas. Primeiro, porque os limites de cada palavra não são claros. Segundo, porque os efeitos co-articulares são mais fortes no discurso contínuo, ou seja, os fonemas são influenciados pelos fonemas anteriores e posteriores. Este efeito ocorre entre palavras.

Discurso discreto (palavras isoladas) - os utilizadores devem proferir as palavras isoladamente, ou seja, com pausas entre elas. Os limites de cada palavra são bem conhecidos o que favorece a exactidão do reconhecimento e limita a procura.

Associados aos sistemas RAF existe normalmente um conjunto de conceitos (alguns deles já introduzidos) que importa aqui definir [Dra95] :

Vocabulário - conjunto de palavras que o sistema consegue reconhecer.

Modelo de fala - representação digital de uma palavra ou frase falada. Cada palavra ou frase do vocabulário deve ter associado um modelo de fala para o sistema reconhecer. Quando o utilizador pronuncia uma palavra, o sistema compara os dados da expressão de entrada com os modelos existentes para fazer a correspondência.

Reconhecimento - processo pelo qual o sistema compara uma expressão do utilizador com o modelo e devolve a palavra pronunciada.

Treino - processo pelo qual uma aplicação recolhe amostras de fala com o propósito de criar e adaptar modelos de fala. Para obter um elevado nível de eficiência em modelos independentes de locutor é sempre aconselhável fazer um treino ou adaptação para cada indivíduo.

Adaptação de fala - processo pelo qual se constrói ou melhora um modelo de fala (muitas vezes por processos de realimentação por intervenção do utilizador).

6.5.1 Pacote de *software* Dragon

O servidor *Rec_Voz* utiliza o pacote de *software Dragon Voice Tools*, um produto comercial da $Dragon^{TM}$ que permite desenvolver aplicações em ambiente DOS e Microsoft Windows. Este *software* suporta reconhecimento de frases ou palavras pronunciadas de

forma discreta e dígitos (zero a nove) falados de forma contínua. Possui modelos independentes do locutor (para a língua inglesa) e permite construir modelos dependentes do locutor.

Os principais componentes do *Dragon Voice Tools* são [Dra95]:

Driver de fala (Dragon Speech Driver (DSD))- programa que gere o interface entre o utilizador e o programa de aplicação, ou seja, trata do processamento e da conversão da voz humana em dados que possam ser utilizados pela aplicação.

Livrarias de interface (Speech Drive API)- funções de interface em linguagem C que podem ser ligadas ao código do programa para controlar o DSD.

Gramática de estados finitos (Finite State Grammar (FSG))- compilador que cria um ficheiro de vocabulário utilizável pela aplicação. O FSG lê um ficheiro que contém as palavras ou frases que se pretendem reconhecer.

Modelos independentes do locutor - representações digitais da fala inglesa de centenas de pessoas. O sistema contém modelos de fala discreta independente do locutor de 110.000 das palavras inglesas mais faladas. Inclui também modelos de fala contínua independente do locutor para dígitos de zero a nove.

Em termos de dispositivos de *hardware*, o funcionamento do *software* apenas necessita de uma placa de som Sound Blaster16 e um microfone.

6.5.2 Aplicação Rec_Voz

O servidor *Rec_Voz* baseia-se numa aplicação para Windows desenvolvida no ISR por um grupo de projecto [LH97], no âmbito da disciplina de projecto e dissertação de licenciatura em Engenharia Electrotécnica, com a colaboração do autor desta dissertação. Esta aplicação foi inicialmente desenvolvida para controlar um robô simulado, tendo sido posteriormente aproveitada para controlo da cadeira RobChair. Os comandos resultantes do processo de reconhecimento são enviados via TCP/IP (utilizando os *WinSock* API) para o servidor RobChair (programa *Gui_Chair*).

A figura 6.9 ilustra de forma simplificada a sequência de eventos executados pela aplicação:

1. O utilizador fala para o microfone;
2. A placa de som converte o sinal eléctrico analógico para a forma digital;
3. O DSD coloca a expressão digital numa fila de espera;
4. O programa de aplicação pede ao DSD para começar o reconhecimento a partir da lista de vocabulário a utilizar pela aplicação;
5. O DSD compara a expressão digital com os modelos activos de fala;
6. O sistema devolve a palavra cujo modelo mais se assemelha com a expressão de entrada (sendo posteriormente enviada ao servidor RobChair);
7. No caso de ainda não existirem modelos da lista de palavras do vocabulário, o utilizador deve proceder ao treino das palavras.

O vocabulário da aplicação *Rec_Voz* consiste numa lista de comandos para controlar a cadeira em português. Uma vez que os modelos independentes do locutor enviados pela Dragon só existem para a língua inglesa, o utilizador deve treinar o conjunto de comandos, repetindo três vezes cada uma das palavras. Os modelos são gravados num ficheiro de utilizador. Depois de criados os modelos, cada utilizador, ao iniciar a aplicação, carrega o respectivo ficheiro. A figura 6.10 mostra a interface da aplicação desenvolvida onde aparece um conjunto de palavras utilizadas para controlar a cadeira.

Eficácia

Em [LH97] foram realizados alguns testes provar a eficácia do reconhecimento de voz. Os testes foram realizados por seis vozes masculinas e quatro femininas em ambientes não ruidosos, ruidosos e com alteração emocional da voz. Para um vocabulário de nove palavras obtiveram-se os seguintes resultados:

- Em ambiente não ruidoso, a eficiência de reconhecimento foi de 100%.
- Em ambiente ruidoso, a eficiência de reconhecimento caiu apenas para 97,8%.
- A alteração emocional da voz dos utilizadores levou a uma eficiência variável de utilizador para utilizador. Um teste específico de voz nasalada revelou que a eficiência não sofreu grandes alterações para nenhum dos utilizadores.

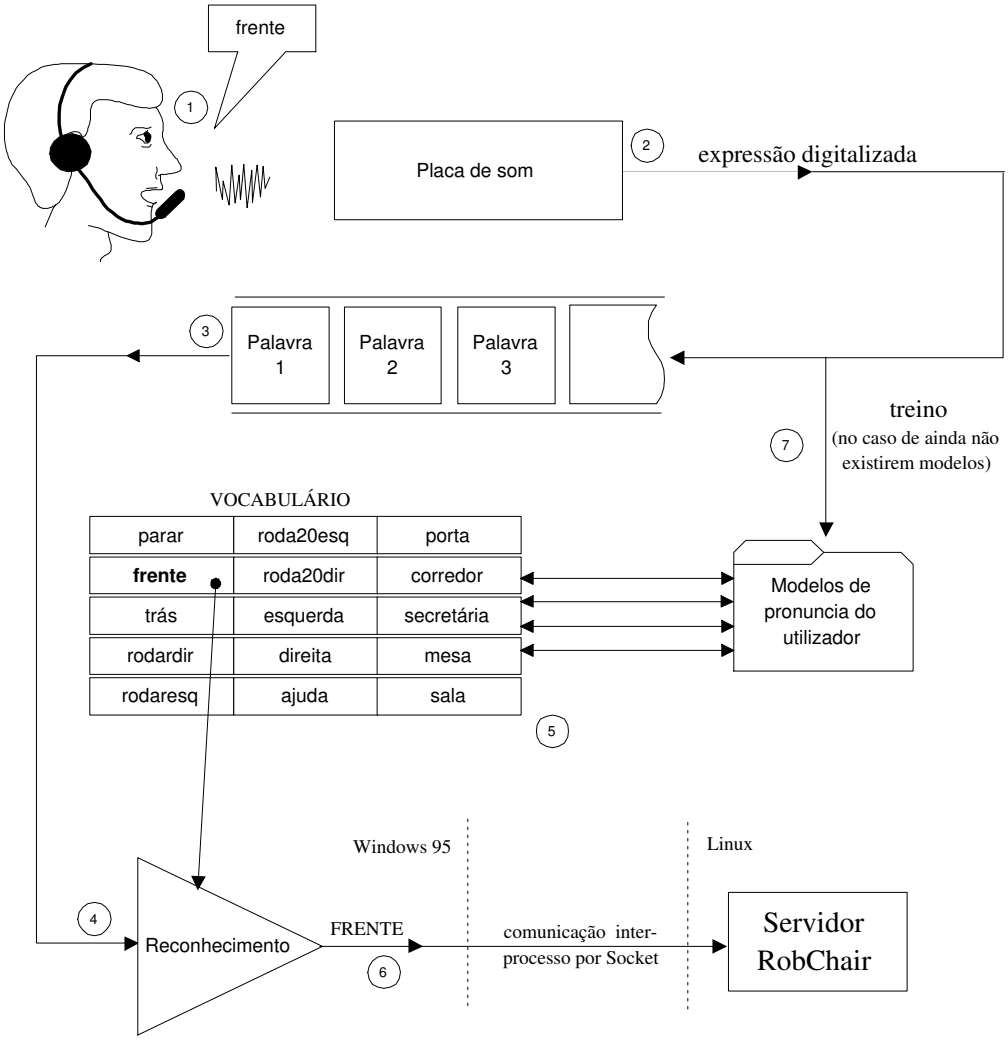


Figura 6.9: Sequência de eventos do sistema de reconhecimento

Figura 6.10: Imagem da aplicação *Rec_Voz* apresentando o conjunto de comandos para condução da cadeira

Note-se, no entanto, que a elevada eficiência prestada pela aplicação deve-se ao número limitado de palavras do vocabulário, assim como à acentuada diferença de pronúncia entre essas palavras. Uma limitação da aplicação (e do sistema) prende-se com o facto deste devolver a palavra cujo modelo mais se assemelha ao comando (mesmo que o modelo exacto não exista). Se, por exemplo, um ruído ambiente for identificado como uma expressão (comando), ou se for pronunciada uma palavra não existente no vocabulário, o sistema devolve uma palavra que, obviamente, não é a correcta. Uma forma de ultrapassar este efeito do ruído passa pela inclusão no vocabulário de um conjunto de ruídos típicos e treiná-los como se palavras se tratassem.

Capítulo 7

RobChair: Arquitectura, Funcionamento e Cinemática

Depois de apresentar os objectivos do sistema RobChair, este capítulo descreve os seus modos de funcionamento e a sua arquitectura de navegação. Pormenoriza-se a camada reactiva uma vez que foi esta a camada implementada. Descreve-se ainda o sistema sensorial e o modelo cinemático da cadeira.

7.1 Objectivos

O sistema RobChair pretende ultrapassar algumas das limitações inerentes às cadeiras de rodas convencionais. Definimos inicialmente os seguintes objectivos primordiais: garantir a acessibilidade da cadeira a um leque mais alargado de potenciais utilizadores; minimizar o esforço do utilizador na condução da cadeira; e assegurar a integridade física do utilizador bem como do próprio sistema. Detalhemos estes três items:

Interface Homem-Máquina (IHM) - o utilizador do sistema RobChair possui duas IHM para conduzir a cadeira: o *joystick* para indivíduos sem problemas nos membros superiores ou com problemas não muito graves (por exemplo, tremura pouco acentuada); e o reconhecedor de fala (descrito na secção 6.5.2) para indivíduos incapazes de utilizar dispositivos de controlo manual (por exemplo, um *joystick*). Contudo, estas duas interfaces não solucionam o problema da condução da cadeira. Os cenários seguintes ilustram algumas das limitações remanescentes:

1. Em ambientes domésticos ou de escritório, o espaço é normalmente muito reduzido e confuso. No caso da cadeira ser controlada através de *joystick* por indivíduos que não conseguem manipular com precisão este dispositivo, a condução torna-se uma tarefa extremamente difícil, comprometendo a segurança do utilizador. Mesmo para indivíduos sem problemas motores, a tarefa não é fácil e sujeita a falha humana.

2. No caso de controlo por voz, os problemas referidos no ponto anterior são ainda mais acentuados. Enquanto que o controlo por *joystick* é contínuo, o controlo por voz faz-se de uma forma discreta. Em ambientes mesmo de baixa complexidade, o utilizador necessitará de fornecer inúmeros comandos (de baixo nível) para realizar simples movimentos. Este facto torna pouco cómoda a utilização da cadeira.

Minimização do esforço do utilizador - o ponto anterior mostrou que as IHM não bastam, por si só, para controlar eficientemente a cadeira. Restam duas soluções:

1. Criar um sistema de controlo partilhado utilizador/máquina. Por outras palavras, o sistema deverá aceitar comandos do utilizador enquanto um algoritmo de desvio local de obstáculos deverá dar origem a manobras de correcção e suavização para seguir o comando do utilizador. Assim, mesmo que o utilizador não forneça o comando mais indicado, o módulo de desvio de obstáculos tem por responsabilidade corrigir o comando dado e fazer com que a cadeira siga a trajectória mais adequada. Desta forma, o utilizador não precisa de controlar o *joystick* com elevada precisão, ou, no caso da IHM por voz, o utilizador necessitará de fornecer um número mais reduzido de comandos.

2. Criar um sistema de navegação que responda autonomamente a comandos de alto nível do utilizador.

Segurança - é garantida pelo módulo de desvio de obstáculos e pelo módulo de detecção de colisões.

7.2 Arquitectura de Controlo

A arquitectura de controlo do sistema de navegação RobChair encontra-se representada na figura 7.2 (diagrama de blocos) e na figura 7.1 (diagrama de camadas). Trata-se de uma arquitectura híbrida de cinco camadas [NPC00]: 1) a camada de controlo é responsável pelos movimentos da cadeira ; 2) a camada reactiva é composta por três comportamentos reactivos; 3) a camada de acção localizada executa tarefas específicas no espaço local do veículo; 4) a camada de raciocínio deliberativo realiza planeamento de alto nível; e 5) a camada de missão, define os objectivos do sistema de navegação.

Propusemos-nos, nesta tese, implementar os módulos de navegação respeitantes às camada 1) e 2). As restantes camadas da arquitectura estão em desenvolvimento no ISR - Coimbra. Segue-se, resumidamente, a descrição das camadas:

Camada de Controlo - traduz os comandos de movimento em acções de controlo para os actuadores. Se houver planeamento local de trajectória, a camada de controlo é também responsável pelo seguimento de trajectória (não implementado no nosso caso).

Camada Reactiva - implementa um controlo partilhado entre utilizador e cadeira. Consiste em dois comportamentos reactivos: *desvio inteligente de obstáculos* e *seguimento de superfícies*. Estes comportamentos baseiam-se apenas em informação actual dos sensores, ou seja, sem recorrer a modelos do ambiente. Esta camada é guiada pelas camadas superiores, que fornecem os objectivos a seguir. Estes assumem essencialmente a forma de comandos de velocidade (linear e angular), mas também de informação cognitiva.

Camada de Acção Localizada - camada intermédia que actua a dois níveis: 1) planeamento de trajectórias locais para desvio de obstáculos (trajectórias óptimas e suaves); 2) execução de tarefas pré-definidas no contexto local do veículo. Duas dessas tarefas, consistem no planeamento de manobras para passagem de portas e aproximação a secretárias ou mesas[1]. Estas tarefas pressupõem um conjunto de acções pré-definidas.

A implementação desta camada depende obrigatoriamente da aquisição de infor-

[1] Estas tarefas são consideradas de elevada complexidade para o utilizador

mação sensorial local na forma de mapas (memória de curto prazo), que poderá ser integrada com informação *a priori* proveniente das camadas superiores (mapas globais).

Camada Deliberativa/Cognitiva - baseia-se em conhecimento do mundo adquirido *a priori*. Este toma a forma de mapas topológicos e mapas geométricos dando a informação da deslocação entre diferentes pontos de um determinado espaço (planeamento global de caminhos). A informação adquirida a *priori* é actualizada e integrada com mapas locais adquiridos no momento.

Camada de Missão - o utilizador ou outro operador humano definem os objectivos do sistema. É uma camada de funções bem definidas e de especial relevância, na medida em que o utilizador é parte integrante do sistema. Este poderá intervir a vários níveis, desde o controlo directo por comandos de velocidade e direcção, a comandos do tipo *vai para cozinha* ou ainda comandos de informação do tipo *a porta encontra-se fechada*.

A arquitectura de navegação RobChair define assim dois modos de funcionamento (figura 7.1): modo semi-autónomo, quando a navegação depende da intervenção do utilizador a vários níveis; e modo inteiramente autónomo, quando o utilizador apenas intervém fornecendo comandos de alto nível. Este último modo também inclui as capacidades do modo semi-autónomo.

7.3 Módulo de navegação Reactiva

A camada reactiva do sistema RobChair é constituída por três comportamentos: *desvio inteligente de obstáculos*, *detecção de colisões* e *seguimento de superfícies* (figura 7.3) [PANA98]:

Desvio inteligente de obstáculos - este comportamento da camada reactiva resulta da fusão entre o comportamento de *desvio de obstáculos* e o comportamento de *seguimento de meta*. Este comportamento é detalhado a seguir.

Detecção de colisões - este comportamento é muito simples e consiste na detecção de colisões com base na informação do pára-choques dianteiro.

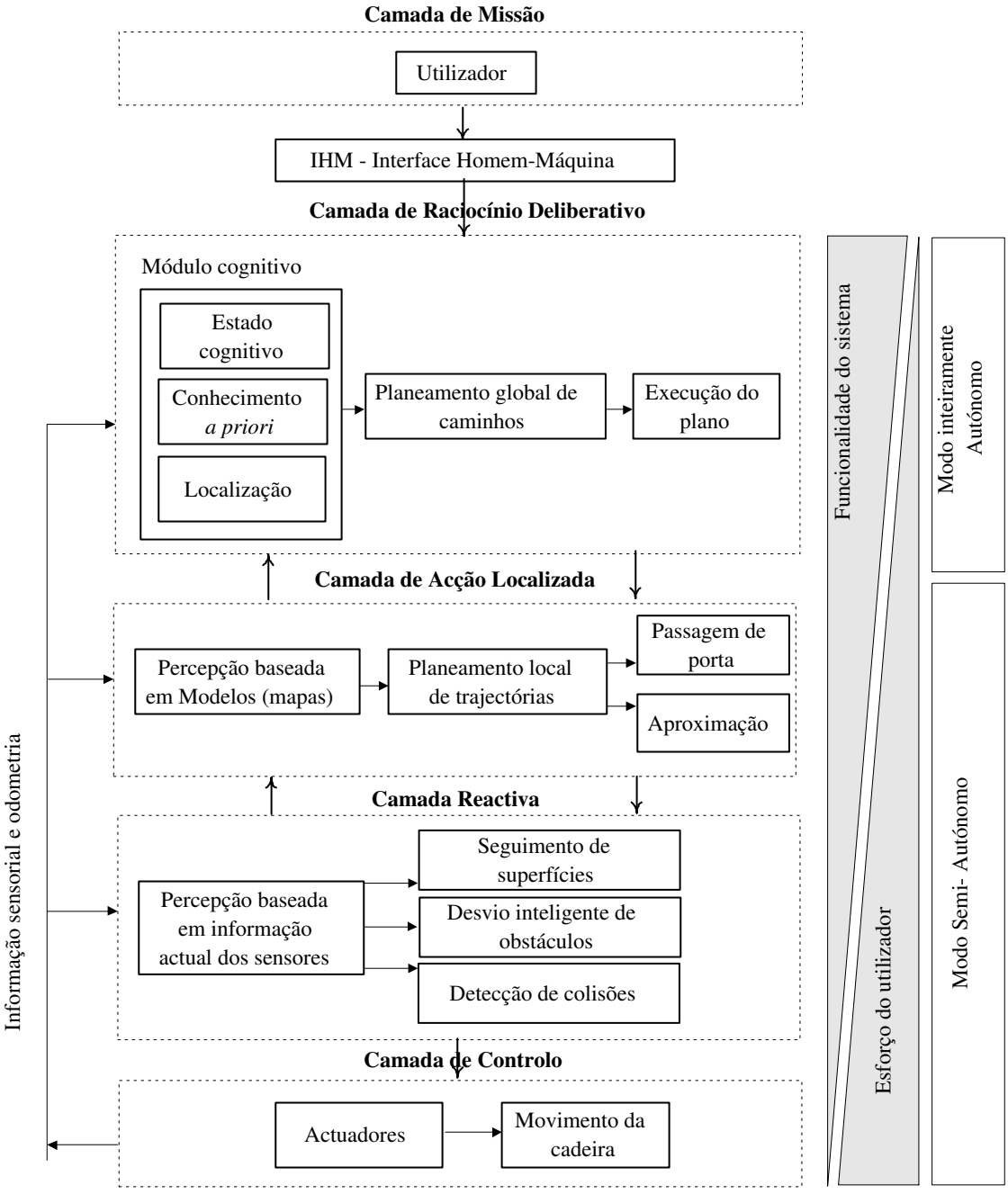


Figura 7.1: Arquitectura RobChair representada por um diagrama de 5 camadas

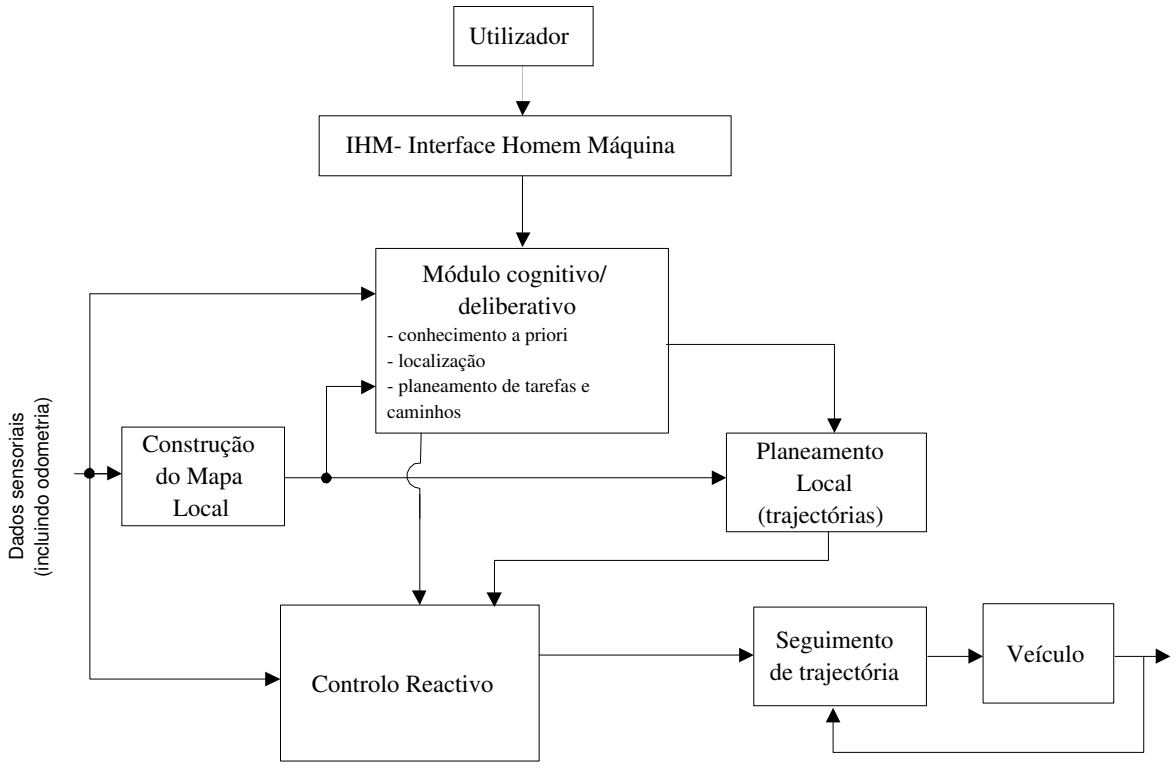


Figura 7.2: Arquitectura híbrida RobChair

Seguimento de superfícies - existem situações em que o seguimento de superfícies (meta) pode melhorar as trajectórias da cadeira e aliviar o esforço do utilizador. Por exemplo, a navegação em corredores curvos ou corredores longos torna a sua utilização vantajosa. Este comportamento foi implementado com um controlador difuso (capítulo 8).

O comportamento de *seguimento de meta* é definido consoante o modo de funcionamento:

- No modo inteiramente autónomo, a meta é fornecida pelas camadas superiores, nomeadamente pelo módulo cognitivo/deliberativo e pelo módulo de planeamento local de trajectória (ver figura 7.2). Estes módulos fornecem ao módulo reactivo um conjunto de posições (x, y) no espaço da cadeira que definem uma dada trajectória. Para além desta informação, podem ainda fornecer dados relativos a restrições cinemáticas e dinâmicas. A camada reactiva deverá seguir a trajectória e ao mesmo tempo reagir em tempo real a estímulos do meio, desviando-se de obstáculos se

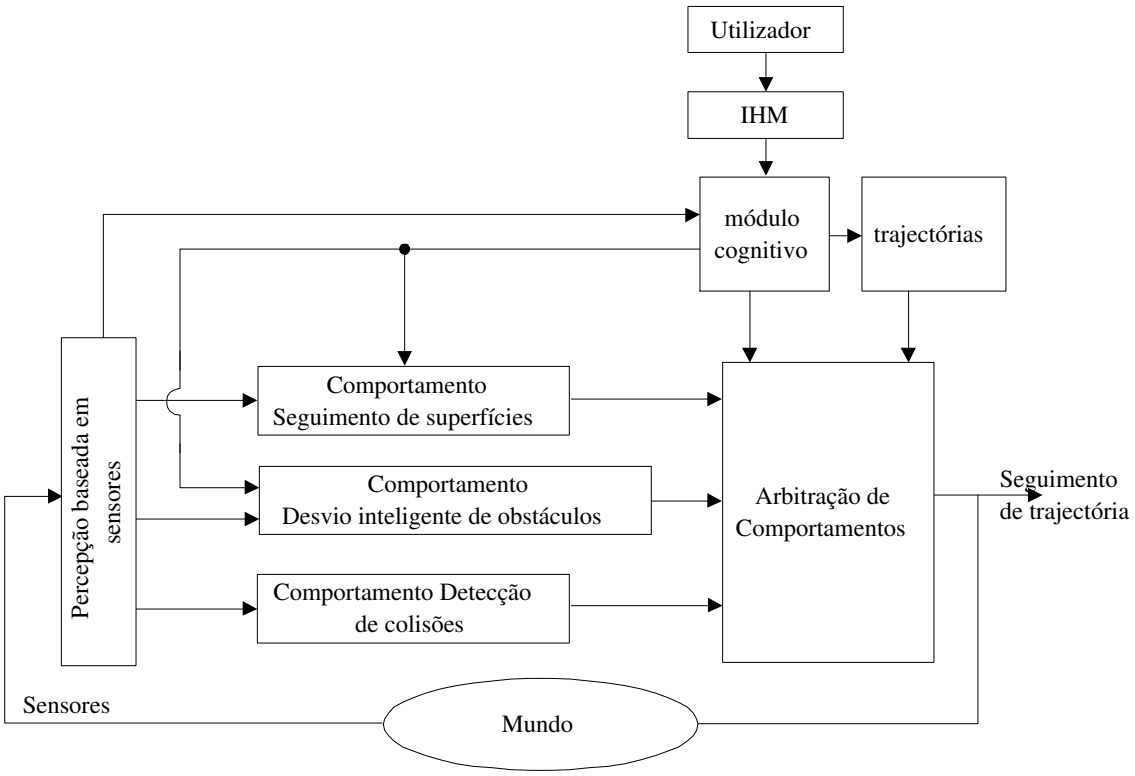


Figura 7.3: Arquitectura reactiva RobChair

necessário.

- No modo actualmente implementado, modo semi-autónomo reactivo, não existe módulo cognitivo artificial. Existe, no entanto, a intervenção activa do utilizador na condução da cadeira. O utilizador, como entidade racional, possui faculdades[2] melhores que qualquer módulo cognitivo artificial, para escolher o melhor percurso, localizar-se, etc. Desta forma, os comandos de condução (comportamento de meta) são fornecidos directamente pelo utilizador através do *joystick* ou através da voz. Contudo, como já foi referido, pretende-se que o controlo da cadeira seja partilhado entre o utilizador e o comportamento de *desvio de obstáculos*, diminuindo o esforço do utilizador. O comportamento de *desvio inteligente de obstáculos* combina as acções do utilizador e as acções do comportamento de *desvio de obstáculos* (dependentes dos estímulos do meio), ou seja, deverá seguir, quando possível, os comandos do utilizador e corrigir as manobras incorrectas. É importante salientar que sem

[2] Assume-se que o utilizador não sofre de incapacidades mentais

a camada reactiva não seria possível a condução por comandos de voz (discretos e difusos).

7.3.1 Mecanismos de Arbitração

Nas arquitecturas comportamentais, os mecanismos de coordenação são uma pedra fundamental para o seu eficiente desempenho. A arquitectura RobChair usa dois mecanismos de coordenação:

Fusão - utilizada para combinar o comportamento de *meta* e o comportamento de *desvio de obstáculos*, resultando no comportamento de *desvio inteligente de obstáculos*. Este comportamento foi implementado através de um controlador difuso (capítulo 8).

Arbitração - utilizada para arbitrar qual dos três comportamentos[3] está activo num determinado instante (apenas um deles pode estar activo). Se o comportamento de *detecção de colisões* estiver activo, isto é, se o pára-choques indicar uma colisão, apenas serão aceites dos outros comportamentos movimentos contrários ao da posição do obstáculo com que colidiu. Significa que este comportamento se sobrepõe aos outros dois. Caso este comportamento não esteja activo, um dos outros dois poderá estar. Por defeito, o comportamento activo é o de *desvio inteligente de obstáculos*. A comutação para o comportamento de *seguimento de superfícies* sucede por ordem do utilizador (que indicará também o lado pelo qual deverá contornar a superfície). Se considerarmos a existência dum módulo cognitivo/deliberativo com capacidade para detectar automaticamente a presença de um corredor longo, a selecção do comportamento de *seguimento de superfície* desencadear-se-á automaticamente.

7.3.2 Escolha do Método de Navegação Reactiva

Após o estudo efectuado no capítulo 3 de todas as abordagens reactivas, escolheu-se o controlo por lógica difusa, pois, para além de não sofrer de algumas das limitações dos outros métodos, permite:

1. execução em tempo real, pois envolve pouco processamento computacional. Esta característica é essencial na navegação reactiva.

[3] Seguimento de superfícies, desvio inteligente de obstáculos e detecção de colisões

2. dispensar um modelo matemático do sistema a controlar. A modelação é feita de uma forma heurística. Isto é particularmente útil para veículos cuja disposição dos sensores e estrutura mecânica do veículo torna difícil retirar um modelo matemático do sistema. Por exemplo, a aplicação do método dos campos potenciais adequa-se a veículos cuja disposição dos sensores é circular, tornando-se a sua utilização mais complexa quando são utilizadas outras disposições de sensores (caso da cadeira RobChair).

3. a implementação fácil de um controlador com várias entradas e saídas.

4. a construção de conjunto de regras heurísticas que definem o algoritmo, com base na experiência e intuição do utilizador.

5. a implementação de mecanismos de fusão de comportamentos aquando da construção das regras, ou na combinação das regras.

7.4 Sistema Sensorial

O módulo de navegação reactiva utiliza 12 sensores de infravermelhos analógicos (IR_a), 12 digitais (IR_d) e dois sensores do pára-choques. Os sensores IR_a estão localizados nas posições $IR0...IR9$ e $IR12$, $IR13$ da figura 7.4 e os sensores IR_d estão localizados nas posições $IR0...IR11$. Os sensores de IR são muito direccionais, daí que tenham de ser colocados de forma cruzada. Se se optasse por uma disposição circular, como no caso dos sonares, criar-se-iam grandes espaços abertos, reduzindo a capacidade de detecção de obstáculos. No caso dos sonares, a escolha da disposição circular deve-se ao facto de estes terem um cone de radiação de cerca de 30^o, permitindo cobrir todo o espaço da zona dianteira. Contudo, estes sensores não serão utilizados no módulo de navegação reactiva. Com a actual disposição dos sensores IR criam-se várias zonas de detecção:

1. **IR0 e IR1**: detecção de obstáculos localizados em frente indicando a possibilidade ou impossibilidade de passagem da cadeira.

2. **IR2 e IR3**: detecção de obstáculos localizados em frente não detectáveis pelos sensores IR0 e IR1.

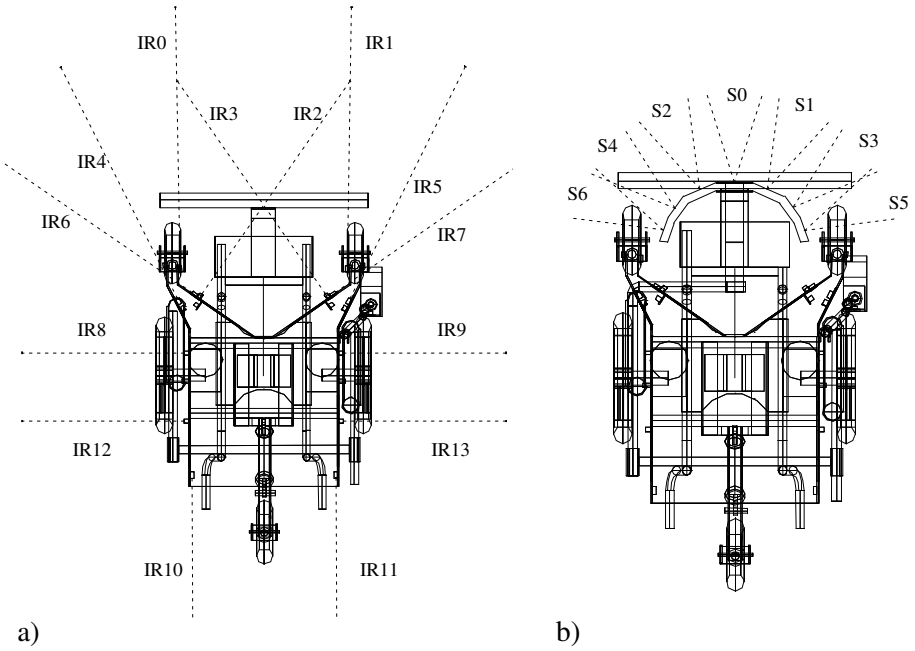


Figura 7.4: a) Disposição dos sensores de infravermelhos e respectivos feixes de emissão (IR0...IR13) b) Disposição dos sensores de ultrassons (S0...S6) e respectivos feixes de emissão

3. **IR4, IR6 e IR5, IR7**: detecção de obstáculos localizados em frente à esquerda e em frente à direita, respectivamente.

4. **IR8 e IR9**: detecção de obstáculos localizados lateralmente à cadeira (no caso dos sensores digitais).

5. **IR8, IR12 e IR9, IR13**: para além de serem utilizados para detecção de obstáculos laterais, fornecem informação suficiente para a cadeira calcular a sua inclinação em relação a superfícies planas. Ou seja, a informação destes sensores será utilizada para o comportamento de seguimento de superfícies.

6. **IR10 e IR11**: detecção de obstáculos localizados na parte traseira da cadeira (apenas sensores digitais).

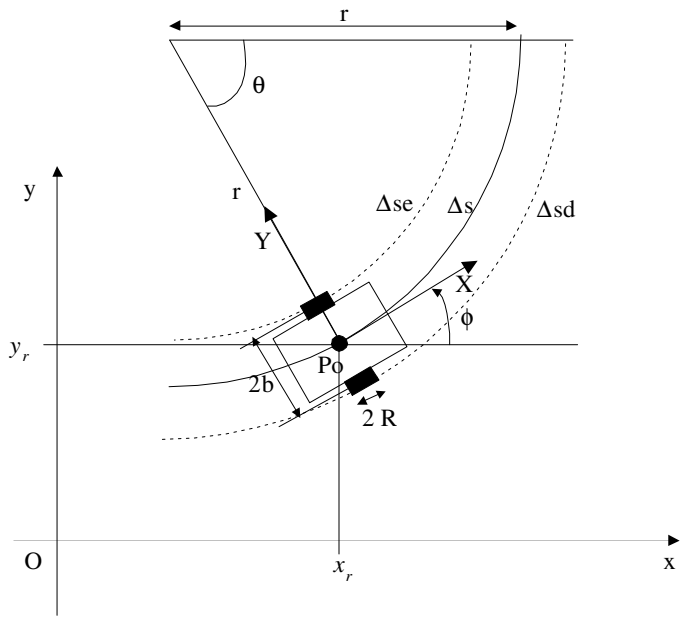


Figura 7.5: Representação do sistema de coordenadas da cadeira RobChair

7.5 Modelo Cinemático

Atendendo à cadeira RobChair representada na figura 7.5, $2b$ é a largura da cadeira e R é o raio da roda. $O-xy$ é o sistema de coordenadas do mundo e $Po-XY$ é o sistema de coordenadas fixo à cadeira. Po é a origem do sistema $Po-XY$, coincidindo com o meio entre a roda esquerda e direita.

Em cada uma das rodas motrizes encontra-se um codificador óptico que permite calcular a velocidade angular de cada uma das rodas. São codificadores de elevada resolução que produzem 5000 pulsos por cada revolução da roda. Desta forma é possível obter informação precisa do deslocamento das rodas permitindo calcular a posição da cadeira (x_r, y_r, ϕ).

A velocidade linear e angular da cadeira pode ser obtida a partir das velocidades angulares das rodas. O deslocamento da cadeira (centro Po) é dado por:

$$\Delta s = r\theta \tag{7.1}$$

em que Δs corresponde ao movimento curvilíneo entre dois instantes de amostragem. O deslocamento da roda esquerda Δse e da roda direita Δsd é obtido pela expressão anterior podendo ser relacionado com o deslocamento angular das rodas esquerda, θe, e direita,

θd, respectivamente:

$$\begin{cases} \Delta se = (r-b)\theta = R\theta_e \\ \Delta sd = (r+b)\theta = R\theta_d \end{cases} \tag{7.2}$$

Somando e subtraindo estas duas expressões obtemos, respectivamente o deslocamento e orientação da cadeira a partir do deslocamento angular das rodas:

$$\Delta s = \frac{R}{2}(\theta_d + \theta_e) \tag{7.3}$$

e

$$\theta = \frac{R}{2b}(\theta_d - \theta_e) \tag{7.4}$$

Derivando em ordem ao tempo, obtemos a velocidade linear, v, e a velocidade angular, w, no referencial $Po - XY$ da cadeira:

$$\begin{cases} v_X = v = \frac{R}{2}(w_d + w_e) \\ v_Y = 0 \\ w = \frac{R}{2b}(w_d - w_e) \end{cases} \tag{7.5}$$

ou na forma matricial:

$$\begin{bmatrix} v_X \\ v_Y \\ w \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ 0 & 0 \\ \frac{R}{2b} & -\frac{R}{2b} \end{bmatrix} \begin{bmatrix} w_d \\ w_e \end{bmatrix} = \frac{R}{2} \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \begin{bmatrix} w_d \\ w_e \end{bmatrix} \tag{7.6}$$

A cadeira desloca-se apenas segundo o seu eixo de simetria X pelo que v_Y será sempre zero. Atendendo a esta restrição, a matriz anterior pode ser apresentada de uma forma mais simplificada, omitindo v_Y:

$$\begin{bmatrix} v_X \\ w \end{bmatrix} = \frac{R}{2} \begin{bmatrix} 1 & 1 \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \begin{bmatrix} w_d \\ w_e \end{bmatrix} \tag{7.7}$$

Calculando a inversa da matriz anterior obtemos w_d e w_e a partir de v_X e w:

$$\begin{bmatrix} w_d \\ w_e \end{bmatrix} = \begin{bmatrix} \frac{1}{R} & \frac{2b}{2R} \\ \frac{1}{R} & -\frac{2b}{2R} \end{bmatrix} \begin{bmatrix} v_X \\ w \end{bmatrix} \tag{7.8}$$

As componentes da velocidade da cadeira segundo o referencial do mundo $O - xy$ são obtidas a partir da transformação de coordenadas (do referencial da cadeira para o

referencial do mundo). Com base na figura 7.5 e partindo da equação 7.7 obtém-se:

$$\begin{bmatrix} v_x \\ v_y \\ w \end{bmatrix} = \begin{bmatrix} \cos(\phi) & 0 \\ \sin(\phi) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_X \\ w \end{bmatrix} \tag{7.9}$$

ou seja,

$$\begin{cases} v_x = v_X \cos(\phi) \\ v_y = v_X \sin(\phi) \end{cases} \tag{7.10}$$

7.5.1 Odometria

Utilizaremos de seguida duas abordagens para actualizar a posição da cadeira: a primeira (figura 7.6a), assumindo que o veículo perfaz movimentos com um determinado raio de curvatura; e a segunda (figura 7.6b), mais simplificada, assumindo que os movimentos do veículo são compostos apenas por movimentos rectilíneos e de rotação pura.

Os deslocamentos de cada uma das rodas obtêm-se a partir das medidas dos codificadores das rodas, através de:

$$\Delta se = \frac{2\pi R N_e}{N_{rev}} \tag{7.11}$$

$$\Delta sd = \frac{2\pi R N_d}{N_{rev}} \tag{7.12}$$

em que N_e e N_d são o número de pulsos medidos na roda esquerda e direita, respectivamente, e N_{rev} é o número de pulsos por revolução da roda.

Primeira abordagem: trajectória curva

Considerando a trajectória percorrida pela cadeira um arco de circunferência tal como está ilustrado na figura 7.5, a partir das equações 7.2 e 7.4 obtém-se θ em função dos deslocamentos percorridos pelas rodas no plano cartesiano:

$$\theta = \frac{\Delta sd - \Delta se}{2b} \tag{7.13}$$

O raio de curvatura r também pode ser obtido em função dos deslocamentos lineares das rodas. A partir das equações 7.1 e 7.3 obtém-se:

$$r = \frac{\frac{\Delta sd + \Delta se}{2}}{\frac{\Delta sd - \Delta se}{2b}} = b\frac{\Delta sd + \Delta se}{\Delta sd - \Delta se} \tag{7.14}$$

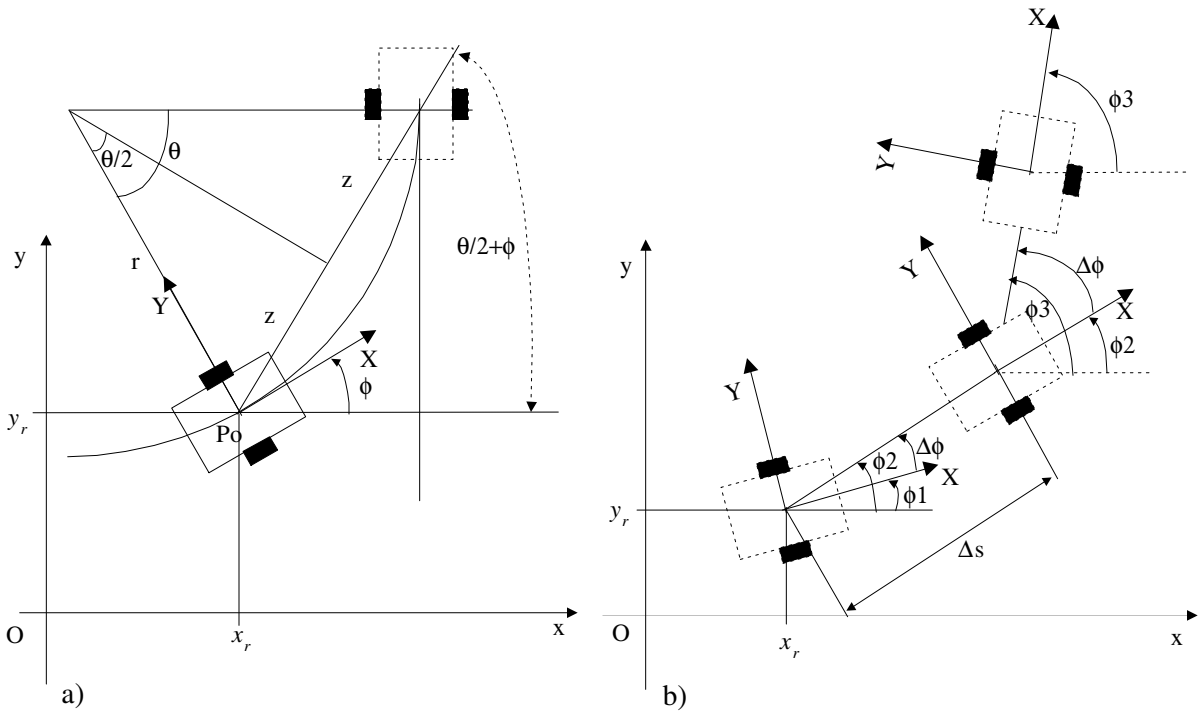


Figura 7.6: a) Posição da cadeira assumindo trajectórias curvas; b) Posição da cadeira assumindo rotações puras e translações

A actualização da posição da cadeira obtém-se por análise da figura 7.6a):

$$\begin{cases} x_r(N) = x_r(N-1) + 2z\cos(\phi(N-1) + \frac{\theta}{2}) \\ y_r(N) = y_r(N-1) + 2z\sin(\phi(N-1) + \frac{\theta}{2}) \\ \phi(N) = \phi(N-1) + \theta \end{cases} \tag{7.15}$$

em que $z = r\sin(\frac{\theta}{2})$. Substituindo pelo valor de r da equação 7.1, obtém-se:

$$\begin{cases} x_r(N) = x_r(N-1) + 2\frac{\Delta s}{\theta}\sin(\frac{\theta}{2})\cos(\phi(N-1) + \frac{\theta}{2}) \\ y_r(N) = y_r(N-1) + 2\frac{\Delta s}{\theta}\sin(\frac{\theta}{2})\sin(\phi(N-1) + \frac{\theta}{2}) \\ \phi(N) = \phi(N-1) + \theta \end{cases} \tag{7.16}$$

Note-se que se o movimento for rectilíneo ($\Delta se = \Delta sd$) então $\theta = 0$ o que resulta numa indeterminação na equação 7.16. Considerando $\frac{\theta}{2}$ muito pequeno ($\frac{\theta}{2} \to 0$), então:

$$\frac{\sin(\frac{\theta}{2})}{\frac{\theta}{2}} \simeq 1 \tag{7.17}$$

Utilizando esta aproximação na equação 7.16 obtém-se:

$$\begin{cases} x_r(N) = x_r(N-1) + \Delta s \cos(\phi(N-1) + \frac{\theta}{2}) \\ y_r(N) = y_r(N-1) + \Delta s \sin(\phi(N-1) + \frac{\theta}{2}) \\ \phi(N) = \phi(N-1) + \theta \end{cases} \tag{7.18}$$

Esta expressão, sempre válida para instantes de amostragem pequenos, utilizou-se para o cálculo da posição da cadeira RobChair (instante de amostragem de 40 ms).

Segunda abordagem: movimentos rectilíneos e rotações puras

Pode-se também considerar, no caso de um modelo cinemático mais simples, que os movimentos do veículo são apenas compostos por rotações puras e translações (movimentos rectilíneos) , como ilustrado na figura 7.6b). Neste caso, considerar-se-á apenas a velocidade linear do veículo na direcção do seu eixo de simetria X (com velocidades angulares das rodas $w_e = w_d$) e a velocidade angular, w, da cadeira em torno de si própria com velocidades angulares simétricas ($w_e = -w_d$). Considerando apenas movimentos rectilíneos:

$$w_e = w_d \tag{7.19}$$

substituindo na equação 7.5, obtém-se:

$$\begin{cases} v_X = \frac{R}{2}(2w_d) = \frac{R}{2}(2w_e) \\ w = 0 \end{cases} \tag{7.20}$$

Considerando movimentos de rotação pura:

$$w_e = -w_d \tag{7.21}$$

substituindo na equação 7.5, obtém-se:

$$\begin{cases} v_X = 0 \\ w = \frac{R}{2b}(2w_d) = \frac{R}{2b}(-2w_e) \end{cases} \tag{7.22}$$

Para calcular a posição da cadeira, sabemos que para movimentos rectilíneos:

$$\Delta s = \Delta se = \Delta sd \tag{7.23}$$

e

$$\Delta\phi = 0 \tag{7.24}$$

Para movimentos de rotação pura:

$$\Delta s = 0 \tag{7.25}$$

e a rotação igual a $\Delta\phi$.

A actualização da posição da cadeira (figura 7.6b) é dada por:

$$\begin{cases} x_r(N) = x_r(N-1) + \Delta s \cos(\phi(N-1) + \Delta\phi) \\ y_r(N) = y_r(N-1) + \Delta s \sin(\phi(N-1) + \Delta\phi) \\ \phi(N) = \phi(N-1) + \Delta\phi \end{cases} \tag{7.26}$$

em que Δs é o deslocamento rectilíneo efectuado e $\Delta\phi$ corresponde ao deslocamento angular relativo às rotações puras.

Capítulo 8

Controlador Difuso

Este capítulo descreve o controlador de navegação reactiva difuso do sistema RobChair. As partes constituintes do controlador são sucessivamente apresentadas.

8.1 Entradas e Saídas do Controlador

O sistema de navegação difuso tem por objectivo controlar o deslocamento linear e angular da cadeira de rodas, enviando comandos de movimento aos actuadores do veículo. A escolha das variáveis de saída do controlador difuso fez-se por analogia aos sinais fornecidos pelo *joystick* e que controlam o veículo em modo manual (ver secção 5.3.2): velocidade linear v, e velocidade angular w.

As variáveis de entrada do controlador difuso estão associadas à leitura dos sensores (figura 8.1a)), e ao objectivo/meta a seguir (figura 8.1b)). No que diz respeito aos sensores, as variáveis referem-se às distâncias medidas: 12 variáveis relativas aos sensores de infravermelhos digitais (d_i $i = 0, \ldots, 11$), 12 variáveis relativas aos sensores de infravermelhos analógicos (da_i $i = 0, \ldots, 11$), e duas variáveis relativas aos sensores de contacto do pára-choques (p_e e p_d), totalizando ao todo 26 variáveis de entrada. Estes sensores foram utilizados de duas formas:

- Numa primeira abordagem, desenvolveram-se algoritmos de desvio inteligente de obstáculos baseados apenas na informação proveniente dos sensores digitais e dos sensores de contacto do pára-choques. Esta abordagem, "semi-difusa"[1], serviu prin-

[1] Dada a natureza das variáveis dos sensores digitais não se pode designar o controlador de difuso

cipalmente para construir a base de regras e validar de forma simples o conceito de controlo partilhado utilizador/máquina.

- Numa segunda abordagem, utilizaram-se todos os sensores. No entanto, os sensores digitais, cujo alcance é superior ao dos sensores analógicos, apenas serviram para definir uma zona de semi-segurança do veículo. Podemos observar esta zona na figura 8.3. Consiste numa área de aproximadamente um metro em redor da cadeira. Se aí for detectado qualquer obstáculo, o controlador do veículo induz imediatamente uma redução de velocidade, mas sem executar qualquer tipo de desvio. Este fica a cargo do comportamento de desvio inteligente de obstáculos implementado pelo controlador difuso que aceita à entrada apenas os sensores analógicos. Os sensores de contacto do pára-choques são utilizados apenas pelo comportamento de detecção de colisões.

Concluímos portanto, que as 26 variáveis relativas aos sensores nunca são utilizadas simultaneamente. Resumidamente: o detector de entrada em zona de semi-segurança (definição de regras simples) utiliza as 12 variáveis dos sensores digitais; o controlador difuso, propriamente dito, que implementa o desvio inteligente de obstáculos, usa as 12 variáveis dos sensores analógicos; e o detector de colisões utiliza as duas variáveis dos sensores do pára-choques (definição de regras simples).

As variáveis objectivo ou meta definem a intenção do utilizador. No caso de controlo por *joystick*, estas são a direcção angular, θ_{meta}, e a intensidade de velocidade, v_{meta}; e no caso de comandos de voz apenas se toma a variável relativa à direcção angular. Os comandos de voz são convertidos em ângulos de direcção discretos de 45º. A figura 8.2 ilustra as variáveis de entrada nos casos de controlo por *joystick* e por comandos de voz.

Somando as variáveis objectivo às variáveis sensoriais resultam 14 variáveis. Supondo que cada variável de entrada é descrita com a ajuda de m termos linguísticos (m define a granularidade de um estado do sistema a controlar), o espaço total de entrada do sistema é de m^{14} regras. Se considerarmos $m = 3$ (um valor normalmente utilizado), serão necessárias 16384 regras. Note-se que este é o número máximo de regras que cobre totalmente o espaço de entrada. Se se optasse por separar cada uma das variáveis em vez de as combinar, ou seja, cada regra apenas com uma variável, reduzir-se-ia o número de regras para $m \times 14$. No entanto, esta metodologia não se adequa ao nosso sistema por

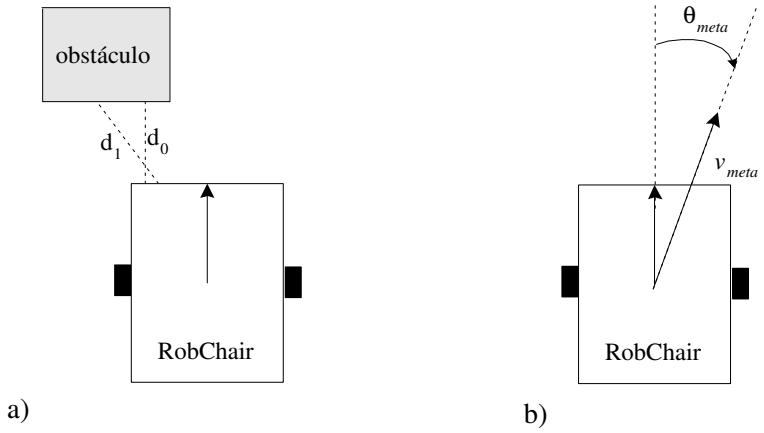


Figura 8.1: Variáveis de entrada do controlador difuso: a) Distância medida pelos sensores (d_i); b) Direcção angular (θ_{meta}) e intensidade de velocidade (v_{meta}) do comportamento objectivo

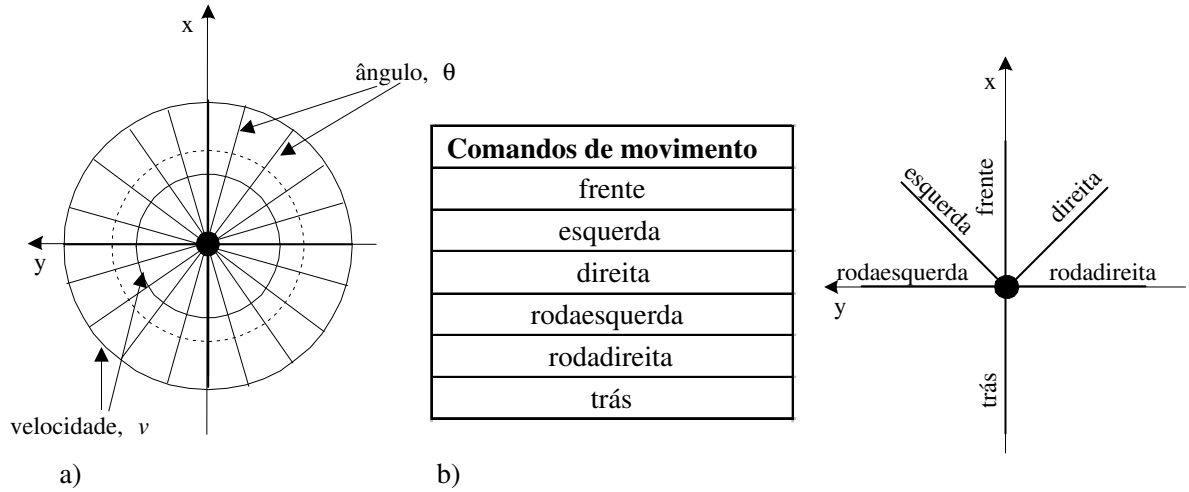


Figura 8.2: Variáveis de entrada do controlador no caso de: a) *joystick*; b) Comandos de voz e respectiva conversão angular

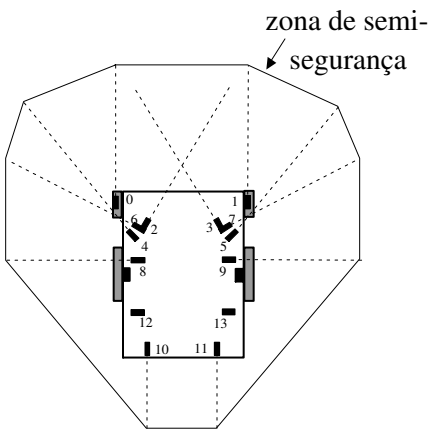


Figura 8.3: Zona de segurança definida pelos sensores de infravermelho digitais

existir uma forte relação entre as variáveis[2]. Para reduzir ainda mais o número de regras, poder-se-ia também ter optado por agrupar as leituras dos sensores em função de várias direcções de detecção, tais como *frente, frente_esquerda, frente_direita, esquerda, direita e trás*, reduzindo assim o espaço de entrada do controlador para apenas 10 variáveis. No entanto, esta modelação simplificada levaria à perda de informação importante relativa à posição mais exacta dos obstáculos, tendo-se, por isso, deixado à entrada todas as variáveis. Por outro lado, como veremos pela implementação, não será necessário um número tão elevado de regras como o referido (16384). Primeiro porque m será inferior a 3 para algumas variáveis e, segundo, porque nem todas as variáveis serão consideradas simultaneamente.

8.2 Módulo de Difusão

O módulo de difusão tem o papel de transformar as variáveis numéricas das entradas em conjuntos difusos manipuláveis pelo controlador. Surgem duas questões imediatamente aquando da implementação: i) como escolher a forma dos conjuntos difusos; e ii) quantos conjuntos serão necessários e suficientes. A escolha das funções de pertença (conjuntos difusos) é subjectiva e não existe uma forma sistemática de as encontrar. No entanto, existem regras que devem ser sempre respeitadas [Jan91]:

[2]Como referido na secção 3.2.7 a combinação das regras requer mais cuidado para obter à saída um valor coerente

- os conjuntos que definem cada um dos termos deverão ser suficientemente largos para maior insensibilidade ao ruído associado às medidas. Os conjuntos deverão cobrir todo o universo de discurso, sendo o número de conjuntos dependente da largura dos mesmos.
- deve haver sobreposição dos conjuntos, caso contrário o controlador pode correr em estados indefinidos, isto é, se houver um intervalo entre conjuntos não existirão regras que se activem para os valores do intervalo.

De acordo com estas regras é recomendado:

- começar com conjuntos triangulares. Os conjuntos laterais esquerdo e direito deverão ser em forma de ombro.
- a sobreposição entre conjuntos deverá ser no mínimo de 50% de forma a fornecer transições suaves das acções de controlo.
- começar com três conjuntos para cada variável.

Com base nestas regras e atendendo à especificidade do nosso controlador, optou-se pela utilização dos seguintes conjuntos difusos (ver figura 8.4):

Singleton difuso - o valor numérico x_0 é transformado num singleton difuso, ou seja, um conjunto em que a função de pertença é 1 para o valor x_0 e zero para todos os outros valores. Este tipo de codificação é utilizado para medidas não ruidosas ou do tipo discreto. No nosso sistema, serão utilizadas para codificar:

1. as entradas relativas aos sensores de infravermelhos digitais e aos sensores de contacto do pára-choques dianteiro, porque estas devolvem uma informação discreta do tipo presença/ausência de obstáculos e contacto/não contacto, respectivamente.
2. as variáveis relativas ao IHM por comandos de voz. Os comandos fornecidos pelo utilizador dão uma informação discreta (com valor de discretização grande) da direcção que pretende seguir.
3. as saídas do controlador. A velocidade linear e angular são desdifusadas de forma discreta. Como veremos na secção 8.4, esta simplificação traz maior rapidez no processamento das regras, sem grande prejuízo nos valores desdifusados.

Em ombro - para codificar as variáveis relativas às medidas dos sensores de infravermelho analógicos. Os conjuntos representam a possibilidade de se encontrar um obstáculo a uma determinada distância. Utilizou-se um módulo de pré-difusão que converte os níveis de tensão em distâncias, evitando assim o uso de conjuntos do tipo exponencial que descrevem a resposta distância vs. tensão dos sensores.

Trapezoidal - para codificar as variáveis relativas à direcção angular e intensidade (velocidade) fornecidas pela IHM por *joystick*. Com estes conjuntos pretende-se definir zonas planas que atribuem o mesmo grau de verdade a uma gama larga de direcção e intensidade. Recordemos que se quer libertar o utilizador da necessidade de manipular o *joystick* com elevada precisão, devendo, por isso, o controlador ser mais ou menos insensível à posição do *joystick* dentro de uma determinada gama.

Para além dos conjuntos atrás referidos, criou-se, para as variáveis de distância da_i, um conjunto rectangular cujo termo linguístico é MUITO PERTO (figura 8.5b)) e que define uma gama de distâncias dentro da qual não deve haver movimento do veículo. A figura 8.5 mostra as funções de pertença e termos linguísticos associados às variáveis de entrada e saída do controlador.

8.3 Lógica de decisão: implicação e operadores lógicos

O módulo de lógica de decisão define os operadores lógicos e mecanismos de inferência. As conectivas utilizadas são os operadores E, OU e NAO e definem as operações mais usuais, ou seja, min, max e $1-\mu$, respectivamente. Utilizou-se o mecanismo de inferência produto-soma para saídas do tipo singleton. A escolha deste mecanismo de inferência baseia-se nos motivos apresentados na secção A.3.2, nomeadamente, o *produto* porque conserva a forma da função de pertença do conjunto de saída[3], e a *soma* porque, desta forma, o resultado final é influenciado por regras diferentes que cheguem à mesma conclusão. Vejamos a aplicação deste mecanismo de inferência através do exemplo seguinte. Suponhamos as seguintes regras genéricas:

R1: SE v PEQUENO E d PERTO ENTÃO w DEVAGAR OU
R2: SE v MEDIO E d PERTO ENTÃO w MEDIO OU

[3]Irrelevante neste caso, uma vez que se trata de saídas do tipo singleton

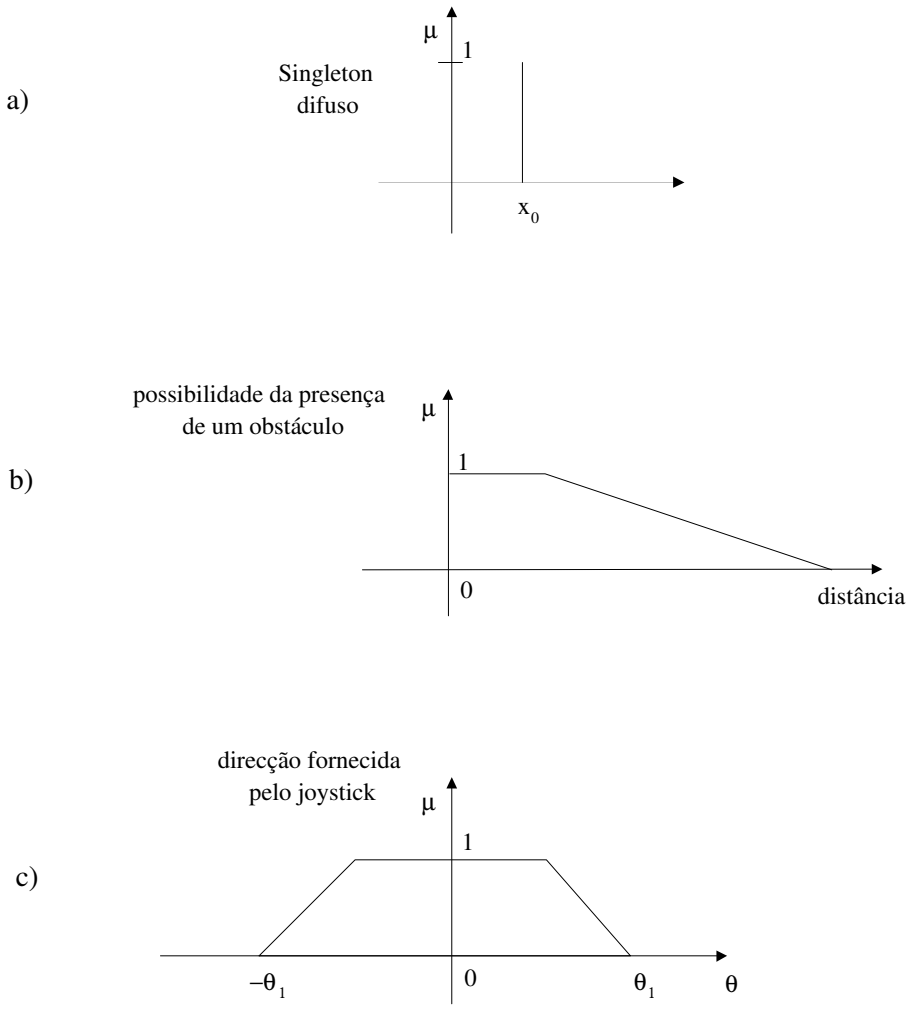
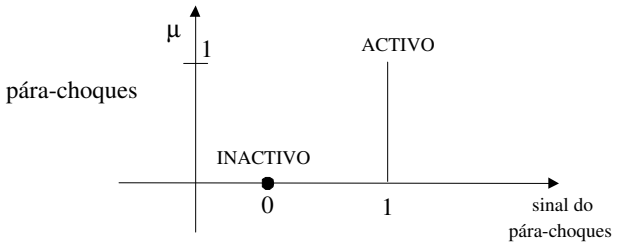
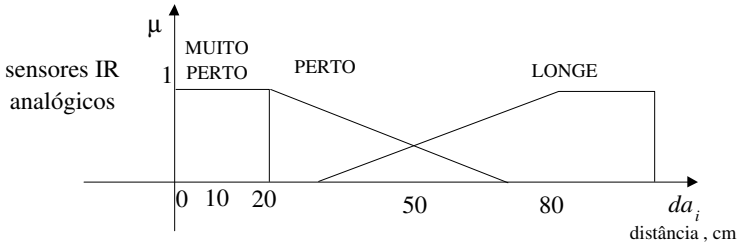


Figura 8.4: Conjuntos difusos utilizados no controlador: a) Singleton difuso; b) Em ombro; c) Trapezoidal

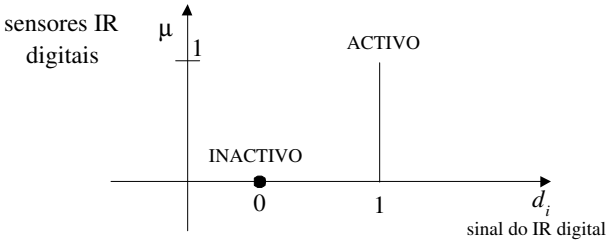
a)


μ
1
ACTIVO
pára-choques
INACTIVO
0
1
sinal do
pára-choques

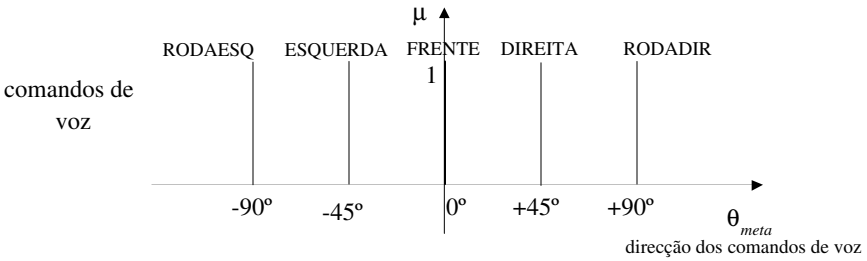
b)


μ
1
MUITO
PERTO
PERTO
LONGE
sensores IR
analógicos
0 10 20
50
80
da_i
distância , cm

c)


sensores IR
digitais
μ
1
ACTIVO
INACTIVO
0
1
d_i
sinal do IR digital

d)


comandos de
voz
μ
RODAESQ
ESQUERDA
FRENTE
DIREITA
RODADIR
1
-90º
-45º
0º
+45º
+90º
θ_{meta}
direcção dos comandos de voz

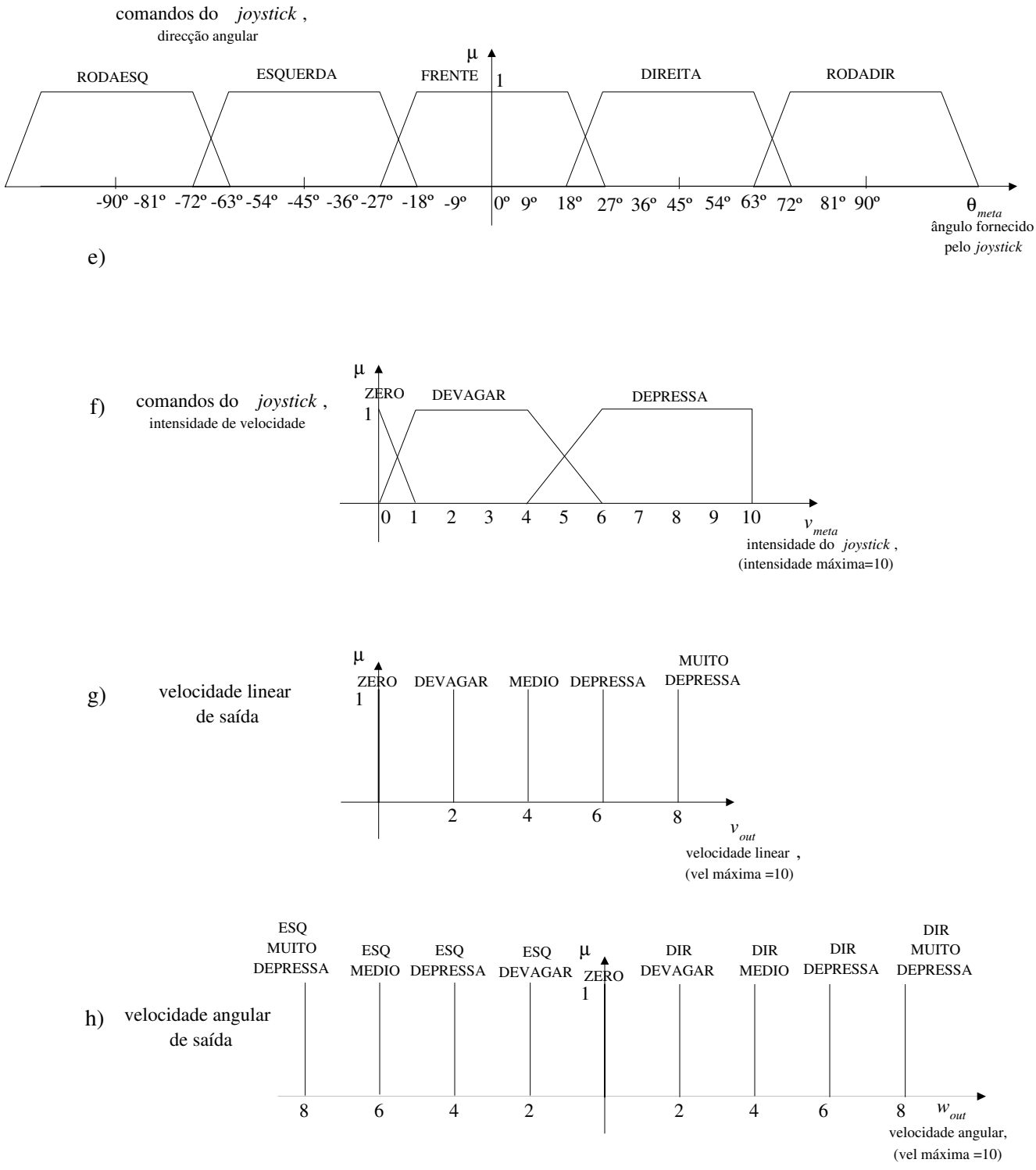


Figura 8.5: Definição das funções de pertença e termos linguísticos das variáveis de entrada e saída do controlador

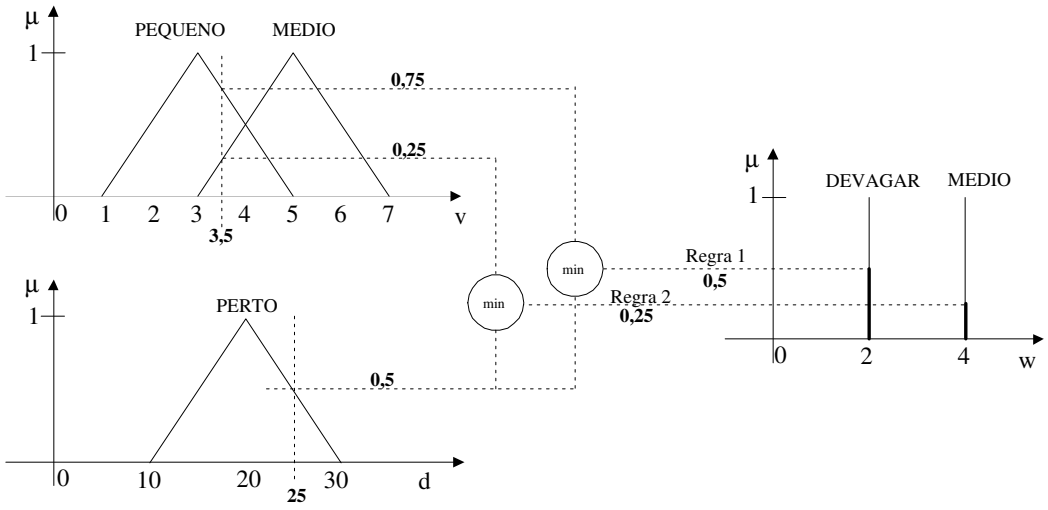


Figura 8.6: Interpretação gráfica do mecanismo de inferência utilizado no controlador RobChair

⋮

Rn: SE v GRANDE E d PERTO ENTÃO w GRANDE

Sendo o processo de inferência do tipo soma-produto, o operador de implicação ($ENTÃO$) define a operação de *produto* e o operador OU, que combina as regras, define a operação *soma*. Uma vez que as variáveis de saída são singletons, o operador $ENTÃO$ poderia definir arbitrariamente a operação *min* sem alterar o resultado. O valor de verdade da premissa é dado pelo mínimo (operador E) dos conjuntos antecedentes. O conjunto de saída é então multiplicado por esse valor. A figura 8.6 interpreta graficamente o mecanismo de inferência supondo o cenário em que são activadas as regras *R1* e *R2* pelos valores de entrada $v = 3.25$ e $d = 25$.

8.4 Módulo de Desdifusão

O resultado da avaliação das regras de controlo são dois conjuntos difusos que representam respectivamente a velocidade angular e a velocidade linear. O módulo de desdifusão tem por objectivo transformar estes conjuntos difusos em valores reais a enviar aos actuadores. Entre os métodos de desdifusão apresentados no apêndice A, escolheu-se o método de centro de gravidade (CDG) porque reflecte, melhor que qualquer outro método, a

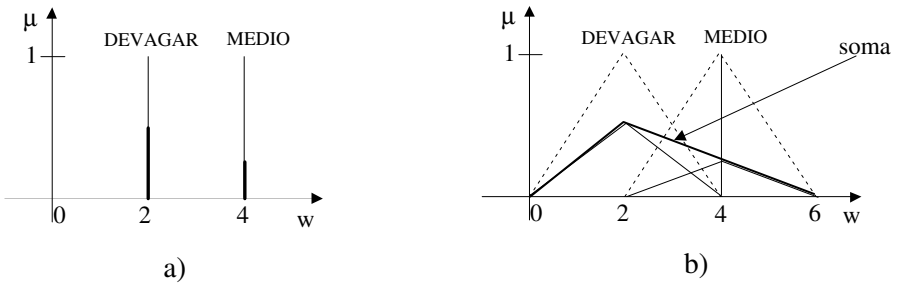


Figura 8.7: Comparação do método de desdifusão CDG aplicado a singletons (a) e conjuntos triangulares (b)

distribuição (forma) do conjunto difuso resultante. O método CDG, será aplicado aos singleton difusos dos conjuntos de saída. A opção de tomar para a saída do controlador RobChair singleton difusos, em vez de outros conjuntos difusos, traz várias vantagens, nomeadamente: simplificação da computação dos mecanismos de inferência e de desdifusão, facilidade em levar os sinais de controlo ao seu limite, se necessário; e finalmente, as regras são escritas de forma mais intuitiva. Por estas vantagens, os singletons à saída têm-se tornado cada vez mais populares. Retomando o exemplo anterior, comparamos o resultado da aplicação do método CDG utilizando à saída singleton difusos e conjuntos triangulares. A figura 8.7 mostra as funções de pertença de saída para cada um dos casos. No primeiro caso (alínea a)), o valor desdifusado é:

$$u = \frac{\sum_i \alpha_i . s_i}{\sum_i \alpha_i} = \frac{0.5 * 2 + 0.25 * 4}{0.5 + 0.25} = 2.66 \tag{8.1}$$

No segundo caso (alínea b)), o valor desdifusado, é dado pela expressão:

$$u = \frac{\sum_i \mu(x_i) x_i}{\sum_i \mu(x_i)} \tag{8.2}$$

Utilizando uma discretização de 0.1 no universo de discurso, o valor de desdifusão obtido, neste caso, também foi de 2.66. Note-se, que o valor poderia ter sido ligeiramente diferente caso se aplicasse outro mecanismo de inferência. Como vemos, pelo gráfico da figura 8.7 e respectiva expressão de desdifusão, as saídas triangulares trazem maior complexidade na soma dos dois conjuntos e exigem maior esforço computacional no cálculo de desdifusão, justificando-se portanto a escolha do uso de singletons para a saída.

8.5 Definição da Base de Regras

As várias partes do controlador definidas até agora são comuns à implementação de todos os comportamentos reactivos. Resta agora definir, para cada comportamento reactivo - desvio inteligente de obstáculos e seguimento de superfícies, a base de regras que traduz o conhecimento heurístico do perito.

Como construir a base de regras? Existem pelo menos 4 formas de encontrar regras de controlo [Jan91]:

1. Com base na experiência de um perito: mediante um questionário organizado, recolhendo assim um conjunto de regras de peso.

2. Por observação das acções de controlo do operador: a partir de um conjunto de exemplos formados por pares de *(observação, acção)*.

3. Construindo um modelo difuso do sistema: através de uma descrição linguística do processo dinâmico a controlar (modelo difuso do processo). Conhecendo o modelo do processo, procura-se extrair as regras que o permitem controlar (invertendo o modelo difuso). Este tipo de abordagem adapta-se mais a situações em que o controlador tem de seguir trajectórias de referência.

4. Aprendizagem: automaticamente, ou então com modificação de uma base de regras já existente. Exemplos de aplicação são o controlador auto-organizável e métodos de aprendizagem baseados em redes neuronais.

Encontrámos as várias regras seguindo a primeira abordagem acima referida.

8.6 Comportamentos

Serão definidas nesta secção as bases de regras para cada um dos comportamentos: desvio inteligente de obstáculos e seguimento de superfícies.

8.6.1 Desvio Inteligente de Obstáculos

A base de regras do comportamento de desvio inteligente de obstáculos combina em cada regra o comportamento de desvio de obstáculos e de seguimento de meta, resultando

uma só saída global. Poder-se-ia também escrever as regras considerando estes dois comportamentos separadamente. Neste caso resultaria uma saída para cada comportamento, obrigando a uma posterior combinação de ambas. No entanto, dada a necessidade de interacção entre os dois objectivos (desvio de obstáculo e seguimento de meta), assegura-se a obtenção de uma resposta mais coerente seguindo a primeira abordagem. A segunda abordagem é mais dependente das técnicas de desdifusão.

A figura 8.8 apresenta um conjunto de cenários que serviram para a construção da base de regras. Expõe-se antes de mais o conjunto de reacções pretendidas para a cadeira. Deseja-se que o controlador perfaça movimentos suaves. Mais especificamente, quando os obstáculos se encontram relativamente longe, a cadeira deverá começar a efectuar uma progressiva trajectória de rotação. Evitam-se assim rotações rápidas e acentuadas que ocorreriam se o veículo se aproximasse demasiado dos obstáculos. Vejamos algumas situações:

Sem obstáculos - o controlador segue os comandos do utilizador, mas com imposição de limites máximos na velocidade linear e angular.

Obstáculos perto em frente - se a indicação do utilizador for para seguir em frente, o controlador deverá reduzir a velocidade linear (ultrapassando a distância de segurança, a velocidade deverá ser zero) e manter velocidade angular zero (figura 8.9a)). Se a indicação do utilizador for virar à esquerda/direita, o controlador deve, ao mesmo tempo que reduz a velocidade linear, rodar mais ou menos depressa, dependendo da distância ser menos ou mais perto, respectivamente.

Obstáculos perto à frente e esquerda ou à frente e direita - Se a indicação do utilizador é para seguir em frente e existem obstáculos do lado frente e esquerdo ou frente e direito (mas não ambos) a velocidade linear deve reduzir (podendo ser zero) e rodar acentuadamente para o lado direito ou esquerdo, respectivamente. Caso a indicação do utilizador seja a de ir ao encontro dos obstáculos, o controlador deve seguir essa indicação (dentro do limite de segurança) (figura 8.9b)).

Obstáculos longe à frente e esquerda ou à frente e direita - Neste caso, o controlador deve efectuar uma rotação suave no sentido oposto ao obstáculo, o que resulta da combinação de uma velocidade linear média e velocidade angular média.

Obstáculos perto laterais - caso a indicação do utilizador seja seguir em frente, havendo obstáculos perto laterais de ambos os lados, a velocidade angular deve ser nula. Se a indicação do utilizador for outra, a velocidade linear deve ser nula e a velocidade angular pequena.

Abordagem simplificada

Como referimos no início do capítulo, numa primeira abordagem, implementou-se um algoritmo de desvio inteligente de obstáculos simples, baseado na informação dos sensores de infravermelhos digitais, sensores do pára-choques e informação das variáveis meta do *joystick* e voz. O algoritmo, apesar de não utilizar formalmente um controlador difuso, serviu para testar e validar um conjunto de regras utilizadas na segunda abordagem. Como podemos visualizar no pseudo-código seguinte, as regras foram construídas de forma encadeada, o que permitiu diminuir consideravelmente o número de regras. Utilizaram-se todos os sensores digitais à excepção dos sensores IR8 e IR9. Eis um dos algoritmos implementados para utilização com *joystick*:

```
partilha_joy_sensor{

  int  vel_lin, vel_ang; //variáveis de saída
  int  j_d, j_a;               //variáveis do joystick
  int  ir[12];                 //variáveis dos sensores ir
                                   digitais
  int  l_bump, r_bump;         //para-choques esquerdo e direito

  vel_lin = j_d;
  vel_ang = j_a;

  Regras:

  if((j_d == 0) AND (j_t == 0)){
    vel_lin = 0;
    vel_ang = 0;
  }
```

```
else
  if((l_bump OR r_bump) AND (j_a > 0))  //análise do para-choques
  {
      vel_lin = 0;
      vel_ang = 0;
  }
  else
    if(left_b OR right_b)
    {
      vel_lin = j_d;
      if(l_bump) vel_ang= 3;
      else vel_ang = -3;
    }
    else
    {                          //análise dos sensores dianteiros
      if((j_d > 0) AND ( ir[1] OR ir[0] OR ir[2] OR ir[3]))
      {
        vel_lin = 0;           //análise dos sensores laterais
        if((NOT ir[10] AND  NOT ir[11]) AND (ir[4] AND ir[5]))
        {
          vel_lin = -1;
          vel_ang = -0;
        }
        if(NOT ir[4]) v_ang = -2;
          else
        if(NOT ir[5]) v_ang = 2;
      }
      else
          if ((j_d > 0) AND ir[4] AND NOT ir[5])
              vel_ang = j_a +2;
          else
          if ((j_d > 0) AND NOT ir[4] AND ir[5])
```

```
            vel_ang = j_a -2;
        else
        if ((j_d > 0) AND (ir[4] OR ir[6]) AND NOT(ir[3] OR ir[5]))
            vel_ang = j_a -3;
        else
        if ((j_d > 0) AND (ir[3] OR ir[5]) AND NOT(ir[4] OR ir[6]))
            vel_ang = j_a +3;
    }
if((j_d == 0) AND (j_a > 0))
  vel_ang = vel_ang + 1;
if(((vel_lin < 0) OR (j_d < 0)) AND (ir[11] OR ir[10]))
  vel_lin = 0;
else if(j_d < 0) vel_lin = j_d;


RCHAIR_ang(vel_ang);  //envia comandos aos actuadores
RCHAIR_lin(vel_lin);
```

Segunda abordagem

Na segunda abordagem, o comportamento de desvio inteligente de obstáculos foi totalmente implementado através do controlador difuso. Utilizam-se apenas os sensores analógicos $IR0, \ldots, IR7$. A tabela 8.1 apresenta as regras que serviram de base à implementação do comportamento de desvio inteligente de obstáculos. As letras F, ESQ, DIR, Z, D, M, D_DEP, D_DEV, E_DEP e E_DEV correspondem aos termos linguísticos já definidos, respectivamente, FRENTE, ESQUERDA, DIREITA, ZERO, DEVAGAR, MEDIO, DIREITA DEPRESSA, DIREITA DEVAGAR, ESQUERDA DEPRESSA e ESQUERDA DEVAGAR.

8.6.2 Seguimento de Superfícies

O controlador difuso para o comportamento de seguimento de superfícies aceita à entrada o erro de orientação, θ_e, da cadeira relativamente à superfície, e também o erro de distância,

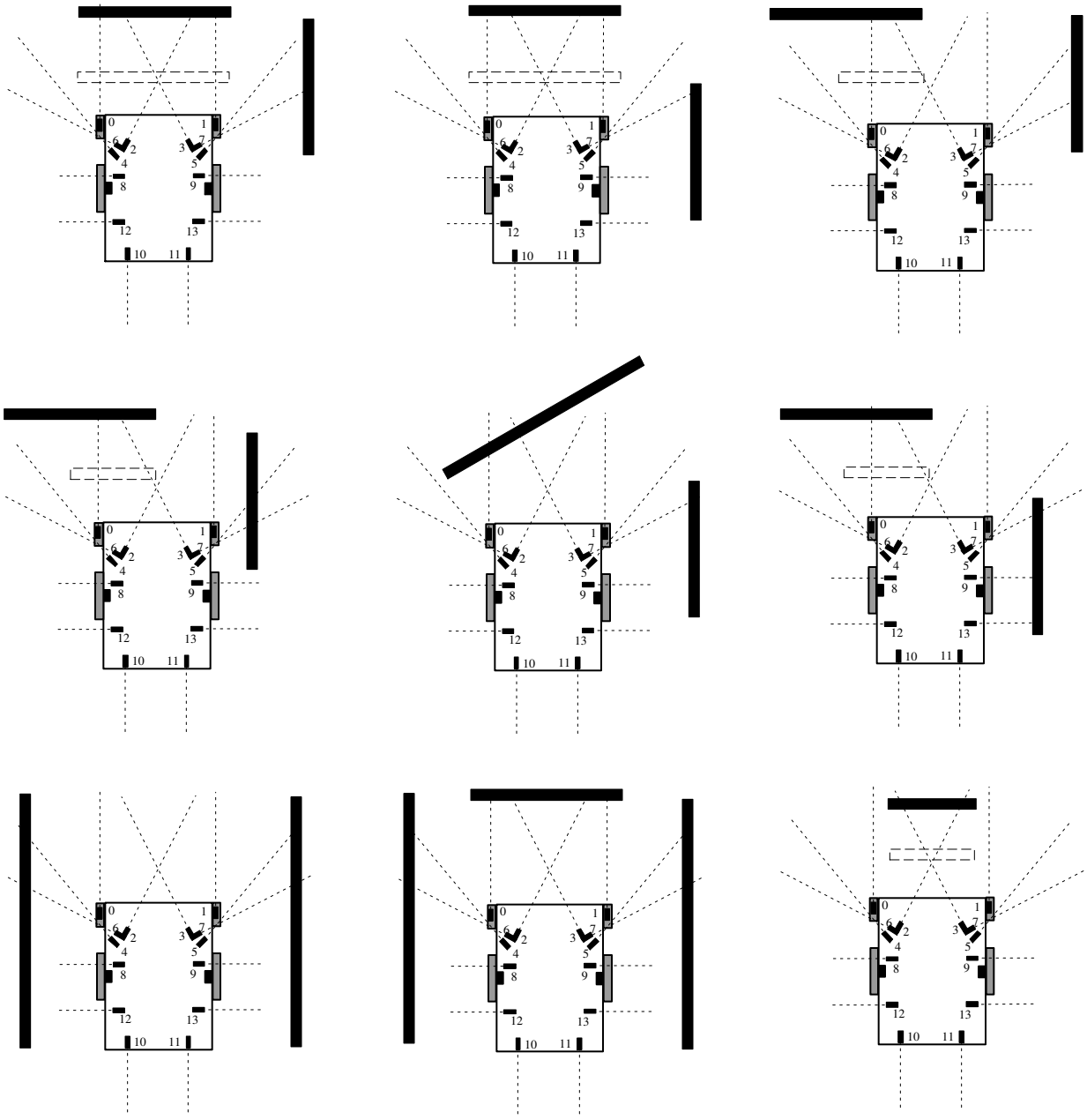


Figura 8.8: Cenários típicos que serviram para construção da base de regras

Regras	Objectivo		Distâncias								Saídas	
	v_{meta}	θ_{meta}	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	v_{out}	w_{out}
R1	D	F	P	P	P	P					D	Z
R2	D	F	L	L	L	L					M	Z
R3	D	ESQ	P	P	P	P	L		L		Z	E_DEP
R4	D	ESQ	L	L	L	L	L		L		D	E_DEV
R5	D	ESQ	P	P	P	P	P		P		Z	E_DEV
R6	D	ESQ	L	L	L	L	P		P		M	E_DEV
R7	D	DIR	P	P	P	P		L		L	Z	D_DEP
R8	D	DIR	L	L	L	L		L		L	D	D_DEV
R9	D	DIR	P	P	P	P		P		P	Z	D_DEV
R10	D	DIR	L	L	L	L		P		P	D	D_DEV
R11	D	F	P	L	L	P		L		L	Z	D_DEP
R12	D	F	P	L	L	P		P		P	D	D_DEV
R13	D	F	P	L	L	P		L		P	Z	D_DEP
R14	D	F	L	P	P	L	L		L		Z	E_DEP
R15	D	F	L	P	P	L	P		P		D	D_DEV
R16	D	F	L	P	P	L	L		P		Z	E_DEP
R17	D	F	P	P							D	Z
R18	D	F			P	P					D	Z

Tabela 8.1: Regras do comportamento de desvio inteligente de obstáculos

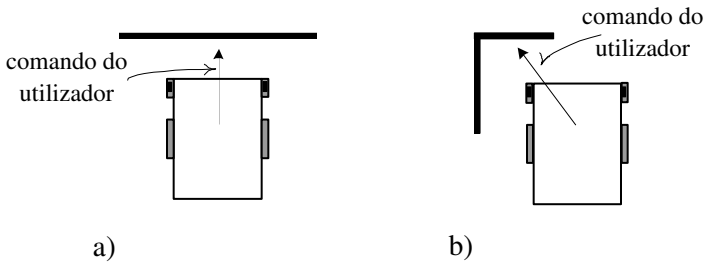


Figura 8.9: Cenários possíveis em que o controlador deve seguir o comando do utilizador sem efectuar desvio de obstáculos

d_e. A orientação é calculada a partir das distâncias medidas pelos sensores laterais IR8, IR12 para o lado esquerdo e IR9, IR13 para o lado direito. As variáveis de saída do controlador são a velocidade linear e angular. A figura 8.10a) ilustra as variáveis de entrada. A distância d_e corresponde a:

$$d_e = d_{actual} - d_{seguimento} \tag{8.3}$$

em que d_{actual} é a distância a que se encontra a cadeira (considera-se $d_{actual} = d_2$) e $d_{seguimento}$ é a distância a que se pretende seguir a superfície. A variável de entrada d_e poderá tomar valores negativos quando a cadeira se encontra demasiado próxima da superfície. A variável θ_e é dada por:

$$\theta_e = \arctan \frac{d_2 - d_1}{L} \tag{8.4}$$

em que d_2 e d_1 são as distâncias medidas pelos sensores IR8 e IR12, respectivamente, e L é a distância entre os dois sensores. Definem-se na figura 8.11 as novas funções de pertença e respectivos termos linguísticos para estas duas novas variáveis de entrada (Z, PN, PP, GP, GN definem respectivamente os termos ZERO, PEQUENO NEGATIVO, PEQUENO POSITIVO, GRANDE POSITIVO, GRANDE NEGATIVO). A tabela 8.2 define as regras que servem para a construção deste comportamento.

Apesar de não se estudar aqui o seguimento de trajectórias, é importante salientar que o funcionamento desse controlador seria muito semelhante ao do seguimento de superfícies. As variáveis de entrada seriam também a distância à trajectória e o erro de orientação relativo à trajectória pré-definida (ver figura 8.10b)).

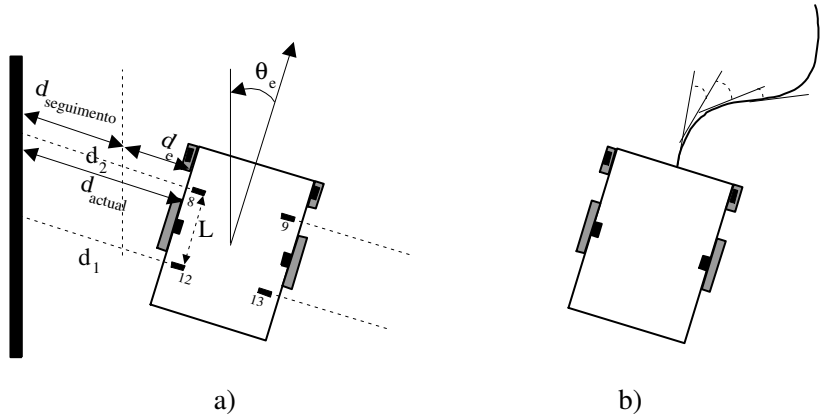


Figura 8.10: Entradas do controlador para: a) Seguimento de superfícies; b) Seguimento de trajectórias

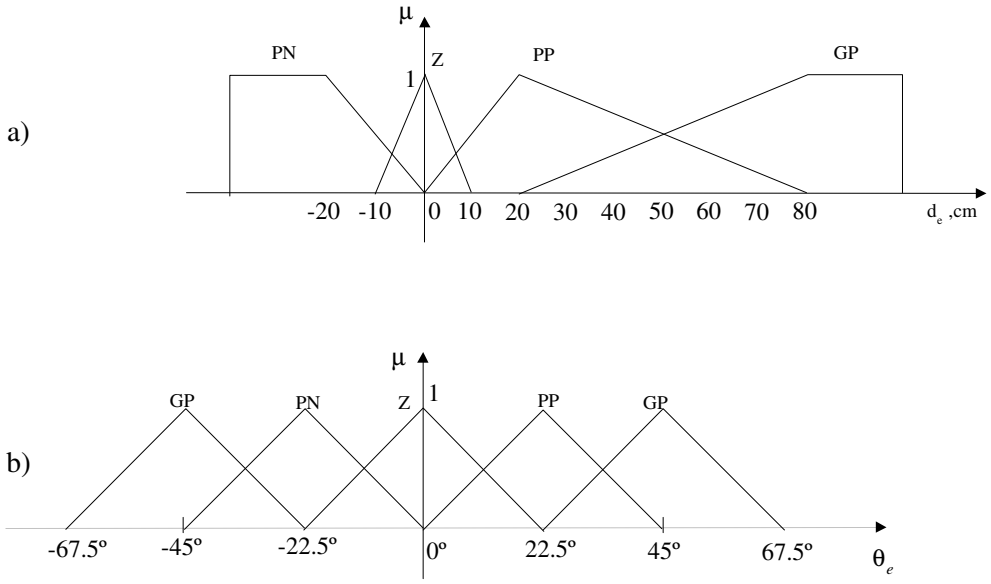


Figura 8.11: Funções de pertença das variáveis de entrada do comportamento de seguimento de superfícies (seguimento pela esquerda)

Regras	d_e	θ_e	v	w
R1	Z	Z	M	Z
R2	Z	PP	D	E_DEV
R3	Z	GP	Z	E_DEV
R4	Z	PN	D	D_DEV
R5	Z	GN	Z	D_DEV
R6	PP	Z	M	E_DEV
R7	PP	PN	D	Z
R8	PP	GN	D	Z
R9	PP	PP	Z	E_DEV
R10	PP	GP	Z	E_DEV
R11	GP	Z	D	E_DEV
R12	GP	PN	D	Z
R13	GP	GN	D	Z
R14	GP	PP	Z	E_DEV
R15	GP	GP	Z	E_DEV
R16	PN	Z	M	D_DEV
R17	PN	PP	D	Z
R18	PN	GP	D	Z
R19	PN	PN	Z	D_DEV
R20	PN	GN	Z	D_DEV

Tabela 8.2: Regras do comportamento de seguimento de superfícies (lado esquerdo)

8.7 Conclusões

Neste capítulo, formalizaram-se dois controladores: um controlador difuso para implementação de um comportamento de desvio inteligente de obstáculos; e um controlador difuso para implementação de um comportamento de seguimento de superfícies. É importante salientar que para a implementação prática destes controladores procederam-se a alguns ajustes e alterações, designadamente:

- Dada a gama limitada dos sensores de infravermelho analógicos ajustaram-se os conjuntos difusos relativos às variáveis de entrada e saída;
- Não se tiveram em conta algumas variáveis relativas aos sensores de infravermelho por se ter constatado que a sua contribuição era diminuta;
- Alteraram-se alguns comandos de direcção por voz.

Capítulo 9

Resultados

Este capítulo apresenta os resultados dos vários desenvolvimentos realizados na tese e discute alguns dos problemas surgidos. O ênfase vai para o sistema de navegação reactivo implementado através do controlador difuso descrito no capítulo 8. Por fim, apresentam-se conclusões e lançam-se novas linhas de desenvolvimento para trabalho futuro.

9.1 Implementações Mecânicas e Sensores

9.1.1 Mecânica

Em termos mecânicos, a cadeira não sofreu quaisquer alterações. Introduziu-se, no entanto, uma base giratória que serviu para o suporte do computador portátil e à qual está acoplada uma estrutura circular com os sensores de ultrassons. Esta possibilita a entrada e saída da cadeira simplesmente girando a base. Por outro lado, o utilizador, ou neste caso, o programador, também pode utilizar todas as funções do computador enquanto conduz a cadeira (figura 9.1).

9.1.2 Sensores

A figura 9.2 mostra os sensores de infravermelho analógicos e os sensores de ultrassons introduzidos na cadeira RobChair. Os sensores de ultrassons, apesar de instalados e testados, nunca foram utilizados. Os sensores de infravermelho analógicos permitem medir distâncias até um limite de 80 cm. No entanto, dada a localização de alguns dos sensores,

Figura 9.1: Base giratória para suporte do computador e ligação dos sonares

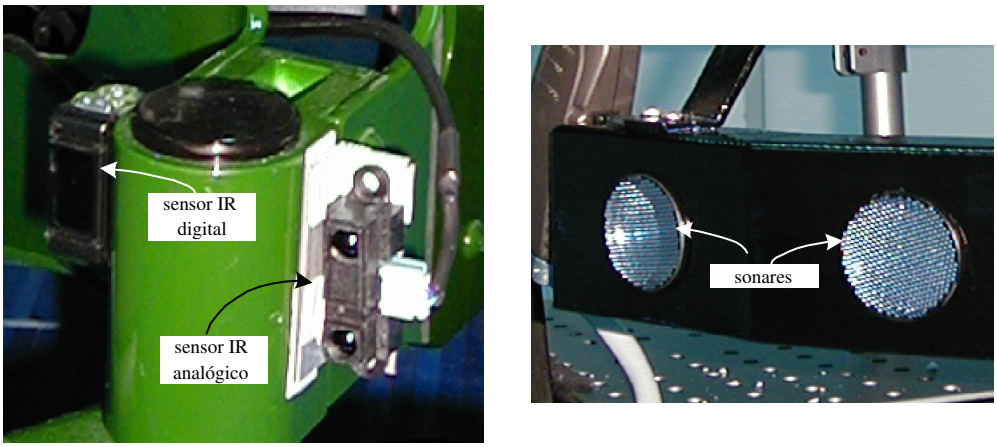
de $\approx 20 - 35$ cm para o interior do limite da cadeira, estes apenas permitem detectar obstáculos até uma distância de $\approx 45 - 60$ cm para além do limite da cadeira. Esta gama mostra-se reduzida para realizar alguns tipos de manobras tal como exemplifica a figura 9.3b).

Uma forma de ultrapassar a gama limitada dos sensores passaria pelo posicionamento dos sensores seguindo os limites da cadeira, aproveitando assim a sua gama máxima.

9.2 Odometria

Foram realizados vários percursos em laboratório para testar a capacidade de auto-localização da cadeira. Nos exemplos das figuras 9.4 e 9.5, a cadeira realizou percursos de ≈ 15 e 25 m, respectivamente. Observa-se que o cálculo da posição da cadeira apresenta um erro muito pequeno. Em termos de posição absoluta, o erro é inferior a 5 cm e em termos de orientação, é inferior a 2^o. Estas experiências foram realizadas com um utilizador, de cerca de 62 Kg, sentado na cadeira. Considerou-se um raio efectivo da roda de 170 mm.

Estes resultados satisfatórios levam-nos a concluir que o sistema odométrico pode ser utilizado com bastante confiança, quando usado com mecanismos de re-localização absoluta aplicados regularmente.



a) b)

Figura 9.2: Vista pormenorizada dos sensores de infravermelho e dos sonares

9.3 Condução Remota da Cadeira

A figura 9.6 mostra um exemplo de comando remoto da cadeira. A experiência foi realizada com a cadeira em linha de vista do utilizador. Apesar dos resultados satisfatórios da odometria, este tipo de informação não é suficiente para comandar remotamente (sem ser em linha de vista) a cadeira em segurança, pois, por um lado, o erro de localização é cumulativo e, por outro lado, podem existir objectos não actualizados no mapa da interface gráfica. Complementando a informação odométrica com a informação visual de câmaras e dotando o sistema com capacidade reactiva, seria possível efectuar tarefas de controlo remoto à distância.

9.4 Desvio de Obstáculos Inteligente

Realizaram-se várias experiências para testar a capacidade reactiva do sistema implementado por um controlador difuso. Todas estas experiências realizaram-se em ambiente de laboratório (figura 9.7). O utilizador conduziu a cadeira através dos comandos de voz descritos na tabela 9.1.

Nos percursos da figura 9.8, o utilizador comanda a cadeira através da voz, não possuindo o sistema capacidade reactiva. Os pequenos círculos que aparecem nas figuras correspondem aos locais onde o utilizador forneceu comandos de condução. Na expe-

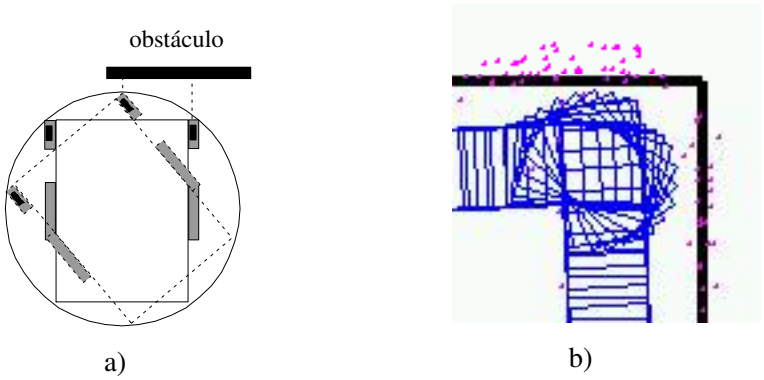


Figura 9.3: a) Círculo descrito pela cadeira aquando de uma rotação; b) Visualização gráfica da rotação da cadeira

Comando	Descrição
Frente/Trás	A cadeira movimenta-se para a frente/trás
Parar	A cadeira pára
Esquerda	A cadeira segue a direcção de 30º para a esquerda
Direita	A cadeira segue a direcção de 30º para a direita
RodaEsquerda	A cadeira roda continuamente para a esquerda
RodaDireita	A cadeira roda continuamente para a direita
Depressa	A velocidade é aumentada
Devagar	A velocidade é diminuída

Tabela 9.1: Comandos de voz à disposição do utilizador

riência da figura 9.8a) o utilizador conseguiu chegar ao local pretendido. Observa-se, no entanto, que teve de fornecer 15 comandos de voz, tendo ocorrido uma colisão frontal e uma lateral. Na experiência da figura 9.8b), o utilizador foi incapaz de passar entre os dois obstáculos iniciais do percurso.

As duas experiências da figura 9.9, foram desta vez realizadas com a capacidade reactiva do sistema. Em ambos os casos, o utilizador forneceu apenas 5 comandos de voz.

Estas experiências demonstram que, a condução da cadeira com capacidade reactiva, exigiu muito menos esforço e atenção por parte do utilizador, descrevendo a cadeira uma trajectória mais suave.

9.4.1 Identificação de Problemas

Foram identificados alguns problemas, que podem conduzir a algumas situações de colisão, nomeadamente:

1. Gama limitada dos sensores de infravermelho - gama insuficiente para alguns tipos de manobras. Este problema foi referido na secção 9.1.2.

2. Número insuficiente de sensores laterais e traseiros - um exemplo corrente de colisão acontece quando a parte dianteira da cadeira já transpôs um obstáculo, tendo de seguida de efectuar uma rotação (figura 9.10). A falta de sensores laterais leva normalmente à não detecção do obstáculo que acabou de ultrapassar.

9.5 Passagem de Porta

A passagem de porta fez-se de forma reactiva, ou seja, sem qualquer planeamento de trajectória. O controlador não actua de forma especial no caso de passar uma porta. Esta é simplesmente vista como o espaço entre dois obstáculos. Com base no comando do utilizador, a cadeira terá de passar entre esses obstáculos. A figura 9.11 ilustra algumas tentativas para passar uma porta com 85 cm de largura. Nas figuras 9.11a) b), o utilizador tenta passar a porta sem ser assistido pelo módulo reactivo. Na primeira experiência, o utilizador não consegue realizar a manobra de passagem. Na segunda experiência, o utilizador atravessa a porta, tendo fornecido 5 comandos. Note-se que o sucesso desta experiência deve-se principalmente à aproximação frontal da cadeira.

Nas experiências das figuras 9.11c) d) o utilizador consegue passar a porta fornecendo apenas um comando de voz. Uma vez mais, se mostra a importância do módulo reactivo.

É importante referir o porquê da dificuldade em passar uma porta usando simples de comandos de voz. Apontam-se os dois principais motivos:

1. O comando de condução deve ser dado no momento exacto. Um pequeno atraso pode levar a que a cadeira tome uma direcção não desejada;

2. A direcção que a cadeira toma não é a direcção fornecida pelo utilizador. Este problema deve-se às rodas livres dianteiras que, após rotações acentuadas, tomam uma posição perpendicular em relação ao sentido da cadeira. Assim, quando após

uma rotação acentuada, o utilizador pretende seguir em frente, a cadeira segue com um desvio superior a 30º.

O módulo reactivo do sistema consegue minimizar estes dois problemas, pois mesmo que os comandos não sejam fornecidos no momento certo, ou quando a cadeira toma direcções incorrectas, o controlador difuso corrige essas manobras.

9.6 Conclusões e Trabalho Futuro

9.6.1 Conclusões

Nesta tese foram apresentados e discutidos os principais métodos de navegação reactiva aplicados em robótica móvel. A capacidade reactiva de um sistema fornece ao veículo formas de reagir em tempo real a estímulos do meio ambiente.

A cadeira RobChair foi equipada com sensores de infravermelho analógicos e codificadores nas rodas, permitindo o desenvolvimento de algoritmos de navegação e o cálculo da posição absoluta da cadeira.

Dada a necessidade de execução em tempo real dos algoritmos, desenvolveu-se uma arquitectura de *software* distribuída. As tarefas de baixo nível são executadas num microcontrolador que comunica com o PC via RS-232. Por sua vez, no PC existem vários processos que comunicam entre si através de memória partilhada. Introduziu-se uma interface gráfica que permite a visualização dos movimentos da cadeira assim como da informação sensorial. Esta interface permite ainda o comando remoto da cadeira.

Desenvolveu-se uma aplicação de reconhecimento de fala que permite comandar a cadeira através de um conjunto de 9 comandos de voz. No entanto, a condução da cadeira usando comandos de voz mostrou-se uma tarefa muito difícil, especialmente em espaços relativamente apertados. Esta dificuldade levou ao desenvolvimento de um módulo de navegação reactiva cujo objectivo foi o de assistir o utilizador na condução da cadeira. O módulo de navegação reactiva foi implementado através de um controlador difuso que combina os comandos do utilizador com a informação sensorial, fornecendo manobras de desvio inteligente de obstáculos.

9.6.2 Trabalho Futuro

Sugerem-se várias linhas de desenvolvimento futuro, entre as quais destacamos as seguintes:

- Testar a integração de novos sensores (infravermelho e ultrassons) para melhorar a capacidade de navegação reactiva;
- Depois de implementado o módulo de navegação reactiva, é agora necessário desenvolver um módulo de planeamento de trajectórias locais e de seguimento de trajectórias. Este módulo poderá ser utilizado para executar autonomamente algumas tarefas, tais como passagem de porta e aproximação a mesas. A implementação deste módulo requer a construção de mapas e, por isso, o uso de sensores de ultrassons (trabalho já iniciado no ISR);
- Introduzir no sistema a capacidade de planeamento global; estudar formas de integrar informação local com informação global; e por fim, combinar planeamento local e capacidade reactiva.

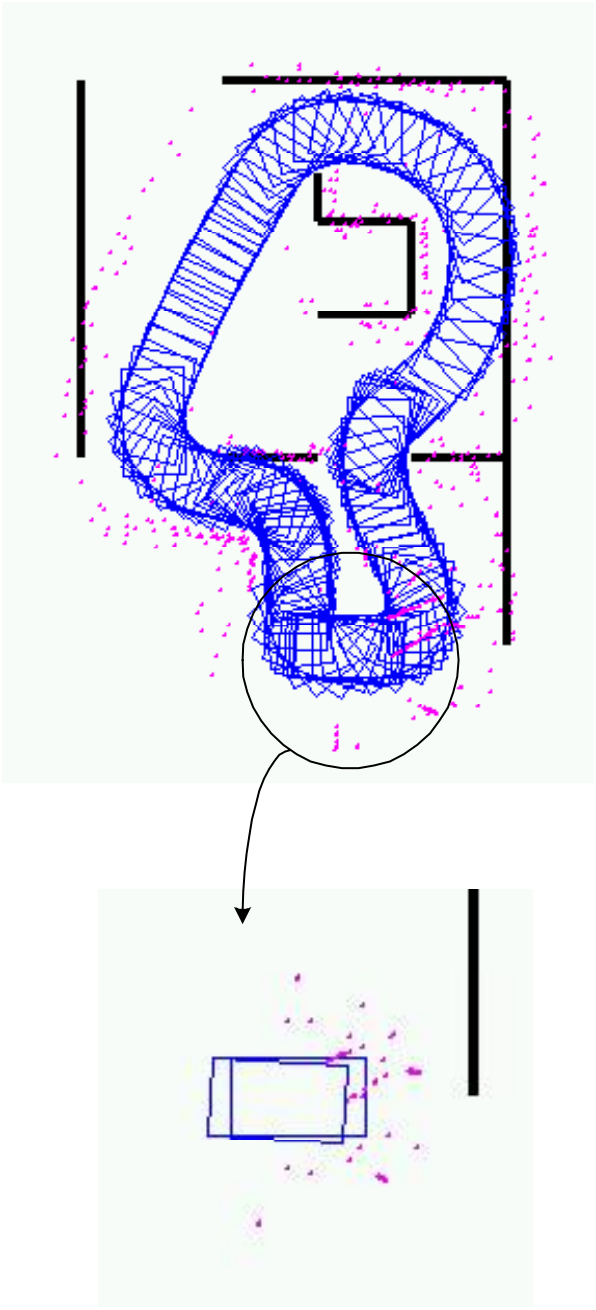


Figura 9.4: Percurso de $\approx$ 15 m realizado pela cadeira (em cima); Vista pormenorizada da diferença entre a posição final real e calculada (em baixo)

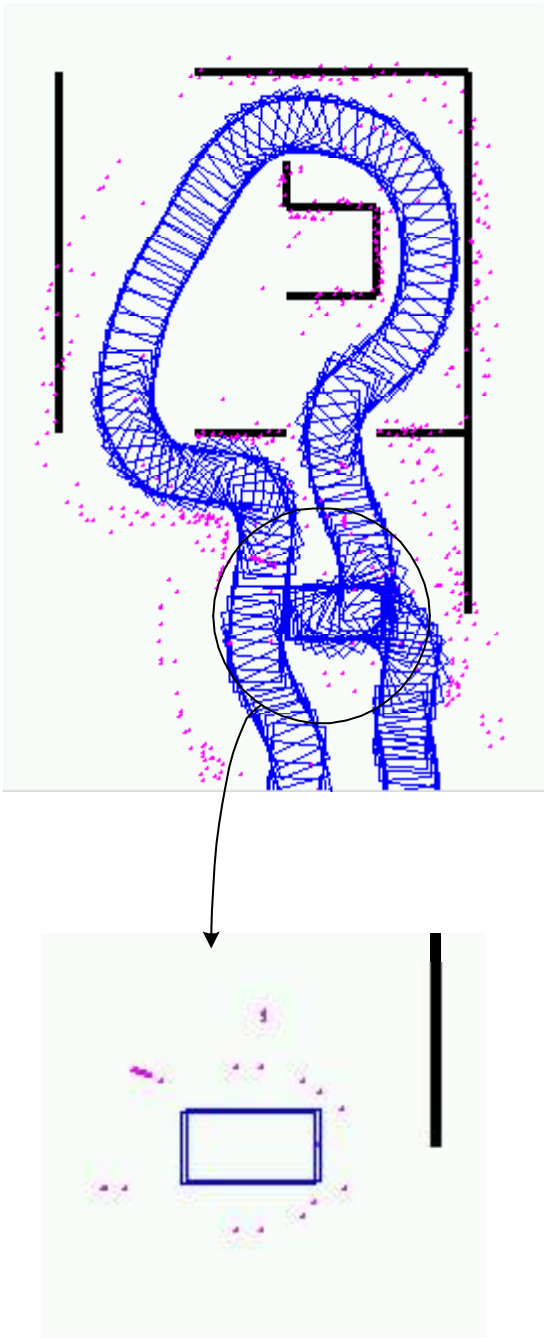
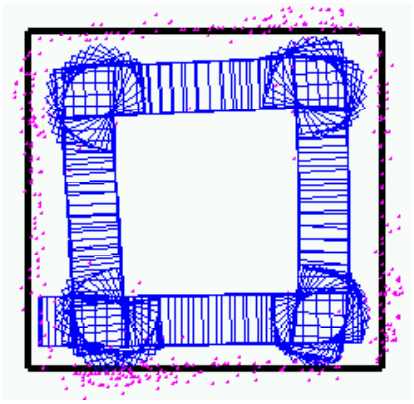


Figura 9.5: Percurso de ≈ 25 m realizado pela cadeira (em cima); Vista pormenorizada da diferença entre a posição final real e calculada (em baixo)



a)

b)

Figura 9.6: a) Exemplo do percurso descrito pela cadeira quando comandada remotamente; b) Operador remoto e ambiente de operação

Figura 9.7: Laboratório onde foram realizados as experiências. Utilizador a conduzir a cadeira com voz

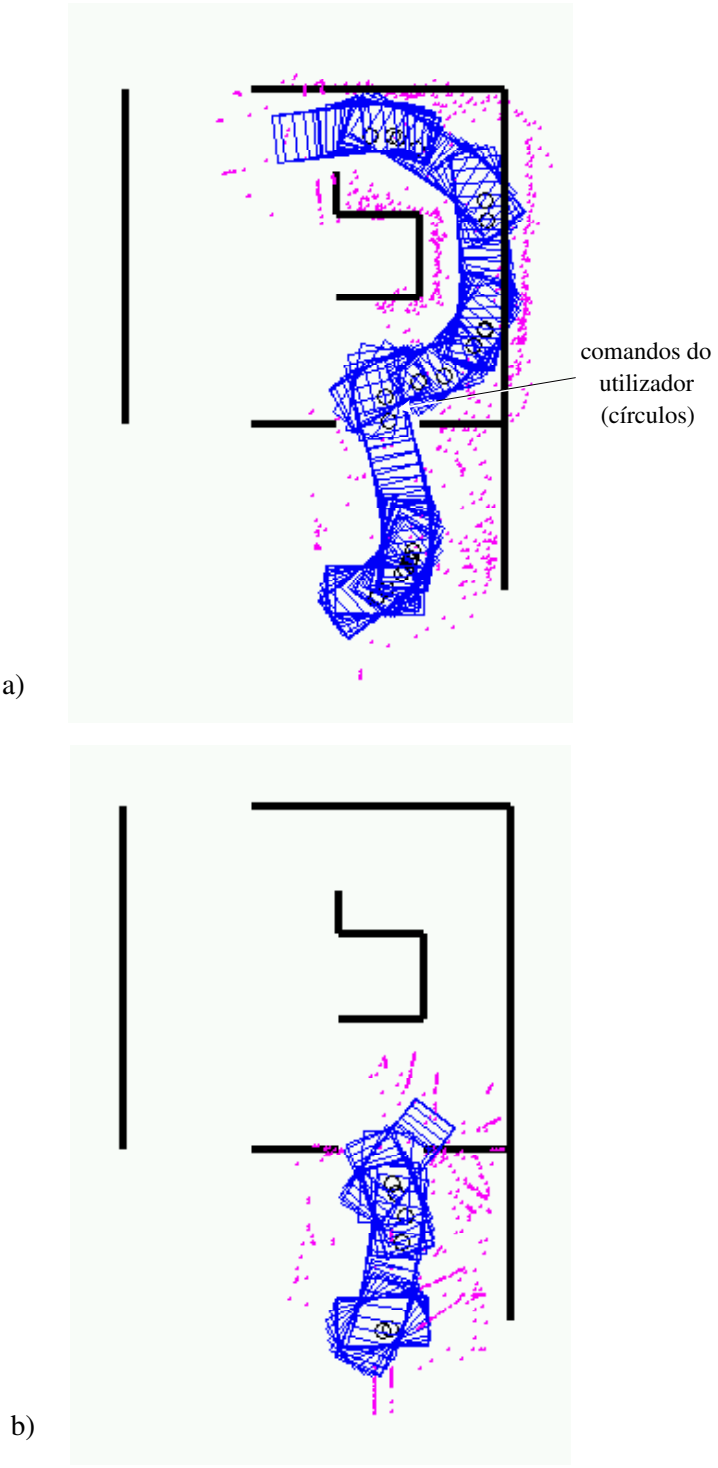
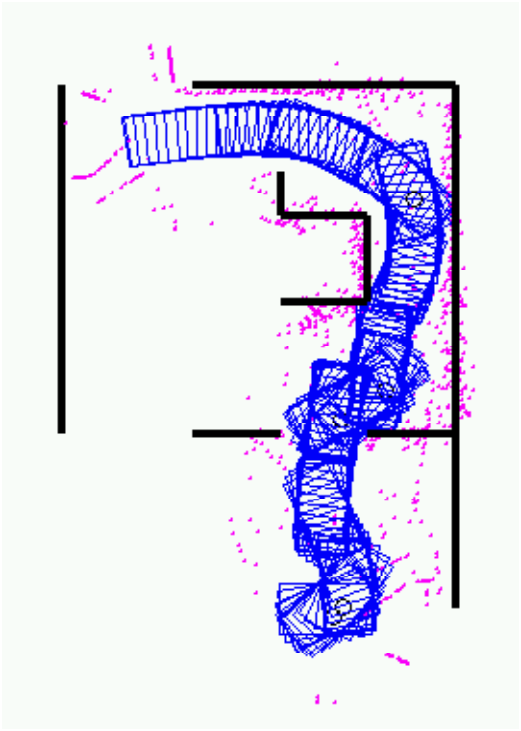


Figura 9.8: Percurso realizado através de comandos de voz, com o sistema sem capacidade reactiva. Os pequenos círculos indicam os pontos em que foram emitidos comandos do utilizador

a)



b)

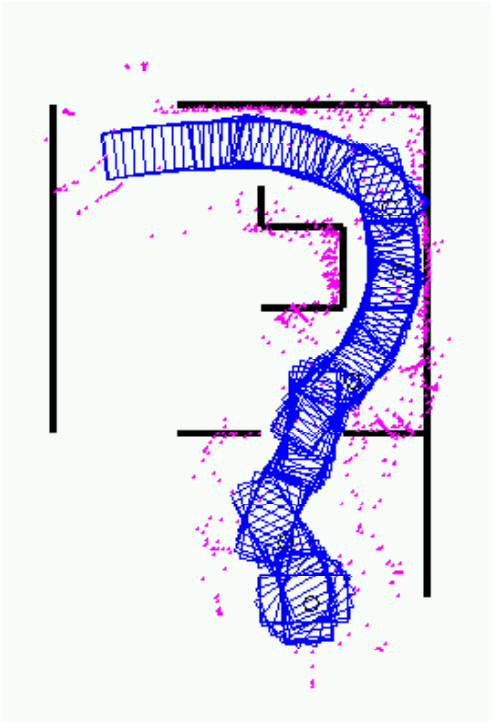


Figura 9.9: O utilizador conduz a cadeira através de comandos de voz, mas com a assistência do controlador difuso. Os pequenos círculos indicam os pontos em que foram emitidos comandos do utilizador

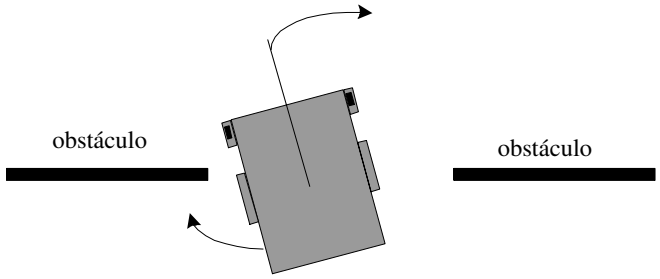


Figura 9.10: Cenário típico de colisão da parte traseira da cadeira

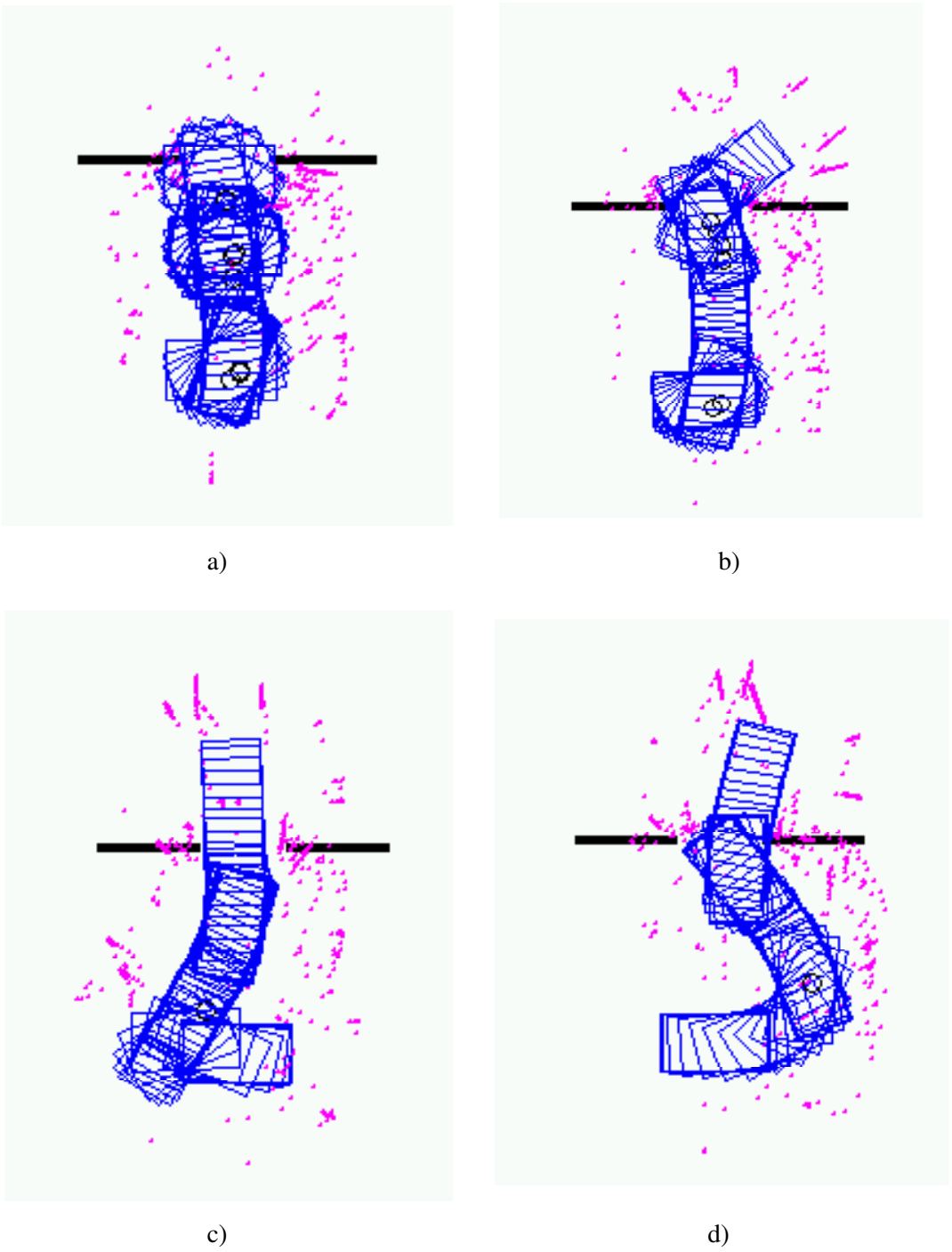


Figura 9.11: Passagem de porta sem capacidade reactiva do sistema (a) e (b); Passagem de porta com capacidade reactiva do sistema (c) e (d). Os pequenos círculos indicam os pontos em que foram emitidos comandos do utilizador

Apêndice A

Controlo por Lógica Difusa

Este apêndice fornece ao leitor o formalismo e ferramentas matemáticas necessários à compreensão da lógica difusa e descreve os vários módulos de um controlador difuso. As referências bibliográfica que constituíram a base destas notas encontram-se em [HMB93] [Ped93] [Jan91] [Ara00].

A.1 Conceito de Lógica Difusa e Controlo difuso

A teoria dos conjuntos difusos e da lógica difusa foi apresentada por Lotfi Zadeh em 1965. Procurou captar e representar a imprecisão das variáveis e dos conceitos que nos rodeiam. Vejamos um exemplo simples ilustrado na figura A.1 que dá uma interpretação difusa e não difusa de temperatura quente de uma sala. Segundo a interpretação clássica, qualquer temperatura acima dos 20º é considerada quente e abaixo dos 20º não-quente. Considerando a interpretação difusa, existe uma transição gradual que transmite a ideia de que aos 20º é 50% verdade que a temperatura da sala está quente, aumentando este grau de verdade para temperaturas superiores. Suponhamos ainda, como outro exemplo, a forma como pensamos quando estamos a conduzir um carro e temos de travar para cortar num cruzamento. Não pensamos "começo a travar 20 metros antes do cruzamento", mas sim "começo a travar à medida que me aproximo do cruzamento".

O controlo difuso utiliza o formalismo da lógica difusa para simular o raciocínio humano. O controlador guia-se por um conjunto de regras verbais apresentadas tipicamente no formato *se-então* do tipo:

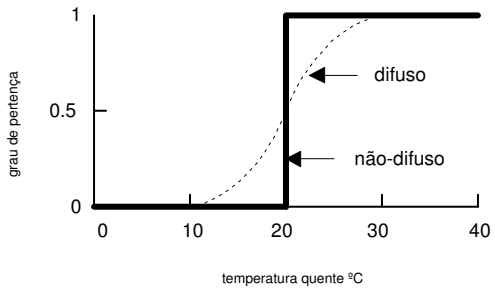


Figura A.1: Interpretação difusa e não difusa de temperatura quente

SE temperatura quente E aumenta ligeiramente ENTÃO aumenta ligeiramente ar fresco

Estas regras são normalmente construídas com base na observação das acções de controlo de operadores e peritos.

A.2 Conjuntos Difusos

Seja U um conjunto clássico ao qual é chamado o universo de discurso e A um sub-conjunto de U, ou seja $A \subset U$. Na teoria clássica de conjuntos, a função de pertença associada a A é definida por:

$$X_A : U \mapsto 0, 1$$
$$X_A(u) = \begin{cases} 1 & se \quad u \in U \\ 0 & se \quad u \notin U \end{cases} \tag{A.1}$$

Definição A.1 (Conjunto Difuso). *Um conjunto difuso num Universo do discurso U é caracterizado pela função de pertença μ_A, que toma valores no intervalo [0,1], nomeadamente $\mu_A : U \mapsto [0, 1]$. Um conjunto difuso A em U pode ser representado como um conjunto de pares ordenados de um elemento genérico u e o seu grau de pertença $\mu_A(u)$:*

$$A = \{(u, \mu_A(u)) : u \in U\} \tag{A.2}$$

Se A é um **conjunto discreto**, pode ser expresso por:

$$A = \mu_A(u_1)/u_1 + ... + \mu_A(u_n)/u_n = \sum_{i=1}^{n} \mu_A(u_i)/u_i \tag{A.3}$$

onde o sinal '+' corresponde ao operador de união da teoria dos conjuntos difusos e não à soma aritmética (a notação '$\sum$' não simboliza o sinal de somatório, indica uma colecção

de pontos e respectivos valores de pertença). A barra oblíqua '/' não significa divisão, mas sim um grau de pertença para um dado valor do universo do discurso.

Se A é um **conjunto contínuo** (u é contínuo) pode ser expresso na forma:

$$A = \int_U \mu_A(u)/u \tag{A.4}$$

Uma vez mais, a notação '$\int$' não simboliza o sinal de integral, indicando uma colecção de pontos e respectivos valores de pertença.

A.2.1 Funções de Pertença

Cada elemento no Universo do discurso tem um grau de pertença associado a um conjunto difuso, podendo este valor ser zero. A função que faz corresponder um número (no intervalo [0,1]) a cada elemento u do Universo é chamada *função de pertença* $\mu(u)$. Existem duas formas alternativas de representar uma função de pertença no computador: contínua ou discreta. Na forma contínua, a função de pertença é uma função matemática podendo assumir a forma de uma curva em forma de sino, curvas em S, em Z (S invertida), triangular ou trapezoidal. Na forma discreta, a função de pertença e o Universo têm de ser implementados como pontos discretos numa lista (vector). Ambas as formas são válidas. A forma contínua é mais exigente em termos de processamento, mas menos exigente a nível de armazenamento de dados do que a forma discreta.

Definição A.2 (Suporte, ponto de inflexão e singleton difuso). *O Suporte $S(A)$, de um conjunto difuso A, é o conjunto de todo o $u \in U : \mu_A(u) > 0$. Em particular, o elemento u de U para o qual $\mu_A(u) = 0.5$ é chamado o* ***ponto de inflexão*** *e o conjunto difuso cujo suporte é um único ponto em U com $\mu_A(u) = 1$ é chamado* ***singleton difuso****, ou seja, é uma variável determinística.*

Tipos de Funções de Pertença

As funções de pertença triangulares têm tido boa aceitação entre os utilizadores de lógica difusa em detrimento das funções quadráticas ou cúbicas. A suavidade introduzida por conjuntos difusos de maior ordem não se reflecte na qualidade de saída do modelo difuso. Se, no entanto, for assumido que a saída do modelo difuso está directamente dependente da forma do conjunto difuso, então deve-se seguir uma metodologia para a escolha da função

de pertença. As curvas em forma de sino são normalmente utilizadas para conjuntos difusos espalhados em torno de um valor central (servem para definir expressões do género "à volta de" ou "perto de"). Apresentam-se a seguida algumas das funções de pertença mais conhecidas.

Curva Gaussiana - (figura A.2) define-se por dois parâmetros: valor em torno do qual a curva é construída (γ) e o valor que indica a largura desta curva em forma de sino (k). A curva Gaussiana não é limitada pelo que se poderá dizer que esta função não possui suporte compacto. O valor da curva para cada ponto u do domínio é dado por:

$$G(u; k, \gamma) = e^{-k(\gamma-u)^2} \tag{A.5}$$

Curva em S (Sigmóide)- (figura A.3) é definida por três parâmetros: o valor com grau de pertença zero (α), o valor com grau de pertença 1 (γ) e o ponto de inflexão (β) que corresponde a 50% de verdade. O valor da curva para cada ponto u do Universo é dado por:

$$S(u; \alpha, \beta, \gamma) = \begin{cases} 0 & u \leq \alpha \\ 2(\frac{u-\alpha}{\gamma-\alpha})^2 & \alpha \leq u \leq \beta \\ 1 - 2(\frac{u-\gamma}{\gamma-\alpha})^2 & \beta \leq u \leq \gamma \\ 1 & u \geq \gamma \end{cases} \tag{A.6}$$

Curva em Z - corresponde ao espelho da função S em relação a um eixo vertical.

Curva-Π (curva em forma de sino)- (figura A.4) é obtida a partir da combinação da curva em S e da curva em Z. A equação da curva é dada por:

$$\Pi(u; \beta, \gamma) = \begin{cases} S(u; \gamma - \beta, \gamma - \frac{\beta}{2}, \gamma) & u \leq \gamma \\ 1 - S(u; \gamma, \gamma + \frac{\beta}{2}, \gamma + \beta) & u > \gamma \end{cases} \tag{A.7}$$

Rectângulo - (figura A.5) a função de pertença é um conjunto rígido (não difuso).

Conjunto em ombro esquerdo e direito - (figura A.6) as funções são definidas respectivamente por:

$$T(u; a, b) = \begin{cases} 1 & u < a \\ \frac{b-u}{b-a} & a \leq u \leq b \\ 0 & u > b \end{cases} \tag{A.8}$$

$$T(u;a,b) = \begin{cases} 0 & u < a \\ \frac{u-a}{b-a} & a \leq u \leq b \\ 1 & u > b \end{cases} \tag{A.9}$$

Trapezoidal - (figura A.7) esta função obtém-se a partir da combinação das funções em ombro. A equação desta função é dada por.

$$T(u;a,b,c,d) = \begin{cases} 0 & u < a \\ \frac{u-a}{b-a} & a \leq u \leq b \\ 1 & b < u < c \\ \frac{d-u}{d-c} & c \leq u \leq d \\ 0 & u > d \end{cases} \tag{A.10}$$

Triangular - (figura A.8) é dada pela seguinte equação:

$$T(u;a,b,c) = \begin{cases} 0 & u < a \\ \frac{u-a}{b-a} & a \leq u \leq b \\ \frac{c-u}{c-b} & c \leq u \leq d \\ 0 & u > c \end{cases} \tag{A.11}$$

Singleton difuso - (figura A.9) o singleton difuso é, como já foi referido atrás, um conjunto difuso cujo suporte é um único ponto em U com $\mu_A = 1$.

Para se obter a representação discreta equivalente às funções de pertença que acabámos de enunciar, substitui-se na função de pertença os valores discretos do Universo. Por exemplo, suponhamos que o Universo é formado pelos 5 valores equidistantes $0, 2, 4, 6, 8$. Se inserirmos estes valores na função de pertença triangular, obtemos $\mu(u) = 0, 0.5, 1, 0.5, 0$.

A.2.2 Variáveis Linguísticas e Termos

Do mesmo modo que as variáveis algébricas tomam números como valores, as variáveis linguísticas difusas tomam palavras ou frases. Ao conjunto de valores que as variáveis podem tomar, chama-se conjunto de **termos**. Cada valor no conjunto de termos é caracterizado por um conjunto difuso definido no universo de discurso onde a variável é definida.

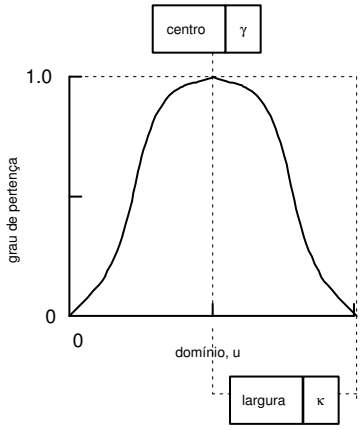


Figura A.2: Função de pertença de curva Gaussiana

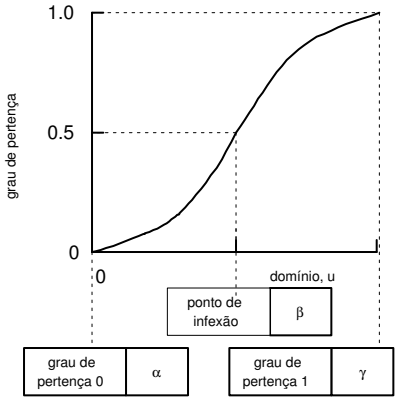


Figura A.3: Função de pertença da curva em S

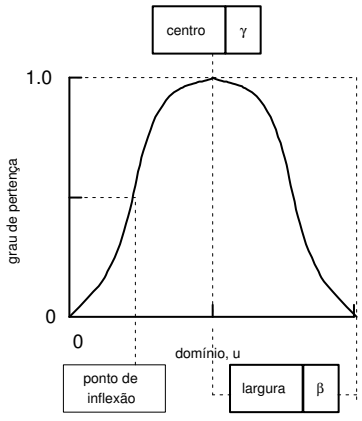


Figura A.4: Função de pertença da curva em Π

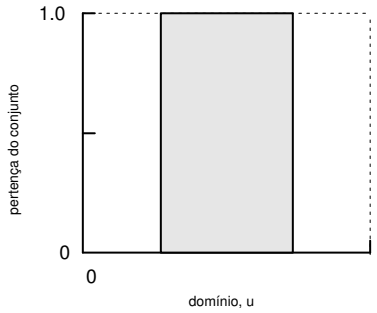


Figura A.5: Conjunto rígido (não difuso)

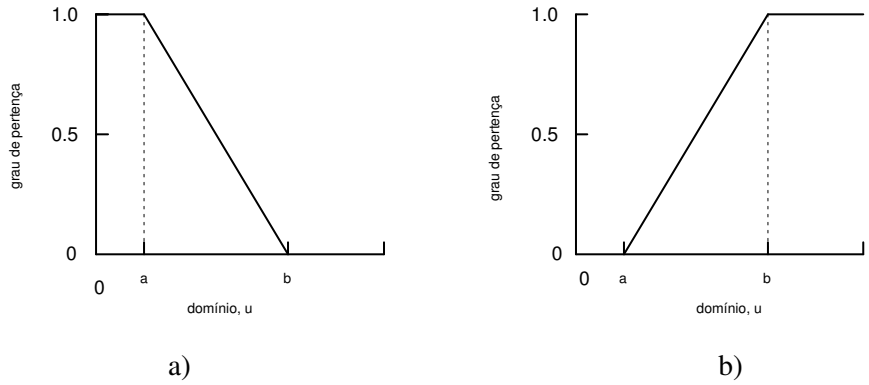


Figura A.6: Função de pertença do conjunto em ombro: a) Ombro esquerdo; b) Ombro direito

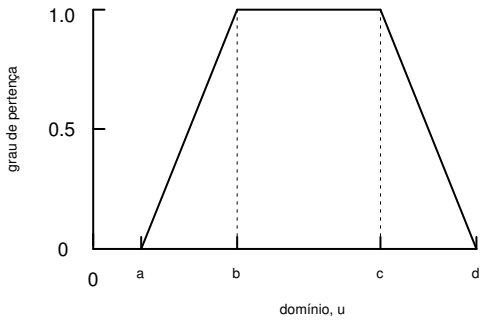


Figura A.7: Função de pertença do conjunto trapezoidal

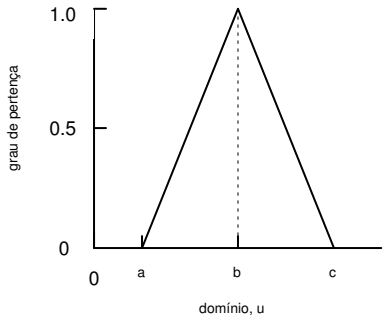


Figura A.8: Função de pertença do conjunto triangular

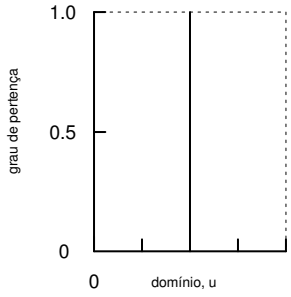


Figura A.9: Função de pertença do singleton difuso

Definição A.3 (Variáveis Linguísticas e Termos). *Uma variável linguística é definida por $(u, T(u), U, S)$, no qual u é o nome da variável; $T(u)$ é o conjunto de termos de u ou o conjunto de nomes de valores linguísticos de u; U é o Universo do discurso; e S é a regra semântica que associa cada termo linguístico de $T(u)$ com um conjunto difuso em u.*

Por exemplo, se a velocidade, u, é a variável linguística, então o conjunto de termos T(velocidade) poderá ser:

$$\text{T(velocidade)} = \text{devagar, médio, depressa}$$

Chama-se **termo primário** ao termo ou conjunto que deve ser definido *a priori*, por exemplo, depressa ou devagar. Se definirmos os termos muito depressa, não muito depressa, chamar-se-ia a estes conjuntos, **conjuntos modificados**.

Definição A.4 (Modificadores Linguísticos). *Um modificador linguístico é uma operação que modifica o significado de um termo. Desta forma, gera-se um conjunto maior de valores para uma variável linguística, a partir de uma pequena colecção de termos primários. Exemplos de modificadores são: muito, mais ou menos, bastante. Por exemplo, as expressões MUITO QUENTE, MAIS OU MENOS QUENTE geram novos conjuntos a partir do termo primário QUENTE. Os modificadores são muitas vezes aproximados pelas operações:*

$$\textit{muito } u \triangleq \mu^2(u)$$
$$\textit{mais ou menos } u \triangleq \mu^{\frac{1}{2}}(u)$$
$$\textit{bastante } u \triangleq \mu(u + C)$$

Operações Elementares com Conjuntos Difusos

Definiremos a seguir as três operações elementares com conjuntos difusos: intersecção ('E'), união ('OU') e complemento ('NAO').

Sejam A, B e C três conjuntos difusos definidos em U com funções de pertença $\mu_A(u)$, $\mu_B(u)$ e $\mu_C(u)$ respectivamente.

Para a apresentação das propriedades destas operações utiliza-se a seguinte notação: $x = \mu_x(u), y = \mu_y(u), z = \mu_z(u), w = \mu_w(u)$.

Definição A.5 (Intersecção ou conjunção -'E'). *A função de pertença* $\mu_{A\cap B}(u)$ *da intersecção (ou mais genericamente da conjunção)* $A \cap B$ *é definida por:*

$$\mu_{A\cap B}(u) = \mu_A(u) t \mu_B(u), u \in U \tag{A.12}$$

onde t corresponde à norma-t e é definida a seguir:

Definição A.6 (Norma-t (norma triangular)). *A norma-t é uma função de dois argumentos* $[0,1] \times [0,1] \mapsto [0,1]$ *que goza das seguintes propriedades:*

comutativa	$xty = ytx$
associativa	$xt(ytz) = (xty)tz$
condições limite	$xt0 = 0$ *e* $xt1 = x$
monotonia	$xty \leq wtz$ *se* $x \leq w$ *e* $y \leq z$

A norma-t inclui um conjunto de operadores entre os quais os mais utilizados são: Mínimo:

$$xty = x \wedge y = min(x, y) \tag{A.13}$$

Produto algébrico:

$$xty = x * y = x.y \tag{A.14}$$

Definição A.7 (União - 'OU'). *A função de pertença* $\mu_{A\cup B}(u)$ *da união (ou mais genericamente da disjunção)* $A \cup B$ *é definida por:*

$$\mu_{A\cup B}(u) = \mu_A(u) s \mu_B(u), u \in U \tag{A.15}$$

onde s corresponde à norma-s e é definida a seguir:

Definição A.8 (Norma-s (co-norma triangular)). *A norma-s é uma função de dois argumentos* $[0,1] \times [0,1] \mapsto [0,1]$ *que goza das seguintes propriedades:*

comutativa	$xsy = ysx$
associativa	$xs(ysz) = (xsy)sz$
condições limite	$xs0 = x$ *e* $xs1 = 1$
monotonia	$xsy \leq wtz$ *se* $x \leq w$ *e* $y \leq z$

A norma-s inclui um conjunto de operadores entre os quais os mais utilizadas são: Máximo:

$$xsy = x \vee y = max(x, y) \tag{A.16}$$

Soma algébrica:

$$xsy = x \dot{+} y = x + y - xy \tag{A.17}$$

Definição A.9 (Complemento - 'NAO'). *A função de pertença de $\mu_A(u)$ do complemento de um conjunto difuso A é dado por:*

$$\mu_{\overline{A}}(u) = \mu_{naoA}(u) = 1 - \mu_A(u)\ , u \in U \tag{A.18}$$

A.2.3 Relações

Em sistemas de controlo, as relações definem-se entre entradas e saídas do sistema. Em sistemas difusos, estas relações ou mapeamentos definem-se entre variáveis difusas (definidas ou não em Universos diferentes) através de regras (implicações) do género:

$$A \Rightarrow B \text{ ou SE } A(u) \text{ ENTÃO } B(v)$$

que ligam o conjunto antecedente (conjunto A) ao consequente (conjunto de saída B). Uma relação difusa para declarações condicionais é dada pelo produto cartesiano entre A e B ($A \times B$), que corresponde a todas as combinações possíveis entre os item de cada Universo. O produto cartesiano é definido por:

$$A \times B = \int_{U \times V} \mu_A(u) t \mu_B(v)/(u, v) \tag{A.19}$$

onde $U \times V$ é a colecção de pares ordenados (u, v), tal que $U \times V = (u, v)/u \in U, v \in V$. Por exemplo, supondo que U=$\{1,\ 2,\ 3\}$; V=$\{1,\ 2,\ 3,\ 4\}$; $\mu_A(u)/u = 1/1 + 0.7/2 + 0.2/3$; $\mu_B(v)/v = 0.8/1 + 0.6/2 + 0.4/3 + 0.2/4$; então utilizando o operador de produto algébrico na equação A.19 obtemos:

$A \times B = 0.8/(1,1) + 0.6/(1,2) + 0.4/(1,3) + 0.2/(1,4) + 0.56/(2,1) + 0.42/(2,2) + 0.28/(2,3) + 0.14/(2,4) + 0.16/(3,1) + 0.12/(3,2) + 0.8/(3,3) + 0.04/(3,4)$

genericamente esta relação pode ser apresentada na forma matricial seguinte:

$$\begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \end{bmatrix} \times [b_1 b_2 \cdots b_m] = \begin{bmatrix} a_1 \wedge b_1 & a_1 \wedge b_2 & \cdots & a_1 \wedge b_m \\ \vdots & & \vdots & \\ a_n \wedge b_1 & a_n \wedge b_2 & \cdots & a_n \wedge b_m \end{bmatrix} \tag{A.20}$$

onde $\wedge$ denota a operação de intersecção. Para o exemplo anterior, teríamos:

'E'	v	0.8	0.6	0.4	0.2
u					
1		0.8	0.6	0.4	0.2
0.7		0.56	0.42	0.28	0.14
0.2		0.16	0.12	0.8	0.04

Definição A.10 (Produto cartesiano). *Sejam $A_1, \cdots, A_n$ sub-conjuntos difusos de respectivamente $U_1, ..., U_n$, então o produto cartesiano de $A_1, \cdots, A_n$ é um sub-conjunto difuso no espaço de produto $U_1 \times \cdots \times U_n$ cuja função de pertença é definida por:*

$$\mu_{A1\times\cdots\times An}(u_1, \cdots, u_n) = \mu_{A1}(u1) \star \cdots \star \mu_{An}(u_n) \tag{A.21}$$

onde '$\star$' representa um operador de norma-t. Utilizando o operador min e produto obteríamos respectivamente:

$$\mu_{A1\times\cdots\times An}(u_1, \cdots u_n) = min(\mu_{A1}(u_1), \cdots, \mu_{An}(u_n)) \tag{A.22}$$

e

$$\mu_{A1\times\cdots\times An}(u_1, \cdots u_n) = \mu_{A1}(u_1).\mu_{A2}(u_2), \cdots, \mu_{A_n}(u_n) \tag{A.23}$$

Definição A.11 (Relação Difusa). *Uma relação difusa, R, é um sub-conjunto difuso de $U_1 \times U_2 \cdots \times U_n$ dada por:*

$$R_{U_1\times U_2\cdots\times U_n} = \mu_R(u_1, \cdots, u_n)/(u_1, \cdots, u_n) \in U_1 \times \cdots \times U_n$$

Definição A.12 (Composição). *Sejam R e S duas relações binárias difusas em $U \times V$ e $V \times W$, respectivamente (considerando R e S representadas por matrizes, o número de colunas de R deverá ser igual ao número de linhas S). A composição é dada por:*

$$R \circ S$$

onde '$\circ$' é o sinal de composição que define a operação 'OU'-'E' (e.g. max-min).

A composição é semelhante ao produto clássico de matrizes, em que a multiplicação é substituída pelo operação 'E' (norma-t) e a soma pela operação 'OU' (norma-s). Supondo R dado pela matriz $m \times p$ e S pela matriz $p \times n$, a composição é dada pela matriz $m \times n$ $T = (t_{ij})$, tal que:

$$t_{ij} = (r_{i1} \wedge s_{1j}) \vee (r_{i2} \wedge s_{2j}) \vee \cdots \vee (r_{ip} \wedge s_{pj}) = \bigvee_{k=1}^{p} (r_{ik} \wedge s_{kj}) \tag{A.24}$$

Os símbolos $\wedge$ e $\vee$ representam as operações 'E' e 'OU' respectivamente. Duas das composições mais utilizadas são a composição $max - min$ e composição $max - produto$. Vejamos um exemplo de aplicação que analisa o grau de parecença entre os familiares da família Donald [Jan91] (o objectivo é descobrir o grau de parecença entre Huey e Donald). Temos os seguintes domínios discretos: $U = \{Hey\}$, $V = \{Dewey, Louie\}$ e $W = \{Donald\}$. A relação P em $U \times V$ é dada por:

	Dewey	Louie
Huey	0.8	0.9

e a relação Q em $V \times W$ é dada por:

	Donald
Dewey	0.5
Louie	0.6

ou seja:
$P = 0.8/(Huey, Dewey) + 0.9/(Huey, Louie)$
$Q = 0.5/(Dewey, Donald) + 0.6/(Louie, Donald)$

Utilizando a composição $max - min$ o grau de parecença entre Huey e Donald é dado por:

$$\begin{bmatrix} 0.8 & 0.9 \end{bmatrix} \circ \begin{bmatrix} 0.5 \\ 0.6 \end{bmatrix} = max\{min(0.8, 0.5), min(0.9, 0.6)\} = \begin{bmatrix} 0.6 \end{bmatrix} \tag{A.25}$$

A.2.4 Implicação e Inferência

Num controlador difuso, o conhecimento vem expresso através de regras difusas do tipo SE-ENTÃO:

SE <proposição difusa> ENTÃO <proposição difusa>

em que as proposições difusas podem ser atómicas ou compostas. Uma proposição atómica é uma afirmação única do tipo:

x é A

onde x é a variável linguística e A é o valor linguístico de x. Uma proposição difusa composta é uma composição de proposições atómicas usando conectivas 'E', 'OU' e 'NÃO' para representar as operações de intersecção, união e complemento, respectivamente. Um exemplo de uma proposição composta é:

x é A E y é B

A regra:

SE u é grande ENTÃO v é grande

corresponde a uma implicação, na medida em que o valor de u implica o valor de v. A implicação descreve uma relação entre estas duas variáveis. A operação de implicação difusa pode ser interpretada por analogia à lógica clássica, segundo a qual:

$$(p \Rightarrow q) \Longleftrightarrow (p \wedge q) \vee \overline{p} \tag{A.26}$$

$$(p \Rightarrow q) \Longleftrightarrow (\overline{p} \vee q) \tag{A.27}$$

Substituindo os operadores $\wedge, \vee,$ pelas operações difusas de intersecção, união e complemento, respectivamente, teremos diferentes interpretações de implicações. Seguindo a expressão A.26 e sabendo que a regra se aplica quando p é V, resulta:

$$(p \Rightarrow q) \Longleftrightarrow (p \wedge q) \tag{A.28}$$

A implicações de Mamdani seguem esta ideia. A implicação:

SE $A(u)$ ENTÃO $B(v)$

traduz-se na relação difusa R em $U \times V$:

$$\mu_R(u,v) = \mu_{A\times B}(u,v) = min\{\mu_A(u), \mu_B(v)\}, u \in U, v \in V \tag{A.29}$$

ou

$$\mu_R(u,v) = \mu_{A\times B}(u,v) = \mu_A(u).\mu_B(v), u \in U, v \in V \tag{A.30}$$

em que $\wedge$ foi substituído pelo operador *min* e pelo operador de *produto*, respectivamente em cada uma das expressões. Estes operadores de implicação são os mais comummente utilizados.

Definição A.13 (Implicação). *Sejam A e B dois conjuntos difusos, não necessariamente no mesmo universo. A implicação A implica B é:*

$$A \Rightarrow B \triangleq A \times B$$

em que '$\times$' representa o produto cartesiano.

Definimos atrás a relação resultante de uma regra. Suponhamos agora um algoritmo composto por N regras:

$$\begin{aligned} &\text{R1: SE } A_1(u) \text{ ENTÃO } B_1(v) \\ &\text{R2: SE } A_2(u) \text{ ENTÃO } B_2(v) \\ &\qquad\vdots \\ &\text{Rn: SE } A_n(u) \text{ ENTÃO } B_n(v) \end{aligned}$$

Para este algoritmo, a relação R é dada pela disjunção das relações de cada uma das regras individuais:

$$\mu_R(u,v) = \bigcup_{i=1,\cdots N} \mu_{Ai}(u) t \mu_{Bi}(v) \tag{A.31}$$

em que t representa uma norma-t. Utilizando os operadores de mínimo e máximo a expressão fica:

$$\max_{i=1\cdots N} min(\mu_{Ai}(u), \mu_{Bi}(v)) \tag{A.32}$$

Em termos de matrizes cada regra é expressa por uma matriz de implicação e o total das regras corresponde a fazer um 'OU' das matrizes de cada regra item a item.

Inferência

De forma a tirar conclusões de uma base de regras é necessário um mecanismo que produza uma saída a partir de uma colecção de regras SE-ENTÃO. Uma das regras de inferência mais conhecida na lógica clássica é chamada *modus ponens*. Esta regra pode ser enunciada da seguinte forma: Se a afirmação $a \Rightarrow b$ é verdade, e se a afirmação **a** é verdade, então pode-se inferir que **b** é verdade. Na lógica difusa esta regra é generalizada em *modus ponens generalizado* (MPG):

Premissa 1: SE x é A ENTÃO y é B
Premissa 2: x é A'

Consequência: y é B'

Os conjuntos A' e B' são, de alguma forma, diferentes de A e B (por exemplo pela aplicação de modificadores). A inferência MPG baseia-se na regra composicional de inferência.

Definição A.14 (Regra composicional de inferência). *Seja R uma relação do universo $U \times V$, e A um conjunto difuso definido em U, então:*

$$A \circ R = b \tag{A.33}$$

é o conjunto de V induzido por A, e '$\circ$' é a operação de composição.

A.3 Controlador Difuso

Um controlador difuso aplica a lógica difusa para controlo de sistemas dinâmicos. É utilizado essencialmente em sistemas para os quais não se possui um modelo matemático satisfatório. A estratégia do controlador difuso é definida por uma base de regras, a qual é de fácil concepção, baseado em conhecimento heurístico. A figura A.10 apresenta a constituição genérica de um controlador difuso.

A.3.1 Módulo de Difusão

O módulo de difusão tem por objectivo transformar as variáveis numéricas em variáveis/conjuntos difusos de modo a poderem ser manipuladas pelo controlador. Esses conjuntos difusos poderão ser quaisquer dos conjuntos já apresentados na secção A.2.1.

A.3.2 Módulo de Inferência

O módulo de inferência é certamente o coração do controlador difuso, pois tem por objectivo calcular os comandos para controlar um dado processo. A definição do módulo de inferência passa por definir uma base de regras, um operador de inferência e as conectivas ('E', 'OU' e 'NAO'). A base de regras é definida por regras do tipo:

$$R^{(l)}\text{: } \underbrace{\text{SE } x_1 \text{ é } A_1^l \text{ E } \cdots \text{ E } x_n \text{ é } A_n^l}_{antecedente} \underbrace{\text{ENTÃO } y \text{ é } B^l}_{consequente}$$

onde:

- $x_1, \ldots, x_n \in U$ são as variáveis linguísticas de entrada e $y \in V$ é a variável linguística de saída
- $A_1^l (i = 1, \ldots, n)$ são conjuntos difusos em U_i
- B^l são os conjuntos de saída em V
- $l = 1, \ldots, M$, em que M é o número de regras do sistema difuso

Utilizando a seguinte base de regras:

R1: SE **x** é A_1 E **y** é B_1 ENTÃO **z** é C_1 OU
R2: SE **x** é A_2 E **y** é B_2 ENTÃO **z** é C_2 OU
$\vdots$
Rn: SE **x** é A_n E **y** é B_n ENTÃO **z** é C_n

apresentam-se os métodos de raciocínio difuso (motor de inferência difusa) mais utilizados:

- **max-min (definido por Mamdani):**
  A conectiva 'E' corresponde ao operador *min* e operação de implicação corresponde ao operador *min*:

$$\mu_{Ci}(z) = min(min(\mu_{Ai}(x_0), \mu_{Bi}(y_0)), \mu_{Ci}(z)) \tag{A.34}$$

  A consequência final C resulta da União (operador max) do resultado de cada uma das regras:

$$\mu_C(z) = max(\mu_{C1}(z), \cdots, \mu_{Cn}(z)) \tag{A.35}$$

- **soma-produto (definido por Mizumoto [Miz]):**
  A conectiva 'E' e a operação de implicação correspondem ao operador de *produto*:

$$\mu_{Ci}(z) = \mu_{Ai}(x_0).\mu_{Bi}(y_0).\mu_{Ci}(z) \tag{A.36}$$

  A consequência final resulta da soma de cada uma das regras:

$$\mu_C(z) = \mu_{C1}(z) + \cdots + \mu_{Cn}(z) \tag{A.37}$$

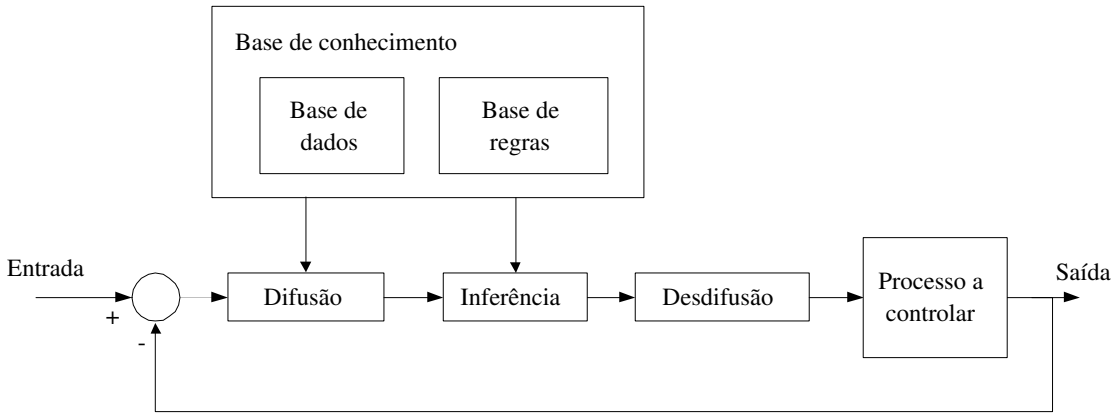


Figura A.10: Elementos de um controlador difuso

- **max-produto:** este controlador utiliza o operador de produto algébrico para a implicação, mas utiliza o operador max para combinar as conclusões de cada uma das regras.

- **Singleton soma-min ou soma-produto:** se as variáveis de saída forem singleton difusos, teremos ainda outra abordagem ao método de Mizumoto. Usa a implicação do tipo produto ou min (não faz diferença uma vez que se trata de singletons) e usa o operador soma para combinar as saídas das regras. O conjunto de saída é dado por:

$$\alpha_1.s_1 + \alpha_2.s_2 + \cdots + \alpha_n.s_n \tag{A.38}$$

  em que os α correspondem aos graus de activação das n regras e $s_1 \cdots s_n$ são as saídas do tipo singleton.

Vejamos a interpretação gráfica destes métodos, para o caso de termos apenas as regras R1 e R2 definidas atrás (ver figura A.11). Os valores x_0 e y_0 correspondem às entradas do controlador. No caso do controlador do tipo Mamdani, o grau de activação das regras R1 e R2 será respectivamente dado por $h_1 = \mu_{A1}(x_0) \wedge \mu_{B1}(y_0)$ e $h_2 = \mu_{A2}(x_0) \wedge \mu_{B2}(y_0)$. Considerando $\mu_{A1}(x_0) = 0.8$, $\mu_{B1}(y_0) = 0.3$, $\mu_{A2}(x_0) = 0.6$ e $\mu_{B2}(y_0) = 0.9$, a regra R1 é activada com grau de verdade 0.3 e R2 com grau de verdade 0.3 e R2 com 0.6, ou seja os conjuntos consequentes são limitados por esses valores através do operador min para a implicação. As funções de pertença da parte consequente das regras são finalmente combinadas através do operador max.

Utilizando o controlador proposto por Mizumoto, o grau de activação das regras é dado pela multiplicação dos graus de activação de cada um dos conjuntos antecedentes. As regras R1 e R2 serão activadas com grau 0.24 e 0.54 respectivamente (através do operador produto). As regras são depois combinadas através da sua soma. Em alguns controladores a operação *min* poderá também ser utilizada para a conectiva 'E' (em vez do produto) sendo o produto utilizado apenas para o operação de implicação.

O controlador proposto por Mizumoto apresenta algumas vantagens em relação ao controlador do tipo Mamdani:

- O nível de activação no controlador Mamdani mantém-se 0.3, desde que o $\mu_{A1}(x_0) \geq 0.3$, ou seja, nesta gama o resultado é independente do nível de activação. Utilizando o controlador do tipo soma-produto, o nível de activação da regra dependerá do nível de activação tanto de μ_A como de μ_B. Preserva-se assim no resultado final a forma da função de pertença da saída (em vez de a cortar) e o grau de activação da regra depende das contribuições do grau de activação de cada um dos antecedentes.

- A utilização do operador *max* para combinar as regras, faz com que o resultado final não seja influenciado pelo número de regras com o mesmo resultado. Mesmo que hajam várias regras a chegar às mesmas conclusões, o resultado da união dos conjuntos resultantes permanece igual. Se se utilizar a soma para combinar ao resultado das regras, cada uma delas contribuirá com um peso. Poderá acontecer que a soma ultrapasse o grau de pertença 1. Nesta situação poder-se-á utilizar uma soma normalizada, no entanto, o valor de desdifusado será igual utilizando a soma normalizada ou não.

- A utilização do controlador do tipo singleton utiliza as mesmas operações que o controlador do tipo Mizumoto. Apresenta algumas vantagens em relação aos outros métodos [Jan91]: 1) é calculado mais facilmente e com maior rapidez; e 2) pode ser mais intuitivo escrever as regras. Não apresenta desvantagens pelo que este tipo de controlador é cada vez mais utilizado.

A.3.3 Módulo de Desdifusão

O módulo de desdifusão tem por objectivo calcular um valor numérico (uma saída não difusa) a partir do conjunto difuso obtido no módulo de inferência. Este valor será então

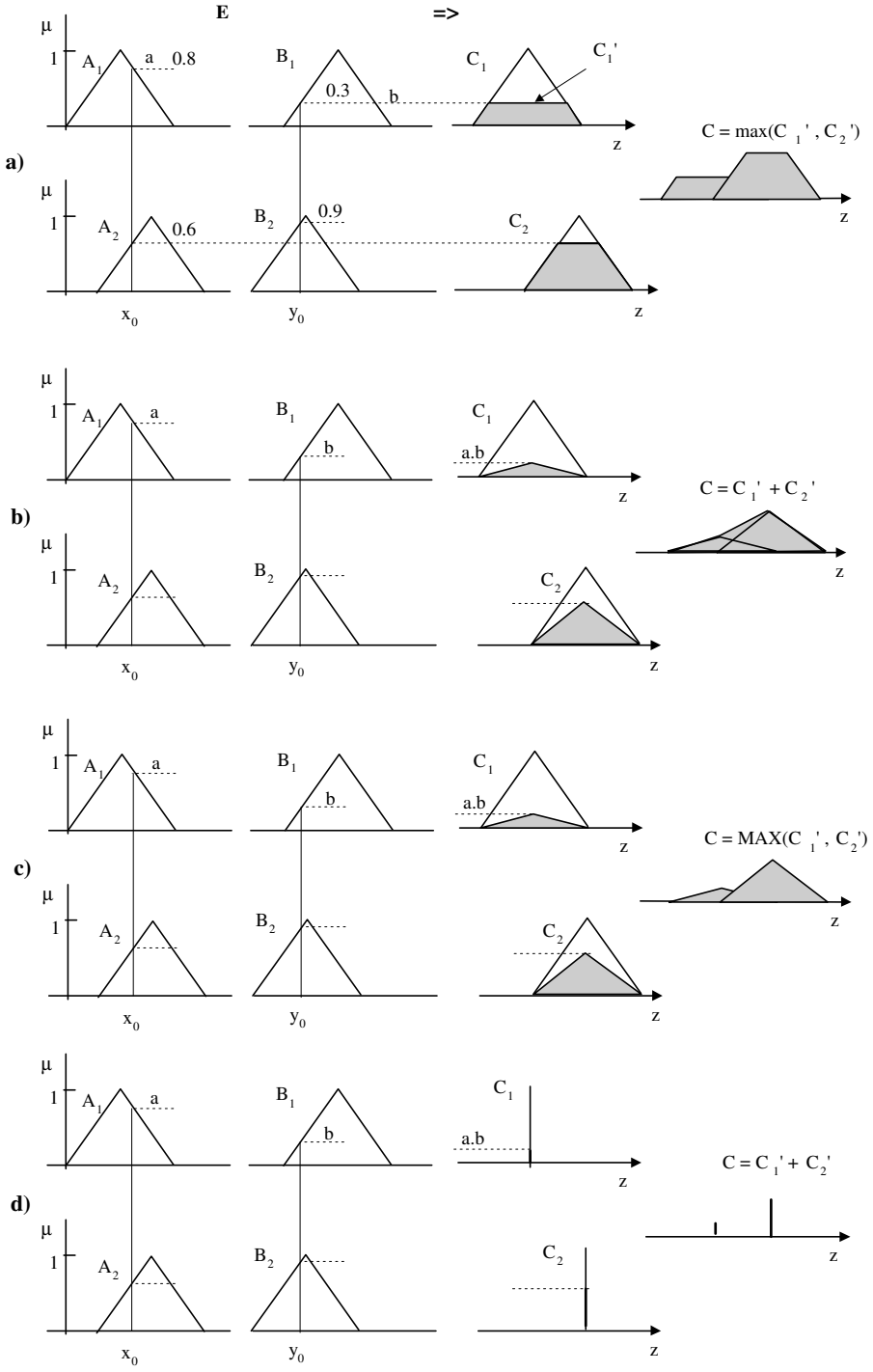


Figura A.11: Interpretação gráfico de métodos de inferência: a) max-min; b) soma-produto; c) max-produto; e d) soma-produto singleton

enviado para controlar o processo. Os métodos de desdifusão mais conhecidos são:

Média dos Máximos(MDM): neste método de desdifusão, escolhe-se o ponto do conjunto com maior grau de pertença, e se existir vários pontos máximos, toma-se a média dos máximos, ou seja:

$$u = \frac{\sum x|\mu(x) = max(\mu(x)))}{m} \tag{A.39}$$

em que m é o número de pontos máximos.

Primeiro dos Máximos e último dos Máximos (PDM, UDM): se o conjunto difuso tiver vários picos, pode ser vantajoso escolher um dos picos. Neste caso o método de desdifusão consiste em escolher o menor dos máximos (PDM) ou o maior dos máximos (UDM).

Centro de Gravidade (CDG) : neste método o valor desdifusado corresponde à soma pesada do conjunto suporte. O valor é dado por:

$$u = \frac{\sum_i \mu(x_i)x_i}{\sum_i \mu(x_i)} \tag{A.40}$$

CDG aplicado a saídas do tipo singleton: neste caso o valor desdifusado corresponde a:

$$u = \frac{\sum_i \alpha_i.s_i}{\sum_i \alpha_i} \tag{A.41}$$

Os dois últimos métodos são os que apresentam melhores resultados pois levam em atenção a distribuição do conjunto difuso.

Apêndice B

Esquemáticos de Hardware

Sensores infravermelho analógicos	Portos			
	A/D	Endereço		Multiplexador
		A	B	
		I/O 1	I/O 0	
IR_a_0	0	0	0	AD0X0
IR_a_1	0	0	1	AD0X1
IR_a_2	0	1	0	AD0X2
IR_a_3	0	1	1	AD0X3
IR_a_4	1	0	0	AD1Y0
IR_a_5	1	0	1	AD1Y1
IR_a_6	1	1	0	AD1Y2
IR_a_7	1	1	1	AD1Y3
IR_a_8	2	0	0	AD2X0
IR_a_9	2	0	1	AD2X1
IR_a_10	2	1	0	AD2X2
IR_a_11	2	1	1	AD2X3

Tabela B.1: Portos de ligação aos sensores de infravermelhos analógicos

Sensores infravermelhos digitais	Portos			
	A/D	Endereço		Multiplexador
		A	B	
		I/O 1	I/O 0	
IR_d_0	3	0	0	AD3Y0
IR_d_1	3	0	1	AD3Y1
IR_d_2	3	1	0	AD3Y2
IR_d_3	3	1	1	AD3Y3
IR_d_4	4	0	0	AD4X0
IR_d_5	4	0	1	AD4X1
IR_d_6	4	1	0	AD4X2
IR_d_7	4	1	1	AD4X3
IR_d_8	5	0	0	AD5Y0
IR_d_9	5	0	1	AD5Y1
IR_d_10	5	1	0	AD5Y2
IR_d_11	5	1	1	AD5Y3

Tabela B.2: Portos de ligação aos sensores de infravermelho digitais

joystick	Portos			
	A/D	Endereço		Multiplexador
		A	B	
		I/O 1	I/O 0	
J_REF	6	0	0	AD6X0
J_VEL	6	0	1	AD6X1
J_ANG	6	1	0	AD6X2

Tabela B.3: Portos de Ligação ao *joystick*: J_REF (referência), J_VEL (velocidade linear) e J_ANG (velocidade angular)

Sensores	Portos			
	A/D	Endereço		Multiplexador
		A	B	
		I/O 1	I/O 0	
BUMP_L	6	1	1	AD6X3
BUMP_R	7	0	0	AD7Y0
R/B BUT_ANG	7	0	1	AD7Y1
FROB	7	1	0	AD6X2
n. u.	7	1	1	AD6X2

Tabela B.4: Portos de Ligação ao pára-choques (L_BUMP e R_BUMP), botões (R/B BUT) e potenciómetro (FROB). (n.u. - não usado)

Sonares	Porto	ECHO
SON1	TPU0	ECHO 0
SON2	TPU1	ECHO 1
SON3	TPU2	ECHO 2
SON4	TPU3	ECHO 3
SON5	TPU4	ECHO 4
SON6	TPU5	ECHO 5
SON7	TPU6	ECHO 6

Tabela B.5: Portos de ligação aos sensores de ultrassons: sinal ECHO

Sonares	Entrada multipl.	Endereço			Saída multiplex.
		C	B	A	
	I/O 5	I/O 4	I/O 3	I/O 2	
SON1	1	0	0	0	INIT0
SON2	1	0	0	1	INIT1
SON3	1	0	1	0	INIT2
SON4	1	0	1	1	INIT3
SON5	1	1	0	0	INIT4
SON6	1	1	0	1	INIT5
SON7	1	1	1	0	INIT6

Tabela B.6: Portos de ligação aos sensores de ultrassons: controlo do sinal INIT

Codificadores		Porto
esquerdo	pulsos	TPU8
	sentido	TPU15
direito	pulsos	TPU6
	sentido	TPU7

Tabela B.7: Portos de ligação aos codificadores

Motores		Porto
esquerdo	vel. linear	TPU12
	sentido	TPU10
direito	vel. angular	TPU11
	sentido	TPU9

Tabela B.8: Portos utilizados para geração dos comandos de direcção

Pino	Sinais do *joystick*
1	Sreen
2	Battery +24V
3	J Bus
4	Joystick Select
5	Speed Pot +Ref
6	Battery sw +24V
7	Joystick Vtop
8	Joystick Vcenter
9	Response Pot Wiper
10	TruCharge
11	Joystick Bottom
12	Response Pot +Ref
13	Speed Pot Wiper
14	Joystick Speed
15	Battery 0V
16	Joystick Direction

Tabela B.9: Pinagem do *joystick*

Bibliografia

[Ara00] Rui Araújo. *Controlo Difuso e Aprendizagem*. Relatório interno do Departamento de Engenharia Electrotécnica - Coimbra, 2000.

[Ark87] R. C. Arkin. Motor schema navigation for a mobile robot: An approach to programming by behavior. *Proceedings of the IEEE Conference on Robotics and Automation*, pages 264–270, 1987.

[Ark90] R. C. Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, 6:105–122, 1990.

[Ark98] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, 1998.

[BBL+94] D.A. Bell, J. Borenstein, S. P. Levine, Y. Koren, and L. Jaros. An assistive navigation system for wheelchairs based upon mobile robot obstacle avoidance. *IEEE Conference on Robotics and Automation*, pages 2018–2022, 1994.

[BHH+95] U. Borgolte, R. Hoelper, H. Hoyer, H. Reck, W. Humann, J. Nedza, I. Craig, R. Vallegi, and A.M. Sabatini. Intelligent control of a semi-autonomous omnidirectional wheelchair. *3rd International Symposium on Intelligent Robotic Systems'95*, pages 113–120, 1995.

[BK89] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man and Cybernetics*, 19(5):1179–1187, 1989.

[BK91] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[BLK+94] D.A. Bell, S. P. Levine, Y. Koren, L. A. Jaros, and J. Borenstein. Design criteria for obstacle avoidance in a shared-control system. *In proc. of the conf. RESNA'94 annual Conference*, pages 581–583, 1994.

[BMG+99] L. Bergasa, M. Mazo, A. Gardel, J. Garcia, A.-Ortuno, and A. Mendez. Guidance of a wheelchai for handicapped people by face tracking. *7th International Conference on Emerging Technologies and Factory Automation*, pages 105–111, 1999.

[Bro86] A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[Car95] J. Carretas. Associação portuguesa de deficientes - cadeiras de rodas. 1995.

[CN00] J. Cruz and U. Nunes. Generating local maps for mobile robots. *Controlo2000: 4th Portuguese Conference on Automatic Control*, pages 412–417, 2000.

[Co.] British Encoder Products Co. Disponível na página: http://www.brit-encoder.com.

[Com98] Nomadic Communications. Mercury user's guide version 2.0, 1998.

[Dra95] Dragon. *Dragon Voice Tools Developer's Kit*. 1995.

[Elf87] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):249–265, 1987.

[fPR] Kiss Institue for Practical Robots. Disponível na página: http://www.kipr.org.

[Gil] Penny & Giles. Disponível na página: http://www.penny-giles-controls.co.uk/.

[HMB93] C.J. Harris, C.G. Moore, and M. Brown. *Intelligent Control - Aspects of Fuzzy Logic and Neural Nets*. World Scientific in Robotics and Automated Systems, 1993.

[HS90] G. Hanke and G. Shafer. Applying optical measurement techniques. *Sensor Review*, 10:30–34, 1990.

[Jan91] J. Jantzen. *Fuzzy Control -lectures notes.* Publication nr. 9109 Technical University of Denmark, 1991.

[KB91] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1398–1404, 1991.

[KBB+94] K. Kawamura, M. Bishay, S. Bagchi, A. Saad, M. Iskarous, and M. Fumoto. Intelligent user interface for a rehabilitation robot. *4th International Conference on Rehabilitation Robotics*, pages 31–35, 1994.

[Kha86] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5:90–98, 1986.

[KMRS97] K. Konolige, K. Mayers, E. Ruffini, and A. Safiotti. The saphira architecture: A design for autonomy. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 9:215–235, 1997.

[KT86] B. Krogh and C.E. Thorpe. Integrated path planning and dynamic steering control for autonomous vehicles. *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1664–1669, 1986.

[KTP+97] N.I. Katevas, N.M. Sgouros. S.G. Tzafestas, G. Papakonstantnou, P. Beattie, J.M. Bishop, P. Tsanakas, and D. Koutsouris. The autonomous mobile robot senario: A sensor-aided intelligent navigation system for powered wheelchairs. *IEEE Robotics and Automation Magazine*, pages 60–70, December 1997.

[Lab] Newton Research Labs. Disponível na página: http://www.newtonlabs.com.

[Lat91] J. C. Latombe. *Robot Motion Planning.* Kluwer Academics Publishers, 1991.

[Lee89] Kai-Fu Lee. *Automatic Speech recognition - The development of SPHINX System.* Kluwer Academics Publishers, 1989.

[LH97] C. Lopes and N. Honório. *Comando por Voz de uma Cadeira de Rodas Robotizada.* Relatório interno do Departamento de Engenharia Electrotécnica - Coimbra, 1997.

[LHCL99] R. Luo, C Hu, T. Chen, and M. Lin. Force reflective feedback control for intelligent wheelchairs. *IEEE International Conference on Intelligent Robots and Systems*, pages 918–923, 1999.

[LLS+94] T. Laengle, T. Lueth., E. Stopp, G. Herzog, and G. Kamstrup. Kantra - a natural language interface for intelligent robots. *3rd IEEE Int. Workshop on Robot and Human Communication, RO.MAN'94*, pages 106–111, 1994.

[Miz] M. Mizumoto. Fuzzy controls under product-sum gravity methods and new fuzzy control methods. *Control Systems*, pages 276–294.

[MNdA99] L. Marques, U. Nunes, and A. T. de Almeida. Sensor optoelectrónico reflectivo com potência de emissão variável e detecção digital. *Anais da Engenharia e Tecnologia Electrónica*, IV(7):64–68, 1999.

[Mot95] Mot. Mc68332 user's manual. 1995.

[NPC00] U. Nunes, G. Pires, and P. Coelho. Assistive navigation control architecture. *Systems and Control: Theory and Applications, N. Mastorakis (Ed.), WSE Press*, pages 38–43, 2000.

[Nye90] Adrian Nye. *X Protocol Manual - Volume Zero*. O' Reilly Associate Inc, 1990.

[Ons95] Ons. Tattletale model 8 - installation and operation manual. 1995.

[PANA98] G. Pires, R. Araújo, U. Nunes, and A. T. Almeida. Robchair: A powered wheelchair using a behaviour-based navigation. *5th IEEE Int. Workshop on Advanced Motion Control (AMC'98)*, pages 536–541, 1998.

[Ped93] W. Pedrycz. *Fuzzy control- second extended edition*. Department of Electrical and Computer Engineering, University of Manitioba, Canadá, 1993.

[Pol] Polaroid. Polaroid datasheet.

[RL99] T. Rofer and A. Lankenau. Ensuring safe obstacle avoidance in a shared-control system. *7th International Conference on Emerging Technologies and Factory Automation*, pages 1405–1414, 1999.

[Ros95] J. Rosenblatt. Damn: A distributed architecture for mobile robot navigation. *AAAI'95 Spring Symposium on Lessons learned for Implemented Software Architectures for Physical Agents*, pages 167–178, 1995.

[RP99] M. Ribo and A. Pinz. A comparison of three uncertainty calculi for building sonar-based ocuppancy grids. *Proceedings of the 7th International Symposium on Intelligent Robotic Systems*, pages 235–243, 1999.

[RSK95] E. Ruffini, A. Safiotti, and K. Konolige. Progress in research on autonomous vehicle motion planning. *Industrial Applications of Fuzzy Logic and Intelligent Systems, IEEE Press*, 1995.

[Saf97a] A. Safiotti. Fuzzy logic in autonomous robotics: behaviour coordination. *Proceedings of the 6th IEEE Int. Conf. on Fuzzy Systems*, pages 573–578, 1997.

[Saf97b] A. Safiotti. The uses of fuzzy logic in autonomous navigation: a catalogue raisonné. *Soft Computing*, 1:180–197, 1997.

[Sie94] B. Siemiatkowska. A highly parallel method fpr mapping and navigation of an autonomous mobile robot. *Proceedings of the IEEE Conference on Robotics and Automation*, pages 2796–2801, 1994.

[SL97] R. Simpson and S. Levine. Adaptive shared control of a smart wheelchair operated by voice control. *IEEE/RSJ IROS'97 Conference*, pages 622–626, 1997.

[SRK93] A. Safiotti, E. Ruspini, and K. Konolige. Blending reactivity and goal-directedness in fuzzy controller. *Proceedings of the Second IEEE Int. Conf. on Fuzzy Systems*, pages 134–139, 1993.

[TBBF98] S. Thrun, A. Bucken, W. Burgard, and D. Fox. Map learning and high-speed navigation in rhino. *A.I.-Based Mobile Robots: Case studies of successful robot systems*, pages 1–24, 1998.

[Tid94] Tide. Technology initiative for disabled and elderly people - bridge phase-synopses. 1994.

[Til90] R. Tilove. Local obstacle avoidance for mobile robots based on the method of artificial potencials. *Proceedings of the IEEE Conference on Robotics and Automation*, pages 566–571, 1990.

[WKK94] P. Wellman, V. Krovi, and V. Kumar. An adaptative mobility system for the disabled. *IEEE Int. Conference on Robotics and Automation*, pages 2006–2011, 1994.

[WSW96] A. Wright, R. Sargent, and C. Witty. Arc development system user's guide - manual edition 1.0 - documents arc version 1.4, 1996.

[YP95] J. Yen and N. Pfluger. A fuzzy logic based extension to payton and rosenblatt's command fusion method for mobile robot navigation. *IEEE Transactions on Systems, Man and Cybernetics*, 25(6):971–978, 1995.

[ZO96] T. Zhao and M. Overmars. *Forms Library - A graphical User Interface Toolkit for X*. 1996.