UNIVERSIDADE DO PORTO

FACULDADE DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

LEVANTAMENTO DAS POSSIBILIDADES DE UTILIZAÇÃO DE FUNÇÕES INTELIGENTES EM ROBÓTICA MÓVEL AUTÓNOMA

(RELATÓRIO DE ESTÁGIO PRODEP)

TRABALHO EFECTUADO POR:

JOÃO ALBERTO VIEIRA DE CAMPOS PEREIRA CLARO

30 DE SETEMBRO DE 1993

PARECER

RELATIVO AO ESTÁGIO REALIZADO POR JOÃO ALBERTO VIEIRA DE CAMPOS PEREIRA CLARO NO ÂMBITO DO PROGRAMA PRODEP

O aluno João Alberto Vieira de Campos Pereira Claro tinha como trabalho proposto para realizar durante o período em que foi bolseiro do PRODEP o Estudo e Desenvolvimento de Métodos de Planificação que se aplicariam num domínio médico.

Porque os dados necessários à aplicação em causa não se encontraram disponíveis em tempo útil da duração da Bolsa, os estudos foram orientados para Planeamento de trajectórias em Robótica Móvel.

O bolseiro realizou um trabalho muito árduo e completo, começando numa extensa recolha bibliográfica sobre os assuntos correlacionados - Visão, Planeamento e Navegação de Robôs móveis - comparando as várias abordagens e opinando sobre os respectivos méritos.

O presente trabalho constitui uma base sólida para uma posterior escolha das melhores abordagens para um funcionamento inteligente de Robôs móveis com autonomia.

Considero, portanto, que embora num domínio diverso do inicialmente previsto, o trabalho realizado excedeu até o que se poderia esperar ser realizado no espaço de tempo coberto pela Bolsa atribuída.

Porto, 30 de Setembro de 1993

Orientador na Universidade

(Eugénio da Costa Oliveira)

Orientador no INEB

(Aurélio Campilho)

PREFÁCIO

O presente trabalho consiste no levantamento das possibilidades de utilização de funções inteligentes em robótica móvel autónoma.

O objectivo principal deste levantamento é a análise de aplicações em diversos tipos de ambientes (completamente desconhecidos, parcial ou totalmente conhecidos, estáticos, dinâmicos), através de um estudo das consequências das características ambientais ao nível da arquitectura do robot, com especial destaque para o impacto na componente navegacional.

Esta análise tem em vista uma posterior aplicação a desenvolver numa plataforma móvel.

Das duas características apontadas, que são a mobilidade e a autonomia, surgem os objectivos de nível mais específico:

. estudo das capacidades de percepção, cognição e acção, inerentes à autonomia desejada;

. conhecimento dos sistemas utilizados correntemente em robótica móvel para a implementação dessas capacidades;

. conhecimento das formas de utilização conjunta dessas capacidades;

. nas diversas aplicações possíveis da robótica móvel autónoma, estudar os sistemas de navegação mais adequados às características dessas aplicações, características entre as quais se salientam o ambiente de operação do robot, filosofias arquitecturais, os modos de obtenção e representação da informação e as tarefas a desempenhar.

Um agradecimento especial ao Prof. Eugénio Oliveira pela disponibilidade manifestada, bem como pelo acompanhamento na realização do trabalho.

ÍNDICE

CAPÍTULO 1

INTRODUÇÃO

1 INTRODUÇÃO

O objecto do nosso estudo são os robots móveis autónomos.

Para o desenvolvimento de uma verdadeira autonomia, a arquitectura de um robot deve integrar as seguintes capacidades:

◇ PERCEPÇÃO, que envolve a interpretação da diversa informação sensorial (visual, sonora, táctil, ...);

◇ COGNIÇÃO, que tem sido o campo privilegiado de investigação da Inteligência Artificial;

◇ ACÇÃO, incluindo navegação através do mundo e manipulação de objectos.

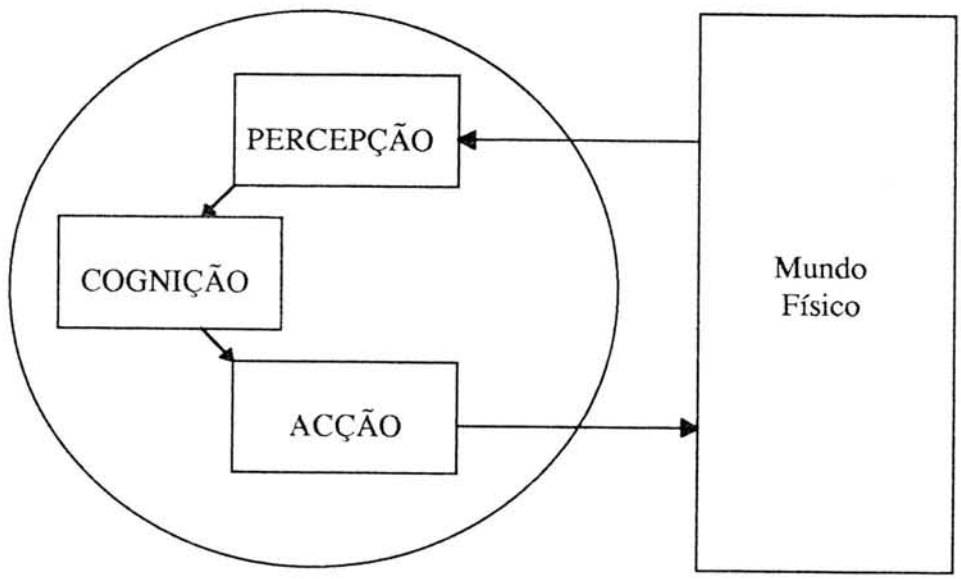


fig. 1.1 Design para um robot autónomo

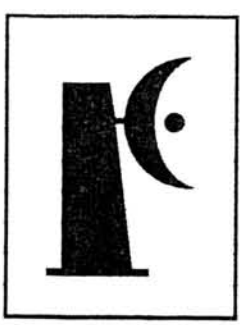
Tendo em conta estas capacidades e as temáticas abordadas na literatura que reunimos, apareceu-nos como natural a seguinte sistematização desta matéria:

- ◇ SENSORES
- ◇ VISÃO
- ◇ REPRESENTAÇÃO ESPACIAL
- ◇ NAVEGAÇÃO
- ◇ ARQUITECTURAS

Nesta perspectiva, colocamos fora do âmbito deste trabalho o estudo do capítulo da mecânica em robótica móvel. No entanto, os aspectos mecânicos não deixam de estar presentes na medida em que a maioria da investigação é, felizmente, feita a pensar em determinadas "test-beds", o que obriga à tomada em consideração das particularidades mecânicas de cada robot, em especial a sua forma de locomoção.

CAPÍTULO 2

SENSORES



2.1 INTRODUÇÃO

A percepção é uma capacidade vital para que um robot móvel possa manobrar no seu ambiente e executar qualquer tipo de tarefa inteligente. Os sensores, dispositivos que permitem essa percepção, devem fornecer a informação em que se baseará a navegação de um robot no seu ambiente. O "sensing" é, aliás, obrigatório devido à acumulação de erros durante a navegação de um robot, que o obriga a renovar o seu conhecimento do mundo.

Na primeira secção deste capítulo apresentaremos uma classificação dos sensores mais usados em robótica móvel e exporemos alguns conceitos acerca desses sensores. Na segunda secção apontaremos algumas referências sobre a utilização de sensores em robótica móvel e faremos algumas considerações acerca dessa mesma utilização.

2.2 UMA CLASSIFICAÇÃO DE SENSORES

Pode ser feita uma divisão dos sensores em duas categorias principais [1]:

◇ ESTADO INTERNO, relacionados com a detecção de variáveis como a orientação do robot ou as distâncias por ele percorridas;

◇ ESTADO EXTERNO, relacionados com a detecção de variáveis como o alcance, a proximidade e o toque.

Por sua vez, os sensores de estado externo podem ainda ser classificados em:

◇ SENSORES DE CONTACTO, que respondem ao contacto físico, como o toque ou o deslizar;

◇ SENSORES DE NÃO-CONTACTO, que se baseiam na resposta de um detector a variações de radiação acústica ou electromagnética.

SENSORES DE ESTADO INTERNO

Em robótica móvel, estes sensores estão relacionados com a odometria, ou seja a determinação das distâncias percorridas pelo robot e da sua orientação. Normalmente esta informação é fornecida por codificadores que se encontram nos veios das rodas dos robots.

SENSORES DE TOQUE

Estes sensores são geralmente usados para detectar choques do robot com obstáculos ou com a fronteira do espaço de trabalho. São úteis à navegação em ambientes estreitos e na detecção de choques inesperados.

Sensores Binários

São dispositivos de contacto, como micro-interruptores ou "bumpers", que constituem tipicamente a "última linha de defesa" do robot.

SENSORES DE ALCANCE

Um sensor de alcance mede a distância entre um ponto de referência (geralmente no próprio sensor) e objectos no campo de operação do sensor.

São usados em aplicações em que se pretenda saber a localização e características gerais da forma dos objectos no espaço de trabalho do robot.

Sensores "Pulsed-Laser"

A medida do tempo que um impulso de luz demora a regressar coaxialmente (i. e., pelo mesmo caminho) de uma superfície reflectora permite obter a distância a essa superfície a partir da relação simples

$$D = c \, . \, T / 2,$$

em que T é o tempo de trânsito do impulso e c a velocidade da luz.

Este sensor deve possuir uma grande precisão na medida do tempo devido ao valor elevado da velocidade da luz.

Sensores Ultrasónicos

A ideia básica é semelhante à do "pulsed-laser", Um "chirp" ultrasónico é transmitido num curto período de tempo. Conhecida a velocidade do som para um meio específico, um cálculo simples envolvendo o intervalo de tempo entre a saída do impulso e o regresso do eco fornece uma estimativa da distância à superfície reflectora.

SENSORES DE PROXIMIDADE

Os sensores de proximidade produzem, em geral, uma saída binária que indica a presença de um objecto dentro de um intervalo de distâncias especificado. São tipicamente usados para detectar obstáculos próximos, permitindo assim evitar colisões com estes.

Sensores Ultrasónicos

O princípio de funcionamento foi já exposto a respeito dos sensores de alcance. Para o funcionamento como sensor de proximidade, é introduzida uma janela de tempo que corresponde à distância pré-especificada para a detecção. Um eco recebido no período de tempo em que a janela está activa indicará a presença de um objecto no intervalo especificado.

Sensores de Proximidade Óptica

Na sua forma mais comum, estes sensores são constituidos por um LED que actua como emissor de luz infra-vermelha e um foto-díodo que actua como receptor. A intersecção do cone de luz emitida com o cone de luz recebida produz o volume de detecção, que possui geralmente uma forma alongada semelhante a um lápis. Este volume define o campo de operação do sensor, visto que uma superfície reflectora que o intersecte é iluminada pela fonte e simultaneamente "vista" pelo receptor.

De uma forma geral, este sensor gera um sinal binário quando a intensidade de luz recebida excede um valor de limiar.

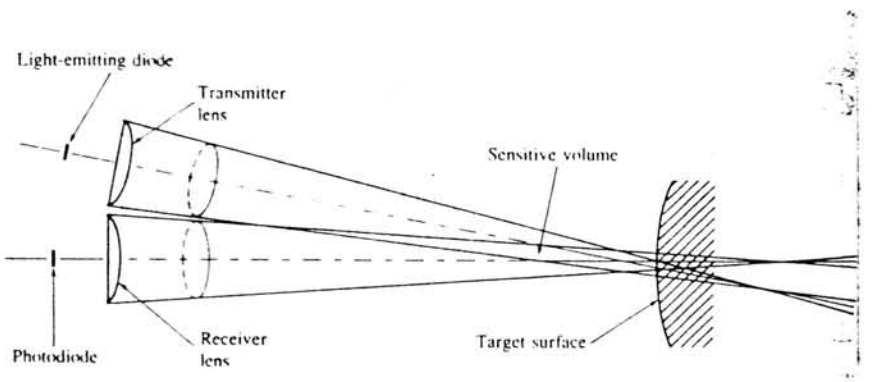


fig. 2.1 Sensor de proximidade óptica

2.3 CONSIDERAÇÕES SOBRE O USO DE SENSORES EM ROBÓTICA MÓVEL

Em robótica móvel vários sensores têm sido usados, como "bumpers", codificadores de veios, sonares [2] [3] [4], sensores de infra-vermelhos [4] [5] e laser [6]. Cada tipo de sensor apresenta, contudo, algumas limitações. Os codificadores de veio, por exemplo, perdem exactidão quando as rodas deslizam e os sonares têm uma largura de feixe grande e são sensíveis a superfícies polidas.

A solução mais frequente consiste em recorrer à redundância sensorial, que permite utilizar as vantagens de uns sensores para compensar as desvantagens de outros. Flynn [4] descreve o uso de dois sensores relativamente baratos, um sensor de infra-vemelhos e um sonar, para a produção de informação para a construção de uma representação do ambiente do robot. O sonar mede bem as distâncias aos objectos, mas tem fraca resolução angular devido à grande largura de feixe. O sensor de infra-vermelhos, embora não seja capaz de grande exactidão na medida de distâncias, tem boa resolução angular na detecção da ausência ou presença de um objecto.

Utilizando ambos os sensores para sondar uma sala, o robot é capaz de construir um mapa melhor. O sonar é bastante exacto na detecção da distância ao objecto mais próximo, enquanto o sensor de infra-vermelhos detecta com grande fiabilidade, por exemplo, arestas de portas que seriam invisíveis ao sonar. Combinando a informação destes dois sensores, o robot constrói um mapa que é convertido para uma representação intermédia, o "curvature primal sketch", que representa a fronteira de um objecto através de pontos que marcam alterações significativas na curvatura dessa fronteira. Unindo estes pontos com segmentos de recta, é simples converter a representação para uma lista de polígonos que pode ser passada a um planeador de trajectórias.

Sobre outras aplicações de sensores, deverão ser consultados os capítulos sobre representação espacial e navegação, onde podem ser encontradas referências sobre a utilização de sensores nas diversas ferramentas de representação espacial e metodologias navegacionais.

BIBLIOGRAFIA

[1] K. S. Fu, R. C. Gonzalez, C. S. G. Lee
"Robotics: Control, Sensing, Vision and Intelligence"
McGraw - Hill
(1987)

[2] H. P. Moravec, A. E. Elfes
"High resolution maps from wide angle sonar"
Proc. 1985 IEEE Int. Conf. Robotics and Automation, pp. 116-121

[3] H. P. Moravec
"Certainty grids for mobile robots"
Proceedings of the Workshop in Space Telerobotics, vol. 1, pp. 307-312

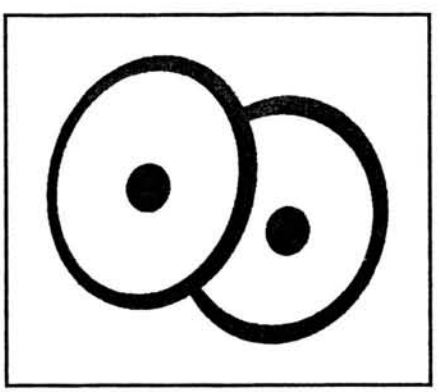
[4] A. M. Flynn
"Combining sonar and infrared sensors for mobile robot navigation"
The International Journal of Robotics Research, vol. 7, no. 6, pp. 5-14
Dec. 1988

[5] G. Giralt, R. Chatila, M. Vaisset
"An integrated navigation and motion control system for autonomous multisensory mobile robots"
Proc. First Int. Symp. Robotics Research, 1983

[6] A. M. Thompson
"The navigation system of the JPL robot"
Proc. IJCAI-5, pp. 335-337, 1979

CAPÍTULO 3

VISÃO



3.1 INTRODUÇÃO

A visão é um dos canais sensoriais mais importantes para os seres humanos. É uma faculdade que nos permite reunir grande parte do conhecimento que utilizamos para resolver problemas.

A visão máquina precisa permite explorar novos campos de aplicações computorizadas como a navegação de robots móveis, tarefas complexas de manufactura, a análise de imagens de satélite, sondas de exploração ou telescópios orbitais ou ainda o processamento de imagens médicas. Interessa-nos de forma especial esta possibilidade de dotar os robots com a capacidade de processar informação visual.

A visão robótica pode ser definida [1] como o processo de extracção, caracterização e interpretação de informação de imagens de um mundo tridimensional. Um robot com capacidades de visão beneficia de uma sofisticação sensorial que lhe permite uma interacção "inteligente" e flexível com o mundo.

Fu, Gonzalez e Lee apresentam-nos em [1] uma divisão do processo de visão robótica em seis áreas principais:

1. "Sensing" - Processo que obtém/forma uma imagem visual. Uma câmara fornece ao computador uma imagem, representada como uma grelha bidimensional de níveis de intensidade. Cada elemento da grelha, a que chamamos pixel, pode guardar um bit único de informação (branco/preto) ou vários bits (informação da medida de intensidade e cor).

2. Pré-processamento - Lida com técnicas como redução de ruído e realce de detalhes. Os valores que caracterizam um pixel são afectados por vários fenómenos, como a cor do objecto, a fonte de luz, o ângulo e distância da câmara, o ruído de imagem provocado pela amostragem, quantificação, transmissão ou perturbações no ambiente durante a aquisição da imagem, etc.. Os efeitos de tais fenómenos podem ser atenuados ou eliminados com as técnicas de pré-processamento que actuam sobre os valores característicos dos pixels.

3. Segmentação - Processo que particiona uma imagem em objectos de interesse. É um dos elementos mais importantes de um sistema automatizado de visão pois é nesta fase que os objectos são extraídos de uma cena para subsequente reconhecimento e análise.

4. Descrição - Lida com a computação de características (tamanho, forma) apropriadas para a diferenciação entre tipos de objectos. Idealmente, os descritores devem ser independentes do tamanho, localização ou orientação do objecto e conter informação discriminatória suficiente para identificar univocamente um objecto.

5. Reconhecimento - Processo que identifica os objectos. As abordagens actuais ao reconhecimento podem ser divididas em duas categorias principais: métodos decisão-teóricos e métodos estruturais. Os primeiros baseiam-se em descritores quantitativos e os segundos em descrições simbólicas e suas relações.

6. Interpretação - Atribui significado a um conjunto de objectos. A nossa compreensão desta área é ainda muito reduzida, o que nos leva a optar pela formulação de restrições e idealizações que tentem simplificar a complexidade desta tarefa.

Estas áreas podem ser agrupadas de acordo com o nível de sofisticação envolvido na sua implementação, sendo assim possível considerar três níveis de processamento: visão de baixo, médio e alto nível. Embora as fronteiras entre os três níveis sejam algo incertas, o enquadramento que estes nos fornecem permite uma útil categorização dos processos envolvidos num sistema de visão.

Na visão de baixo nível incluiremos os processos que não requerem inteligência, o que na divisão apresentada corresponderá aos dois primeiros. O nível médio seria constituido pelas três funções seguintes, ou seja as que extraem, categorizam e rotulam os componentes da imagem resultante do tratamento de baixo nível. Finalmente a interpretação da imagem constitui o nível mais elevado, do qual, como já foi referido, a nossa compreensão se limita praticamente a vagas especulações.

As subdivisões apresentadas não pretendem de modo algum modelizar o sistema de visão humano, nem se pretende que sejam implementadas independentemente. Trata-se apenas de uma abordagem que permite, de acordo com a nossa compreensão da visão e com as ferramentas de que dispomos, realizar sistemas de visão robótica que explorem o melhor possível essas mesmas ferramentas e compreensão.

No caso mais particular das aplicações de robótica móvel, o seu carácter mais específico permite-nos colocar restrições aos níveis da concepção e implementação de um sistema de visão móvel. Efectivamente, é possível definir e caracterizar um sistema de visão robótica móvel em função da sua relação com os restantes módulos da arquitectura do robot e do meio em que ele irá operar. Com o objectivo de tentar sistematizar a investigação que se tem realizado neste domínio, exporemos na segunda secção deste capítulo alguns conceitos básicos sobre os dispositivos sensores mais comuns nos sistemas de visão. De seguida procuraremos apresentar uma perspectiva conceptual destes sistemas e finalmente, na quarta secção, deter-nos-emos mais em particular nos aspectos implementacionais.

Como foi referido inicialmente, a visão máquina é um campo muito vasto, não sendo todo ele relevante para o nosso estudo. Mesmo dentro da visão robótica há áreas que ultrapassam as particularidades da visão móvel. O âmbito deste estudo implica que apenas haja incidência sobre o campo da visão robótica móvel, sem perder de vista que o material exposto tem o seu principal valor como material de base. Um estudo profundo só poderá ser levado a cabo recorrendo bibliografia muito mais específica, alguma da qual poderá ser encontrada no final do capítulo. Essa bibliografia aponta, por sua vez, outras referências que deverão ser consultadas de modo a reunir um pormenor de estudo que permita abarcar o "state of the art" da investigação em visão robótica móvel.

3.2 DISPOSITIVOS SENSORES

Os dispositivos sensores mais frequentemente adoptados nos sistemas de visão móvel são uma única câmara ou um par stereo. Mais raras, mas todavia interessantes, são algumas aplicações que utilizam outras configurações, como por exemplo um robot do INRIA que possui um sistema trinocular [2] ou ainda sistemas de visão omnidireccional que utilizam uma lente "fish-eye"[3], um espelho esférico ou um espelho cónico [4].

Do âmbito deste estudo exclui-se naturalmente o tratamento dos dispositivos físicos que constituem uma câmara, bem como dos processos de amostragem espacial e quantificação de amplitude, directamente relacionados com a digitalização da imagem. Estes assuntos são pertença do âmbito mais geral da visão máquina e estão já plenamente explanados em variada documentação, pelo que não serão objecto de referência.

Da informação disponível em relação aos sistemas equipados com uma câmara ou com um par stereo, e ainda sobre a utilização da visão omnidireccional, é importante destacar alguns aspectos mais relevantes que permitam basear algumas das observações feitas mais adiante.

UTILIZAÇÃO DE UMA CÂMARA

Na figura 3.1 podemos observar um modelo de uma câmara, com o seu sistema de coordenadas.

Esta câmara está montada num dispositivo que lhe permite descrever ângulos de "pan" e "tilt". Define-se como "pan" o ângulo entre os eixos x e X e como "tilt" o ângulo entre os eixos z e Z.

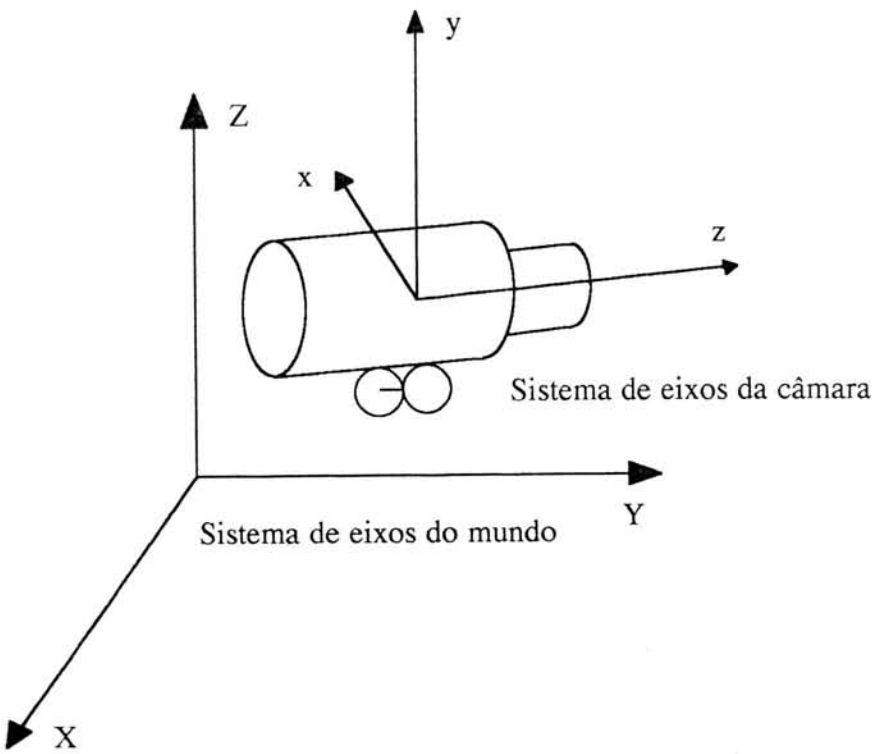


fig. 3.1 Geometria do processo de formação de imagens

A figura 3.2 mostra um modelo do processo de formação da imagem. O plano de imagem coincide com o plano xy do sistema de coordenadas da câmara e o eixo óptico (estabelecido pelo centro da lente) com o eixo z. O centro do plano de imagem é na origem e o centro da lente nas coordenadas $(0,0,\lambda)$, em que λ é o comprimento focal da lente.

Os parâmetros mais importantes de uma câmara são o seu comprimento focal, os seus "offsets" e os ângulos de "pan" e "tilt". Estes parâmetros podem ser medidos directamente, mas frequentemente torna-se mais conveniente determinar um ou mais desse parâmetros usando a própria câmara como dispositivo de medida.

É esse o caso da visão móvel. Este processo, chamado de calibração da câmara, utiliza um conjunto de pontos da imagem cujas coordenadas do mundo são conhecidas, o que lhe permite computar os parâmetros da câmara. Um conjunto de equações para este processo de calibração pode ser encontrado em [1] na secção dedicada à calibração de câmaras.

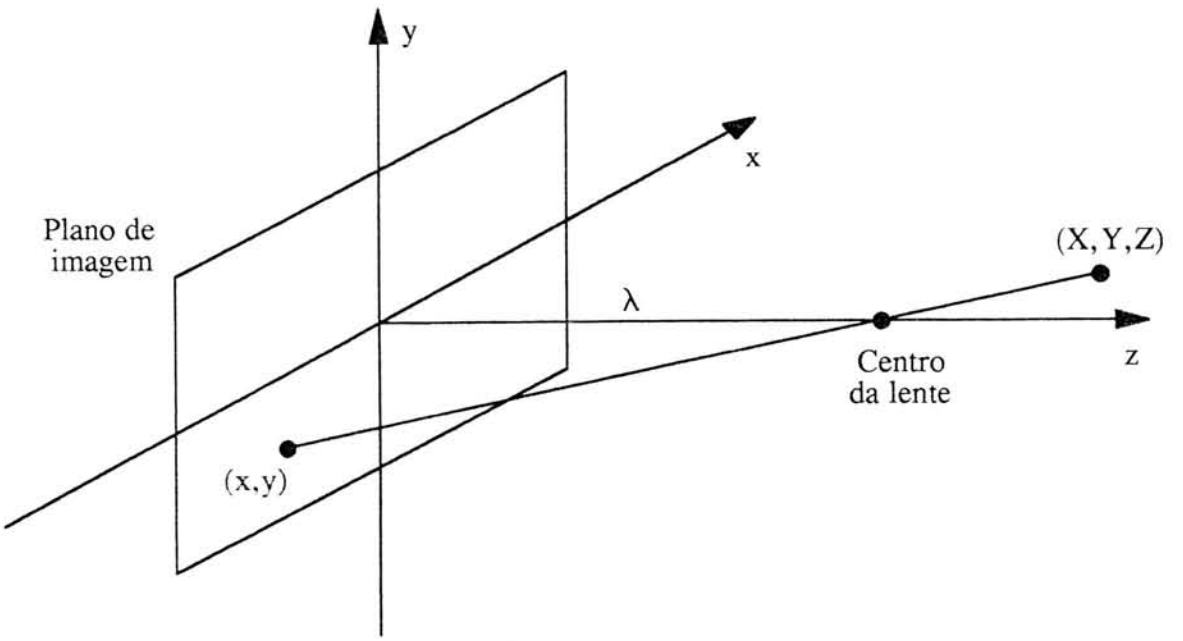


fig. 3.2 Modelo do processo de formação da imagem

UTILIZAÇÃO DE UM PAR STEREO

O uso de um par stereo permite ultrapassar o problema de o mapeamento de uma cena tridimensional num plano de imagem ser uma transformação de muitos para um. A informação de profundidade pode ser obtida através do uso de técnicas stereo.

Estas técnicas envolvem a obtenção de duas vistas diferentes de um objecto. A distância que separa os centros das duas lentes é chamada linha de base. Assumindo que as câmaras são idênticas e que os seus sistemas de coordenadas estão perfeitamente alinhados, diferindo apenas na localização das suas origens, e dados os pontos imagem de um ponto real é possível determinar as suas cordenadas no mundo real.

Numa demonstração exposta em [1], e de acordo com a figura 3.3, obtém-se para a profundidade do ponto a seguinte expressão:

$$Z=\lambda - \lambda B / (x_2 - x_1)$$

em que λ é a distância focal, B a linha de base e x_2 e x_1 as coordenadas x dos pontos imagem.

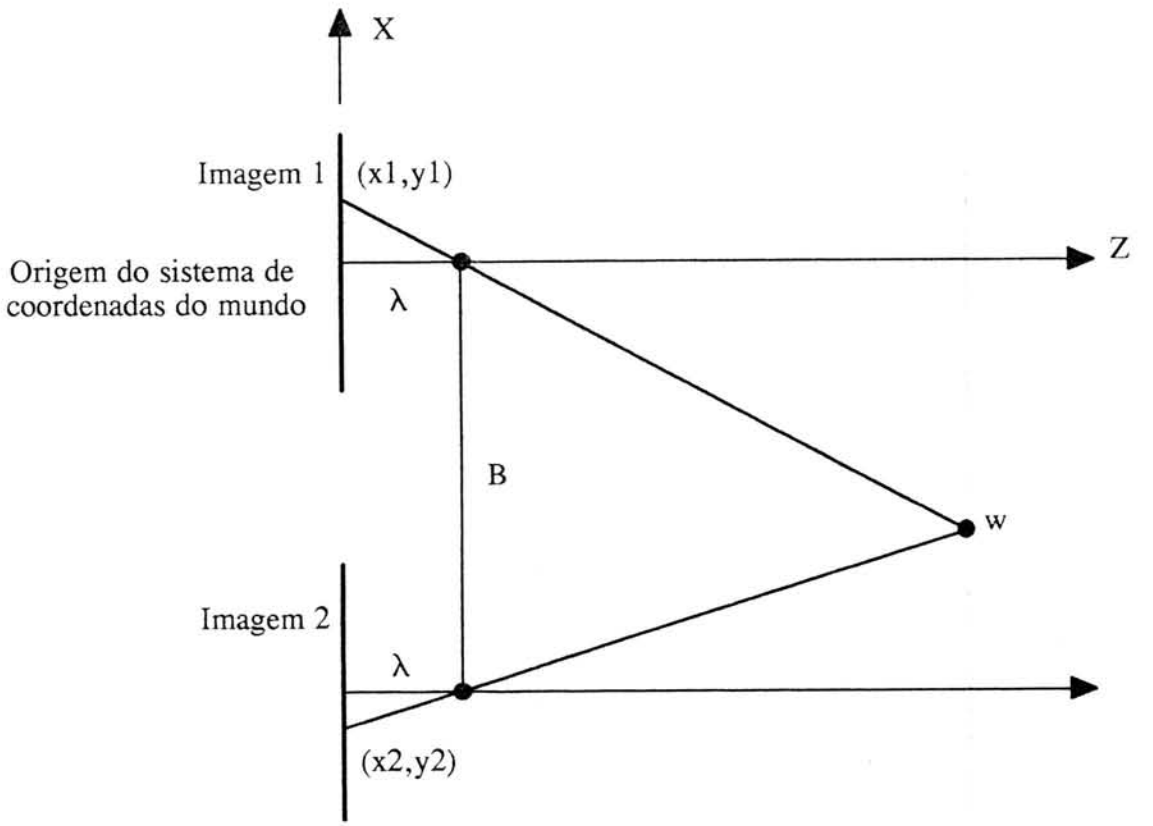


fig. 3.3 Vista de cima do processo de formação de imagem na visão stereo

A tarefa de maior dificuldade nos algoritmos stereo é encontrar dois pontos correspondentes em imagens diferentes da mesma cena. Uma abordagem possível é escolher um ponto numa pequena região de uma das imagens e tentar encontrar a melhor região correspondente na outra imagem através de técnicas de correlação. No caso de cenas com características proeminentes, como cantos ou arestas, uma abordagem de correspondência de características fornecerá, de um modo geral, uma solução mais rápida para o estabelecimento de correspondências.

Ao montar um par stereo, não é possível obter para as câmaras uma configuração com planos focais verticais exactamente coplanares e uma linha de base precisamente horizontal. As câmaras são montadas tão perto quanto possível da configuração ideal e então proceder-se-á à sua calibração. Kriegman *et al.* [5] adoptaram um processo de calibração que utiliza um gráfico de calibração pendurado numa parede a uma altura precisa (figura 3.4).

A zona que nesta figura é representada a cinzento tem, na realidade, cor preta, enquanto o interior dos quadrados tem cor branca.

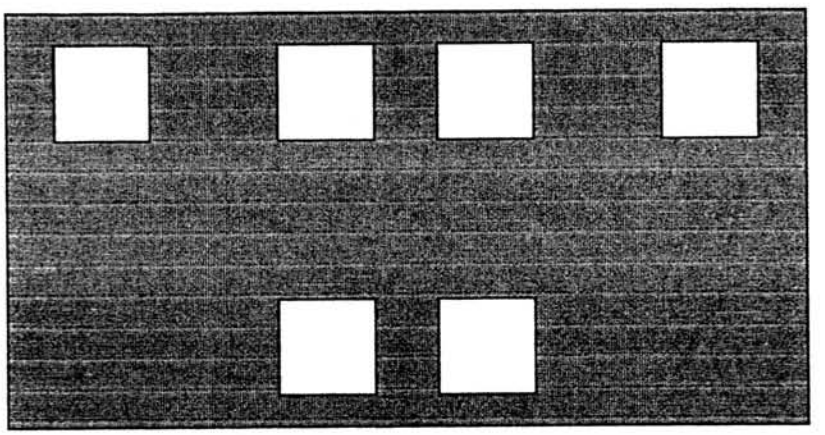


fig. 3.4 Aspecto do gráfico de calibração

São três as razões pelas quais as câmaras devem ser calibradas:

1. As linhas epipolares correspondentes ao horizonte para cada câmara precisam de ser conhecidas para o processo de correspondência.

2. A posição relativa das câmaras e a distorção quadrática são necessárias para calcular a localização tridimensional das correspondências.

3. Para comparar intensidades entre as duas câmaras, deve ser determinada a função de transferência de intensidade relativa entre as duas câmaras.

Para cada câmara, a linha do horizonte é determinada a partir do alinhamento dos quatro quadrados na parte de cima do gráfico, o qual é pendurado de forma a que os seus centros fiquem precisamente sobre o plano do horizonte do robot.

Os parâmetros importantes da transformada relativa são a separação da linha de base e o ângulo de vergência (ângulo formado pelos eixos focais das duas câmaras, que idealmente seriam paralelos) porque erros nestes podem conduzir a erros grandes na localização das correspondências. Conhecendo o tamanho e espaçamento horizontal dos quadrados brancos, a separação da linha de base e o ângulo de vergência são calculados, bem como a distorção de segunda ordem das lentes.

Finalmente, como o facto de a linha de base ser estreita conduz a uma geometria de iluminação/reflexão praticamente idêntica na visão do gráfico, a função de transferência de

intensidade relativa é determinada a partir da média dos níveis de cinzento nas mesmas regiões pretas e brancas do gráfico. São retidos os dois parâmetros de uma relação linear.

UTILIZAÇÃO DE IMAGENS OMNIDIRECCIONAIS

Para a aquisição de vistas omnidireccionais do ambiente têm sido estudados métodos que usam câmaras rotativas, lente "fish-eye", espelho esférico ou espelho cónico.

Estes métodos apresentam diversos inconvenientes que limitam a sua aplicação em implementações de visão móvel. Na utilização de câmaras rotativas, por exemplo, o processo de formação da imagem omnidireccional leva muito tempo, o que torna esse método não aplicável a problemas de tempo real como evitar colisões com objectos móveis. Relativamente à utilização de lentes "fish-eye" e espelhos esféricos ou cónicos, verifica-se uma distorção de imagem que obriga a um pré-processamento de imagem mais pesado do que o necessário com os sistemas normais.

No entanto, para aplicações como navegação visual existem já implementações que usam imagens omnidireccionais, nomeadamente com recurso a espelhos cónicos. Y. Yagi, Y. Nishizawa & M. Yachida [4] desenvolveram um sensor de imagem omnidireccional, que denominaram COPIS (COnic Projection Image Sensor). O princípio de funcionamento do COPIS pode ser observado na figura 3.5.

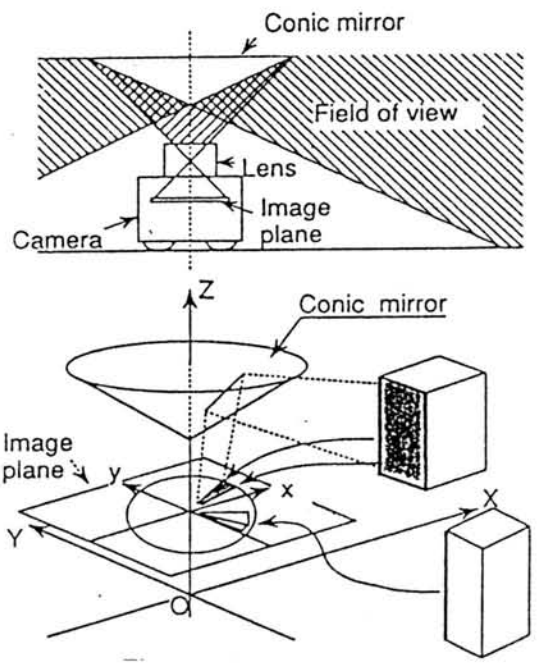


fig. 3.5 O processo de formação de imagem no COPIS

O azimute de cada ponto na cena aparece na imagem como a sua direcção a partir do centro da imagem. Obtendo o ângulo de azimute a partir de dois pontos, a localização relativa entre o robot e o ponto pode ser calculada por triangulação.

Na implementação descrita em [4], Yagi, Nishizawa e Yachida recorrem a um sensor ultra-sónico para obter informação de alcance, que complementa a informação de azimute obtida da sequência de imagens, permitindo verificar se uma superfície candidata é uma superfície verdadeira ou oca.

3.3 PERSPECTIVA CONCEPTUAL

3.3.1. BREVE PERSPECTIVA HISTÓRICA

Um dos primeiros robots móveis a usar visão para localizar objectos e navegar foi o robot Shakey [6]. O seu desenvolvimento foi levado a cabo pelo SRI International Artificial Intelligence Center por volta do ano de 1984. O robot Shakey utilizava um conjunto de vários sensores e visão.

Outro dos projectos de robótica móvel pioneiros na utilização de sistemas de visão foi o Stanford Cart, desenvolvido por Moravec [7]. Utilizava visão pura e trabalhava com base em características pontuais da imagem para escolher uma trajectória livre de obstáculos.

Quer o projecto do SRI, quer o de Stanford, usavam imagens relativamente isoladas. Outra característica comum era o facto de nenhum deles tentar a comprensão geral da cena. Este é talvez um dos motivos que contribuiram para o sucesso no funcionamento de ambos os robots. Em qualquer dos dois projectos ficou demonstrada a utilidade de sistemas simples e orientados-à-tarefa.

O passo seguinte de grande importância na investigação em visão robótica móvel foi o aumento do interesse por sistemas de visão em tempo real e orientados-à-tarefa motivado em grande parte pelo projecto do Autonomous Land Vehicle (ALV) DARPA [8] para o Exército do Estados Unidos. O ALV é um veículo que opera em exteriores, onde as restrições ambientais são muito menores que em ambientes interiores (edifícios, fábricas, etc.). Os ALVs são guiados visualmente ao longo de uma estrada, utilizando para tal recursos computacionais consideráveis. Os seus sistemas de visão são em geral sistemas 3-D em tempo real e que processam uma sequência densa de imagens.

Em Stanford, Triendl e Kriegman [5] desenvolveram um robot que usa visão stereo sobre uma fatia 2-D do mundo, ou seja, o mundo é representado bidimensionalmente, sendo

todos os objectos projectados no plano do chão. Este robot constrói um modelo do que vê à medida que explora.

No INRIA, Ayache e Faugeras [9] foram os responsáveis pelo projecto de um robot que usa stereo 3-D e constrói modelos do mundo globalmente consistentes. Este projecto apresenta, no entanto, a desvantagem de os seus sistemas de visão e mapeamento não funcionarem em tempo real. Esta arquitectura, tal como a anterior, usa computadores off-board e destina-se a operar em ambientes interiores.

No MIT, têm sido conduzidos diversos projectos. Brooks e Flynn [10] projectaram um sistema de navegação que usa uma combinação de visão stereo e movimento. Trata-se de um sistema auto-calibrante, em que é relevante a ideia apresentada pelos dois investigadores de que a precisão métrica não é um requisito principal para sistemas de visão concebidos para mobilidade. O sistema de Horswill e Brooks [11] persegue objectos, não utilizando uma representação centralizada. Sarachik [12] concebeu um sistema para navegação entre salas, que coloca mais ênfase nos aspectos topológicos (ligados à noção básica de um espaço não quantitativo, e em que se consideram só as relações de posição dos objectos) do que nos aspectos geométricos (ligados às propriedades e dimensões de linhas, superfícies e volumes) do ambiente do robot.

Os projectos do MIT são caracterizados por evitar a utilização de mapas ou modelos ambientais e optar por uma abordagem mais descentralizada e comportamental à produção de comportamentos complexos. Além disso, de toda a investigação aí realizada, ficou demonstrado que a precisão fotogramétrica não é necessária para sistemas de visão orientados à tarefa.

O conhecimento da localização exacta de um robot no ambiente em que se desloca é um dos difíceis problemas que a investigação em robótica móvel tenta resolver. Dessa inexactidão na localização resulta a dificuldade em construir modelos ambientais globalmente consistentes. É devido à imperfeição dos sensores (em particular as câmaras, no caso da visão) e da interpretação dos seus dados que se torna necessário levar em consideração a incerteza e tentar reduzi-la para que seja possível melhorar a construção de modelos, a tomada de decisões e o planeamento de acções.

O paradigma tradicional dos sistemas visão-acção em Inteligência Artificial tem sido tentar efectuar "compreensão de imagens" e construir um modelo "preciso" do mundo. Tal modelo necessita de ser preciso se for usado como base para planeamento dos movimentos do robot por um período superior ao curto prazo.

Um esquema alternativo é amostrar o mundo. Se esta amostragem for frequente, então as percepções não necessitam de ser muito precisas, visto que as acções podem facilmente ser corrigidas à medida que a situação do robot no mundo evolui.

Nesta breve panorâmica da investigação em visão robótica móvel é possível verificar que a tendência tem sido a colocação de ênfase na simplicidade e rapidez dos sistemas de visão, de acordo com os princípios de orientação-à-tarefa e funcionamento em tempo real. As perspectivas que apresentaremos com maior detalhe colocam-se também dentro desta tendência. Ilustram de forma excelente estes dois princípios, sendo, do mesmo modo, representativas das principais linhas de orientação da actual investigação em visão móvel: a visão stereo e o "structure-from-motion" aplicados ao movimento em ambientes interiores e em estradas.

3.3.2. VISÃO MÓVEL EM AMBIENTES INTERIORES

Os sistemas de visão são de vital importância para as aplicações em ambientes complexos, dinâmicos e previamente desconhecidos. Se estes ambientes forem estruturados, o robot pode à partida possuir informação sobre o que esperar num determinado ambiente. Esta informação pode ser agrupada sob a forma de modelos genéricos que descrevem exactamente o que esperar no ambiente. Uma vez dadas as restrições deste modelo, a visão e possivelmente mais "sensing" podem ser usados para instanciar o modelo de um ambiente em particular.

Em Stanford, Kriegman, Triendl e Binford têm utilizado o robot MARS (Mobile Autonomous Robot Stanford) como plataforma de várias experiências sobre a mobilidade em edifícios [5].

Num corredor ou armazém, todas as arestas observáveis relevantes são horizontais ou verticais. As linhas não verticais (chão-parede, chão-porta, etc.) frequentemente estão fora do campo de visão, têm um contraste demasiado baixo e estão mal definidas ou obstruidas por mobília. Por outro lado, as arestas verticais no mundo projectam-se em arestas verticais num plano de imagem vertical e verifica-se que fornecem informação suficiente para instanciar um modelo do ambiente.

Na figura 3.6 podemos observar uma possível instanciação do modelo genérico de um corredor com todas as características observáveis.

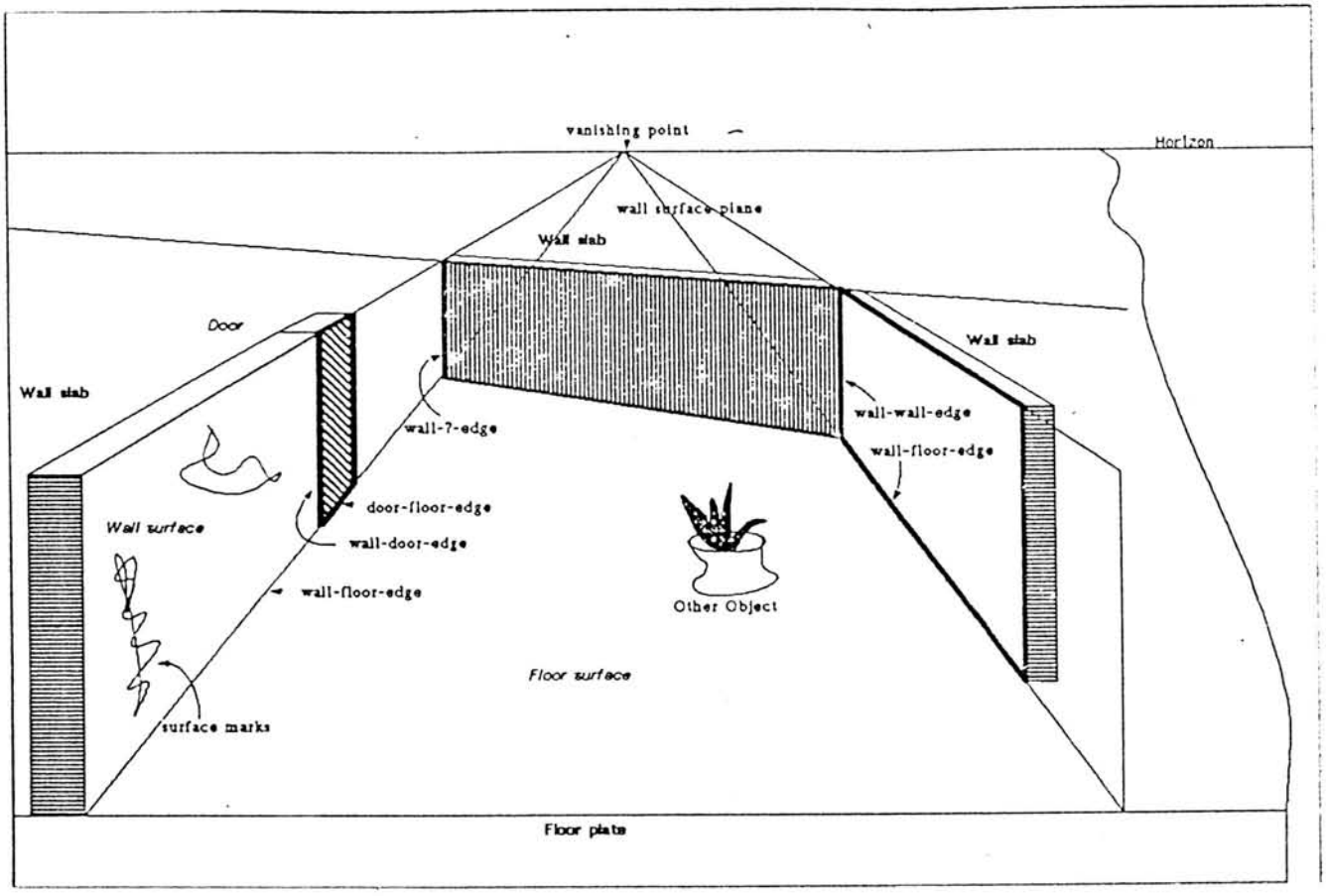


fig. 3.6 Possível instanciação do modelo genérico de um corredor

Na implementação descrita em [5] o modelo genérico do edifício está implícito no código de instanciação do modelo, conforme poderá ser verificado pelo que referiremos de seguida. No entanto, Kriegman e Binford [13] desenvolviam já em 1989 uma representação explícita de modelos genéricos. Naturalmente, o "sensing" e os métodos para instanciação do modelo apresentados em [5] deverão ser apenas um subconjunto dos deriváveis do modelo genérico explícito.

Para a obtenção do modelo instanciado, os intervalos entre arestas vizinhas são considerados portas ou paredes, desde que o seu aspecto seja semelhante em ambas as imagens do par stereo de câmaras com que o robot está equipado, e a intensidade varie suavemente. As direcções das possíveis paredes são agrupadas com pesos proporcionais aos seus comprimentos. A direcção mais proeminente é considerada a direcção de uma parede na cena, determinando-se assim as localizações prováveis das "espinhas" das paredes. A "espinha" de uma parede não é na realidade uma parede, mas a sua localização esperada.

Após a determinação das direcções das paredes, acrescentam-se ao modelo as portas e paredes.

Uma porta pode ser vista como o intervalo entre duas arestas que:

1. estão afastadas entre 60 a 140 cm, ou seja uma largura maior que uma pessoa , mas não demasiado,
2. têm elevado contraste,
3. têm alterações complementares de nível de cinzento, por exemplo escuro no interior e claro no exterior, indicando uma porta uniforme numa parede uniforme,
4. se encontram perto de uma "espinha".

Os intervalos perto de "espinhas" que não forem reconhecidos como portas e tiverem uma intensidade razoavelmente uniforme entre as arestas limite são classificados como paredes. Finalmente, as arestas verticais que não são reconhecidas como fronteiras de uma parede ou porta poderão ser marcas na parede, caso se encontrem perto de uma parede, ou pertencerem a obstáculos, caso se encontrem afastadas das paredes.

A pesquisa de arestas pode ser efectuada apenas ao nível do horizonte da câmara, pois todas as arestas verticais relevantes para o modelo cruzam um plano horizontal à altura da câmara.

Este sistema de visão construído à volta da detecção de arestas permite ainda a exploração de espaço livre numa estratégia de planeamento de movimento que não utiliza modelo. A projecção das correspondências stereo e um pequeno conjunto de pressupostos permitem a determinação de um polígono-fronteira do espaço livre. Uma estratégia simples de planeamento permite ao robot explorar com segurança. A partir da representação da fronteira seria possível invocar um planeador de trajectórias bidimensional para encontrar uma trajectória para um determinado objectivo. No entanto, dada a possibilidade de nem todos os obstáculos ou o objectivo terem sido vistos, o mais apropriado seria a aplicação de métodos locais de navegação e evitar de obstáculos.

O segundo robot móvel do SRI [14] possui um sistema de visão que detecta estruturas tridimensionais proeminentes numa sequência densa de imagens. O robot está equipado com uma única câmara. O seu sistema de visão foi concebido por Wells para lhe permitir navegar visualmente num corredor e é usado apenas para a navegação entre objectos estáveis, o que permite estreitar os seus objectivos à estimação das localizações de características estáveis no mundo. Os objectos não estáveis, ou seja, em movimento rápido, são detectados por outros sensores.

O sistema escolhido é, portanto um sistema baseado em características, cujas características são segmentos de recta da imagem e do mundo. Os segmentos de recta apresentam-se vantajosamente como um compromisso prático entre as curvas e os pontos. As características pontuais são de análise simples, mas as características pontuais proeminentes podem ser dispersas, particularmente em ambientes construidos pelo homem. As cenas industriais e culturais contêm normalmente características lineares proeminentes que podem ser detectadas facilmente e com fiabilidade. Embora essas cenas frequentemente tenham também características curvas significativas, tais características são muito mais difíceis de analisar do que pontos ou linhas.

O vector principal do funcionamento deste sistema de visão é a estimação de características estáticas do mundo a partir da sua observação numa sequência de imagens, à medida que a câmara se move.

Wells descreve e implementa uma formulação do "structure-from-motion" (SFM) utilizando segmentos de linha. O SFM é um problema que consiste em recuperar uma estrutura 3-D (estruturas discretas, como pontos e linhas) a partir de uma sequência de imagens 2-D (conjunto 2-D de tais pontos ou linhas). Uma abordagem mais pomenorizada do SFM será exposta na secção dedicada aos aspectos implementacionais.

As características do mundo são estimadas em relação a um sistema de coordenadas global. O facto de as poses da câmara serem baseadas na odometria implica que os dados resultantes não sejam globalmente consistentes e não devam, por conseguinte, ser interpretados como um mapa cartesiano global.

Efectivamente, e tal como referimos na breve perspectiva histórica introdutória a esta secção, a inexactidão na localização torna difícil a construção de modelos globalmente consistentes, pois, à medida que o robot se move, os erros na sua localização vão-se acumulando. Como a pose da câmara é derivada da pose do robot, sendo esta por sua vez determinada a partir da odometria, torna-se evidente que a acumulação de erros na localização do robot se propaga à localização das características do mundo, do que resulta a inconsistência do modelo global.

As características estimadas são, no entanto, consistentes relativamente à pose do robot na altura em que são estimadas, e isto é tudo o que é necessário para permitir a navegação do robot.

3.3.3. VISÃO MÓVEL EM AMBIENTES EXTERIORES

Em ambientes exteriores destaca-se a investigação em veículos de estrada autónomos guiados visualmente, deslocando-se a alta velocidade. Como é evidente, os algoritmos de um sistema de visão para tais aplicações deverão ser rápidos e eficientes, sendo muito frequentemente necessário estabelecer um compromisso entre a profundidade da análise e a velocidade de resposta.

O MARF (Maryland Road Finder) [15] é um sistema desenvolvido na Universidade de Maryland, capaz de seguir estradas simples com intersecções. Foi projectado para conduzir um veículo autónomo por estradas desconhecidas, desviando-se de obstáculos. Constrói um modelo parcial do seu ambiente tridimensional a partir de imagens da estrada, modelo esse que contém os objectos relevantes do ambiente, ou seja, os limites esquerdo e direito da estrada, a linha central ou as valas paralelas à estrada.

O sistema de visão baseia-se na detecção de segmentos. Após a determinação de segmentos contíguos em imagens sucessivas, é possível reproduzir analiticamente as imagens seguintes sem necessidade de interpretar imagens do mundo real. O intervalo de tempo até nova leitura de imagens depende de a trajectória actual da estrada ser rectilínea ou curvilínea.

Um comportamento básico requerido de um veículo autónomo de estrada é passar de uma faixa para outra de modo a evitar um obstáculo ou ultrapassar outro veículo. Um passo inicial em direcção ao desenvolvimento da capacidade de decidir se uma mudança de faixa é praticável será desenvolver um localizador de faixas múltiplas que trabalhe em tempo real. Wershofen e Graefe [16] têm desenvolvido um trabalho neste sentido com o apoio do Ministério da Pesquisa e Tecnologia alemão e da indústria automobilística alemã, no âmbito do projecto PROMETHEUS.

Serão necessárias diferentes abordagens ao problema da detecção e localização de faixas, conforme uma estrada real se enquadre num dos três tipos básicos seguintes:

1) estradas com linhas marcadoras das fronteiras de cada faixa,

2) estradas parcialmente marcadas,

3) estradas não marcadas.

Nas situações em que apenas as duas fronteiras da estrada possam ser detectadas, a localização das faixas é determinada pela divisão geométrica da imagem em faixas. Em estradas parcialmente marcadas o reconhecimento de faixas pode basear-se primeiramente na detecção das fronteiras marcadas e adicionalmente na detecção de fronteiras não marcadas, sendo estas frequentemente as transições entre a estrada e o solo contíguo. Numa estrada completamente marcada, o reconhecimento pode ser reduzido ao reconhecimento das bem definidas marcações das faixas.

O localizador de múltiplas faixas é constituído pelas seguintes partes básicas:

- detector para a faixa do veículo,
- localizador para a faixa do veículo,
- um detector e um localizador para a faixa da esquerda,
- um detector e um localizador para a faixa da direita.

A faixa do veículo estará sempre presente, sendo as suas linhas de fronteira usadas como linhas de referência para a pesquisa de faixas adicionais em ambos os lados. Se uma faixa adicional for detectada, as suas marcações são localizadas. Se não houver detecção de faixas adicionais, a sua procura será repetida em cada imagem subsequente.

3.3.4. COMENTÁRIO

Estão assim apresentadas as principais ramificações da investigação actual em visão móvel. As principais aplicações são em sistemas robóticos que operam em edifícios, no caso dos ambientes interiores, e em estradas, no caso dos ambientes exteriores. Foi também possível detectar nas aplicações apresentadas a influência no sistema de visão da relação

com os restantes módulos da arquitectura do robot, em especial os sistemas de representação do mundo, navegação e o restante sistema de "sensing".

Realizam-se ainda, actualmente, importantes pesquisas em veículos robóticos para exploração submarina e exploração planetária. São projectos de grande dimensão e com um elevado grau de secretismo em relação às metodologias, tecnologias e resultados, o que combinado com a sua actualidade não nos permitiu obter qualquer tipo de informação relevante sobre os seus sistemas de visão.

3.4 PERSPECTIVA IMPLEMENTACIONAL

3.4.1. DETECÇÃO DE ARESTAS

Uma das observações mais importantes acerca do material exposto na secção anterior é a importância que os algoritmos de detecção de arestas desempenham nos sistemas de visão móvel. A ideia subjacente à maioria das técnicas de detecção de arestas é a computação de um operador de derivação local. Este conceito pode ser ilustrado com a ajuda da figura 3.7 [1].

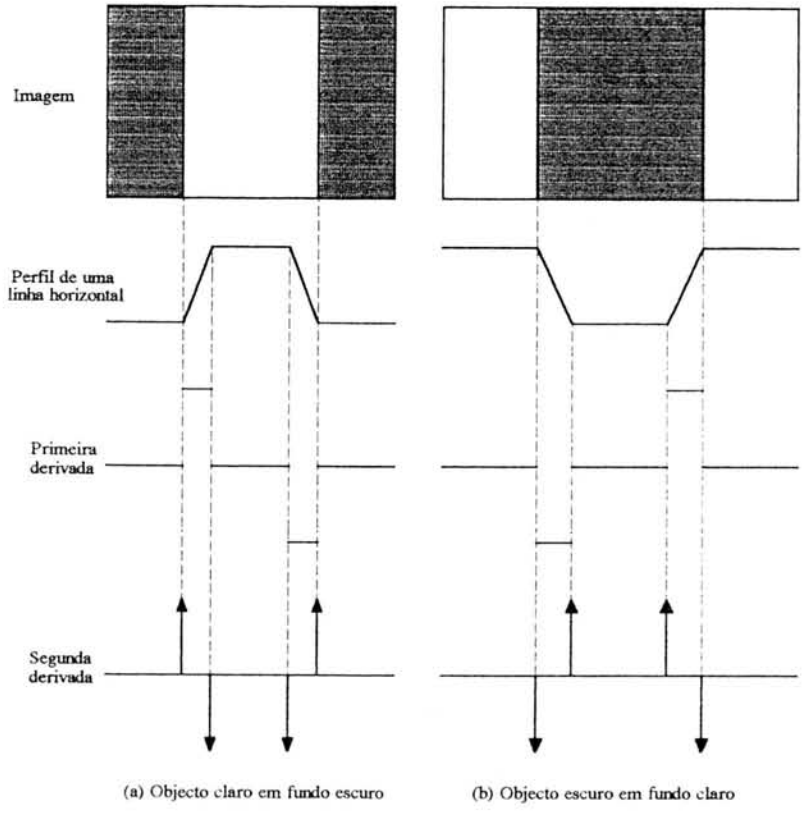


fig. 3.7 Elementos da detecção de arestas por operadores de derivação

Na parte (a) desta figura podemos observar uma imagem de um objecto claro sobre um fundo escuro, o perfil de intensidade ao longo de uma linha de "scan" horizontal da imagem, e as primeira e segunda derivadas. É observável no perfil de intensidade que uma aresta não é modelada como uma repentina mudança de intensidade, mas como uma rampa, sendo tal representativo do facto de que em imagens digitais, como resultado da amostragem, as arestas são geralmente ligeiramente mal definidas.

A primeira derivada de uma aresta assim modelada é zero em todas as regiões de intensidade constante e asume um valor constante durante uma transição de intensidade. A segunda derivada é sempre zero, excepto no início e término de uma transição de intensidade. Assim, o módulo da primeira derivada pode ser usado para detectar a presença de uma aresta, enquanto o sinal da segunda derivada permite determinar se um pixel de uma aresta fica no lado claro (objecto) ou escuro (fundo) de uma aresta. Para um objecto escuro em fundo claro podemos chegar a conclusões semelhantes.

Para uma aresta com qualquer direcção definimos um perfil perpendicular à orientação da aresta e interpretamos os resultados de modo semelhante à discussão precedente. A primeira derivada de qualquer ponto da imagem pode ser obtida usando o módulo do gradiente nesse ponto, enquanto a segunda derivada é dada pelo Laplaciano. Em [1] demonstra-se como determinadas máscaras podem ser usadas para computar o módulo do gradiente e o Laplaciano.

Embora o Laplaciano responda a transições em intensidade, raramente é usado por si só para detecção de arestas. O Laplaciano é um operador de derivação de segunda ordem, o que implica que, tipicamente, tenha uma sensibilidade inaceitável ao ruído. É este o motivo pelo qual lhe é delegado o papel secundário de estabelecer se um pixel pertence ao lado escuro ou claro de uma aresta.

No localizador de faixas de Wershofen e Graefe [16], o detector de linhas opera em duas fases. Primeiro, procura arestas individuais que possam correponder à aresta direita ou esquerda de uma marcação de faixa. De seguida tenta agrupar arestas adjacentes numa linha que corresponde a uma marcação de faixa.

1. Para a detecção de arestas individuais na imagem, é usado o método da correlação controlada. Este método é uma generalização do bem conhecido método da correlação, detectando características numa imagem através da correlação de uma pequena área da imagem com um protótipo da característica procurada. Este protótipo é usualmente designado máscara. A correlação é efectuada apenas segundo uma

direcção de pesquisa unidimensional. O resultado da correlação controlada é uma função discreta, chamada função de correlação. Um extremo da função de correlação indicará a localização da característica procurada.

Por motivos de eficiência, o detector de linhas usa máscaras ternárias (com elementos de valor +1, 0 e -1). A figura 3.8 mostra uma típica máscara ternária, tal como implementada no detector. Esta máscara em particular é selectiva para uma aresta com uma orientação de 120 graus. Para outras orientações são usadas outras máscaras, em quadrados de 16 por 16 pixels. A direcção de pesquisa é uma linha horizontal.

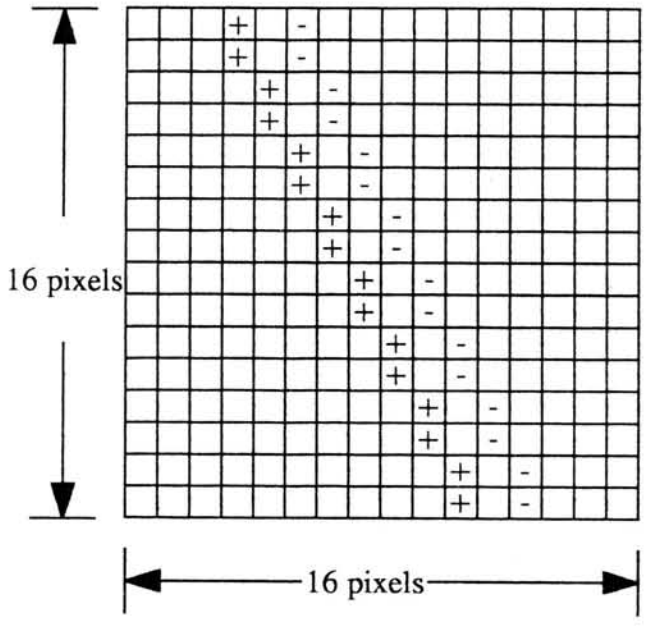


fig. 3.8 Máscara ternária para detecção de uma aresta com uma orientação de 120 graus

2. Uma marcação de faixa pode ser descrita como :

   . uma aresta no lado esquerdo com uma alteração do nível de cinzento de escuro para claro e

   . uma aresta do lado direito com uma alteração do nível de cinzento de claro para escuro.

Essas mudanças de nível de cinzento correspondem a dois extremos da função de correlação com sinais opostos e localizados perto um do outro. Se existir exactamente

um tal par no espaço de pesquisa, o detector de linhas retorna as coordenadas das arestas detectadas.

Em Kriegman *et al.*[5], a detecção de arestas incide apenas sobre as arestas verticais, agindo o algoritmo de detecção do seguinte modo:

1. Aplicação de um filtro de média vertical a ambas as imagens no horizonte. O horizonte, tal como referimos anteriormente, é obtido na calibração. A aplicação deste filtro tem os seguintes efeitos:

   . as arestas verticais mantêm a sua acuidade,

   . as arestas oblíquas perdem definição e as arestas horizontais desaparecem,

   . o efeito do ruído de imagem nas arestas verticais é reduzido,

   . o efeito de pequenas marcas e obscuridades é reduzido,

   . o efeito do mau alinhamento das câmaras no processo de correspondências é menor porque uma maior área efectiva de imagem será comparada.

2. Um modelo de aspecto de aresta é aplicado à linha de imagem filtrada. Este modelo compara um excerto local da imagem à imagem que teria sido criada se a câmara estivesse a observar uma aresta ideal. Para tal é usado o filtro espacial criado pela "pipeline" constituida pela lente, câmara, digitalizador e filtro de média.

Os parâmetros que este detector permite obter acerca de uma aresta são a sua qualidade, a sua posição, os níveis de cinzento à esquerda e à direita e uma estimativa do erro de localização.

Wells utiliza um detector de arestas dirigido por segmentos de imagem protótipo [14]. O detector encontra conjuntos de segmentos candidatos próximos de cada protótipo, requerendo que os segmentos candidatos tenham o mesmo sentido de gradiente ou "polaridade de contraste"que os seus predecessores. Usa um estimador de segmentos de linha 3-D que permite inferir segmentos de linha do mundo a partir de sequências de segmentos de linha de imagem correspondentes.

São três as fases de operação do sistema:

1. Prospecção ("Prospecting"). Os segmentos de prospecção são os primeiros segmentos protótipo usados. São gerados de modo a que a componente de detecção de características possa encontrar novas características de imagem.

2. "Bootstrapping". Durante esta fase, mantém-se uma pequena árvore de possíveis sequências de correspondência, ou seja das hipóteses alternativas sobre o sucessor de um segmento. Quando a árvore atinge uma profundidade mínima, usa-se o estimador de segmentos 3-D para gerar uma estimativa da característica do mundo, bem como uma medida de consistência para cada sequência na árvore. Se a sequência mais consistente estiver acima de um limiar de consistência mínima, é promovida a "actualização sequencial"; se tal não acontecer é rejeitada.

Os protótipos usados nesta fase são segmentos detectados na imagem precedente.

3. Actualização Sequencial ("Sequential Updating"). Nesta fase as novas características da imagem são envolvidas em estimativas de características do mundo à medida que chega cada imagem nova. Os segmentos protótipo são gerados tomando as projecções centrais (relativas ao centro de projecção da câmara) das estimativas anteriores dos segmentos 3-D no plano de imagem, usando a nova pose da câmara. A característica de imagem detectada que se aproxime mais do protótipo é, se suficientemente próxima, usada como sucessor.

O sistema usa vários subsistemas de prospecção, "bootstrapping" e actualização sequencial para manter a localização de um conjunto de características ambientais.

Ocasionalmente, o sistema encontrará um conjunto de microcorrespondências aparentemente consistente, que conduzirá à sobrevivência de uma hipótese incorrecta até à fase de actualização sequencial. No entanto, tais hipóteses falham rapidamente, sujeitas ao requisito de consistência a longo prazo.

Verifica-se que nas aplicações mais recentes em robótica móvel, em deterimento da computação de um operador de derivação local, os investigadores têm optado pela comparação com protótipos. Tal prende-se, sem dúvida, com motivos de eficiência dos algoritmos, dado que um dos principais requisitos dos sistemas de visão móvel é a operação em tempo real.

É também devido a este requisito que os três sistemas discutidos utilizam a detecção de características baseada em predição. É assim possível aplicar os detectores de arestas a pequenas áreas da imagem, eliminando a necessidade de um processamento global da imagem, o que é evidentemente vantajoso em termos de operação em tempo real.

No sistema de Kriegman *et al.* [5] a detecção das arestas verticais é aplicada apenas ao excerto de imagem correspondente ao horizonte das câmaras.

No sistema de Wells [14] a aplicação da predição é inerente à forma sequencial como opera o sistema, gerando os segmentos protótipo a partir de estimativas que envolvem os segmentos anteriores.

Wershofen e Graefe [16] recorrem a particularidades do problema da mobilidade em estradas que permitem praticar no seu sistema de visão a predição da localização das marcações das faixas. Efectivamente, ao conduzir numa auto-estrada o aspecto das faixas é bastante previsível. Numa situação típica pode assumir-se que as faixas têm uma largura constante e são rectas ou moderadamente curvas. As marcações das faixas aparecem como duas linhas (praticamente) rectas cujas localizações e declives aproximados na imagem são mais ou menos predizíveis. Esta é a base para encontrar as marcações das faixas na imagem.

Para a faixa do veículo o detector opera do seguinte modo:

1. Dois detectores de linhas são colocados na parte inferior da imagem e a partir daí, seguindo uma trajectória vertical, eles iniciam a pesquisa das linhas brancas que marcam a faixa própria do veículo. Um procura a marcação direita e outro a esquerda. Logo que uma linha seja detectada, ela é seguida em direcção ao topo da imagem.

2. Subsequentemente, a faixa deve ser localizada numa distância pré-determinada de "look-ahead". Assume-se que existe uma relação conhecida entre a largura aparente da faixa na imagem e a distância de "look-ahead", o que é justificável pois os parâmetros que regem a projecção da faixa na imagem (e.g. comprimento focal, ângulo da câmara, largura da câmara) são relativamente bem conhecidos. Os detectores de linhas seguem as linhas até aos pontos da imagem em que a distância horizontal entre as linhas tem um certo valor que corresponde aproximadamente à

distância de "look-ahead" pré-determinada. As coordenadas das marcações nesses pontos são então passadas ao localizador da faixa.

Estas duas etapas pertencem à fase de inicialização, que é efectuada apenas sobre uma imagem. Poderão ser repetidas mais tarde em situações em que seja necessário reinicializar o sistema.

Após a (re)inicialização, a pesquisa das linhas em cada nova imagem é baseada na esperança de que elas aparecerão em regiões bem definidas da imagem, chamadas espaços de pesquisa. Inicialmente os espaços de pesquisa são centrados nas coordenadas dadas pelo detector da faixa do veículo. A coordenada vertical dos espaços de pesquisa é dada pela distância de "look-ahead" e não será alterada durante a localização. As coordenadas horizontais, contudo, serão continuamente ajustadas de forma a manter as marcações da faixa centradas nos espaços de pesquisa. Para conseguir isto, gera-se a localização esperada das linhas na imagem seguinte através de uma filtragem passa-baixo das coordenadas obtidas pelos detectores de linhas em imagens anteriores.

Devido à possibilidade sempre presente de o localizador perder o seu alvo, a sua operação é continuamente monitorizada:

. Se um detector de linhas não consegue detectar a sua linha por um período de tempo superior a um certo intervalo, o seu espaço de pesquisa é reposicionado. A nova localização é obtida a partir da localização da marcação de faixa encontrada pelo detector de linhas oposto e da largura nominal da faixa do veículo.

. Se ambos os detectores de linhas não fornecerem coodenadas por um período de tempo superior a um certo intervalo, a localização da faixa do veículo é interrompida temporariamente.

. A distância entre as linhas localizadas é continuamente comparada com a largura nominal da faixa do veículo. Se a diferença exceder um certo limite, assume-se que a faixa do veículo já não está a ser localizada correctamente, e a localização é interrompida temporariamente.

Sempre que a localização seja interrompida, é automaticamente accionado o detector para reinicializar o processo. Para tal, o sistema de visão conta com um processo de (re)inicialização suficientemente curto, para dotar o sistema de maior robustez, permitindo lidar com situações como marcações interrompidas e entradas ou saídas de túnel (em que há rápidas alterações do nível de iluminação que impedem a câmara de fornecer imagens úteis).

A pesquisa de faixas adicionais, uma à esquerda e outra à direita, é iniciada em simultâneo com a localização da faixa do veículo. O pressuposto principal desta operação é as faixas adicionais terem a mesma largura que a faixa própria do veículo. É colocado um detector de linhas à esquerda e outro à direita da faixa própria. Ao invés do que acontece com a faixa própria, os espaços de pesquisa não são centrados em torno das marcações detectadas e a largura da faixa detectada não é verificada.

3.4.2. ALGORITMOS STEREO

Kriegman, Triendl e Binford apresentam [5] um algoritmo stereo que utiliza arestas, níveis de cinzento, correlação de intensidades e propagação de restrições, ilustrando de forma excelente os principais aspectos a considerar numa implementação.

As correspondências stereo são determinadas em três fases: primeiro, propõem-se correspondências com base em informação local das arestas, de seguida confirmam-se ou negam-se estas correspondências considerando os intervalos entre correspondências vizinhas e mais globalmente, ligam-se séries consistentes de vizinhanças.

Com todos os possíveis pares de arestas que respeitam as restrições stereo, a primeira etapa do algoritmo propõe as correspondências que são semelhantes do lado direito ou esquerdo da aresta ou tenham um perfil de intensidade semelhante. Estas correspondências são ordenadas de acordo com uma estimativa inicial da qualidade da correspondência. É importante salientar que nas correspondências propostas são incluidas as correspondências de arestas oclusivas para as quais o lado do objecto da aresta é semelhante mas os fundos diferem.

De seguida são comparadas as curvas de intensidade de nível de cinzento entre correspondências vizinhas em cada imagem. As curvas de intensidade são interpoladas e reamostradas para um número constante de amostras. Determina-se então a correlação normalizada. A função de comparação de níveis de cinzento retorna este resultado pesado pelo desvio padrão dos níveis de cinzento, pela diferença em nível de cinzento médio e pela diferença em comprimento do intervalo. Se a função de comparação de níveis de cinzento tiver um valor grande, confirmam-se as correspondências propostas e estabelece-se uma ligação entre estas correspondências. Como efeito lateral, uma correlação alta num intervalo entre arestas resulta frequentemente de um objecto sólido no modelo, como uma parede ou uma porta fechada.

Finalmente, é produzido um conjunto de grafos orientados nos quais os nós são correspondências propostas e os arcos são as ligações de vizinhança determinadas acima. Todos os caminhos através de cada grafo formam um conjunto de correspondências consistentes. Quando um nó não tiver arcos de entrada e saída, é formado outro grafo. Tal pode dever-se a uma oclusão, à fronteira da imagem ou a uma omissão num fase prévia. A qualidade de um caminho num grafo é determinada pela soma das qualidades das correspondências ao longo do caminho.

Estabelece-se uma ligação entre caminhos em diferentes grafos se forem consistentes, ou seja se cada aresta só tiver uma correspondência. Para cada caminho ligado, determina-se uma medida de qualidade também através de um somatório. Se houver mais que um caminho consistente de alta qualidade, o algoritmo stereo retorna um conjunto de potenciais correspondências. Para resolver qualquer ambiguidade que ainda reste podem ser utilizadas, se disponíveis, correspondências de movimento.

No sistema de visão em que é usado este algoritmo, o detector de arestas (referido na subsecção anterior), as correspondências da primeira fase do algoritmo stereo e a função de comparação de níveis de cinzento estão implementadas em C. As partes restantes estão implementadas em LISP. Os algoritmos correm num VAX 11/750 e o tempo de processamento é de aproximadamente 1 segundo por par stereo.

3.4.3. MÉTODOS "STRUCTURE FROM MOTION"

O problema "structure from motion" (SFM) consiste, como já referimos, na recuperação de uma estrutura 3-D a partir de uma sequência de imagens 2-D, mais concretamente, na recuperação de informação 3-D sobre um conjunto de estruturas discretas, como pontos ou linhas, a partir de um conjunto 2-D de tais pontos ou linhas.

As imagens 2-D são formadas a partir do mundo 3-D através de um processo de projecção. O SFM pretende tomar a informação 2-D e e recuperar a informação 3-D original, invertendo o efeito do processo de projecção. Os dois tipos de processos de projecção são a projecção perspectiva e a projecção ortográfica. A projecção perspectiva é um modelo realista do processo de formação da imagem, enquanto a projecção ortográfica permite modelos de fácil resolução aplicáveis em alguns casos simples. Os diferentes processos de projecção estão relacionados com diferentes classes de algoritmos SFM.

As entradas dos algoritmos SFM são informação sobre posições 2-D de pontos ou linhas correspondentes que emergem da projecção da mesma informação física 3-D na

sequência de "frames". Possivelmente é também fornecida informação sobre o movimento. Os algoritmos produzem informação sobre estrutura ou movimento 3-D.

FORMULAÇÕES DO SFM

Os métodos SFM usam frequentemente formulações que utilizam o eixo e ângulo de rotação do objecto ou observador para gerar uma matriz de rotação, matriz esta que é uma função não linear do eixo e do ângulo. Como o problema SFM é uma função linear da matriz de rotação, com dados típicos que são equações lineares de translações puras, o resultado é um conjunto de equações não lineares transcendentes.

Alternativamente, pode-se remover a restrição de usar uma matriz de rotação e usar uma matriz arbitrária. Neste caso a formulação será linear, contudo não haverá garantias de que a matriz arbitrária tenha as propriedades correctas.

Têm sido usadas outras restrições não lineares que recorrem a propriedades 3-D como a invariância dos ângulos que ligam as linhas 3-D e a invariância do comprimento de segmentos de linha 3-D. Estas restrições resultam em equações cujos parâmetros desconhecidos são o movimento ou valores das estruturas. Tais equações são satisfeitas pela solução correcta que conduz ao movimento e estrutura correctos, mas também por muitas outras soluções incorrectas, entre as quais soluções fisicamente impossíveis.

UMA TAXONOMIA

Existe uma multidão de equações, métodos, formulações e soluções na literatura. Em [17], C. P. Jerian e R. Jain apresentam uma vastíssima bibliografia sobre o SFM. O seu trabalho consistiu em classificar e analisar os métodos existentes, e mostrar as relações matemáticas existentes entre eles.

Com efeito, os diferentes métodos podem ser relacionados por certas alterações, algumas das quais preservam as qualidades essenciais do sistema original, enquanto outras não. Por exemplo, pode-se eliminar uma variável através de manipulações algébricas, ou eliminar uma equação necessária que se revele difícil de resolver, ou estabelecer um pressuposto restritivo acerca do domínio do problema, obtendo-se um problema mais simples.

Jerian e Jain desenvolvem uma taxonomia que usa as seguintes classificações de alterações das equações do SFM:

1. Alterações neutras, que eliminam variáveis de forma a obter equações mais simples. Não criando quaiquer soluções falsas adicionais, este tipo de alterações afecta, no entanto, o peso das várias equações num ajuste de mínimos quadrados.

2. Alterações de aumento da dimensão, cuja popularidade reside no facto de conduzirem a equações lineares. Contudo, o facto de serem conseguidas ignorando restrições torna-as menos fiáveis na presença de ruído do que os métodos que usam apenas alterações neutras.

3. Alterações de redução da dimensão, aplicáveis em conjunto com qualquer um dos outros métodos através da eliminação de variáveis que, à partida, se sabe serem irrelevantes para um caso particular. O SFM completamente geral requer cinco variáveis mas, se o movimento se limitar a um plano conhecido, bastam uma única variável de rotação e uma única de translação.

4. Alterações de extensão finita, que são usadas para substituir restrições complexas por restrições mais simples sem introduzir quaisquer variáveis ou dimensões extra. A solução correcta está entre a solução das equações simplificadas, mas estas podem ter um número de soluções múltiplo do número de soluções das equações originais.

Para obter um bom SFM, Jerian e Jain apontam dois factores fundamentais:

1. Usar um bom algoritmo que evite as armadilhas dos métodos tradicionais não lineares e os riscos dos métodos lineares.

2. Obter quantidade e tipo de movimento que permitam vistas suficientemente diferentes e um bom condicionamento do problema de estrutura.

3.4.4. A Incerteza no Processo de Visão

N. Ayache e O. D. Faugeras publicaram um trabalho [9] que constitui uma forte referência no capítulo da incerteza no processo de visão. Nesse trabalho, conduzem uma profunda reflexão e análise dos problemas de construção e actualização de uma representação 3-D do ambiente de um robot móvel que usa como principal modalidade sensorial a visão.

A principal motivação do trabalho por eles desenvolvido é a necessidade de obter uma representação quer da geometria quer da incerteza. Apresentam uma ferramenta extremamente bem adaptada à resolução da maioria dos problemas que essa necessidade levanta: o Filtro de Kalman Extendido (Extended Kalman Filter - EKF). Esta ferramenta é também utilizada por Kriegman *et al.* no já referido trabalho [5].

A maioria das modalidades de "sensing" partem de pixels que são então convertidos em estruturas 3-D. Torna-se evidentemente necessário considerar a presença de ruído desde os pixels até à geometria 3-D.

Se o ruído está presente, deve ser avaliado, ou seja, precisamos de modelos de ruído sensorial. Esse ruído necessita ainda de ser reduzido, redução esta que pode ser obtida de variadas formas:

1. Para sensores em posições fixas, é possível repetir as medidas para obter melhores estimativas.

2. Para sensores móveis, dadas as medidas efectuadas numa determinada posição, determinar qual o melhor movimento, de modo a reduzir a incerteza e aumentar o conhecimento do ambiente de uma forma compatível com a tarefa em mãos.

3. Para vários sensores diferentes, combinar as suas medidas de uma forma que tenha significado.

Assim, surgem vários problemas que requerem maior atenção:

1. A relação entre o ruído sensorial e a incerteza geométrica.

2. A representação da informação geométrica tendo em conta a descrição não apenas da geometria, mas também da incerteza nessa geometria.

3. A combinação da informação geométrica incerta produzida por diferentes sensores.

Ayache e Faugeras definem bem quais os problemas que tentam resolver, relacionados com um robot que se movimenta em interiores, usando visão e odometria.

A. Construir Descrições 3-D Locais do Ambiente

As fontes principais de informação 3-D são o Stereo e o SFM, enfrentando ambas as modalidades problemas semelhantes. Ayache e Faugeras concentram-se no Stereo.

O principal problema na construção de descrições 3-D locais do ambiente é o modo como se propaga a incerteza geométrica quando construimos primitivas mais complexas a partir de primitivas mais simples:

. O conhecimento imperfeito das posições dos pixels implica que a posição do ponto 3-D possa variar numa área com forma de diamante (figura 3.9).

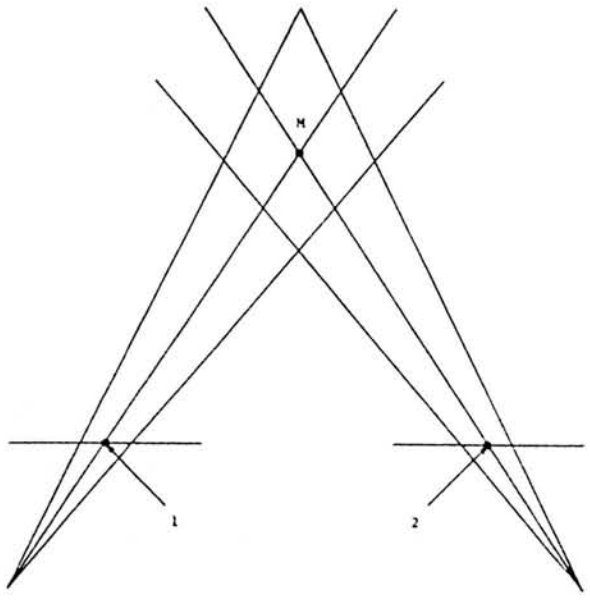


fig. 3.9 Efeito do ruído do pixel sobre a reconstrução 3-D

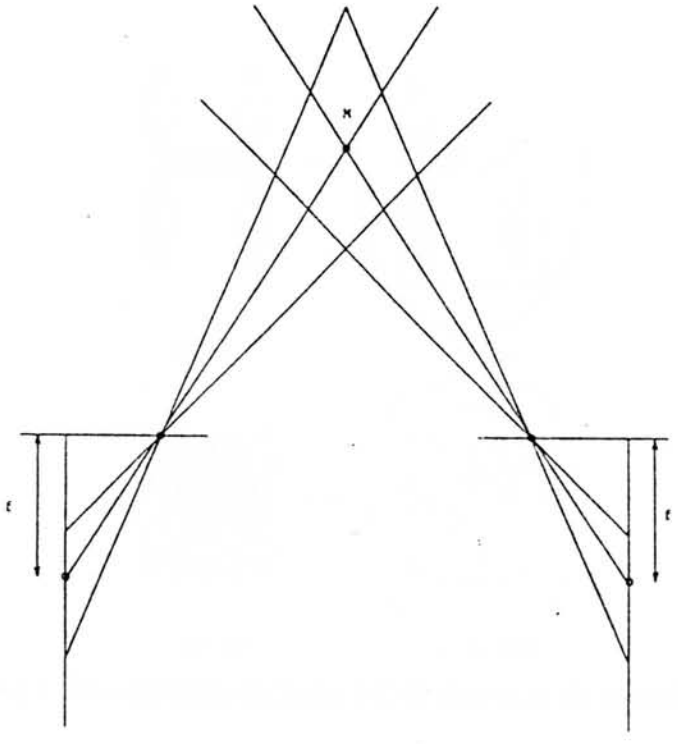


fig. 3.10 Efeito dos erros de calibração sobre a reconstrução 3-D

. Efeito semelhante têm os erros de calibração (figura 3.10), nomeadamente as incertezas resultantes do cálculo de parâmetros intrínsecos como o comprimento focal e parâmetros extrínsecos como a posição e orientação relativas das câmaras.

. Outro exemplo de propagação é apontado na figura 3.11: a propagação pixel-2D-3D. Ao agrupar pixels em rectas, a incerteza dos pixels propaga-se às linhas 2-D. No "matching" dos segmentos de linha 2-D para reconstruir os segmentos 3-D, a incerteza de calibração e a incerteza 2-D convertem-se em incerteza 3-D.

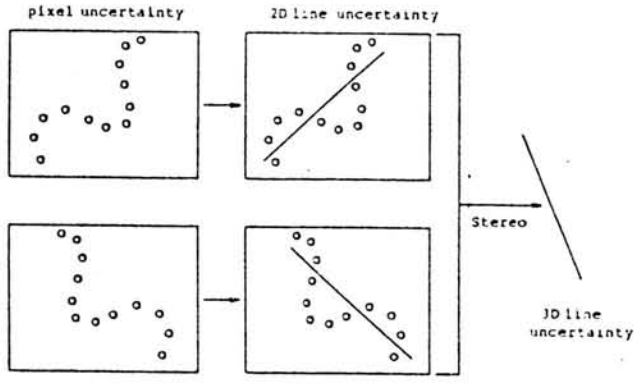


fig. 3.11 Da incerteza do pixel à incerteza da linha 3-D

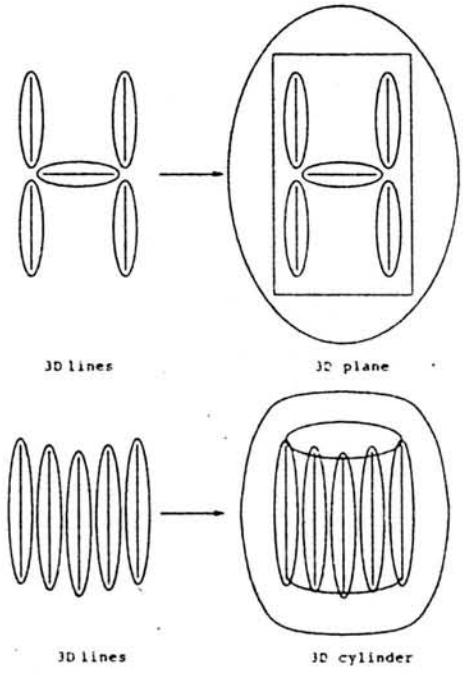


fig. 3.12 Da incerteza da linha 3-D à incerteza da superfície 3-D

. Um outro conjunto de exemplos desta propagação pode ser observado na figura 3.12, onde segmentos de linha coplanares ou cocilíndricos são agrupados. A questão

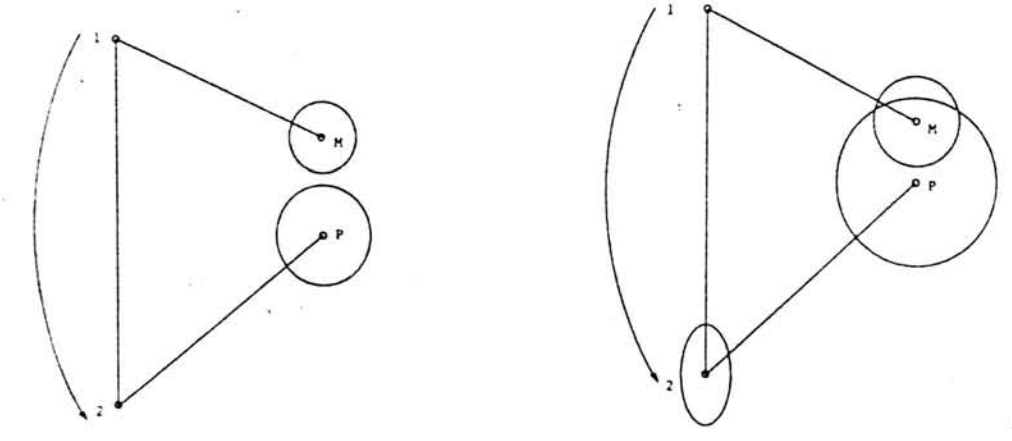
é, novamente, qual a incerteza do plano ou do cilindro? (A incerteza é representada na figura, de um modo simbólico, por elipses).

Resultam, de tudo isto, ainda mais duas questões:

1. Como representar primitivas geométricas?
2. Como representar a incerteza nessas primitivas?

B. Actualizar Informação de Posição e Movimento

Numa situação em que a medida de um ponto físico é feita de duas posições diferentes, se for tomada em conta a incerteza do deslocamento, a combinação dessa incerteza e da incerteza da medida pode evitar que um mesmo ponto possa ser visto como 2 pontos diferentes (figuras 3.13 (a) e (b)). As medidas podem ainda ser usadas para produzir melhores estimativas das posições.



(a) Estimativa errada do movimento (b) Estimativa de deslocamento e incerteza

fig. 3.13 Medida de um ponto a partir de duas posições

C. Fusão de Entidades Geométricas

O veículo móvel efectua medidas de um ponto físico M a partir de n posições ($1..n$) fornecendo, para cada medida, um ponto M_i, $i=1, \ldots, n$ e a incerteza no sistema de coordenadas associado ao robot. Está também disponível a incerteza do deslocamento. Descobrindo que $M_1, \ldots, M_n$ são instanciações do mesmo ponto, podemos melhorar as estimativas dos deslocamentos e reduzir a sua incerteza.

É possível reduzir a incerteza em, digamos, M_1, combinando as n medidas e produzindo um ponto $\mathcal{m}$, fusão de M_1, ..., M_n, bem como a sua incerteza associada. Os pontos M_1, ..., M_n podem então ser apagados da representação do ambiente. O que fica é a o ponto $\mathcal{m}$ expresso no sistema de coordenadas ligado à posição 1, por exemplo, e os deslocamentos de 1 para 2, 2 para 3, etc., que nos permitem expressar $\mathcal{m}$ nos outros sistemas de coordenadas.

Obtém-se assim um "esquecimento inteligente" e evitamos que a representação do ambiente cresça demasiado.

D. Descobrir Relações Geométricas "Interessantes"

O interesse neste problema reside em caracterizar a probabilidade de existir uma determinada relação geométrica entre um determinado número de entidades geométricas e usar essa informação para obter melhores estimativas dessas entidades e reduzir a sua incerteza. As relações típicas serão de perpendicularidade e paralelismo.

E. Descobrir Entidades Semânticas

Combinando geometria e uma descrição a priori do ambiente é possível realizar agrupamentos semânticos. Por exemplo, se tivermos vários segmentos coplanares e o plano correspondente for vertical podemos assumir que se trata de uma parede; segmentos dessa parede que formem rectângulos podem ser vistos como partes de janelas ou de portas.

A metodologia proposta em [18] apresenta as seguintes características salientes:

. Representação:

1) As primitivas geométricas são usadas para descrever o ambiente, enquanto os movimentos são descritos por deslocamentos rígidos.

2) A incerteza é modelada por uma função de densidade de probabilidade destes parâmetros.

3) As relações entre entidades geométricas são representadas por equações algébricas nos seus parâmetros.

. Algoritmos: detecção de relações geométricas, cálculo e actualização dos parâmetros das entidades geométricas (para primitivas e deslocamentos) feitos por algoritmos de predição-e-verificação recursivos, incluindo o Filtro de Kalman Extendido. Estes algoritmos encontram-se melhor detalhados em [18], [19] e [20].

Os resultados experimentais expostos em [9] comprovaram a validade da maioria das aproximações feitas por Ayache e Faugeras em vários casos práticos. No entanto os autores não deixam de reconhecer que muito trabalho teórico e experimental é ainda necessário para alargar a sua abordagem a uma classe mais vasta de problemas que não permitem essas aproximações.

Akio Kosaka e Avi Kak [21] propõem também um sistema de visão que mantém um modelo de incerteza e mantém ainda conhecimento sobre o crescimento da incerteza à medida que o robot se move. Este trabalho, fortemente inspirado pelos trabalhos já referidos de Ayache e Faugeras [9] e de Kriegman *et al.* [5], apresenta no entanto em relação a eles diferenças substanciais.

Ao invés de Ayache e Faugeras, Kosaka e Kak concentram-se na navegação autónoma num ambiente já conhecido, via CAD ou outras representações geométricas, enquanto os primeiros partem do pressuposto de que não existe à partida um modelo do ambiente.

Enquanto Kriegman *et al.* focam o seu interesse em processos stereo ou na construção de modelos, Kosaka e Kak preocupam-se com o estabelecimento de correpondências entre "landmarks" derivados dos modelos genéricos dos corredores e características de imagens monoculares. As incertezas estimadas são usadas para limitar a região da imagem que deve ser analisada e restringir fortemente as escolhas de características de imagem candidatas às "landmarks" que o robot espera ver.

BIBLIOGRAFIA

[1] K. S. Fu, R. C. Gonzalez, C. S. G. Lee
"Robotics: Control, Sensing, Vision and Intelligence"
McGraw - Hill
(1987)

[2] N. Ayache, O. D. Faugeras
"Building, registrating, and fusing noisy visual maps"
Int. Conf. on Computer Vision, pp. 73-82
(1987)

[3] Z. L. Cao, S. J. Oh, E. L. Hall
"Dynamic omnidirectional vision for mobile robots"
Journal of Robotic Systems, Spring '86, pp. 5-

[4] Y. Yagi, Y. Nishizawa, M. Yachida
"Obstacle avoidance for mobile robot integrating omnidirectional image sensor COPIS and ultrasonic sensor"
ICARV '92 - vol.3, pp. RO 11.7.1-11.7.5

[5] D. J. Kriegman, E. Triendl, T. O. Binford
"Stereo vision and navigation in buildings for mobile robots"
IEEE Trans. on Robotics and Automation, vol. 5, no. 6, Dec. 1989, pp. 792-803

[6] N. J. Nilsson
"Shakey the robot"
Tech. Rep. 323, SRI International Artificial Intelligence Center
(1984)

[7] H. P. Moravec
"The Stanford cart and the CMU rover"
Proc. IEEE, vol. 71, no. 7, Jul. 1983, pp. 872-884

[8] J. Lowrie *et al.*
"Autonomous land vehicle annual report"
Tech, Rep., US Army ETL
(1985)

[9] N. Ayache, O. D. Faugeras
"Maintaining representations of the environment of a mobile robot"
IEEE Trans. on Robotics and Automation, vol. 5, no. 6, Dec. 1989, pp. 804-819

[10] R. A. Brooks, A. M. Flynn, T. Marill
"Self calibration of motion and stereo vision for mobile robots"
R. C. Bolles and B. Roth, Eds.,
The Fourth International Symposium on Robotics Research
Cambridge, MA: MIT Press, 1988, pp. 275-286

[11] J. D. Horswill, R. A. Brooks
"Situated vision in a dynamic world: chasing objects"
Proc. 7th Nat. Conf. on Artificial Intelligence, pp 796-800
(1988)

[12] K. B. Sarachik
"Characterising an indoor environmente with a mobile robot and uncalibrated stereo"
IEEE Int. Conf. on Robotics and Automation
(1989)

[13] D. J. Kriegman, T. O. Binford
"Generic models for robot navigation"
IEEE Int. Conf. on Robotics and Automation, 1988

[14] W. M. Wells, III
"Visual Estimation of 3-D line segments from motion - a mobile robot vision system"
IEEE Trans. on Robotics and Automation, vol. 5, no. 6, Dec. 1989, pp. 820-825

[15] L. S. Davis
"Visual navigation at the University of Maryland"
Intelligent Autonomous Systems, vol. 1, pp. 1-

[16] K. P. Wershofen, V. Graefe
"A real-time multiple lane tracker for an autonomous road vehicle"
S. G. Tzafestas (ed.), Robotic Systems, pp. 333-340
Kluwer Academic Publishers
(1992)

[17] C. P. Jerian, R. Jain
"Structure from motion - a critical analysis of methods"
IEEE Trans. on Systems, Man and Cybernetics.,
vol. 21, no. 3, May/June 1991, pp. 572-588

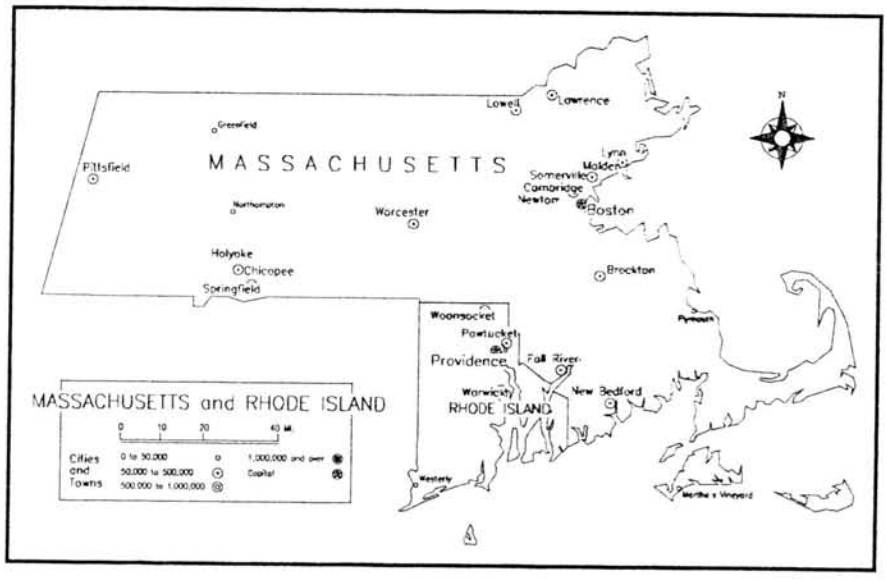
[18] N. Ayache, O. D. Faugeras, F. Lustman, Z. Chang
"Visual navigation of a mobile robot"
IROS '88

[19] N . Ayache
"Construction et fusion de répresentations visuelles tridimensionelles: applications à la robotique mobile"
Thèse d'Etat, Université de Paris-Sud, May 1988, INRIA Int. Rep.

[20] N . Ayache
"Vision stéreoscopique et perception multisensorielle - application à la robotique mobile"
Inter-editions
(1989)

CAPÍTULO 4

REPRESENTAÇÃO ESPACIAL



4.1 INTRODUÇÃO

O conhecimento do ambiente do robot é vital para a execução das suas tarefas. Diversas representações têm sido desenvolvidas com este objectivo básico de permitir a navegação do robot. No entanto, o desenvolvimento dessas representações responde a motivações diferentes, que estão na base da divisão em secções que utilizaremos para expor o material que incluimos neste capítulo.

Inicialmente apresentaremos o Espaço de Configuração [1] [2] [3], abordagem que surge directamente motivada pelo problema do planeamento de trajectórias para objectos poligonais rígidos num espaço populado por objectos poligonais. Esta abordagem consiste de uma representação explícita do espaço proibido ao objecto móvel, sendo o espaço livre representado como o complementar desse espaço proibido.

Seguidamente, debruçar-nos-emos sobre representações explícitas do espaço livre, obtidas a partir de uma descrição inicial do ambiente de trabalho do robot e dos obstáculos desse ambiente: Cones Generalizados [4], Diagramas de Voronoi [5] [6] [7] [8] [9] e Triangulação [10] [11] [12] [13].

A secção final será dedicada a representações que podem ser construidas pelo próprio robot a partir de informação sensorial, os mapeamentos: Quadtree [14] [15] e Grelhas de Ocupação ou Grelhas de Incerteza [16]. Faremos também referência a um algoritmo de aquisição de terreno a partir de informação sensorial [17].

Algumas das ideias aqui expostas são também referidas no capítulo sobre navegação, devido à forte interligação entre as ferramentas de representação espacial com que um robot está dotado e as suas metodologias navegacionais. No entanto o nosso propósito de sistematização leva-nos a apresentá-las neste capítulo, contribuindo assim para uma melhor visão deste importante aspecto da investigação em robótica móvel.

4.2 ESPAÇO DE CONFIGURAÇÃO

Lozano-Pérez introduziu no início da década de 80 uma das mais importantes ferramentas de representação espacial: o Espaço de Configuração [1] [2]. Esta ferramenta produz uma grande simplificação no problema do planeamento de trajectórias, simplificação essa que tem permitido estudar uma grande diversidade de problemas navegacionais.

A configuração de um objecto rígido é um conjunto de parâmetros independentes que caracterizam a posição de cada ponto no objecto. Associando um sistema de coordenadas local a um objecto rígido, como um polígono planar, podemos especificar a configuração do polígono pela posição x,y da origem do sistema de coordenadas local, conhecida como ponto de referência, e um valor θ indicando a rotação do sistema de coordenadas local em relação ao sistema global.

O espaço de todas as configurações de um objecto é o seu espaço de configuração.

Um ponto no espaço de configuração, um ponto de configuração, representa uma posição particular do ponto de referência do objecto e uma orientação dos eixos do objecto.

O espaço de configuração para polígonos planares é tridimensional, enquanto para poliedros sólidos tem dimensão seis: três dimensões translacionais e três rotacionais.

Devido à presença de obstáculos imóveis, algumas regiões do espaço de configuração não são alcançáveis; estas regiões são os obstáculos de configuração. Consequentemente, no espaço de configuração, o objecto móvel é reduzido a um ponto de configuração, enquanto os obstáculos imóveis são expandidos de modo a preencherem todo o espaço em que a presença do ponto de configuração implique uma colisão do objecto com os obstáculos.

Na figura 4.1 podemos observar o espaço de configuração para um objecto móvel quadrangular, sem rotação.

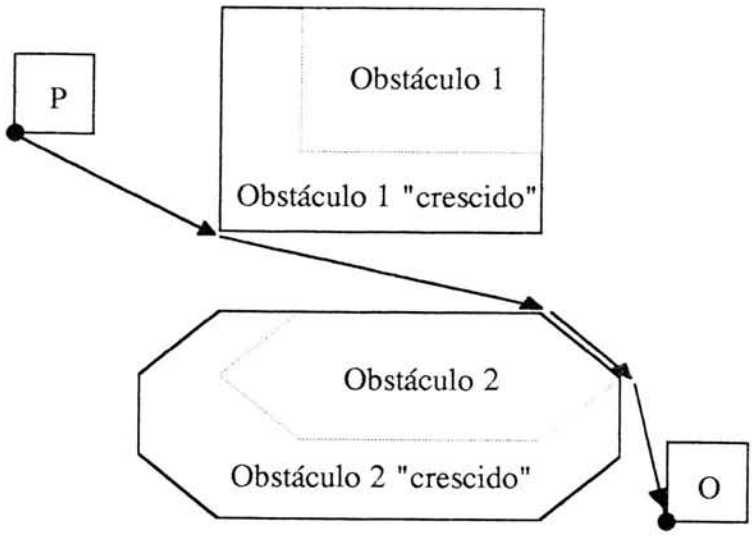


fig. 4.1 Construção dos obstáculos do Espaço de Configuração (2-D sem rotação)

Esta representação apresenta a importante vantagem de permitir um planeamento de trajectórias não geométrico, sendo utilizada em múltiplas abordagens ao problema navegacional. Numa das mais curiosas aplicações do espaço de configuração, a expansão dos obstáculos é feita a um nível local, tendo em conta, além da posição e da orientação, a própria velocidade do objecto móvel [3].

4.3 REPRESENTAÇÕES DO ESPAÇO LIVRE

4.3.1. CONES GENERALIZADOS

Brooks utiliza em [4] uma representação do espaço livre por cones generalizados sobrepostos. Como estes são descrições de volumes varridos, constituem representações que captam os efeitos essenciais da translação e rotação de um corpo através do espaço.

A sua descrição desta representação é restrita ao problema 2-D em que os objectos são polígonos convexos ou colecções de polígonos convexos.

Um cone generalizado é formado varrendo uma secção transversal ao longo de uma curva no espaço, chamada espinha, e deformando-a de acordo com uma regra de varrimento.

Brooks considera uma especialização 2-D dos cones generalizados, em que as espinhas são linhas rectas e as secções transversais são segmentos de recta perpendiculares à espinha com raio esquerdo e raio direito. A regra de varrimento controla independentemente as amplitudes dos raios esquerdo e direito.

O espaço livre é representado como cones generalizados sobrepostos. "Freeways" naturais, regiões alongadas através das quais um objecto pode ser movido, são representadas como cones generalizados, que se sobrepões nas intersecções dessas "freeways".

A figura 4.2 ilustra alguns dos cones generalizados que descrevem o espaço livre num espaço de trabalho com três obstáculos.

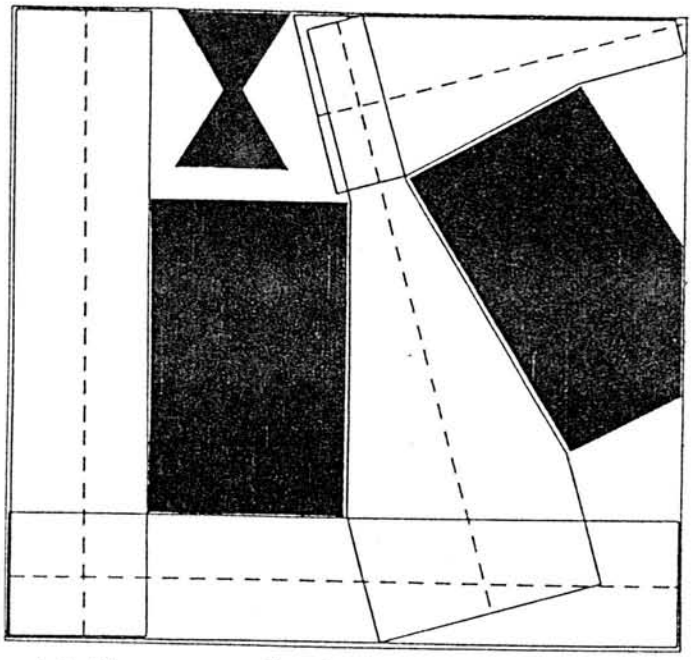


fig. 4.2 Cones generalizados gerados por dois obstáculos e fronteira do espaço de trabalho

A representação do espaço livre é construida examinando todos os pares de arestas de polígonos obstáculo. Se as arestas definirem uma "freeway" natural através do espaço então são usadas para construir um cone generalizado.

Cada aresta tem um lado "cheio" e um lado "livre", em que o lado "livre" é a parte de fora do polígono ao qual a aresta está associada. Cada aresta tem uma normal que aponta para fora, para o lado "livre".

Duas arestas serão aceites como cone generalizado candidato se preencherem os seguintes requisitos:

1. Pelo menos um vértice de cada aresta deve estar no lado "livre" da outra.

2. O produto escalar das duas normais que apontam para fora deve ser negativo.

A segunda condição assegura que os lados "livres" das duas arestas estão essencialmente em frente um do outro.

Dado um par de arestas candidato, é construida uma espinha para o cone generalizado. Essa espinha será a bissectriz do espaço que está no lado "livre" de ambas as arestas. Assim, se as arestas forem paralelas a espinha será paralela e equidistante a ambas; se as arestas não forem paralelas, então a espinha é a bissectriz do ângulo que elas formam.

O cone generalizado ocupa a superície entre a duas arestas. Em cada vértice das duas arestas (se o vértice estiver no lado "livre"da outra aresta) o cone é extendido paralelamente à espinha.

O cone generalizado assim definido pode não ficar inteiramente no espaço livre. Pode haver outros obstáculos que o intersectem. Cada obstáculo é comparado com o cone generalizado. Se a intersecção for vazia, então nada mais será necessário fazer. Se não, então a intersecção é projectada normalmente na espinha do cone generalizado (figura 4.3).

O resultado da comparação de todos os obstáculos com o cone generalizado é um conjunto de regiões da espinha em que nenhum obstáculo intersecta o cone numa fatia normal à espinha. Cada fatia que inclua partes das duas arestas originais é então aceite como um cone generalizado descrevendo parte do espaço livre.

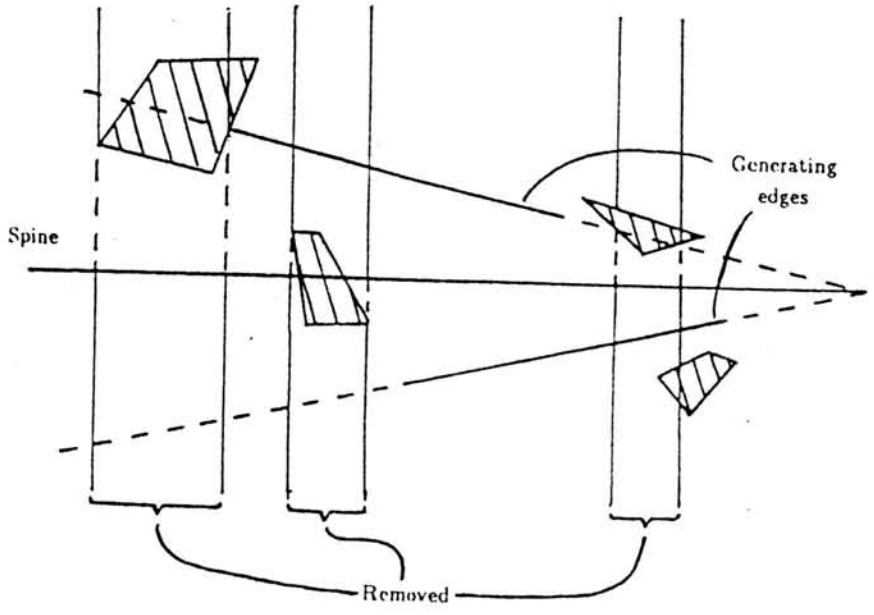


fig. 4.3 Fatias de um cone generalizado removidas devido a obstáculos

Na figura 4.4 encontra-se a representação completa usada para os cones que descrevem partes do espaço livre. A espinha recta é parametrizada no intervalo $t \in [0, 1]$, em que 1 é o comprimento do cone. Se os lados do cone não forem paralelos à espinha então $t=0$ corresponde ao comprimento maior.

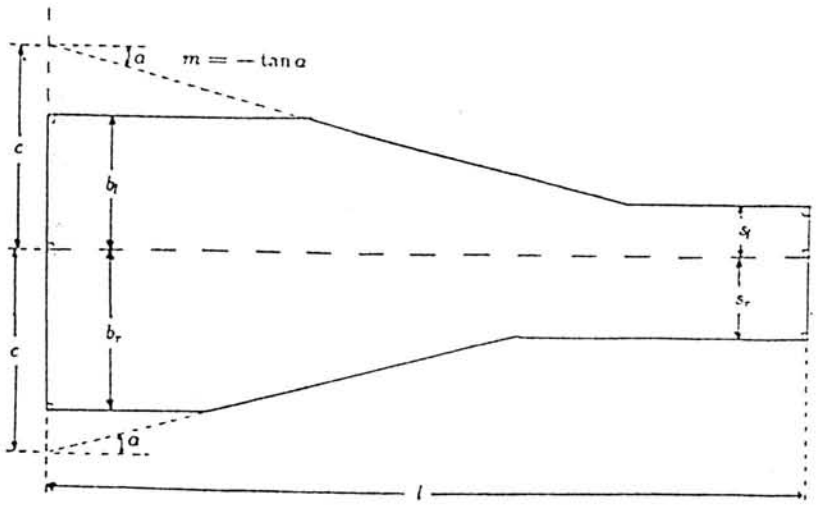


fig. 4.4 Definição do cone generalizado

Em ambos os lados, esquerdo e direito, os raios máximos alcançados ao longo do comprimento do cone ocorrem em t=0, e são denotados b_l e b_r, respectivamente, descrevendo o extremo "grande" do cone. Os raios mínimos ocorrem em t=l e são denotados, s_l e s_r, descrevendo o extremo "pequeno" do cone.

Se $b_l=s_l$ e $b_r=s_r$, então os dois lados do cone são paralelos à espinha. Se não, então existe um estreitamento simétrico do cone (que pode começar e acabar em diferentes valores de t na esquerda e na direita), em que os raios esquerdo e direito das partes em estreitamento do cone são dados pela expressão mt+c, em que m e c são constantes. Acontece sempre $m \leq 0$ e $c \geq 0$.

As sete constantes l, b_l, b_r, s_l, s_r, m e c especificam completamente a forma e tamanho do cone generalizado. Adicionalmente, a sua localização e orientação devem ser determinados ao calcular estes parâmetros.

4.3.2. DIAGRAMAS DE VORONOI

Os diagramas de Voronoi generalizados permitem a obtenção de uma representação bastante refinada do espaço livre.

Um diagrama de Voronoi generalizado (DVG) é a localização de pontos que são equidistantes de duas ou mais fronteiras de obstáculos, incluindo a fronteira do espaço de trabalho [5]. Para um espaço de trabalho poligonal populado com obstáculos poligonais, o DVG consiste de uma rede de segmentos de linha rectos e parabóilicos. Vários algoritmos têm sido propostos para computar um DVG [6] [7] [8].

A utilização dos DVGs tem sido intensa em diversas áreas, mas só recentemente se iniciou a sua utilização para o planeamento de movimento num espaço de trabalho robótico. Em [8] e [9] foram expostas as bases teóricas do método DVG, mostrando que a pesquisa de uma trajectória livre de colisões pode ser restrita ao DVG.

Na figura 4.5 podemos observar um DVG de um ambiente com três obstáculos poligonais.

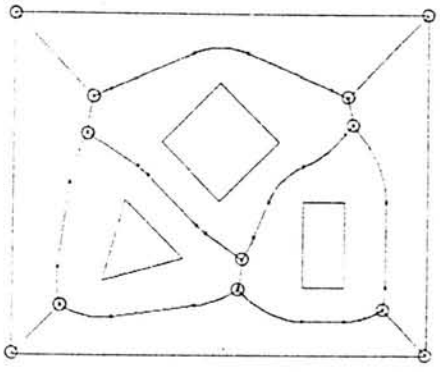


fig. 4.5 Um diagrama de Voronoi generalizado

Os DVGs constituem uma boa representação do espaço livre, mesmo em ambientes bastante populados, permitindo o planeamento de trajectórias bem afastadas dos obstáculos quando possível e a obtenção de trajectórias mais curtas do que os métodos que usam "freeways", devido à presença de arcos parabólicos em torno das esquinas [5].

4.3.3. TRIANGULAÇÃO

A triangulação do espaço livre é usada por diversas abordagens basicamente devido à grande simplicidade da figura e à consequente facilidade no seu processamento. Como tal, apresentamos nesta secção referências sobre alguns dos algoritmos de triangulação expostos na literatura que encontrámos.

Definição 1: Seja P um polígono simples fechado de n vértices, definido por uma lista V_1, V_2, ..., V_n dos seus vértices em ordem inversa ao movimento dos ponteiros do relógio. Seja Π a região de P, união de P com o seu interior.

Definição 2: Os lados de P são os segmentos de recta cujos extremos são os pontos V_i e V_{i+1}, para $1 \leq i \leq n$ e $V_{n+1}=V_1$, denotados $S(V_i, V_{i+1})$.

Definição 3: Uma diagonal de P é um segmento de recta cujos extremos são vértices de P e que ficam inteiramente no interior de P, denotadas $D(V_i, V_j)$, e $|i-j| > 1$.

$D(V_i,V_j) \cap \Pi = D(V_i, V_j)$.

Teorema 1: Qualquer região poligonal pode ser expressa como a união de um número finito de regiões triangulares que se intersectam apenas nos seus lados, se se intersectarem de todo.

Teorema 2: Qualquer região poligonal com n vértices pode ser decomposta num conjunto de n-2 regiões triangulares cujos vértices são os vértices de P.

Dadas as coordenadas planares de n vértices de um polígono simples fechado P ordenados à volta de P, a triangulação de P é encontrar n-3 diagonais de P que não se intersectem e que dividem o interior de P em n-2 regiões triangulares.

O método de triangulação mais conhecido e, porventura, mais utilizado é o método de Delaunnay [10]. Tang e Jarvis apresentam em [11] um método desenvolvido com base na triangulação de Delaunnay, bem como um método desenvolvido pelo próprios autores que apresenta diversas vantagens em relação ao primeiro, nomeadamente a nível de velocidade e aplicabilidade para actualizações em áreas locais.

Algoritmo 1 - Triangular o interior de um polígono sem buracos

. Para um polígono com n lados, se n=3, não é necessária qualquer triangulação. Se $n \geq 3$, calcular os ângulos interiores dos n vértices.

. Entre o conjunto de segmentos de recta criados pela ligação do vértice com o maior ângulo interior com todos os outros n-1 vértices, escolher aquele que não atravesse nenhuma das arestas do polígono e esteja mais perto da bissectriz do maior ângulo interior. Usar este segmento para separar o polígono em duas partes.

. Repetir o algoritmo 1 para estes novos polígonos mais pequenos.

Algoritmo 2 - Triangular o interior de um polígonno com um ou mais buracos poligonais (não necessariamente convexos)

. Se o número de buracos=0, chamar o algoritmo 1.

. Se não, tal como no algoritmo 1, encontrar a linha que mais perto esteja da bissectriz do maior ângulo interior. Se esta linha não intersectar nenhum dos buracos, separar o polígono em dois e repetir o algoritmo 2. Se cortar um ou mais buracos, separar o polígono num quadrilátero e num polígono como na figura 4.6. Repetir o algoritmo 2 para estes dois novos polígonos.

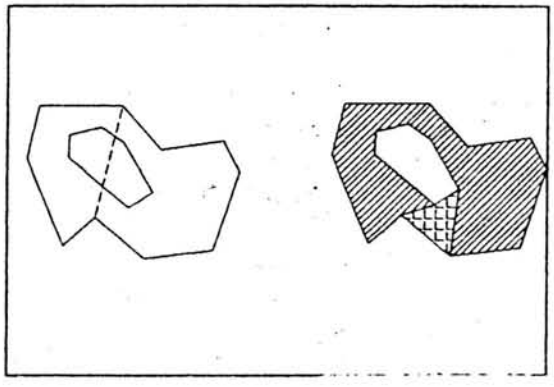


fig. 4.6 Divisão de um polígono com um buraco

Previamente à utilização deste método, os autores usavam uma metodologia baseada na relação de dualidade diagrama de Voronoi-triangulação de Delaunnay:

. Os vértices de todos os obstáculos são usados para gerar um diagrama de Voronoi. De seguida a triangulação de Delaunnay associada é efectuada explorando a relação de dualidade. Os vértices de todos os triângulos gerados são vértices dos obstáculos, mas

nem todas as arestas coincidem com as arestas dos objectos. Esta inconsistência permite que algumas arestas dos triângulos cruzem arestas dos obstáculos, o que não é aceitável para o propósito do algoritmo.

. Todos os triângulos residindo inteiramente dentro de um obstáculo são eliminados, tal como os triângulos cujas arestas cruzam arestas de obstáculos. Assim, no espaço livre existirão algumas áreas que não estão cobertas por quaisquer triângulos.

. Se qualquer destas áreas tiver a forma de um polígono convexo, a triangulação de Delaunnay é novamente executada para essa área.

. Se não for convexa, verificações de visibilidades mútuas entre vértices são feitas para dividir a área em polígonos convexos. Então a triangulação de Delaunnay é repetida para cada um dos polígonos convexos obtidos.

Nas figuras 4.7 e 4.8 mostramos, respectivamente, a triangulação obtida com o método apresentado pelos autores e o método baseado na triangulação de Delaunnay.

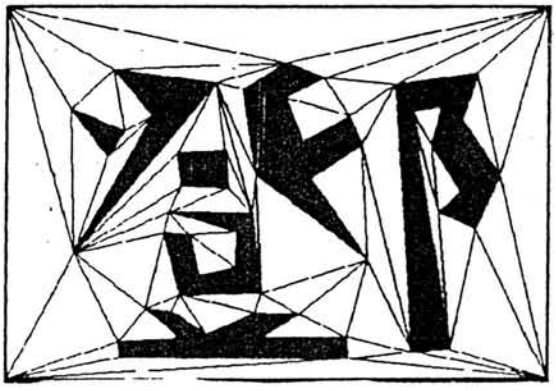


fig. 4.7 Triangulação pelo método de Tang e Jarvis

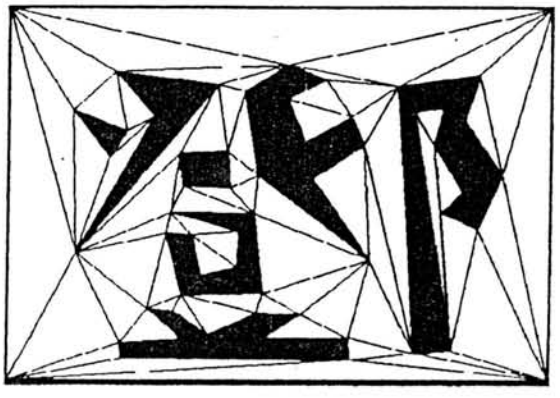


fig. 4.8 Triangulação pelo método de Delaunnay

O método apresentado pelos autores forma alguns triângulos mais estreitos, o que apresenta algumas desvantagens para o planeamento de trajectórias, nomeadamente a produção de trajectórias muito oscilantes e passagens muito perto de obstáculos. No entanto o uso de um algoritmo de pesquisa de uma trajectória óptima pode ser usado para evitar estes triângulos.

Por outro lado, apresenta maior velocidade e, tratando-se de uma abordagem recursiva com informação de entrada que muda em cada iteração, apresenta ainda uma maior aplicabilidade para actualizações locais.

Além das referências já apresentadas, outros algoritmos poderão ser encontrados em [12] e [13].

4.4 MAPEAMENTOS

4.4.1. INTRODUÇÃO

Se em ambientes conhecidos a representação do mundo pode ser fornecida a priori ao robot, o mesmo não sucede em ambiente desconhecidos, em que para possuir um modelo do mundo, o próprio robot terá de o construir. Uma abordagem ideal a essa modelização deverá incluir:

- ◇ um mapeamento multi-resolução da vizinhança do robot;
- ◇ procedimentos para inserir informação de variados sensores, como sonar, visão stereo, sensores de proximidade e outros;
- ◇ procedimentos para actualizar um modelo reflectindo as incertezas resultantes do movimento impreciso do robot e da própria informação sensorial;
- ◇ procedimentos para extrair conclusões a partir dos mapas, como a detecção de corredores, a descoberta de pontos de observação com boas vistas sobre regiões ainda não observadas e a identificação de características maiores, como por exemplo portas e secretárias.

4.4.2. QUADTREES

As quadtrees são estruturas hierárquicas em árvore usadas para representar áreas em Sistemas Gráficos. São geradas pela divisão sucessiva de uma região 2-D em quadrantes.

Cada nó da quadtree tem quatro elementos, um para cada quadrante na região. Se todos os elementos num quadrante tiverem a mesma cor (quadrante homogéneo) o correspondente elemento do nó guarda essa cor.

Além disso, é activada uma flag a indicar que o quadrante é homogéneo. Se o quadrante for heterogéneo, faz-se a sua divisão em quadrantes e o elemento correspondente assinala na flag a heterogeneidade do quadrante e guarda o apontador para o nó seguinte na quadtree.

Para a representação do espaço ocupado e do espaço livre, podemos usar uma versão mais simples desta estrutura, como a apresentada em [14]. Cada nó da árvore corresponde a uma região e é classsificado de acordo com a posição da região correspondente em relação a um conjunto de obstáculos: exterior (nó "branco"), interior (nó "preto") e intersecção (nó "misto"). Os nós mistos, correpondentes a uma região de intersecção têm quatro nós filhos correspondentes às suas quatro subregiões.

A quadtree com obstáculos e regiões livres pode ser obtida a partir de uma rápida conversão de uma imagem real do espaço de trabalho. A imagem seria obtida por uma câmara colocada numa posição do tecto de modo a reduzir ao máximo a distorção geométrica e maximizar a nitidez da imagem.

Num ambiente interior, a cor do chão pode normalmente ser bem distinguida da cor de um obstáculo. Deste modo, a imagem real pode ser processada eficazmente para a imagem binária, cujos pixels são classificados em região livre (chão) ou região de obstáculo, através da utilização de informação de cor.

A imagem binária é rapidamente convertida para a quadtree, fazendo a fusão recursiva de quatro regiões vizinhas com a mesma classificação. Assim, a quadtree pode representar a alocação de regiões livres e ocupadas no espaço de trabalho num bom intervalo de tempo.

Na figura 4.9 apresentamos um espaço de trabalho e a correspondente quadtree.

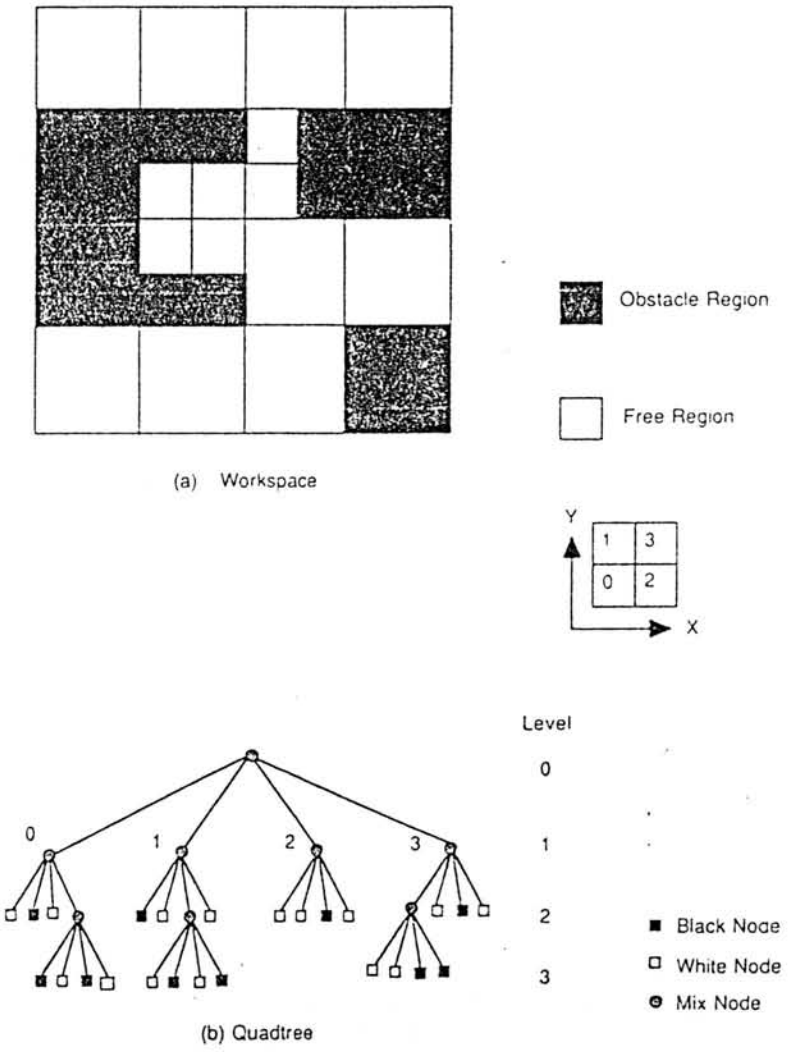


fig. 4.9 (a) Espaço de trabalho e (b) quadtree correspondente

Um esquema de codificação octree divide regiões de espaço 3-D em octantes, guardando oito elementos de informação em cada nó da árvore. Aos elementos individuais de um espaço 3-D chama-se elementos de volume ou voxels. Quando todos os voxels num octante têm a mesma cor, o seu valor é guardado no elemento correspondente do nó. As regiões vazias são representadas por voxels do tipo "void". Qualquer octante heterogéneo é subdividido em octantes e o elemento do nó correspondente aponta para o nó seguinte da octree. Cada nó da octree pode ter 0 a 8 descendentes imediatos.

As representações quadtree e octree apresentam as seguintes vantagens e inconvenientes:

VANTAGENS

◇ representação compacta do espaço ocupado, livre e desconhecido com acesso rápido a esta informação;

◇ grandes partes do ambiente podem ser ignoradas para análise;

◇ a possibilidade de dispor em qualquer altura de uma representação multi-resolução;

INCONVENIENTES

◇ decomposição rígida do espaço com uma origem implícita;

◇ a estrutura em grelha imposta não é conveniente para rotações e conduz frequentemente a perdas na precisão espacial;

◇ a dificuldade de fundir representações locais numa árvore global quando o deslocamento relativo das origens locais é desconhecido.

Leon Piotrowski [15] implementou um sistema de represenatação que usa codificação quadtree da informação, obtida por um sensor de alcance do tipo "pulsed-laser". Na figura 4.10 podemos observar o resultado de um scan típico do ambiente do robot.

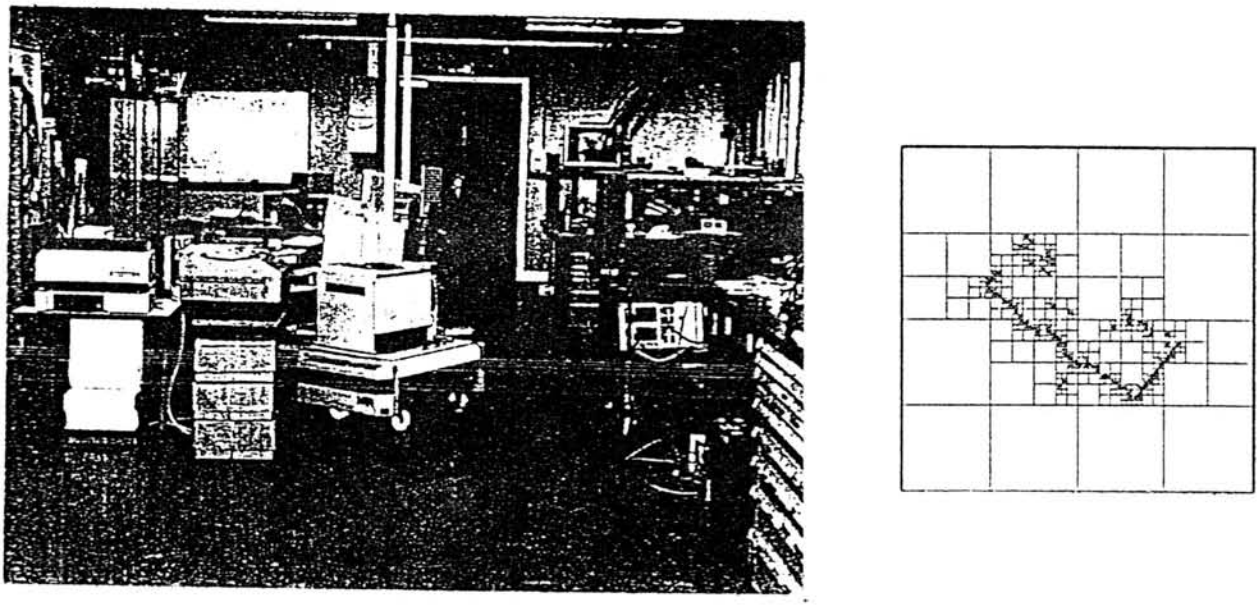


fig. 4.10 Ambiente do robot (à esquerda) e resultado de um scan típico (à direita)

Trata-se de um scan de 360 graus com medidas de distância tiradas em intervalos de 1 grau. Esta representação tem 9 níveis de resolução espacial, o que implica uma precisão máxima de 256 x 256 elementos. Cada medida de distância é guardada na folha correspondente na altura da aquisição. O espaço é representado como uma grelha de células, em que cada uma mapeará uma determinada área no caso de uma representação 2-D ou volume numa representação 3-D.

4.4.3. GRELHAS DE OCUPAÇÃO ((IN)CERTEZA)

O espaço é representado como uma grelha de células, em que cada uma mapeará uma determinada área no caso de uma representação 2-D ou volume numa representação 3-D.

Em algumas implementações, cada célula é caracterizada por uma única estimativa da probabilidade de estar ocupada. Em outras implementações essa caracterização é feita por dois parâmetros: a probabilidade estimada de estar vazia e a probabilidade estimada de estar ocupada. As células cujo estado seja desconhecido têm ambas as probabilidades nulas. Se ambas as probabilidades forem altas, estamos perante dados inconsistentes.

Esta representação permite a fácil implementação de procedimentos de actualização incremental a partir de várias fontes sensoriais. Para obter uma resolução adequada da área ou volume vizinho e cobertura suficiente para planeamento de longo alcance, poderá manter-se uma hierarquia de mapas que cubram as áreas ou volumes pretendidos.

Do trabalho de Moravec [16], podemos observar na figura 4.11 a construção de uma grelha de certeza por um robot guiado por sonar que atravessa o laboratório. Os eixos estão em pés. Cada ponto da trajectória é um paragem que permitiu ao sonar fazer novas leituras. As células estão em branco se a probabilidade de ocupação for baixa, têm pontos (•) se essa probabilidade for desconhecida e um x se for alta. Podemos observar ainda na mesma figura as trajectórias que foram sendo planeadas à medida que a grelha ia sendo gerada incrementalmente.

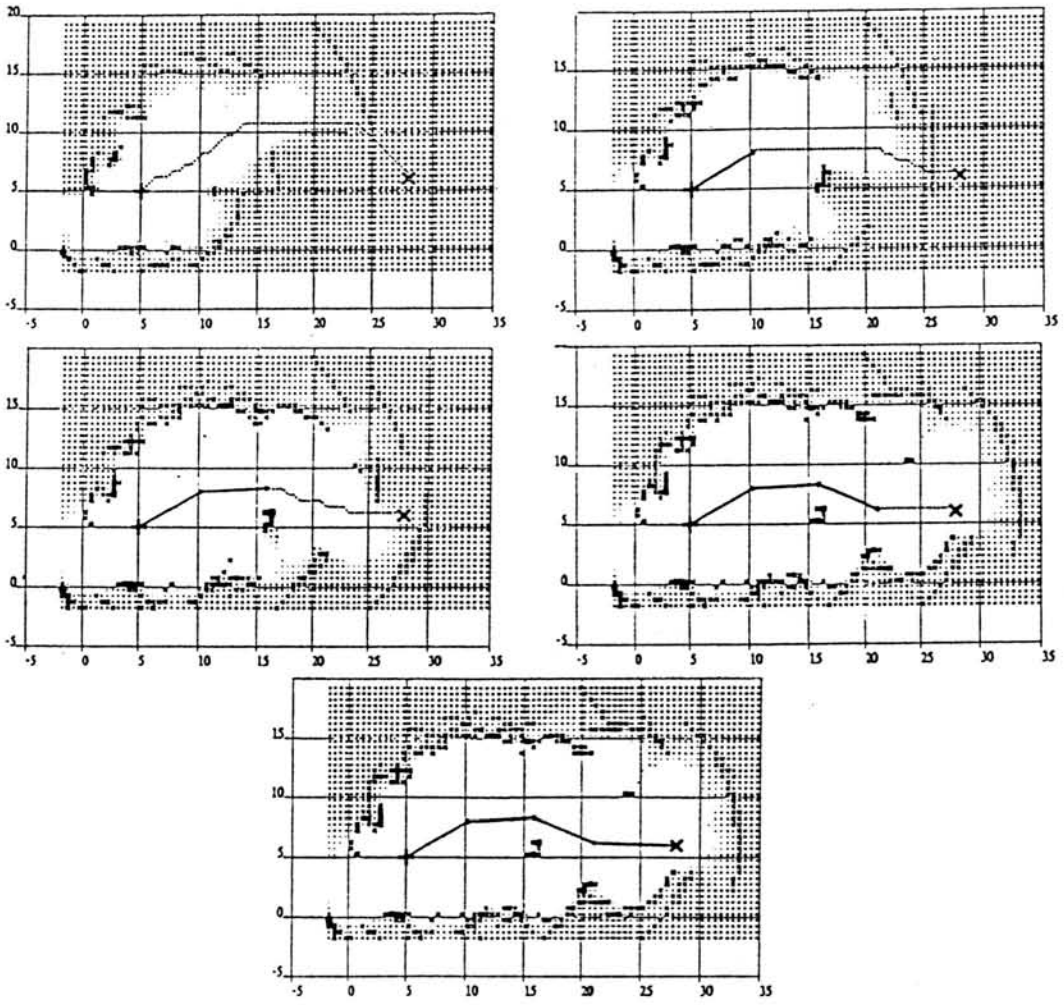


fig. 4.11 Mapeamento por sonar e navegação

4.4.4. MAPEAMENTO DE CENAS DESCONHECIDAS

Numa ambiente basicamente inalterável, um mapa apropriado pode ser preparado previamente por um operador humano e dado ao robot. Mas, a maioria das cenas industriais e comerciais em que um robot tem que operar, mudam com demasiada frequência para que a preparação de mapas por humanos seja prática. Uma alternativa atraente para estes casos é deixar o próprio robot explorar e fazer um mapa da cena.

Equipado com sensores e algoritmos, o robot pode periodicamente explorar e actualizar o mapa de uma cena com pouca ou nenhuma assistência humana. Outra vantagem de um mapa

gerado pelo robot é que uma característica da cena representada no mapa será facilmente identificável pois o mesmo conjunto de sensores efectua quer a extracção das características (para o mapa) quer a sua identificação.

Sankaranarayanan e Masuda [17] apresentam um algoritmo geral para o problema da aquisição de terreno, problema este que pode ser formulado do seguinte modo: requer-se a obtenção de uma descrição completa das coordenadas das fronteiras de todos os objectos numa cena planar esconhecida. Um ponto móvel equipado com sensores apropriados (visão, tácteis, etc.) é usado para explorar a cena. Através dos seus sensores o robot pode detectar e seguir a fronteira de um objecto, podendo os objectos assumir qualquer forma e tamanho. O robot, conhecendo as coordenadas exactas da sua posição num determinado referencial, pode armazenar um grande número de pontos das fronteiras dos objectos.

Estes autores trabalharam a partir do trabalho de Lumelsky [18] sobre este mesmo problema. O desenvolvimento que efectuaram fez sobressair os dois principais aspectos do problema: a selecção no grafo e a pesquisa no grafo. Várias regras para o movimento entre objectos formam a parte do algoritmo que lida com a selecção no grafo. A parte de pesquisa é formada por um procedimento de pesquisa óptimo, combinado com heurísticas.

Para a parte de selecção no grafo, os autores propõem uma regra geral de partida de um objecto, com um parâmetro α real e positivo. O comportamento do algoritmo pode ser controlado de cauteloso a aventureiro, variando o parâmetro α. Como compromisso entre os dois comportamentos extremos, é proposta a utilização de um α igual ao raio de visibilidade R_V.

Outras referências anteriores que lidam com a exploração de um ambiente desconhecido de objectos poligonais são os trabalhos de Brooks [19], Turchen e Wong [20] e Rao *et al.* [21].

BIBLIOGRAFIA

[1] T. Lozano-Pérez
"Automatic planning of manipulator transfer movements"
IEEE Trans. Syst., Man, Cybern., vol. SMC-11, pp. 681-698, Oct. 1981

[2] T. Lozano-Pérez
"Spatial planning: a configuration space approach"
IEEE Trans. Comput., vol. C-32, pp. 108-120, Feb. 1983

[3] F. Wallner, T. C. Lueth, F. Langinieux
"Fast local path planning for a mobile robot"
ICARV '92, vol. 3, pp. RO 10.1.1-10.5.5

[4] R. A. Brooks
"Solving the find-path problem by good representation of free space"
IEEE Trans. Syst., Man, Cybern.,
vol. SMC-13, no. 3, Mar/Apr 1983, pp. 190-197

[5] O. Takahashi, R. J. Schilling
"Motion planning in a plane using generalized Voronoi diagrams"
IEEE Transactions on Robotics and Automation,
vol. 5, no.2, Apr. 1989, pp. 143-150

[6] Fortune
Proc. 2nd ACM Symp. on Computational Geometry
1986

[7] D. G. Kirkpatrick
"Efficient computation of continuous skeletons"
Proc. 20th IEEE Annu. Symp. on Foundations of Computer Science,
pp. 135-145, Oct. 1979

[8] C. O'Dunlaing, M. Sharir, C. K. Yap
"Retraction: a new approach to motion planning"
Proc. 15th ACM Symp. on Theory of Computing, pp. 207-220, 1983

[9] C. O'Dunlaing, C. K. Yap
"A retraction method for planning the motion of a disc"
J. Algorithms, vol. 6, pp. 104-111, 1985

[10] F. Preparata, M. Shamos
Computational Geometry: an Introduction
Springer-Verlag, 1985

[11] K. W. Tang, R. Jarvis
"Collision-free path finding amongst polygonal obstacles using efficient free space triangulation"
ICARV '92, vol. 3, pp. RO 11.1.1-11.1.5

[12] R. E. Tarjan, C. J. Van Wyk
"An O(nloglogn)-time algorithm for triangulating a simple polygon"
SIAM Journal of Computing, vol. 17, no. 1, Feb. 1988

[13] S. W. Sloan, G. T. Houlsby
"An implementation of Watson's algorithm for computing 2-dimensional Delaunnay triangulations"
Adv Eng. Software, vol. 6, no. 4, pp. 192-197
1984

[14] H. Noborio, T. Naniwa, S. Arimoto
"A quadtree-based path-planning algorithm for a mobile robot"
Journal of Robotic Systems, vol. 7, no.4, pp. 55-574
1990

[15] L. Pietrowski
"Environment representation by a mobile robot using quadtree encoding of range data"
S. G. Tzafestas (ed.), Robotic Systems, pp. 375-382, 1992

[16] H. P. Moravec
"Certainty grids for mobile robots"
Procedings of the Workshop in Space Telerobotics, vol. 1, pp. 307-312

[17] A. Sankaranarayann, I. Masuda
"Sensor based terrain acquisition: a new, hierarchical algorithm and a basic theory"
IROS '92, vol. 3, pp. 1515-1523

[18] V. J. Lumelsky, S. Mukhopadhyay, K. Sun
"Dynamic path planning in sensor-based terrain acquisition"
IEEE Transactions on Robotics and Automation, vol. 6, no. 4, Aug. 1990

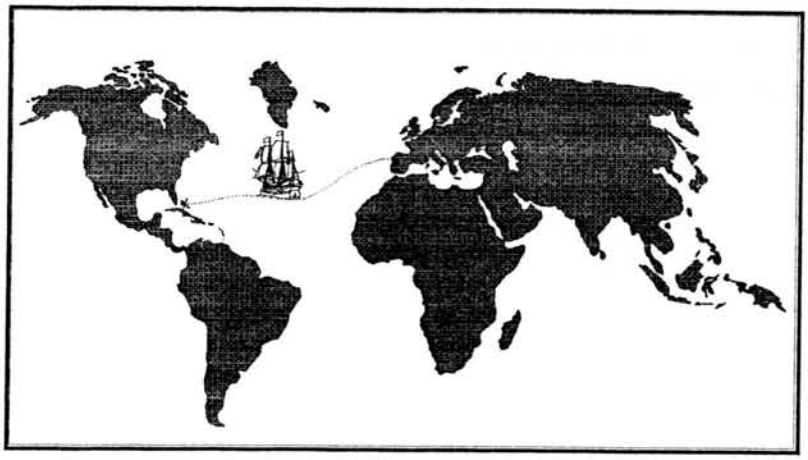
[19] R. A. Brooks
"Visual map making for a mobile robot"
Proc. IEEE Int. Conf. on Robotics and Automation, 1985

[20] M. Turchen, A. Wong
"Low level learning for a mobile robot: environmental model acquisition"
Proc. 2nd Int. Conf. AI and its Applications, 1985

[21] N. Rao, S. Iyengar, C. Jorgensen, C. Weisbin
"On terrain acquisition by a finite-sized mobile robot in a plane"
Proc. IEEE Int. Conf. on Robotics and Automation, 1987

CAPÍTULO 5

NAVEGAÇÃO



5.1 INTRODUÇÃO

Os sistemas de navegação para robots móveis são a área de investigação em robótica móvel que mais trabalho tem produzido. Entre os factores que influenciam esse grau de produção, o mais importante será, evidentemente, o facto de a navegação ser o principal objectivo de um robot móvel.

Os robots móveis são construidos para operar num tipo pré-determinado de ambientes (armazéns, fábricas, exteriores em exploração planetária, etc.), sendo tal imposto, em parte, pelas próprias limitações físicas do robot. As características desses ambientes e as características das tarefas que o robot desempenhará determinam, em conjunto com as tecnologias e metodologias disponíveis, a definição e concepção da arquitectura do robot.

A concepção de sistemas de navegação é encorajada pela possibilidade de teste em simulações corridas em computador. Se a simulação tiver bons resultados, então o algoritmo poderá ser implementado num robot real. De qualquer modo, as simulações são efectuadas considerando desde logo uma determinada plataforma e as suas especificidades a nível dos sistemas de representação do mundo, "sensing" e do próprio hardware motor do robot.

Para a navegação de um robot móvel são necessários três elementos:

. Processamento de informação sensorial

. Planeamento de trajectória

. Controlo de trajectória

Os sensores fornecem informação sobre o estado interno do robot (tacómetros para a a velocidade e codificadores para os ângulos) e sobre o ambiente que envolve o robot (sistemas de visão, dispositivos de medida de distâncias e mesmo detectores de colisões).

Essa informação é usada para planear ou replanear uma trajectória que, finalmente, será executada pelo controlo.

Os objectivos que nos propusemos com este trabalho levam-nos a excluir deste capítulo o estudo dos mecanismos de actuação do robot, ou seja, a sua estrutura mecânica. Concentramo-nos nas funções de planeamento e controlo do movimento e na forma como estas funções utilizam a informação disponível, quer a interna quer a externa ao robot.

No entanto as particularidades mecânicas do robot estão presentes na medida em que influenciam as funções de planeamento e controlo, impondo restrições ao tipo de movimentos que pode ser executado e sendo fonte de incertezas que devem ser consideradas.

Na literatura sobre robótica móvel surgem diversas abordagens aos sistemas navegacionais, resultantes de diferentes pespectivas sobre alguns dos factores que os condicionam:

. a filosofia que preside à arquitectura do robot,

. o modo como a informação sobre o estado interno do robot e o seu ambiente é obtida e representada,

. as tarefas que o robot deve desempenhar,

. o meio em que o robot operará,

. o tipo de planeamento utilizado.

Numa tentativa de categorização de mais alto nível, estas abordagens podem ser agrupadas de acordo com as características do ambiente de trabalho do robot.

AMBIENTES CONHECIDOS

O problema de encontrar uma trajectória tem duas abordagens principais: os métodos de pesquisa em grafos e os métodos de campo potencial.

Na abordagem da pesquisa em grafos, é criado um gráfico ou um grafo que mostra o espaço livre e o espaço proibido no ambiente do robot. A trajectória é gerada juntando os espaços livres ou contornando as zonas proibidas.

Entre estas abordagens situam-se o Espaço de Configuração [1] [2] [3], os Cones Generalizados [4], a Triangulação do Espaço Livre [5] [6] [7], o Método do Grafo de Vértices [8], os Diagramas de Voronoi [9], Formas Convexas Intersectantes [10], Representação Quadtree [11] [12] e Mundo de Rectângulos [13].

A abordagem de campo potencial usa uma função escalar para descrever quer os objectos, quer o espaço livre. Mais especificamente, o gradiente negativo do campo potencial é precisamente a direcção em que se deve mover o robot de modo a evitar os obstáculos.

São diversos os trabalhos de investigadores que utilizam esta abordagem [14] [15] [16]. Mais à frente neste capítulo, exploraremos dentro desta abordagem os trabalhos de Hwang e Ahuja [17] e Ikazami e Ozono [18]. Recentemente, surgiram trabalhos que permitem ao robot lidar com objectos cuja forma, velocidade e direcção são assumidos a priori [19] [20] [21]. A principal vantagem da abordagem de campo potencial é oferecer uma maneira relativamente rápida e eficiente de encontrar caminhos seguros em torno dos obstáculos.

Ainda dentro dos ambientes conhecidos debruçar-nos-emos sobre uma outra abordagem, baseada numa analogia electromagnética [22].

AMBIENTES CONHECIDOS COM OBSTÁCULOS INESPERADOS

Para ambientes conhecidos, mas com obstáculos inesperados, as abordagens caracterizam-se normalmente por:

1. planeamento de uma trajectória no ambiente conhecido, à semelhança das abordagens anteriores,
2. lidar com obstáculos inesperados, produzindo uma alteração na trajectória do robot que lhe permita evitar o obstáculo (contornando o obstáculo ou tomando nova direcção),
3. após alteração na trajectória, replanear a trajectória até ao objectivo ou regressar à trajectória inicial.

Em [23] é exposto um algoritmo que se enquadra exactamente neste tipo de abordagem, com uma clara distinção entre o planeamento da trajectória e o evitar local de

obstáculos. R. C. Arkin apresenta em [24] um planeador hierárquico em que é também posta em prática esta abordagem.

Uma outra abordagem [25] parte de algum conhecimento a priori sobre o ambiente de trabalho do robot para construir um Mapa Topológico, mapa este que é utilizado para o planeamento de trajectória. Esta abordagem lida com o aparecimento de obstáculos inesperados em posições aleatórias através do uso de um modelo estatístico que se baseia em distribuições pressupostas previamente e é actualizado dinamicamente através de informação sensorial.

AMBIENTES DESCONHECIDOS OU DINÂMICOS

O planeamento global de uma trajectória para um robot móvel num ambiente desconhecido coloca-se como um problema de difícil resolução. A principal dificuldade consiste em modelar um ambiente possivelmente dinâmico e incerto de um modo que permita encontrar uma solução possível.

Para resolver este problema, a maioria das abordagens adoptam sensores "on-board" activos, responsáveis pela aquisição de informação sobre a vizinhança do robot. Os modelos do ambiente são então construidos dinamicamente para permitir determinar uma trajectória à medida que o robot se move.

Entre os métodos propostos temos a Grelha de Ocupação [26] [27] [28] e o Grafo de Estado [29]. Os modelos probabilísticos são construidos com base nas células de um grelha e actualizados dinamicamente com informação sensorial. A trajectória é encontrada através da minimização da probabilidade de encontrar obstáculos.

Estes métodos envolvem, contudo, um grande esforço computacional na manutenção das grelhas para mapeamento do ambiente, obrigando o robot a lidar com enormes quantidades de informação.

A utilização de métodos rápidos permite operar em ambientes em que são necessárias reacções rápidas para evitar obstáculos. Na abordagem que apresentaremos [30], a utilização de um Espaço de Configuração Local, juntamente com alguns conceitos do campo de distâncias e um modelo de grelha permite obter um método computacionalmente muito eficiente, com procedimentos de mapeamento e planeamento de trajectória simples, sem recorrer a equações matemáticas complexas. Adicionalmente, a taxa de repetição do algoritmo pode ser relativamente baixa.

NAVEGAÇÃO REFLEXIVA

Num lugar aparte nesta classificação, colocamos uma arquitectura desenvolvida por R. C. Arkin, influenciada por considerações cibernéticas [31], cuja metodologia navegacional, a navegação reflexiva, se baseia em *esquemas*.

Um *esquema* motor é definido como a unidade básica do comportamento motor a partir da qual se podem construir acções complexas. Consiste quer do conhecimento de como agir quer do processo computacional pelo qual é estabelecido, permitindo uma abordagem simples da navegação de baixo nível.

Esta arquitectura, AuRA (Autonomous Robot Architecture), será descrita parcialmente neste capítulo (a parte respeitante à navegação), sendo essa descrição completada no capítulo sobre arquitecturas.

Na próxima secção descreveremos de uma forma geral o problema do planeamento de trajectórias. Nas secções seguintes exploraremos com maior pormenor as abordagens que referimos nesta introdução.

5.2 O PLANEAMENTO DE TRAJECTÓRIA

A versão básica do problema do planeamento de trajectória para um robot pode ser descrita da seguinte forma:

A um objecto, definido como o robot, é dada uma localização inicial S, uma localização objectivo G e um conjunto de objectos chamados obstáculos dentro do ambiente de trabalho.

Planear uma trajectória solução entre S e G, evitando os obstáculos e de modo a que o robot se possa mover ao longo dela.

Os algoritmos desenvolvidos para resolver este problema dividem-no, essencialmente, em três tarefas e lidam com elas sequencialmente ou em paralelo, de acordo com as arquitecturas computacionais dos algoritmos:

1. Modelização e tradução

O ambiente físico de trabalho do robot e os obstáculos desse ambiente são modelizados e traduzidos como um modelo matemático, ou seja, um grupo de dados numéricos num formato apropriado, que seja acessível ao sistema computacional.

Na introdução apontámos diferentes abordagens a esta tarefa. Por exemplo, os métodos de pesquisa em grafos modelizam o ambiente como um grafo que contem nós e arestas, os métodos de campo potencial tratam a totalidade do sistema como um campo com gradientes de colisão, etc..

2. Pesquisa e planeamento

É prucurada e planeada de modo numérico uma trajectória solução, ou um conjunto contínuo de informação que ligue os elementos de informação correspondentes a S e G. Como exemplo, para a pesquisa em grafos há dois métodos bastante populares: o método de Dijkstra [32] e o método A* [33].

3. Interpretação

A trajectória solução é então interpretada como uma sequência de operações do robot, sequência essa que constitui o seu comportamento ao longo dessa trajectória.

Existe ainda um conjunto de factores que influenciam de forma diferente os diversos algoritmos:

1. Espaço livre

Referimo-nos à distância entre o robot e a fronteira do espaço livre. Por exemplo, nos nós do grafo de visibilidades (Espaço de Configuração) essa distância é mínima, enquanto nos diagramas de Voronoi é máxima.

2. Convergência ou correcção

Consiste em garantir uma trajectória solução, desde que exista uma no ambiente. Alguns algoritmos podem deixar de procurar uma possível trajectória solução devido a uma restrição na resolução.

3. Optimização

Normalmente, os algoritmos tentam optimizar uma determinada função de custo, como por exemplo o caminho Euclideano mais curto, a trajectória com menor tempo de operação ou a trajectória com menor número de viragens.

4. Eficiência

A eficiência dos algoritmos tem a ver com a complexidade computacional, o tempo esperado de corrida e a quantidade de informação que é necessariamente armazenada pelos algoritmos.

5.3 AMBIENTES CONHECIDOS

5.3.1. MÉTODOS BASEADOS EM PESQUISA EM GRAFOS

5.3.1.1. ESPAÇO DE CONFIGURAÇÃO

A primeira abordagem para resolver o problema do planeamento de trajectórias que vamos explorar consiste essencialmente na transformação do movimento de um objecto através de um espaço de obstáculos num problema equivalente, mas mais simples, de planear o movimento de um ponto através de um espaço de obstáculos alargados.

Este espaço construido artificialmente é conhecido como Espaço de Configuração ou C-space. Foi introduzido por Lozano-Pérez [1] e permite um enquadramento eficaz para investigar uma variedade de problemas de planeamento de movimento de robots.

Esta técnica é muito eficaz quando aplicada a um movimento puramente translacional num plano em que a trajectória óptima pode ser encontrada. Se os obstáculos forem modelizados como colecções de poliedros convexos, as restrições de posição podem ser formuladas em termos da posição dos vértices do objecto móvel em relação aos planos das superfícies dos obstáculos.

Embora a estejamos a discutir para uma representação 2-D, esta técnica pode facilmente ser generalizada para lidar com obstáculos 3-D.

Consideremos o problema de mover um objecto pontual da posição de partida P para o posição objectivo O evitando os obstáculos (figura 5.1).

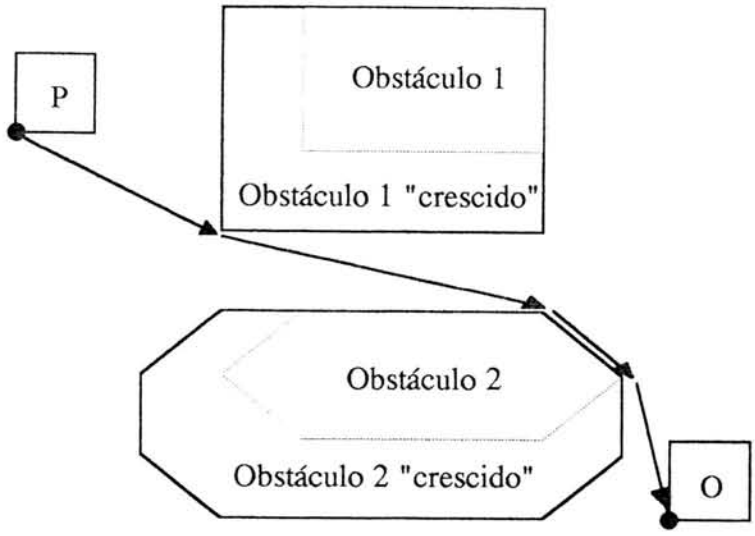


fig. 5.1 Construção dos Obstáculos do Espaço de Configuração (2-D)

A principal propriedade desta trajectória é que é composta de linhas rectas ligando os pontos P e O via uma sequência de vértices de obstáculos. No caso do movimento planar com objectos poligonais arbitrários, a trajectória mais curta livre de colisões que liga dois pontos acessíveis tem sempre esta propriedade.

Constrói-se um grafo não direccional, denotado GV(N,L), em que os nós são todos os vértices dos obstáculos. As ligações existem entre pares de vértices sempre que uma linha recta que ligue os dois vértices não se sobreponha a qualquer obstáculo. GV(N,L) é chamado o grafo de visibilidades pois os vértices ligados no grafo podem ver-se uns aos outros.

A trajectória mais curta livre de colisões entre P e O no plano é o caminho mais curto (com menores custos/distâncias) no GV(N,L) a partir do nó correspondente a P até ao nó correspondente a O. Este método de encontrar trajectórias livres de colisões encontrando o caminho mais curto num grafo de visibilidades é chamado algoritmo VGRAPH. A simplicidade do algoritmo VGRAPH deriva do facto de o objecto móvel ser um ponto.

Davis e Camacho, seguindo de muito perto estas considerações, apresentam em [3] uma implementação em PROLOG para a geração de trajectórias no caso de um objecto poliédrico que se move entre objectos poliédricos.

O algoritmo transforma os obstáculos de modo a estes representarem as localizações das regiões proibidas para um ponto de referência arbitrário do objecto móvel. Uma

trajectória deste ponto de referência que evite todas as regiões proibidas é uma trajectória livre de colisões.

As trajectórias são procuradas num grafo de visibilidades, que indica, para cada vértice (nó) dos obstáculos transformados, que outros vértices podem ser alcançados com segurança. Em 3-D as visibilidades são determinadas através da verificação de interferência entre objectos, primeiro em 2-D e depois em 3-D. A visibilidade em 2-D implica visibilidade em 3-D. O facto de não haver visibilidade em 2-D não implica que não haja visibilidade em 3-D.

Para encontrar uma trajectória é utilizado o algoritmo A*, usando como informação heurística a distância aritmética entre dois pontos.

Também baseada no Espaço de Configuração é uma outra abordagem de Brooks e Lozano-Pérez [2], que permite encontrar um caminho livre sem colisões para um objecto poligonal rígido que se move por entre um espaço preenchido por obstáculos poligonais. O caminho é determinado desde uma posição e orientação iniciais até uma posição e orientação finais.

Ao objecto móvel é associado um eixo local. Deste modo a localização do objecto pode ser totalmente definida a partir da posição (x,y) desse eixo, conhecida como ponto de referência, e de um ângulo indicando a rotação desse eixo relativamente ao eixo global. O objecto móvel é reduzido a um ponto e os obstáculos são expandidos, de acordo com os princípios do Espaço de Configuração.

O Espaço de Configuração é dividido em rectangulóides com arestas paralelas aos eixos. Esse rectangulóides são classificados do seguinte modo:

- vazios, se o seu interior não intersectar nenhum obstáculo;

- cheios, se todos os seus pontos pertencerem a um obstáculo;

- mistos, se houver pontos dentro de obstáculos e outros fora.

Procurar-se-á então encontrar um caminho livre entre o ponto inicial e o ponto final utilizando apenas as células vazias. Caso tal não seja possível, ter-se-á de determinar esse caminho incluindo também as células mistas. Estas células são então cortadas paralelamente a um dos eixos e cada divisão é classificada. De seguida tentar-se-á novamente encontrar um caminho que apenas contenha células vazias. O algoritmo continua iterativamente até se obter sucesso ou se determinar que não há solução possível.

Os objectos pertencentes ao espaço de trabalho são definidos por inequações relativamente complexas, pois estas incluem o facto de o objecto móvel poder rodar no espaço. Deste modo, a configuração dos obstáculos depende da orientação do robot.

Dado que o algoritmo se aplica apenas a objectos convexos, os objectos móveis não convexos devem ser decompostos na união de vários objectos convexos que partilhem o mesmo ponto de referência. Deste modo mantém-se a interligação do objecto inicial. O caminho é, igualmente, calculado para o ponto de referência através do Espaço de Configuração.

O espaço é dividido em células. De modo a optimizar esta divisão utilizam-se rectângulos de tamanhos diferentes, procurando assim que as células vazias tenham o maior tamanho possível. Para tal são usados os valores limite da configuração dos obstáculos. Divide-se, assim, primeiro em x e depois para cada coluna produzida pelos valores de x, em y (figuras 5.2 e 5.3).

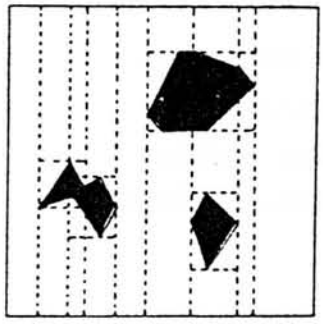


fig. 5.2 Espaço de configuração sem rotação com obstáculos dividido em células rectangulares

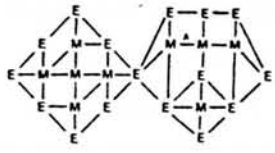


fig. 5.3 Grafo de conectividade para as células da figura 5.2. A ligação com "A" não é na realidade incluida no grafo pois a sua travessia é impossível.

O algoritmo compara cada célula com o Espaço de Configuração. Como os obstáculos são definidos através de inequações, basta ir comparando toda a superfície da célula. Se todas as inequações são verificadas então a célula é classificada como cheia; se nenhuma se verificar então é classificada como vazia; senão é classificada como mista. A cada célula é associado um intervalo de valores dentro do qual pode variar a orientação do robot, dependendo desse intervalo de valores a rotação a impôr ao robot.

A representação do espaço é feita a partir de um grafo de células vazias e mistas. Os nós representam as células, enquanto os arcos representam as possíveis ligações entre estas.

Na construção do grafo não se deve ligar todas as células vizinhas. Apenas devem ser representadas as ligações realmente possíveis. Se o robot não se puder deslocar da célula C1 até à célula vizinha C2, então esta ligação deve ser retirada do grafo. Deste modo consegue-se simplificar o grafo e aumentar a velocidade de processamento.

A divisão de uma célula mista deve igualmente ser feita com cuidado, de modo a tirar o melhor partido desta estratégia. Assim, devemos cortá-la por uma das restrições, o que nos dá mais garantias de arranjar uma célula vazia e outra mista ou cheia.

Este algoritmo é um pouco complexo, podendo tornar-se lento à medida que a dificuldade dos problemas aumenta. As resoluções conseguidas variam entre as dezenas de segundos e as dezenas de minutos. O algoritmo poderá beneficiar de optimizações relacionadas com a diminuição do número de células e com o uso de objectos mais simples.

5.3.1.2. CONES GENERALIZADOS

O algoritmo apresentado por Brooks [4] baseia-se na utilização de representações do espaço que captem os efeitos essenciais da translação e rotação de um corpo através do espaço. O espaço livre é representado como a sobreposição de cones generalizados, visto estes serem descrições de volumes varridos.

Brooks considera uma especialização 2-D dos cones generalizados, em que as espinhas são linhas rectas e as secções tranversais serão segmentos de recta perpendiculares à espinha, com raio esquerdo e raio direito. A regra de varrimento controla independentemente as amplitudes dos raios esquerdo e direito. A figura 5.4 ilustra alguns dos cones generalizados que descrevem o espaço livre à volta de três obstáculos num ambiente de trabalho limitado.

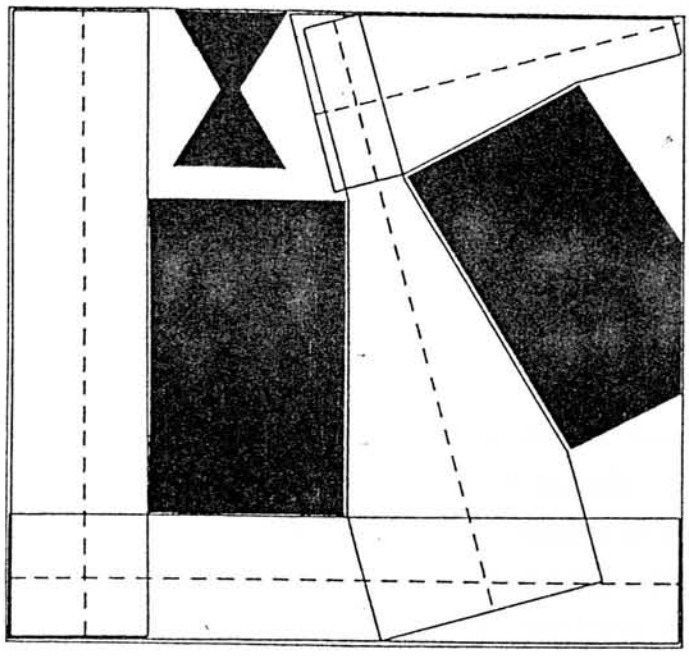


fig. 5.4 Cones generalizados gerados por dois obstáculos e fronteira do espaço de trabalho

O modo como se obtém esta representação está exposto no capítulo dedicado às representações espaciais.

O algoritmo para encontrar uma trajectória livre de colisões procede do seguinte modo:

1. Computação dos cones generalizados que descrevem o espaço livre.

2. Os cones são examinados par a par, para determinar se e onde se intersectam as suas espinhas.

3. A cada ponto de intersecção das espinhas é associado um subconjunto de $[0, 2\pi]$ que descreve as orientações para o objecto móvel para as quais se garante que o objecto está inteiramente contido no cone generalizado. O método que permite obter este intervalo encontra-se exposto detalhadamente em [4].

4. Finalmente, determina-se uma trajectória entre as posições inicial e final, seguindo as espinhas dos cones generalizados e mudando de cone para cone nos pontos de intersecção das espinhas. Se o objecto se mantiver com uma orientação válida em cada ponto da espinha, então a trajectória será livre de colisões.

O grafo construido tem como nós os pontos das espinhas dos cones generalizados que correspondem a pontos de intersecção, juntamente com um sub-intervalo de $[0, 2\pi]$ de orientações.

Os arcos do grafo surgem de dois modos:

. aqueles que correspondem à transferência de um ponto de intersecção para outro na mesma espinha e

. aqueles que correspondem à transferência de uma espinha para outra no ponto de intersecção comum.

A conectividade de todos os pares de nós candidatos é verificada. É suficiente verificar, para os arcos intracones, se os intervalos de orientação dos dois nós têm uma intersecção não vazia. Tal garante que é possível mover o objecto de um ponto para o outro usando qualquer orientação dentro do intervalo de intersecção. Para os pares de nós intercones é também suficiente verificar uma intersecção não vazia dos seus intervalos de orientações, visto que os pontos coincidem no espaço.

O grafo é pesquisado com o algoritmo A*, usando como função de custo a distância viajada através do espaço.

Uma pesquisa com sucesso do grafo tem como resultado toda uma classe de trajectórias livres de colisões, classe essa que partilha uma trajectória comum para a origem do objecto móvel. O intervalo de orientações associado a cada nó restringe as orientações válidas nesse ponto, sendo também necessário assegurar que não é assumida nenhuma orientação inválida durante a translação e rotação ao longo da espinha de um cone generalizado.

É bastante provável que as rotações tenham custos elevados, pelo que valerá a pena pesquisar a classe de trajectórias resultante da pesquisa no grafo para determinar a trajectória óptima em termos de mínima rotação necessária. Tal poderia ser efectuado usando novamente o algoritmo A*.

Para cada par de nós adjacentes na trajectória inicial obtém-se a intersecção dos seus intervalos de orientações, ou seja um conjunto de orientações possíveis. Cada orientação válida para um determinado nó origina, juntamente com esse nó, um novo nó no grafo. A adjacência é herdada da adjacência dos nós originais e o algoritmo A* é usado para pesquisar o novo grafo.

Brooks apresenta um método mais simples que consiste em restringir as rotações aos nós do grafo, ou seja, o objecto mantém uma orientação fixa durante cada translação. A orientação em cada ponto é o ponto médio das orientações dadas pela intersecção dos intervalos de orientações permitidas do nó actual e do nó subsequente. Deste modo, as travessias dos arcos intracones são isentas de rotações e as travessias dos arcos intercones são puras rotações. Podemos observar dois resultados da aplicação deste método mais simples nas figuras 5.5 e 5.6.

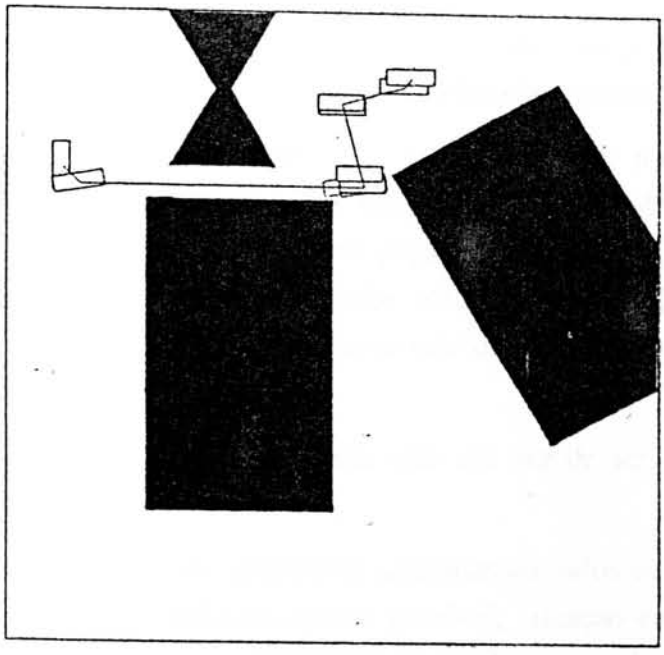


fig. 5.5 Trajectória encontrada pelo algoritmo

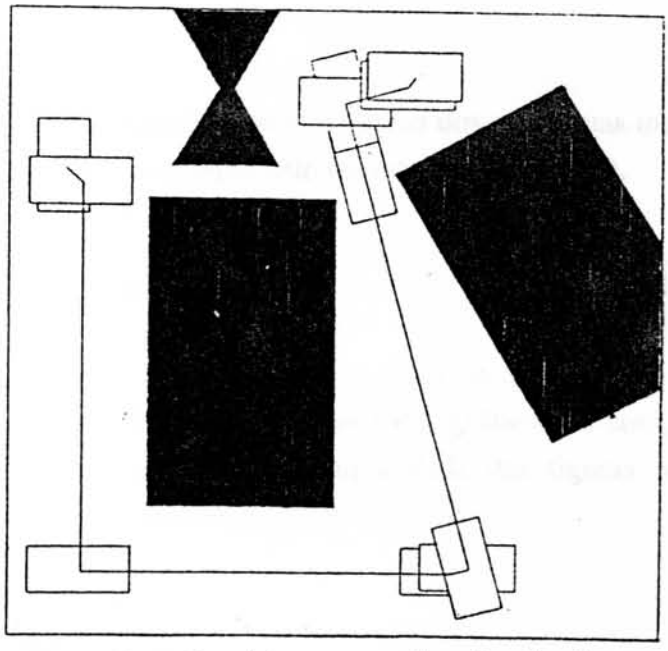


fig. 5.6 Trajectória encontrada pelo algoritmo

A principal desvantagem do algoritmo apresentado é o facto de as trajectórias serem restritas a seguirem as espinhas dos cones generalizados escolhidos para representar o espaço livre. Tipicamente, tal não funciona bem em ambientes com pesadas restrições de espaço por haverem cones generalizados insuficientes para fornecer uma escolha adequada de trajectórias.

No entanto em ambientes mais libertos o algoritmo é extremamente rápido, apresentando as seguintes vantagens:

1. As trajectórias encontradas e a quantidade de computação necessária são completamente independentes do sistema de coordenadas escolhido.

2. Os obstáculos afectam a representação do espaço livre apenas na sua própria localidade. Assim, obstáculos adicionais espacialmente separados de uma trajectória livre encontrada quando eles não estavam presentes não podem afectar a capacidade de encontrar essa trajectória livre quando estão presentes. Este problema surge frequentemente por exemplo nos algoritmos que dividem o espaço livre em células rectangulóides.

3. O espaço livre é representado explicitamente em vez de ser representado como o complemento do espaço ocupado.

4. As trajectórias tendem a ser igualmente afastadas de todos os objectos, em vez de serem tão próximas dos obstáculos quanto possível, situação em que a instanciação das trajectórias por dispositivos mecânicos se revela mais sujeita a falhas devido a imperfeições mecânicas.

Brooks apresenta ainda algumas ideias e alguns dos problemas inerentes à continuação desta investigação, no sentido de alargar este método ao espaço 3-D.

5.3.1.3. Triangulação do Espaço Livre

Os algoritmos de triangulação são muito úteis na fase de pré-processamento para muitos problemas, como reconhecimento de padrões, gráficos de computadores, geometria computacional, etc.. Tal advém quer da simplicidade das figuras resultantes, quer das facilidades de processamento posterior.

Neste capítulo o nosso maior interesse reside em focar o modo como as vantagens dos triângulos e algoritmos de triangulação podem ser aplicadas à resolução do problema do planeamento de trajectória para um robot.

Tang e Jarvis [5] apresentam uma abordagem que representa o espaço livre entre obstáculos por uma rede de triângulos que se tocam, mas não se sobrepõem, e cujos vértices são os vértices dos obstáculos poligonais. O algoritmo de triangulação proposto por estes autores encontra-se exposto no capítulo sobre representação espacial.

Cada nó da rede é um triângulo, cujas arestas serão uma aresta de um obstáculo ou um segmento de recta que liga dois vértices de obstáculos mutuamente visíveis (aresta "livre"). Uma trajectória viável entre quaisquer dois pontos no espaço livre é formada pela ligação dos pontos médios das arestas "livres" dos triângulos adjacentes, e ainda dos pontos de partida e objectivo. As alterações de orientação do robot só são permitidas nestes pontos médios. A imposição destas restrições ao algoritmo de planeamento de trajectória reduz grandemente o esforço computacional.

O processo de planeamento divide-se em dois passos:

1. Pesquisa de uma rota que ligue os triângulos inicial e final, juntando nós adjacentes da rede de triângulos.

2. Verificação de colisões na trajectória produzida pelas ligações dos pontos médios da rota encontrada (ver descrição do modo como é feita esta verificação em [5]).

Uma trajectória pode não ser viável se durante os testes de colisão se verificar que o robot não pode passar em determinada junção, junção essa que determinará triângulos que não constituem zonas de passagem. Consequentemente as correspondentes arestas serão marcadas como "não livres" e o processo de planeamento será repetido até se encontrar uma trajectória viável ou se conclua que não existe uma trajectória. Na maioria das situações, um destes resultados deverá ser obtido num curto período de tempo.

Uma vantagem que esta algoritmo apresenta em relação aos trabalhos de Brooks e Lozano-Pérez é o facto de ignorar as dimensões e orientação do veículo na fase de formação da rede de triângulos, evitando assim processos que consomem muito tempo, como o crescimento dos obstáculos ou o cálculo de áreas mínimas de segurança. A forma e orientação do objecto móvel só são tomadas em consideração na fase de formação da trajectória, o que permite uma poupança substancial nos custos computacionais.

Em particular, o método concebido por Brooks [4], que já apresentámos anteriormente, envolve a conversão do espaço livre numa rede de elementos de informação com os ângulos de rotação permissíveis em cada nó. Esta abordagem tem duas desvantagens principais:

1. Torna a estrutura de dados muito complicada e consequentemente o espaço livre pode não ser completamente representado. Aliás, isto mesmo é referido pelo próprio Brooks, em especial em relação aos ambientes com pesadas restrições espaciais.

2. Os algoritmos de planeamento tornam-se mais complexos.

Tang e Jarvis reivindicam para o seu método a ultrapassagem destas dificuldades através da abordagem inversa do problema. Não consideram inicialmente as dimensões físicas do veículo, de modo que uma rede de triângulos pode representar totalmente o espaço livre. Depois, então, utilizam as características úteis da rede para localizar, e consequentemente acelerar, o teste de colisões.

Liegeois e Mognard apresentam [6] um trabalho também dentro desta área, mas com características algo distintas. Os autores apresentam como motivação para o seu trabalho o facto de, na investigação em robótica móvel, pouca atenção se ter dado, até ao presente, aos veículos em superfícies irregulares. Com efeito grande parte do trabalho de investigação é dedicado aos veículos autónomos em mundos 2-D com obstáculos, casos em que a velocidade é baixa e é possível assumir que é constante num piso liso. Deste modo a dinâmica do veículo é ignorada e o problema do planeamento de trajectória torna-se um problema geométrico.

O problema dos veículos em superfícies irregulares, com potenciais aplicações de relevo (vigilância, combate a incêndios, agricultura, exploração planetária, etc.), apresenta, contudo, um acréscimo de complexidade teórica em relação ao problema 2-D:

1. O terreno é irregular e não homogéneo, e as suas características podem variar de acordo com o tempo.

2. Os veículos podem usar outros actuadores que não motores eléctricos, por exemplo motores de combustão.

3. As velocidades dos veículos podem ser altas.

4. O índice de desempenho pode deixar de ser o comprimento da trajectória e passar a ser o tempo ou o consumo de combustível, com restrições adicionais como risco, estabilidade, etc..

O método apresentado em [6] baseia-se numa discretização do ambiente representado por curvas de nível e na pesquisa em grafos.

Inicialmente, a partir da representação por curvas de nível é feita uma triangulação do terreno, com a preocupação de preservar a informação inicial mais importante.

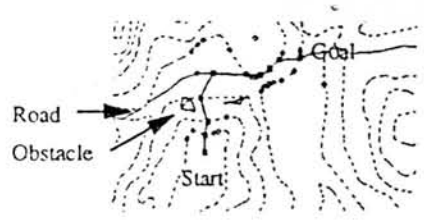
O planeador de movimento terá a responsabilidade de computar a trajectória em tempo óptimo entre um ponto de partida e um ponto objectivo (ambos acrescentados ao modelo triangulado do terreno), trajectória essa que segue arestas da triangulação. O planeador fornece ainda as correspondentes velocidades e mudanças, contando para tal com um modelo da dinâmica do veículo, que no caso particular da aplicação de Liegeois e Moignard se encontra exposto em [6].

É, portanto, construido um grafo que tem como nós os vértices das triangulações e como condição de adjacência a existência de uma aresta da triangulação entre nós. O algoritmo de pesquisa associado é o A*, com uma função de custo $f=g+h$, em que g é o tempo calculado entre o ponto de partida e o nó (vértice) actual e h uma estimativa do tempo mínimo necessário para atingir o objectivo.

No caso geral da aplicação considerada o problema pode não ser simétrico, pois à mesma parte da trajectória podem ser asssociados dois custos diferentes, conforme o sentido do movimento. Tal deve-se ao facto de se poderem usar estratégias diferentes conforme o veículo esteja numa descida ou não.

Liegeois e Moignard discutem ainda o problema da escolha da função heurística h, bem como as vantagens e inconvenientes de usar uma função de custo do tipo $f=g+kh_0$. Não nos deteremos em qualquer consideração sobre estes problemas, por tal se desenquadrar do esquema de abordagem que temos vindo a aplicar.

Na figura 5.7 apresentamos um exemplo da aplicação deste planeador.



Edge	Speed	Gear
T750-T719	V=108.44 km/h	rapport 3
T719-T706	V=150.00 km/h	rapport 4
T706-T683	V=150.00 km/h	rapport 4
T683-T650	V=150.00 km/h	rapport 4
T650-T550	V=150.00 km/h	rapport 4
T560-T526	V=58.99 km/h	rapport 2
T526-T548	V=50.00 km/h	rapport 1
T548-T542	V=46.44 km/h	rapport 1
T542-T541	V=48.04 km/h	rapport 1

fig. 5.7 Uma trajectória com respectivas velocidades e mudanças

Conscientes das limitações desta sua primeira abordagem, os autores propõem algumas melhorias, entra as quais a suavização das trajectórias, tomar em consideração a estabilidade dinâmica do veículo e o uso de modelos dinâmicos mais sofisticados do que o usado na sua implementação.

Ko *et al.* apresentam uma implementação do problema do planeamento de trajectória num labirinto [7]. Utilizam um algoritmo de triangulação [34], bem como a abordagem do Espaço de Configuração [1], reduzindo o robot a um ponto. Os lados dos triângulos são classificados como diagonais ou fronteiras, coincidindo estes últimos com a fronteira do espaço livre.

O segundo passo desta abordagem consiste em determinar o centro do círculo inscrito em cada região triangular. As coordenadas destes centros vão constituir os nós do grafo de triangulação para pesquisa da trajectória solução.

O terceiro passo será definir ligações (segmentos) entre estes nós de forma a completar o grafo. Dois centros C_i e C_j são ligados por um segmento E_{ij} se as correspondentes regiões triangulares T_i e T_j forem vizinhas, ou seja, se tiverem um lado diagonal comum (figura 5.8). Esta escolha de ligações garante o afastamento das fronteiras do espaço livre. A demonstração deste facto encontra-se em [7].

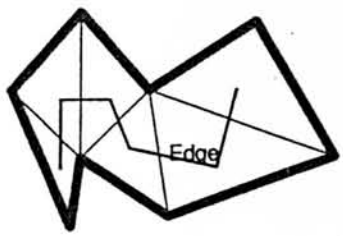


fig. 5.8 Ligações entre centros vizinhos

O grafo de triangulação é uma árvore binária, se virmos o nó de partida C_S como a raiz e o nó objectivo C_g como o destino. A pesquisa da solução numa árvore binária terá a vantagem de ser mais eficiente.

Como exemplo da aplicação deste algoritmo apresentamos um labirinto e respectiva triangulação na figura 5.9 e a trajectória solução na figura 5.10.

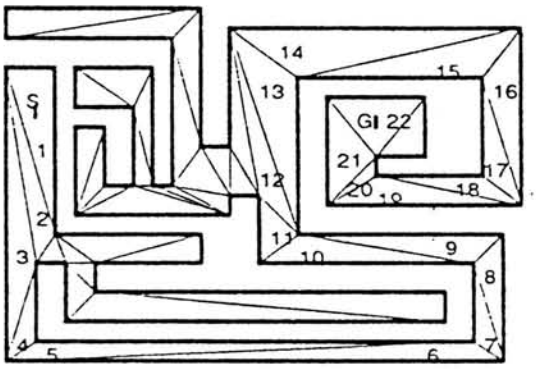


fig. 5.9 Labirinto e respectiva triangulação

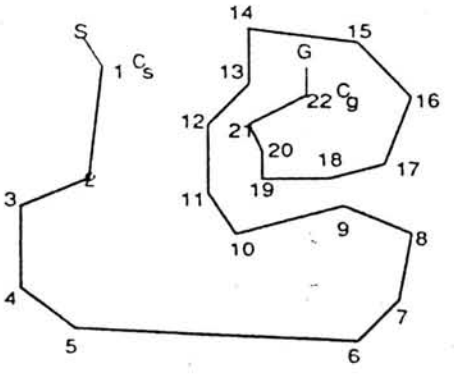


fig. 5.10 Trajectória solução

5.3.1.4. DIAGRAMAS DE VORONOI

O Diagrama de Voronoi Generalizado (DVG), ferramenta de representação do espaço livre cuja descrição se encontra no capítulo deste trabalho dedicado à representação espacial, é a localização de pontos que são equidistantes de duas ou mais fronteiras de obstáculos, incluindo a fronteira do espaço de trabalho. Para um espaço de trabalho poligonal populado com obstáculos poligonais, o DVG consiste de uma rede de segmentos de linha lineares e parabólicos.

As bases teóricas dos DVGs permitem demonstrar que se existir uma trajectória livre de colisões entre o ponto de partida e o ponto objectivo, então existe uma trajectória livre de colisões que atravessa o DVG, com excepção das fases inicial e final, que correspondem a entrar e sair do DVG. Deste modo, a pesquisa de uma trajectória livre de colisões pode ser limitada ao DVG, simplificando de maneira considerável o problema da pesquisa.

Uma vez obtida a representação DVG do espaço livre, pode encontrar-se facilmente a trajectória mais curta do DVG com um raio adequado, usando técnicas de pesquisa em grafos. As heurísticas de movimento dependerão do tamanho e forma do objecto móvel.

Takahashi e Schilling apresentam um algoritmo de planeamento de trajectória para um rectângulo, num espaço de trabalho planar com obstáculos poligonais, usando os DVGs [9].

O DVG é convertido para um grafo equivalente de nós e arcos de modo a permitir a condução de uma pesquisa eficiente. No grafo do DVG aparecem três tipos de nós: os nós de junção, os nós terminais e os pseudonós:

. Um nó de junção é gerado onde se intersectarem três ou mais arcos do DVG.

. Um nó terminal corresponde a um "beco sem saída" de um arco do DVG. Aparecem quando estão presentes vértices côncavos. Como o espaço de trabalho pode ser visto como um buraco dentro de um obstáculo grande, os vértices de um espaço de trabalho rectangular são nós terminais, como pode ser visto na figura 5.11.

. Os pseudonós são o nó de partida e o nó objectivo, introduzidos no grafo perto das posições de partida e objectivo, respectivamente. Estes nós representam os pontos de entrada e saída, nos quais o objecto móvel entra e sai do grafo.

Os arcos do grafo são linhas que ligam pares de nós. Cada arco é representado por segmentos de linha fragmentados entre *pontos via*. Na tabela 5.1 apresentamos a estrutura de dados utilizada para representar um arco. Cada ponto via é caracterizado pelos

parâmetros (x, y, R), em que (x, y) representa as coordenadas do ponto via e R é o raio do DVG no ponto via. As curvas parabólicas suaves podem ser aproximadas arbitrariamente controlando o espaçamento entre os pontos via.

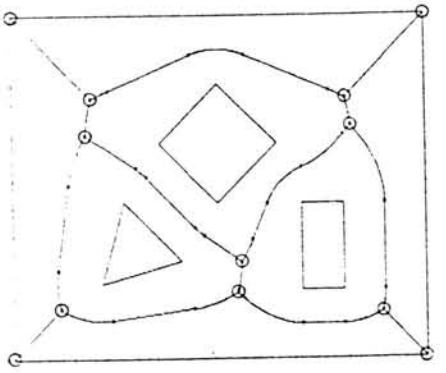


fig. 5.11 Um grafo DVG

Campo	Descrição	Tipo
1	número do nó de partida	inteiro
2	número do nó final	inteiro
3	comprimento do arco	real
4	raio mínimo do arco	real
5	apontador para lista de pontos via	apontador

tabela 5.1 Representação de um arco de um grafo DVG

A formulação apresentada em [9] não se dirige ao planeamento de movimento preciso. Como tal, não é feita nenhuma tentativa de mover um objecto que esteja em contacto com um obstáculo. O planeador move o objecto para um ponto perto do seu objectivo, altura em que poderá ser usado um planeador mais preciso.

Assim, quando as posições de partida e objectivo do objecto móvel não estão em contacto com qualquer obstáculo do espaço de trabalho, a seguinte heurística simples funciona razoavelmente bem no movimento do objecto para dentro e para fora do grafo DVG:

1. É construida uma linha através do centro do objecto móvel, normal à aresta do obstáculo mais próximo. O objecto móvel afasta-se do obstáculo mais próximo, seguindo esta linha, até atingir o grafo DVG. Este ponto de intersecção é o nó de partida ou objectivo, conforme apropriado.

2. Quando o objecto se move para o grafo, da posição de partida para o nó de partida, roda de modo a que, quando atingir o grafo, o seu eixo principal esteja alinhado com o DVG. De modo semelhante, quando o objecto se move para fora do grafo, do nó objectivo para a posição objectivo, ele roda de forma a assumir a orientação apropriada na posição objectivo.

Vários algoritmos de pesquisa em grafos podem ser aplicados para a pesquisa da trajectória. A técnica utilizada em [9] começa por examinar os nós adjacentes ao nó de partida. Os nós adjacentes a estes são investigados e a pesquisa continua a expansão de todos os ramos como subgrafos até cada subgrafo atingir um "beco sem saída" ou o nó objectivo. Se o raio mínimo de um arco for inferior a metade da largura do objecto móvel, é inevitável uma colisão. Consequentemente estes arcos estreitos são considerados "becos sem saída".

Quando um nó adjacente ao subgrafo A já tiver sido atingido pelo subgrafo B, e o comprimento total para o subgrafo A exceder o comprimento para o subgrafo B, o subgrafo A é rejeitado. O subgrafo mais curto que atinja o nó objectivo é então tomado como a trajectória mais curta livre de colisões do DVG.

O critério usado para pesquisar o grafo é simplesmente a trajectória mais curta que satisfaça um limiar mínimo de raio. Nenhum esforço é desenvolvido para avaliar a dificuldade de percorrer essa trajectória. Em alguns casos poderia haver uma trajectória ligeiramente mais longa que não obrigasse o objecto móvel a passar tão perto dos obstáculos. Um modo de determinar uma tal trajectória seria aumentar o raio mínimo para um raio algo superior a metade da largura do objecto móvel. Isto permitiria encontrar trajectórias mais seguras. Contudo, quando a única trajectória existente fosse uma trajectória extremamente estreita, o algoritmo poderia não encontrar qulquer trajectória.

Em espaços de trabalho com muitos vértices de obstáculos, uma parte significativa do tempo total de execução é gasto na construção do DVG. Assim, não parece haver qualquer motivo para não usar a trajectória mais curta, a não ser que haja dificuldade na execução do movimento ao longo dessa trajectória.

Uma vez obtida a trajectória mais curta, é necessário planear o movimento real do objecto móvel ao longo dessa trajectória. Aqui os detalhes do tamanho e forma do objecto móvel devem ser explorados de forma a construir técnicas heurísticas para a execução desse movimento.

Para simplificar estas heurísticas, Takahashi e Schilling assumem que o objecto móvel é um rectângulo ou pode ser envolvido por um rectângulo. De certo modo, esta restrição não é muito severa, pelo menos para objectos poligonais convexos. Na realidade, à medida que um objecto se aproxima de uma forma circular, o planeamento de movimento simplifica-se pois as rotações são desacopladas das translações.

Assim, o caso mais interessante ocorre quando o objecto é alongado e é aqui que um envolvimento rectangular pode ser uma aproximação eficaz da forma e tamanho do objecto. Evidentemente se o objecto móvel não for convexo e a única trajectória possível for uma em que seja necessário explorar o espaço entre o objecto e o seu envolvimento rectangular, então a abordagem proposta falhará.

A. Heurística para Trajectórias Largas

Sejam W e L respectivamente a largura e o comprimento do rectângulo móvel, com $W \leq L$. Dois pontos de referência P_1 e P_2 são localizados ao longo do seu eixo principal, a uma distância de L/4 de cada extremo (figura 5.12).

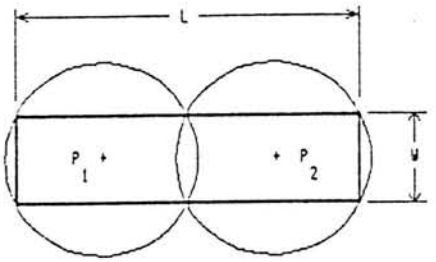


fig. 5.12 A heurística da trajectória larga

A ideia base desta heurística é ter os dois pontos de referência a percorrer a trajectória encontrada, à semelhança das rodas da frente e de trás de um automóvel. O ponto da frente P_2 é avançado incrementalmente ao longo da trajectória e P_1 é determinado pela intersecção do DVG com um círculo de raio L/2 centrado em P_2.

Esta heurística será tentada se o raio mínimo da trajectória satisfizer a condição se ser superior ao raio comum dos dois círculos centrados em P_1 e P_2 e que contêm totalmente o objecto rectangular:

$$R > (W^2 / 4 + L^2 / 16)^{1/2}.$$

Os dois factores que impedem os pontos P_1 e P_2 de seguirem exactamente a trajectória do DVG são o facto de o movimento ser planeado usando uma aproximação do DVG exacto, em que os arcos são representados por uma sequência de segmentos de recta entre pontos via e o facto de o movimento entre os pontos discretos ao longo da trajectória ser assumido como rectilíneo.

Seria possível reduzir o espaçamento entre os pontos via para evitar colisões nos pontos discretos ou reduzir os incrementos na trajectória para evitar colisões entre estes pontos. Em vez disso, procede-se a uma verificação do movimento apenas nos pontos discretos e no caso de ocorrer uma colisão, passa-se a uma heurística mais sofisticada.

Dois exemplos da aplicação desta heurística estão ilustrados nas figuras 5.13 e 5.14.

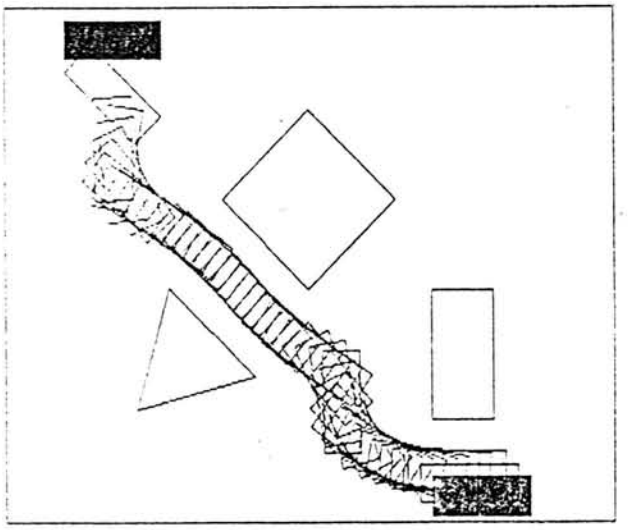


fig. 5.13 Uma trajectória planeada com a heurística da trajectória larga

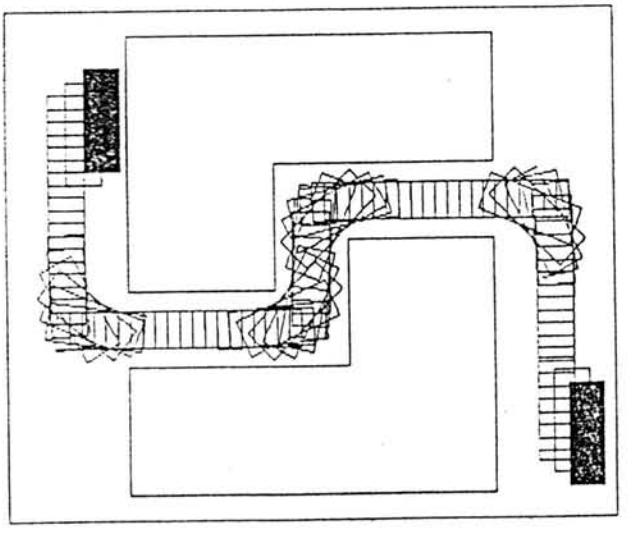


fig. 5.14 Uma trajectória larga com viragens acentuadas

B. Heurística para Curvas Apertadas

Como a pesquisa de trajectórias é efectuada no conjunto de trajectórias com $R > W/2$, a condição da heurística para trajectórias largas pode não ser satisfeita.

Se tal efectivamente não acontecer, é tentada uma heurística de movimento mais sofisticada baseada numa representação local do espaço livre. Tal pode ser necessário, por exemplo, se o objecto móvel tiver de passar através de um corredor estreito em que tenha que se desviar para um dos lados do DVG para contornar uma curva apertada.

Consideremos a figura 5.15. C_0 é a posição da cauda do objecto móvel no DVG e C_3 a correspondente posição para a cabeça. Para representar o espaço livre em ambos os lados do objecto móvel, é introduzido um sistema local de coordenadas L-D. A origem deste sistema de coordenadas é localizada em C_0 e o eixo L é alinhado com o segmento C_0C_3.

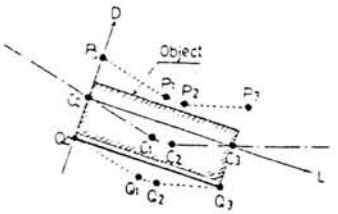


fig. 5.15 A heurística da curva apertada

Os pontos do DVG entre C_0 e C_3 são pontos via. Cada ponto C_k do DVG é expandido num par de pontos $\{P_k, Q_k\}$ gerados pelo movimento em ambos os sentidos segundo o eixo D de uma distância R_k, sendo R_k o raio DVG no ponto C_k. Aplicando esta operação a cada ponto C_k gera-se um polígono de espaço livre que é uma representação local do espaço livre em embos os lados do objecto móvel. No caso da figura 5.15 esse polígono é $\{P_0, P_1, P_2, P_3, Q_3, Q_2, Q_1, Q_0\}$.

O problema reduz-se agora a localizar o objecto móvel dentro deste polígono. Visto que o objecto está fixo no eixo L, há dois graus de liberdade: a orientação do objecto e a posição do objecto ao longo do eixo D.

Na figura 5.15 ilustra-se um método para estabelecer a orientação do objecto móvel. Constroem-se dois segmentos P_0P_3 e Q_0Q_3. Se um destes segmentos se encontrar inteiramente dentro do polígono de espaço livre, o eixo principal do objecto móvel é alinhado com esse segmento. De seguida examinam-se as distâncias entre Q_0Q_3 e cada um dos vértices do lado oposto do polígono, para determinar o vértice mais próximo, neste caso P_2. Essa distância é a mínima largura de espaço livre para a orientação escolhida para o

B. Heurística para Curvas Apertadas

Como a pesquisa de trajectórias é efectuada no conjunto de trajectórias com $R > W/2$, a condição da heurística para trajectórias largas pode não ser satisfeita.

Se tal efectivamente não acontecer, é tentada uma heurística de movimento mais sofisticada baseada numa representação local do espaço livre. Tal pode ser necessário, por exemplo, se o objecto móvel tiver de passar através de um corredor estreito em que tenha que se desviar para um dos lados do DVG para contornar uma curva apertada.

Consideremos a figura 5.15. C_0 é a posição da cauda do objecto móvel no DVG e C_3 a correspondente posição para a cabeça. Para representar o espaço livre em ambos os lados do objecto móvel, é introduzido um sistema local de coordenadas L-D. A origem deste sistema de coordenadas é localizada em C_0 e o eixo L é alinhado com o segmento C_0C_3.

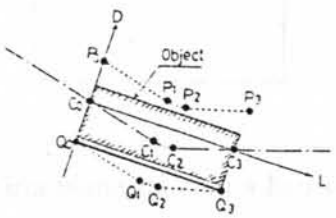


fig. 5.15 A heurística da curva apertada

Os pontos do DVG entre C_0 e C_3 são pontos via. Cada ponto C_k do DVG é expandido num par de pontos $\{P_k, Q_k\}$ gerados pelo movimento em ambos os sentidos segundo o eixo D de uma distância R_k, sendo R_k o raio DVG no ponto C_k. Aplicando esta operação a cada ponto C_k gera-se um polígono de espaço livre que é uma representação local do espaço livre em embos os lados do objecto móvel. No caso da figura 5.15 esse polígono é $\{P_0, P_1, P_2, P_3, Q_3, Q_2, Q_1, Q_0\}$.

O problema reduz-se agora a localizar o objecto móvel dentro deste polígono. Visto que o objecto está fixo no eixo L, há dois graus de liberdade: a orientação do objecto e a posição do objecto ao longo do eixo D.

Na figura 5.15 ilustra-se um método para estabelecer a orientação do objecto móvel. Constroem-se dois segmentos P_0P_3 e Q_0Q_3. Se um destes segmentos se encontrar inteiramente dentro do polígono de espaço livre, o eixo principal do objecto móvel é alinhado com esse segmento. De seguida examinam-se as distâncias entre Q_0Q_3 e cada um dos vértices do lado oposto do polígono, para determinar o vértice mais próximo, neste caso P_2. Essa distância é a mínima largura de espaço livre para a orientação escolhida para o

objecto. Se esta distância for maior que a largura W do objecto, este pode ser localizado dentro do polígono de espaço livre. O objecto é localizado paralelamente a Q_0Q_3 e centrado no espaço entre P_2 e Q_0Q_3.

Como ilustração esta heurística, apresentamos um exemplo na figura 5.16.

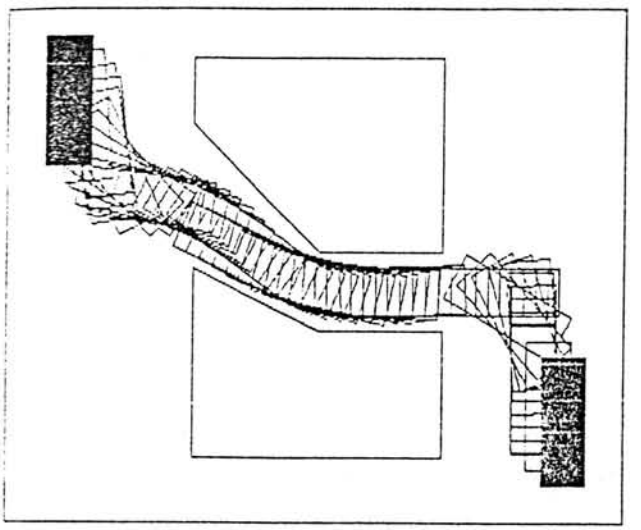


fig. 5.16 Uma trajectória planeada com a heurística da curva apertada

C. Heurística da Passagem Estreita

Se a heurística da curva apertada falhar, ou não for aplicável, é tentado um segundo método para estabelecer a orientação do objecto móvel no polígono de espaço livre. Este método é ilustrado na figura 5.17.

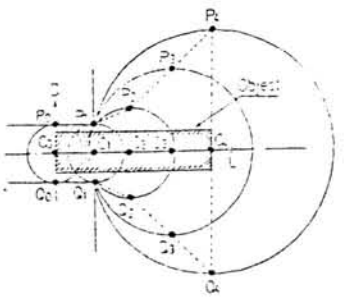


fig. 5.17 A heurística da passagem estreita

O polígono de espaço livre é caracterizado pelos vértices $\{P_0, P_1, P_2, P_3, P_4, Q_4, Q_3, Q_2, Q_1, Q_0\}$. Neste caso, nem o segmento P_0P_4, nem o segmento Q_0Q_4 ficam inteiramente dentro do polígono de espaço livre. Consequentemente a heurística da curva apertada não é aplicável.

Na heurística da passagem estreita, determina-se nesta altura a secção mais estreita do DVG, que para o caso exposto na figura 5.17 será a secção C_0C_1. O eixo principal do objecto móvel é alinhado com o DVG na secção C_0C_1. Se o raio mínimo do DVG ao longo de C_0C_1 for maior que W/2, então o objecto pode ser localizado no polígono de espaço livre. O objecto móvel é então alinhado com o segmento C_0C_1 e centrado em relação ao raio mais pequeno ao longo do segmento C_0C_1.

Esta heurística é tipicamente usada para entrar e sair de pequenas passagens entre obstáculos, de onde advém o nome heurística da passagem estreita. Um exemplo que ilustra esta heurística está na figura 5.18, em que se pode destacar o modo como o objecto atravessa a última passagem antes de rodar para a posição final.

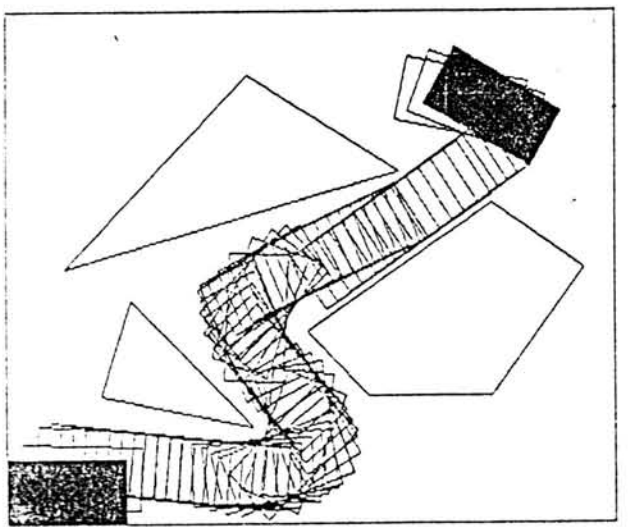


fig. 5.18 Uma trajectória planeada com a heurística da passagem estreita

D. Heurística da Inversão

As três primeiras heurísticas podem não ser aplicáveis em viragens muito agudas na trajectória, tipo "gancho de cabelo". Os ângulos agudos numa trajectória DVG surgem tipicamente em nós de junção onde se encontram três ou mais arcos.

Esta heurística usa uma abordagem em que os papéis dos dois pontos de referência são invertidos algures na trajectória. Para ilustrar esta técnica considere-se o problema da figura 5.19.

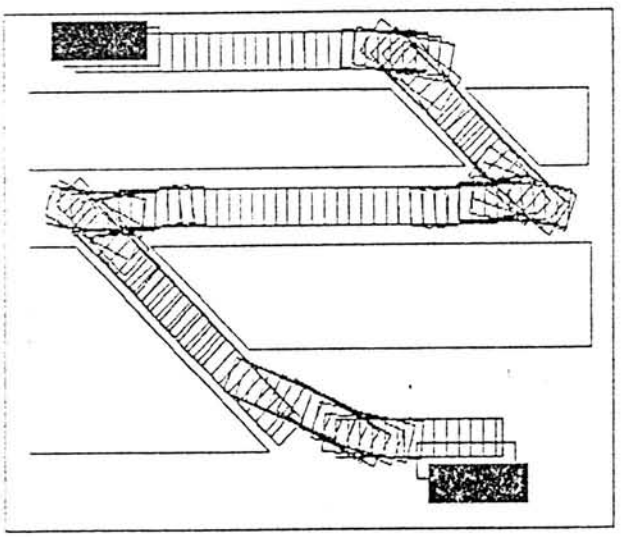


fig. 5.19 Uma trajectória planeada com a heurística da inversão

Quando se tenta usar as três primeiras heurísticas neste problema, são detectadas colisões em dois pontos onde não existe espaço suficiente para a rotação. Estes pontos correspondem a nós de junção que têm pelo menos um arco adicional que não pertence à trajectória mais curta do DVG. Estes arcos suplementares são usados para implementar uma estratégia alternativa de viragem.

Quando se encontra um nó de junção nestas condições, o objecto móvel viaja num arco suplementar até a cauda do objecto móvel atingir o nó de junção. Os papéis da cabeça e cauda são então trocados e o objecto continua o movimento na porção restante da trajectória original.

No algoritmo geral de planeamento de movimento, o método usado para escolher a heurística a aplicar é o seguinte:

1. Se a trajectória mais curta do DVG satisfizer a restrição de trajectória larga, é tentada a heurística da trajectória larga. Esta é a heurística mais simples e é frequentemente a adequada. Neste caso, a trajectória é verificada à procura de colisões

2. Se for detectada uma colisão ou não for respeitada a restrição de trajectória larga, então é tentada a heurística da curva apertada.

3. Se esta falhar ou não for aplicável, então é tentada a heurística da passagem estreita.

4. Como último recurso é usada a heurística da inversão.

Em cada caso as trajectórias são validadas nos pontos discretos da trajectória, mas as colisões no movimento entre esses pontos não são verificadas.

Takahashi e Schilling apresentam ainda uma série de comparações entre o seu algoritmo e outros, como os já apresentados algoritmos de Brooks e Lozano-Pérez [2] e Brooks [4]. Utilizando exemplos semelhantes aos apresentados por estes autores, conseguem obter tempos de computação bastante inferiores e com trajectórias de qualidade superior (mais suaves e com comprimento inferior).

Em relação ao trabalho de Brooks, usando um exemplo (figura 5.20) apresentado em [4], Takahashi e Schilling conseguem obter uma trajectória semelhante, mas com uma ligeira melhoria no comprimento da trajectória devido ao arredondar de esquinas com arcos parabólicos. Enquanto no algoritmo de Brooks as rotações só são efectuadas nas intersecções das "freeways", o movimento que podemos observar na figura 5.20 é bastante suave com o objecto a rodar à medida que efectua as translações.

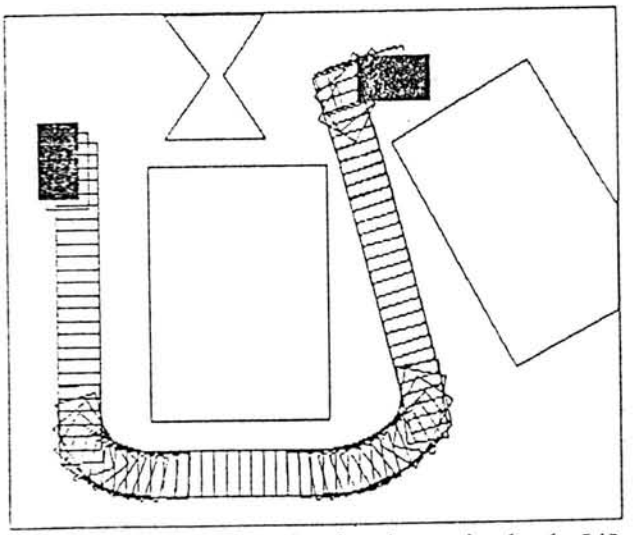


fig. 5.20 Um exemplo simples retirado de [4]

Também em termos de tempo de execução, a técnica DVG parece ser algo mais rápida que o método de Brooks. O tempo de execução citado em [4] era ligeiramente inferior a um minuto para problemas de complexidade moderada usando uma máquina LISP sem hardware floating-point. Para o exemplo da figura 5.20, o tempo de execução foi de

12.1s, usando o Turbo Pascal 4.0, num microcomputador Zenith Z-248 (máquina a 8 Mhz compatível IBM AT). A construção do DVG levou 7.5s, tendo os restantes 4.6s sido usados para planear uma trajectória no DVG.

Para a comparação com o trabalho [2] de Brooks e Lozano-Pérez, foi também usado um exemplo apresentado no trabalho destes (figura 5.21). O tempo total de execução foi 16.4s (7.5s + 8.9s), solução que na mesma máquina já utilizada por Brooks terá levado dezenas de minutos a ser encontrada.

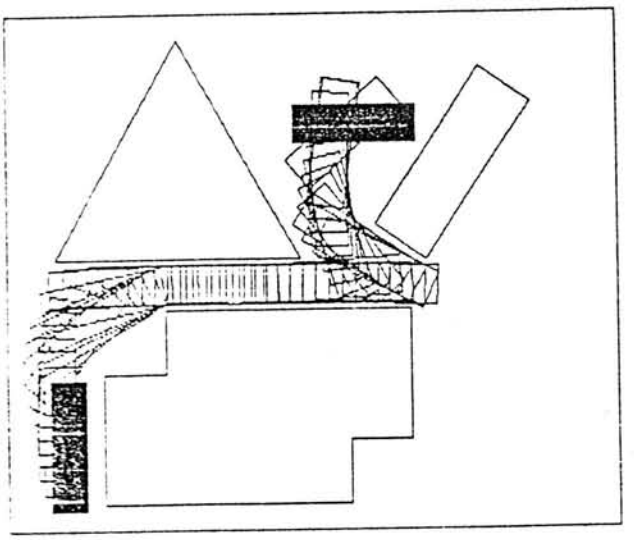


fig. 5.21 Um exemplo complexo retirado de [2]

Finalmente apresenta-se uma aplicação a um problema labiríntico de treze obstáculos (figura 5.22). O tempo de execução foi 67.1s (54.9s + 12.2s).

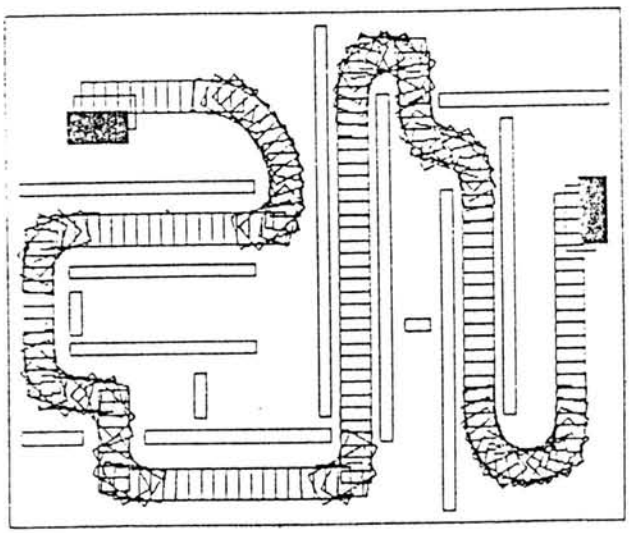


fig. 5.22 Um exemplo com um labirinto

5.3.1.5. REPRESENTAÇÃO QUADTREE

A abordagem proposta por Noborio *et al.* [12] consiste num algoritmo de planeamento de trajectória que escolhe uma trajectória razoável livre de colisões entre o ponto de partida e o ponto objectivo a partir de uma representação quadtree do espaço de trabalho do robot. Esta ferramente encontra-se descrita no capítulo dedicado à representação espacial.

A maioria dos algoritmos de planeamento de trajectória não funcionam bem em ambientes em que a forma e localização dos obstáculos se alterem flexivelmente ao longo do tempo. Entre os motivos que contribuem para tal, podemos salientar:

1. A utilização de dados fixos construidos a partir de um conjunto de obstáculos poligonais. A estrutura do polígono não é adequada ao registo automático de formas de obstáculos na memória do computador via informação sensorial. Deve ser utilizada uma estrutura de dados mais adequada, que possa ser construida directamente a partir de informação sensorial, como por exemplo, uma imagem real obtida a partir de uma câmara.

2. A conversão do conjunto de obstáculos poligonais para os dados fixos requer muito tempo de cálculo. Tal verifica-se para alguns dos algoritmos que vimos anteriormente, como os cones generalizados de Brooks [4] ou os algoritmos que usam espaço de configuração [1] [2] [4]. Com efeito, a construção do espaço de configuração consome muito tempo em qualquer estrutura de dados e com este problema se depararam Khambampati e Davis [11], que usam em conjunto a ferramenta quadtree e o espaço de configuração.

   Por outro lado, estes algoritmos investigam directamente os seus dados fixos, com um grande número de nós como espaço de pesquisa e consequentemente requerem muito tempo de cálculo para escolher uma trajectória razoável a partir desses dados.

No algoritmo proposto por Noborio *et al.* a quadtree expressa o próprio ambiente do robot, enquanto no algorirmo de Khambampati e Davis a quadtree expressa o espaço de configuração do espaço de trabalho, estando assim dependente da escala do robot móvel. No primeiro caso a construção da representação a partir de uma imagem real obtida por uma câmara colocada no tecto é mais rápida. Como a quadtree representa sempre os obstáculos e a sua localização em tempo real, o algoritmo proposto pode funcionar bem mesmo com alterações no ambiente.

Uma outra diferença reside no facto de o espaço de configuração implicar que a representação seja preparada para cada robot móvel, enquanto em Noborio *et al.* a representação é preparada para o ambiente de trabalho em si, o que permitirá poupar capacidade de memória e tempo de cálculo em situações com vários robots.

O algoritmo proposto apresenta um ponto de defeito: é necessário verificar as colisões entre o espaço de movimento do robot e os obstáculos. Para ultrapassar esta dificuldade, o algoritmo escolhe como trajectória livre de colisões uma sequência de regiões livres cujo tamanho seja maior ou igual ao tamanho do robot.

Na quadtree, as regiões de obstáculos e as regiões livres mais pequenas que o robot são vistas como regiões proibidas (figura 5.23). O algoritmo poderá escolher uma trajectória razoável composta de outras regiões livres da quadtree. A sequência de regiões livres inclui pelo menos uma trajectória livre de colisões para o robot móvel, ligando os pontos centrais das regiões livres vizinhas, através de segmentos horizontais, verticais e arcos de quadrante.

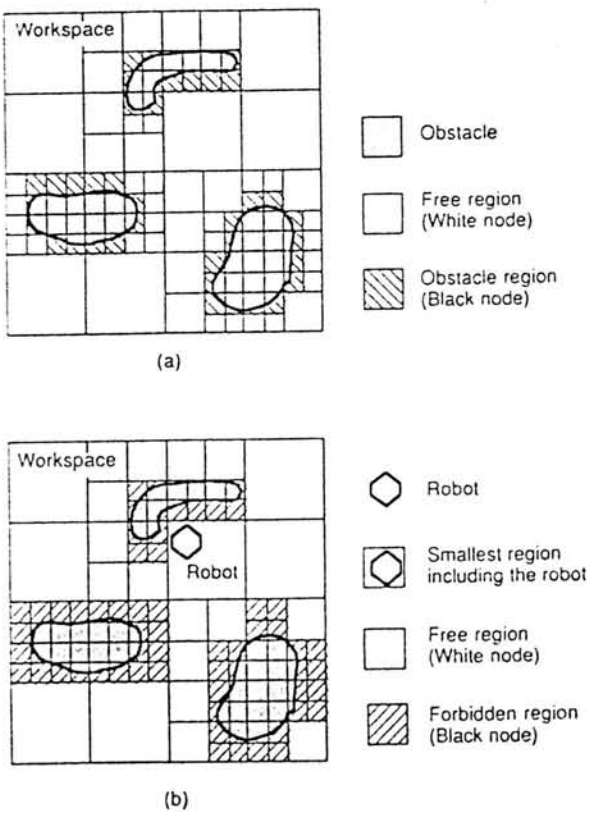


fig. 5.23 (a) Regiões livres e de obstáculos numa quadtree
(b) Regiões livres e proibidas para um robot numa quadtree

A quadtree tem sempre um grande número de nós e consequentemente não é trivial escolher a trajectória livre de colisões em tempo de cálculo económico. Para conseguir tal, constrói-se sobre a quadtree um grafo de trajectórias tão pequeno quanto possível. Neste grafo, os nós correspondem a posições no espaço de trabalho (centros de regiões do nível explorado da quadtree) e cada arco entre dois nós ao segmento de linha entre as posições correspondentes (figura 5.24).

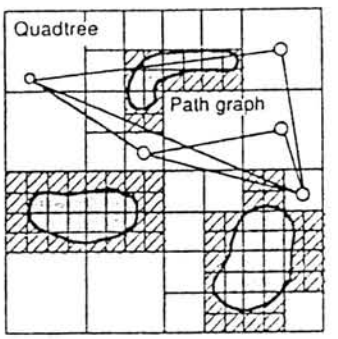


fig. 5.24 Um grafo de trajectórias na quadtree

Para manter o grafo pequeno, o algoritmo efectua a sua construção sem tomar em consideração se o segmento intersecta as regiões proibidas. Apenas mais tarde modificará gradualmente partes do grafo cujo segmento intersecte as regiões proibidas.

Se o segmento L correspondente a um arco não intersectar qualquer região proibida, é atribuida ao arco a seguinte informação: (1) um sinal *zero*, equivalente à não intersecção e (2) um custo definido pela distância Euclidiana do segmento. Se acontecer o contrário, será atribuida a seguinte informação: (1) um sinal *um*, equivalente à intersecção, (2) um ou dois pontos intermédios para evitar cada conjunto de intersecções com regiões proibidas e (3) um custo definido pela soma mínima do comprimento de dois segmentos que liguem os extremos do segmento L através de um dos pontos intermédios (figura 5.25).

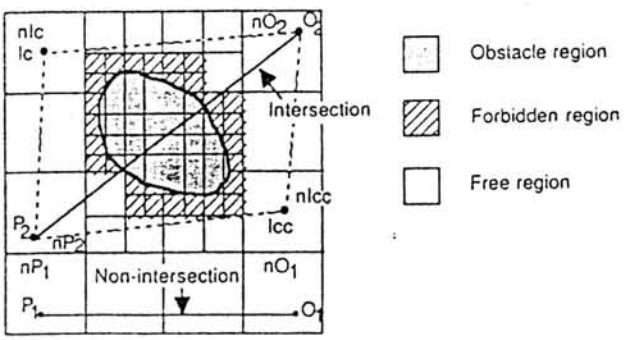


fig. 5.25 Atribuição de uma sinal binário e de um custo para cada arco

Inicialmente será necessário definir o nível de pesquisa na quadtree. O nível de pesquisa é definido como o nível máximo de um nó para o qual o tamanho da região correspondente seja maior ou igual ao tamanho do robot. Consideramos que o nível da árvore aumenta com a sua profundidade. Os nós mistos (regiões de intersecção) são vistos como nós pretos (regiões proibidas), e consequentemente não são investigados.

O procedimento para detectar intersecções é o seguinte:

[Passo 1] Na quadtree, seleccionar os nós *nP* e *nO* correspondentes aos extremos *P* e *O* do segmento L e fazer *n=nP*.

[Passo 2] Encontrar o nó vizinho *n'* de *n* ao longo do vector entre os pontos *P* e *O* [35].

[Passo 3] Registar o nó *n* na sequência S, e fazer *n=n'*.

[Passo 4] Se *n=nO*, passar para [Passo5]. De outro modo, voltar a [Passo2].

[Passo 5] A intersecção ocorre se a sequência S incluir um nó preto, caso em que ao arco é atribuido o sinal *um*. De outro modo, o sinal será *zero* (figura 5.26).

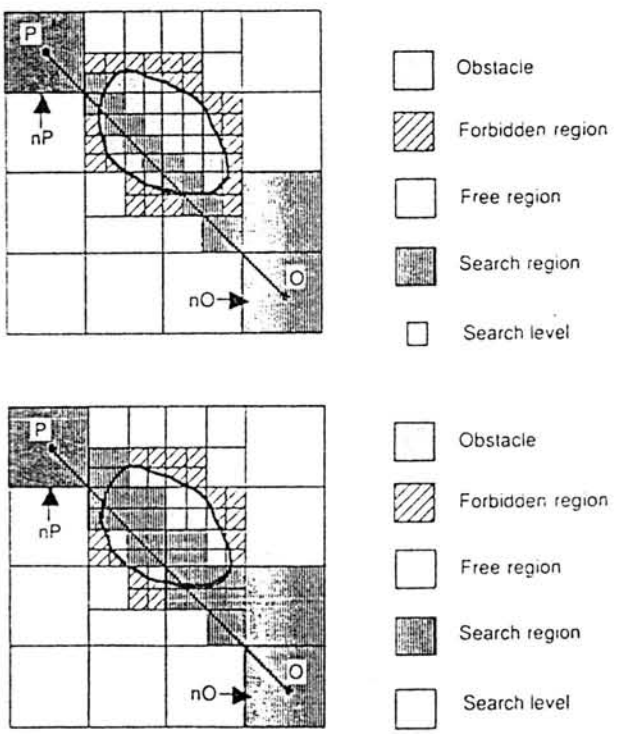


fig. 5.26 Detecção de interferências entre o segmento L e a região proibida na quadtree

Se o segmento intersectar um conjunto de regiões proibidas, devem ser escolhidos pontos intermédios à volta desse conjunto de forma a evitá-lo. Para escolher esses pontos intermédios, o algoritmo detecta as regiões livres (nós brancos) à volta das regiões proibidas entre duas regiões particulares: a região livre em que o segmento entra no conjunto proibido e a região livre em que sai desse conjunto (figura 5.27). Essa detecção é feita quer no sentido dos ponteiros do relógio, quer no sentido inverso.

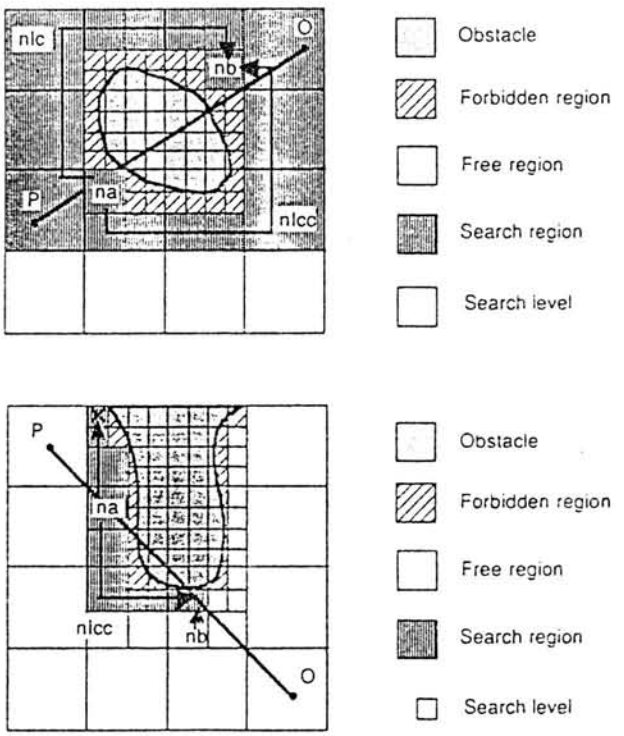


fig. 5.27 Selecção de nós intermédios

Os candidatos a essas regiões particulares são escolhidos da sequência S da seguinte forma:

. Um nó branco é registado numa fila Q se na sequência S preceder imediatamente um nó preto.

. Um nó branco é registado numa fila Q' se na sequência S for imediatamente precedido de um nó preto.

As filas Q e Q' integram, respectivamente, os candidatos a regiões livres de entrada e saída.

O procedimento para determinar os pontos intermédios é:

[Passo 1] Retirar os nós que estão à cabeça de Q e Q' e denotá-los *na* e *nb*, ou seja, $a \leftarrow 0$ e $b \leftarrow 0$. Todos os nós em cada fila estão ordenados de *nP* para *nO*.

[Passo 2] Determinar todos os nós brancos à volta do conjunto de regiões proibidas, partindo de *na* para *nb*, no sentido dos ponteiros do relógio e no sentido inverso, com o auxílio do algoritmo de Samet [36]. Escolher um ou dois nós (*nIc* e *nIcc*) mais afastados do segmento L. Se o nó *nb* não for encontrado em nenhum dos sentidos, não é possível obter qualquer nó intermédio. Assim, retira-se o nó que esteja à cabeça da fila Q', que se toma como nó *nb*, ou seja, $a \leftarrow a$ e $b \leftarrow b+1$, e volta-se a [Passo2].

[Passo 3] Se uma das filas estiver vazia, termina. Caso contrário, retira-se o nó que esteja à cabeça da fila Q', que se toma como nó *nb*, isto é $b \leftarrow b+1$, retira-se a parte superior da fila Q para obter *na* ($a \leftarrow b$) e volta-se a [Passo 2].

O procedimento pode escolher um ou dois pontos intermédios para cada conjunto intersectado de regiões proibidas, mesmo quando o segmento L intersecta vários conjuntos proibidos, como no caso da figura 5.28.

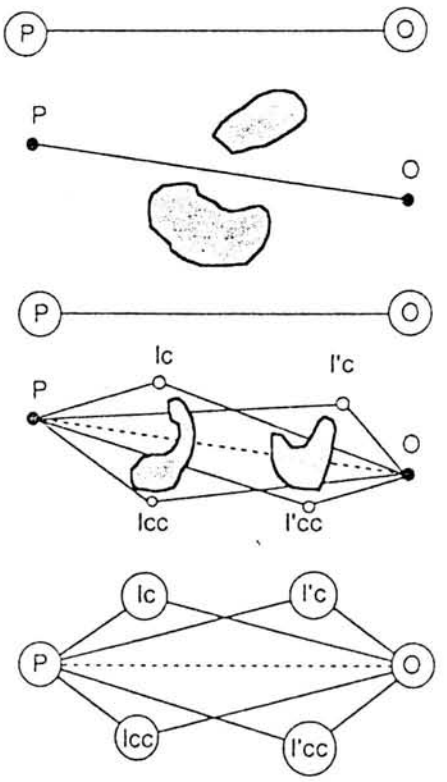


fig. 5.28 Selecção de nós intermédios quando o segmento L colide com vários conjuntos de regiões proibidas

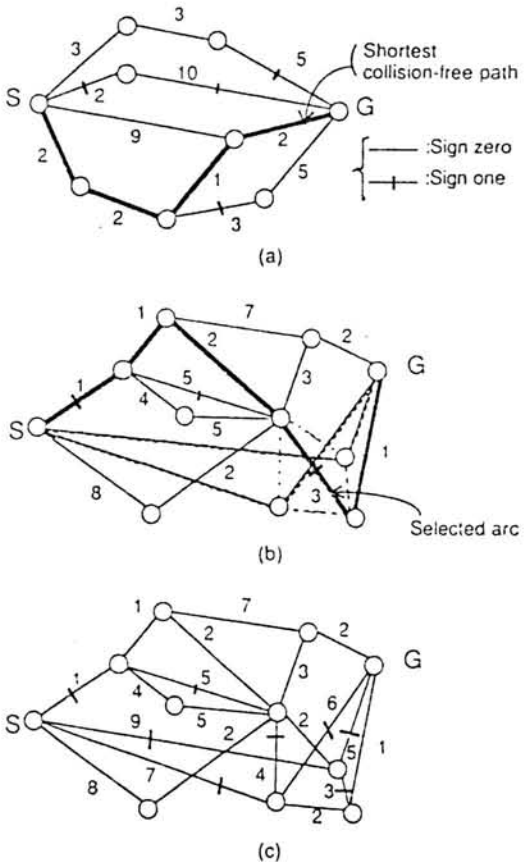


fig. 5.29 Expansão do grafo de trajectórias

O procedimento para selecção da trajectória mais curta no grafo de trajectórias é o seguinte:

[Passo 1] Criação de um grafo de trajectórias inicial com um arco que liga os pontos de partida e objectivo. Os procedimentos de verificação de intersecções e determinação de pontos intermédios lidam com este arco para determinar a sua informação.

[Passo 2] Utilização do algoritmo de Dantzig [37] para selecção da trajectória óptima (mais curta) no grafo de trajectórias. Neste algoritmo o custo de uma trajectória é a soma dos custos de todos os arcos ao longo dessa trajectória e a trajectória óptima é aquela com custo mínimo.

[Passo 3] Se a trajectória mais curta escolhida consistir apenas de arcos com o sinal zero, a trajectória é livre de colisões e o algoritmo termina (figura 5.29 (a)).

[Passo 4] Senão, escolhemos um arco com sinal 1 e custo máximo da trajectória mais curta e juntamos os nós intermédios preparados para o arco do grafo de trajectórias (figura 5.29 (b)).

[Passo 5] O arco escolhido é eliminado do grafo e os novos arcos são acrescentados, ligando cada nó intermédio aos dois nós extremos do arco eliminado e aos nós de partida e objectivo. Os procedimentos de detecção de colisões e determinação de pontos intermédios determinam a informação de cada um dos novos arcos antes de estes serem acrescentados ao grafo de trajectórias. O grafo foi renovado (figura 5.29 (c)) e regressa-se a [Passo 2].

Noborio *et al.* apresentam também várias comparações com outros algoritmos de planeamento de trajectórios, em que o algoritmo proposto demonstra maior rapidez.

5.3.1.6. MUNDO DE RECTÂNGULOS

E. Palma-Villalon e P. Dauchez [13] tentam encontrar um sistema simples e flexível que permita mover um robot móvel no interior de um ambiente industrial.

O uso de marcas e linhas no chão é muito pouco flexível e como tal deve ser rejeitado. A construção de um robot totalmente autónomo, que consiga encontrar o seu caminho através de sensores apresenta a desvantagem da sua complexidade, tendo em conta que se pretende utilizar o robot num ambiente fabril.

O desenvolvimento de um algoritmo de navegação está muito dependente do tipo de representação do mundo que se entender utilizar. Alguns autores usam mapas muito precisos, muito completos, o que implica um maior gasto de tempo na sua produção mas permite o uso de algoritmos mais simples. Por outro lado, pode-se escolher uma representação mais simples, mas com a desvantagem de aumentar a complexidade dos algoritmos utilizados na navegação.

Estes autores propõem o uso de um modelo bastante simples, cujo planeador não é, no entanto, muito complicado. Esta escolha baseia-se no facto de uma representação sofisticada apresentar falta de flexibilidade para lidar com modificações do ambiente. A representação proposta pode ser facilmente actualizada garantindo a consistência do modelo em qualquer momento.

Dado tratar-se da representação de um ambiente industrial, seria possível considerar duas representações simples dos objectos:

. um círculo a rodear o objecto;

. um rectângulo a rodear o objecto.

Os autores optaram pela segunda hipótese já que os rectângulos se assemelham mais aos objectos característicos existentes em fábricas. Um objecto mais complexo pode ser aproximado pela combinação de rectângulos (figura 5.30).

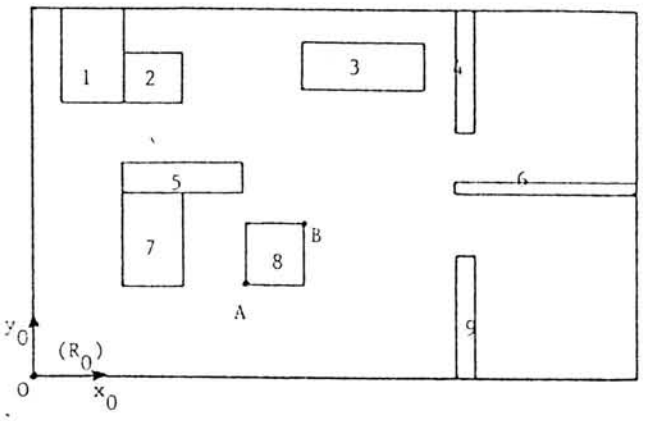


fig. 5.30 Representação do mundo através de rectângulos

Os lados dos rectângulos são unicamente paralelos a dois eixos. Deste modo qualquer objecto pode ser univocamente representado por dois pontos: os extremos opostos do rectângulo (x_{min}, y_{min}) e (x_{max}, y_{max}).

De modo a ter em conta o tamanho do robot, utiliza-se o espaço de configuração, sendo os rectângulos aumentados do tamanho do robot enquanto o robot é reduzido a um ponto.

A representação do mundo é feita apenas por uma matriz (figuras 5.31 e 5.32) que contém, para cada elemento, uma lista dos objectos que total ou parcialmente o intersectam, ou seja, os objectos que inclui. Como tal, um objecto pode pertencer a mais do que um elemento da grelha.

A informação guardada é muito simples, consistindo apenas em números que representam os obstáculos. A definição dos obstáculos, ou seja, os seus nomes e as coordenadas dos seus pontos representativos são guardados noutra lista.

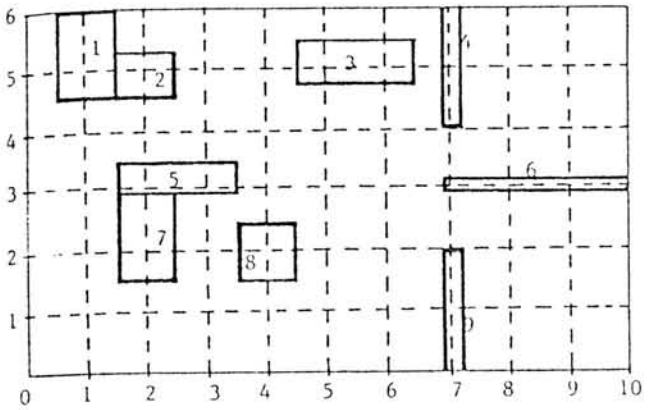


fig. 5.31 Sobreposição de uma grelha no ambiente

(1)	(1,2)	(2)	( )	(3)	(3)	(3,4)	(4)	( )	( )
(1)	(1,2)	(2)	( )	(3)	(3)	(3,4)	(4)	( )	( )
( )	(5)	(5)	(5)	( )	( )	(6)	(6)	(6)	(6)
( )	(5,7)	(5,7)	(5,8)	(8)	( )	(6)	(6)	(6)	(6)
( )	(7)	(7)	(8)	(8)	( )	(9)	(9)	( )	( )
( )	( )	( )	( )	( )	( )	(9)	(9)	( )	( )

fig. 5.32 Representação matricial das relações espacias para o ambiente da fig. 5.31

O planeamento da trajectória tem como principal objectivo encontrar um caminho livre desde o ponto de partida até ao objectivo. No entanto, outras características, como o tamanho, duração e energia consumida, podem ser consideradas na escolha desse caminho.

Para a obtenção dos possíveis pontos de passagem até ao objectivo vai sendo construida uma árvore com todos os caminhos possíveis, de modo a escolher o melhor de entre eles. Usa-se para essa escolha o algoritmo A* com uma função de custo apropriada, como referiremos mais à frente. De modo a determinar os possíveis pontos de passagem, o planeador encontra localmente as direcções proibidas, paredes, e as direcções privilegiadas, os buracos (figura 5.33).

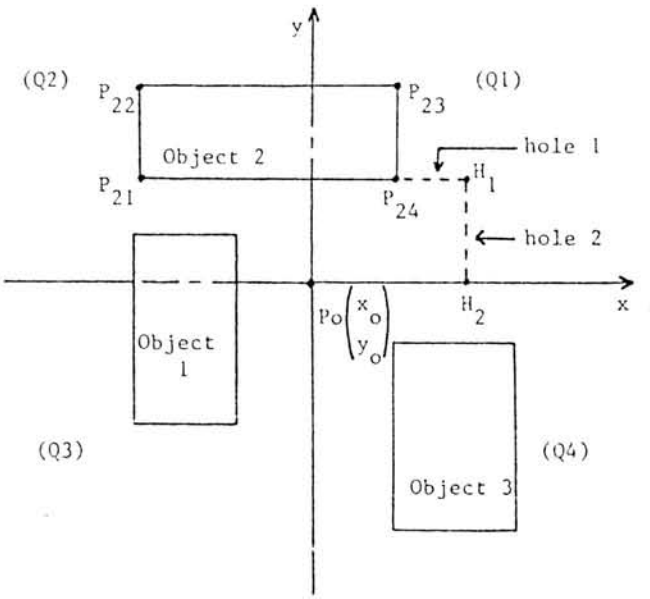


fig. 5.33 Paredes e buracos locais

Apoiando-nos na figura 5.33, a determinação das paredes é feita do seguinte modo:

1. Encontrar a célula a que o ponto actual P_o pertence;

2. Abrir esta célula e as oito adjacentes, ou seja determinar quais os objectos que lhes pertencem;

3. Encontrar todas as paredes na direcção $x > x_o$, ou seja, se houver algum x_{imin} nessa direcção então o segmento $[(x_{imin}, y_{imin}), (x_{imin}, y_{imax})]$ é uma parede;

4. Encontrar todas as paredes na direcção $y > y_o$, ou seja, se houver algum y_{imin} nessa direcção então o segmento $[(x_{imin}, y_{imin}), (x_{imax}, y_{imin})]$ é uma parede;

5. Proceder de igual modo nas direcções $x < x_o$ e $y < y_o$.

Após termos determinado todas as paredes, os buracos são encontrados quadrante a quadrante, de um modo que exemplificaremos apenas para o 1º quadrante:

1. Guardamos a parede na direcção $y > y_o$ mais próxima de P_o (segmento horizontal);

2. Faz-se o mesmo na direcção $x > x_o$. Se não existir nenhuma parede, é criada uma (representada únicamente por um ponto que dista de P_o do comprimento de uma célula).

3. Com estas duas paredes, são determinados como buracos os limites das paredes mais próximos de P_o.

Só os buracos são mantidos e passarão a constituir possíveis pontos de passagem. Outros pontos poderiam ter sido escolhidos, porém, segundo os autores esta é a escolha que permite a obtenção de melhores resultados.

Tendo determinado todos os pontos de passagem, aplicamos o algoritmo A*, o qual nos dará um caminho livre entre o ponto de partida e o objectivo. A função de custo utilizada tem em conta dois parâmetros: o tamanho do percurso e o seu número de mudanças de direcção. Deste modo tenta-se optimizar o consumo de energia do robot, já que as mudanças de direcção são muito custosas.

5.3.2. MÉTODOS BASEADOS EM CAMPO POTENCIAL

Hwang e Ahuja apresentam em [17] um algoritmo de planeamento de trajectória para o "classical mover's problem" em três dimensões usando uma representação de campo potencial dos obstáculos. A sua abordagem poderia ser englobada nos métodos de pesquisa em grafo, visto utilizar um grafo e efectuar pesquisa neste. Preferimos colocá-la nesta secção por ilustrar bem os princípios da abordagem de campo potencial.

Este problema consiste em, dados um robot rígido e um espaço populado com obstáculos rígidos, encontrar um movimento ligando as configurações inicial e objectivo do robot.

Na abordagem de campo potencial assume-se que os obstáculos têm cargas eléctricas. O campo potencial escalar resultante é usado para representar o espaço livre. As colisões entre os obstáculos e o robot são evitadas por uma força repulsiva entre eles, que é simplesmente o gradiente negativo do campo potencial.

Vários motivos conduzem ao uso desta representação:

1. O campo potencial pode ser usado para obter uma representação global do espaço, de forma que um planeamento pouco refinado pode ser feito a nível global.

2. Um campo potencial contínuo dá uma boa indicação acerca das distâncias aos obstáculos e respectivas formas, podendo as alterações na posição e orientação do robot ser fetas de um modo contínuo e suave.

3. Na detecção de colisões, a complexidade combinatória da detecção de intersecções efectuada com representações geométricas é evitada. Tal é conseguido pela eliminação da necessidade de fazer essa detecção explicitamente, devido ao uso de um campo potencial que fornece informação da distância aos objectos.

4. O uso de uma definição apropriada de potencial num ponto permite eliminar a influência de obstáculos que não se encontrem na vizinhança do ponto.

Hwang e Ahuja desenvolvem uma função de potencial que tem uma expressão analítica, sendo portanto de computação eficiente. Essa função preenche os requisitos necessários para o planeamento de trajectórias: tem um valor máximo dentro da região do obstáculo e decresce de acordo com o inverso da distância fora dela.

Na figura 5.34 podemos observar um gráfico da função potencial, obtido para um objecto triangular.

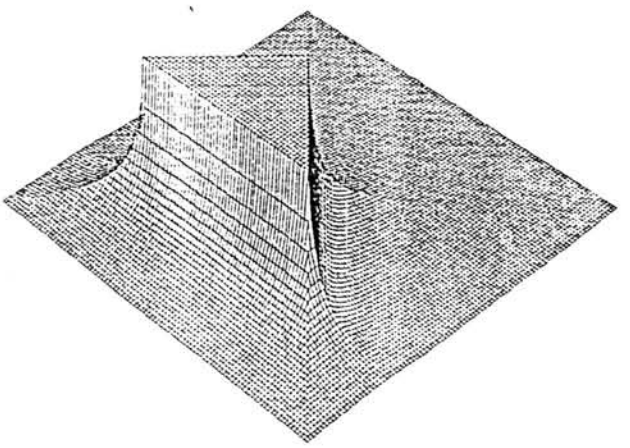


fig. 5.34 A função de potencial devida a um obstáculo triangular

Quando há múltiplos obstáculos presentes, o potencial em qualquer ponto é dado pelo máximo dos potenciais devidos a obstáculos individuais. É crucial usar o máximo em vez da soma, pois no caso de esta ser usada como potencial combinado, pequenos máximos locais de potencial podem aparecer no espaço livre afastado dos obstáculos.

Os vales de potencial mínimo (VPM) podem ser definidos como o conjunto de pontos sela e pontos mínimos locais. Por outras palavaras, se $x \in$ VPM, então há pelo menos uma direcção ao longo da qual o potencial atinge o seu mínimo em x. É desejável ter máximos locais de potencial apenas nas regiões ocupadas pelos obstáculos de modo a que a estrutura VPM capte a estrutura topológica do espaço livre. Em [17] demonstra-se que o VPM possui esta propriedade.

Como o número de pontos do VPM é infinito, o cálculo de um VPM é uma tarefa intensiva. Para limitar esse cálculo, obtemos uma aproximação linear "piecewise" do VPM como um grafo cujos nós são certos pontos ao longo do VPM e cujas arestas correspondem a segmentos de recta que ligam os nós. Hwang e Ahuja apresentam um algoritmo que gera este grafo, começando com os pontos de partida e objectivo, e recursivamente localizando nós pela determinação de locais de baixo potencial ao longo de um círculo (esfera) centrado em cada ponto.

Nas experiências realizadas, o algoritmo VPM correu em menos que um minuto para 2-D e cerca de 10 minutos para 3-D num computador Sun 3/260. É especialmente lento quando há um canal estreito e longo no espaço livre, mas apresenta uma grande robustez a pequenas variações nas formas dos obstáculos.

Esta representação VPM é usada para planeamento global. Um planeador global escolhe do grafo VPM a trajectória mais curta entre os nós de partida e objectivo, com uma estimativa heurística mínima da probabilidade de colisão. Para encontrar a trajectória de custo mínimo pode ser usada programação dinâmica, o algoritmo de Dijkstra, etc..

Uma função de custo é usada para denotar o comprimento e dificuldade de uma trajectória a ser percorrida pelo robot. Uma trajectória é dada como uma sequência de nós e arestas que os ligam:

. O comprimento e custo de uma aresta é a distância Euclideana entre os nós extremos da aresta.

. O custo de um nó mede a dificuldade de colocar o robot na localização do nó e é medido em termos da distância ao obstáculo mais próximo. Definimos que o custo de um nó será infinito (em termos de implementação, um número muito grande) se a distância aos obstáculos for inferior a metade da largura do robot, visto que o robot não poderá passar pelo nó em nenhuma orientação. Se a distância for maior que metade da dimensão maior do robot, este poderá atravessar o nó em qualquer orientação, pelo que o custo do nó será nulo. Para os valores das distâncias intermédias, uma curva monótona ligando infinito e zero pode ser usada. Ao calcular o custo do nó, a distância pode ser aproximada pelo inverso do potencial para reduzir o tempo de cálculo.

O custo total de uma trajectória será a soma dos custos de todos os nós e arestas dessa trajectória.

A trajectória de custo mínimo do grafo é uma sequência de nós relativamente distantes. Esta trajectória é interpolada com um certo número de pontos, de forma a que a distância entre pontos adjacentes seja inferior a um limiar. Este limiar representa a resolução do algoritmo: se o robot não colidir com obstáculos em dois pontos adjacentes, assume-se que se pode mover entre esses dois pontos sem colisões.

A trajectória interpolada serve como estimativa inicial de uma trajectória para o ponto de referência do robot, que será o centro do volume do robot. As orientações iniciais do robot ao longo da trajectória são determinadas pelo alinhamento do eixo mais longo do robot com a direcção da trajectória, minimizando assim o volume varrido pelo robot e consequentemente a probabilidade de colisão.

Em 3-D o grau de liberdade restante é escolhido de forma a que o segundo eixo maior do robot fique no plano tangente à superfície de potencial mínimo. Este alinhamento é

Se o planeador local não conseguir encontrar uma trajectória livre de colisões, a aresta na qual a colisão inevitável ocorre é removida e o planeador global procura novamente uma trajectória com custo mínimo.

Hwang e Ahuja apresentam vários exemplos clássicos resolvidos com o seu algoritmo, bem como uma análise do seu desempenho. Os tempos de computação para a resolução dos exemplos apresentados são inferiores a 5 minutos para exemplos 2-D e entre 5 e 30 minutos para os 3-D, num computador Sun 3/260. Estes tempos são coniderados bastante curtos em comparação com tempos de computação de horas obtidos para esses exemplos com outras abordagens.

Duas características essenciais deste algoritmo contribuem para o seu sucesso:

1. A abordagem efectua eficazmente uma análise multi-resolução do espaço livre. O primeiro passo de extrair todas as trajectórias topologicamente distintas entre o ponto de partida e o destino requer um cálculo relativamente simples, o do campo potencial. De seguida, efectuando um cálculo mais complexo de heurísticas, selecciona candidatos prováveis a melhor trajectória solução. Finalmente, usando o planeador local, a trajectória candidata é modificada de forma a evitar colisões em espaços estreitos. Este tipo de organização da computação, efectuando cálculos mais complexos em partes escolhidas, mais pequenas, do espaço minimiza o esforço computacional total.

2. As configurações viáveis em regiões estreitas são encontradas usando os valores dos potenciais. As orientações com potencial mínimo têm localmente máximos de afastamento dos obstáculos e consequentemente melhores hipóteses de passagem através das regiões estreitas.

Apesar de a decomposição do problema e as heurísticas usadas serem perfeitamente justificáveis e justificadas, o facto de o algoritmo ser heurístico implica que não consiga encontrar soluções para determinados problemas.

Para o algoritmo funcionar, deverão ocorrer determinadas condições:

1. O VPM deve fornecer boas estimativas iniciais de trajectórias solução. Em casos em que os obstáculos e o robot tenham formas complicadas e se encontrem "engatados", o VPM entre os obstáculos pode não se assemelhar a uma trajectória solução de forma nenhuma.

2. A rotação do robot em torno do seu ponto de referência deve produzir alterações importantes na configuração do robot.

A principal motivação para este algoritmo foi desenvolver um algoritmo de planeamento de trajectórias que fosse um compromisso entre os algoritmos heurísticos e os algoritmos exactos. Através da exclusão de um pequeno conjunto de problemas e aumentando marginalmente o tempo de cálculo sobre os algoritmos heurísticos, o algoritmo apresentado é capaz de resolver um largo conjunto de problemas num tempo muito mais curto que os algoritmos exactos. Evidentemente um planeador de trajectórias deve ser escolhido para corresponder à complexidade do problema, à solução desejada e aos requisitos de velocidade computacional. Se por um lado este algoritmo não se adequa a um robot móvel cilíndrico que se movimenta no chão liso de uma fábrica, por outro talvez corresponda à complexidade exigida por um submarino que se movimenta num meio subaquático.

Em [18] Ikazami e Ozono propõem uma abordagem em que toda a informação geométrica é transformada num valor de distância que representa o conceito de menor distância a uma linha fronteira. A este modelo chama-se modelo de valor de distância.

São definidas quatro forças que actuam sobre o robot:

1. Força repulsiva F_R, gerada pelos obstáculos, com a qual se pretende evitar os obstáculos. A direcção da força é normal à superfície equipotencial gerada pelos obstáculos.

2. Força atractiva F_A, gerada pelo objectivo. É sempre dirigida para o objectivo e o seu valor é inversamente proporcional ao quadrado da distância a ele. Para contornar alguns problemas, quando a distância é grande e o valor da força é pequeno opta-se por manter este valor constante.

3. Força rotacional F_T. É usada para fugir a mínimos locais. A sua direcção é tangencial à superfície equipotencial e o seu valor igual a F_R.

4. Força Virtual Atractiva F_V. É uma força imaginária utilizada para manter o robot em movimento. Tem a mesma direcção e valor de F_A e sentido oposto.

No modo de navegação básica, a força de navegação é dada por $F_B = F_R + F_A$, em que o valor de F_R é menor ou igual ao de F_A. Nesta forma de navegação, o robot pode ser conduzido a uma paragem.

No modo de "wall following", o robot vai contornando os obstáculos depois de os encontrar, mantendo uma distância constante a estes. No entanto mantém esse contorno indefinidamente. A força de navegação é neste modo dada por $F_W = F_R + F_V + F_T$.

No modo complementar (figura 5.35) utiliza-se uma troca entre os dois modos anteriores para contornar as desvantagens que ambos apresentam. Assim, quando o robot se encontra num mínimo local passamos do modo básico para o "wall following". Para impedir que o robot contorne indefinidamente o obstáculo, quando for atingido um máximo local faz-se a troca inversa.

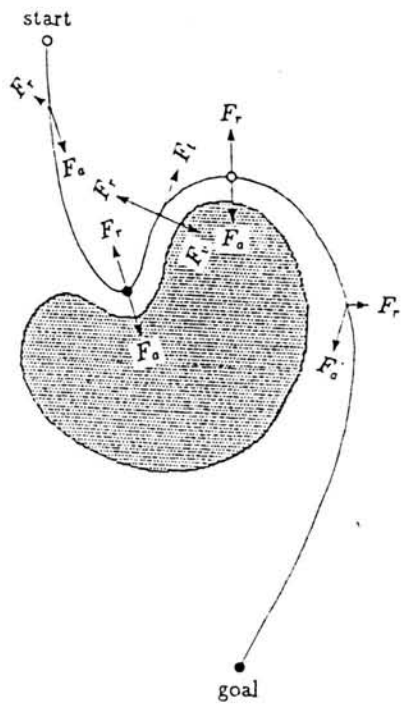


fig. 5.35 O modo complementar

O movimento é iniciado no modo básico, que dirige o robot até ao obstáculo, sem colisões. Atingindo um mínimo local passamos para o modo "wall following", no qual contornamos o obstáculos. Quando se chega a um máximo local, o robot regressa ao modo básico e continua em direcção ao objectivo.

Este método apresenta algumas dificuldades, e, nomeadamente, não garante a convergência para a solução correcta na presença de um mínimo local no campo potencial, nem garante que a trajectória escolhida seja a mais adequada. No entanto permite que a obtenção de uma solução seja feita em tempo prático.

O problema do planeamento de trajectórias em ambientes estruturados pode ser resolvido por meio de uma analogia electromagnética. Em [22] Petridis e Tsiboukis expõem uma abordagem baseada nesta analogia.

O ambiente do robot é representado por uma área de trabalho 2-D dentro da qual existem obstáculos com uma forma arbitrária. O robot e o alvo são representados por condutores perfeitos enquanto a área de trabalho é representada por uma folha de condutividade finita. A representação dos obstáculos é feita por regiões de condutividade zero.

A aplicação de uma diferença de potencial entre o robot e o alvo transforma o problema do planeamento de trajectórias no problema de encontrar as linhas de fluxo do campo estabelecido. A trajectória mais curta entre o robot e o alvo, evitando obstáculos, é especificada pela linha de fluxo mais curta do campo eléctrico estabelecido.

Devido à geometria complicada do problema, este não pode ser resolvido analiticamente. Para o tratamento numérico usa-se um método de elementos finitos em que a região condutora é subdividida num conjunto finito de subregiões (normalmente triângulos), dentro das quais o valor do campo é aproximado por funções de interpolação apropriadas que permitem conhecer os valores nos vértices do elemento. A utilização de algoritmos que permitem desenhar, de um modo contínuo, os tubos de corrente com extremos na posição do robot e no alvo, facilita a determinação do tubo mais curto de fluxo.

Na figura 5.36 mostra-se o resultado da aplicação do algoritmo desenvolvido por estes autores para um determinado "layout". O tempo total de execução foi de 65 segundos, num 386SX com coprocessador 387.

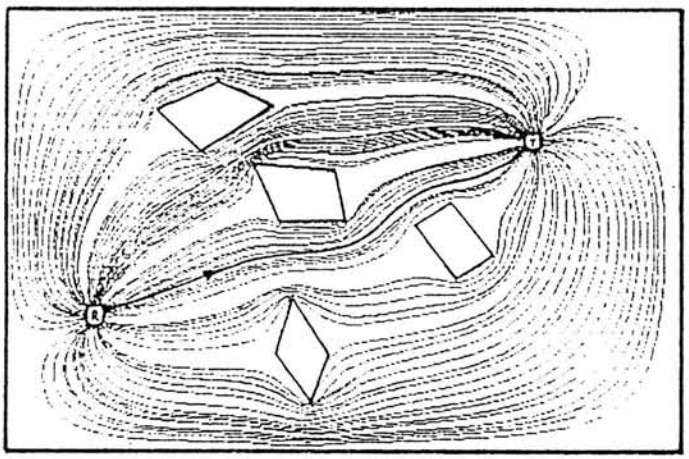


fig. 5.36 Resultado da aplicação do algoritmo baseado na analogia electromagnética num determinado "layout"

5.4 AMBIENTES CONHECIDOS COM OBSTÁCULOS INESPERADOS

A abordagem proposta por Sinha e Benmounah [23] consiste no pré-planeamento de uma trajectória enquanto o robot está estacionário (antes de agir) e em evitar localmente obstáculos desconhecidos enquanto segue a trajectória planeada.

O planeamento de trajectória é feito por um algoritmo muito simples. Quando há um obstáculo entre os pontos de partida e objectivo a trajectória toma o lado do obstáculo que permita obter a trajectória mais curta e ao mesmo tempo evitar o obstáculo.

Quando o robot detecta a presença de um obstáculo inesperado, deve evitá-lo e continuar com a trajectória planeada. Assim, se a trajectória mais curta tomar o lado esquerdo do obstáculo conhecido, o evitar do obstáculo desconhecido deve também ser feito tomando o lado esquerdo. Com efeito, se o espaço entre os obstáculos conhecido e desconhecido for inferior ao tamanho do robot, uma colisão do robot com o obstáculo conhecido será praticamente inevitável.

Em [23] é também exposto um outro método para evitar obstáculos inesperados, que seria parar e activar um mecanismo que detecte a distância relativa às arestas extremo do obstáculo. Após comparação das distâncias, o robot toma o lado mais próximo para se desviar do obstáculo.

Sinha e Benmounah reivindicam que a sua abordagem se insere no estudo de um certo número de esquemas de navegação para aplicação em diferentes ambientes. De qualquer modo, a abordagem destes autores peca por um certo simplismo, visto que não são apresentadas quaisquer indicações sobre situações com múltiplos obstáculos, nem é tomada em consideração a fronteira do espaço de trabalho. Consequentemente desconhecemos, por exemplo, como é que o robot lidaria com uma situação em que não lhe fosse possível contornar o obstáculo e retomar a trajectória planeada.

utilizado é o A* - 3. A função de custo é f=g+ ĥ, onde g é o custo do caminho até um determinado ponto, e ĥ é o custo heurístico desse ponto até ao objectivo.

O navegador não é capaz de garantir um caminho óptimo, pois não toma em consideração obstáculos não modelados, nem prevê a posição de objectos móveis.

O percurso inicialmente calculado, consiste em linhas entre vértices de polígonos adjacentes, o que resulta num percurso em ziguezague. Este percurso pode ser melhorado suavizando o caminho à volta dos obstáculos e paredes.

Este método pode ainda ser utilizado em regiões com diferentes tipos de terreno. Isto constitui uma grande vantagem, sendo poucos os algoritmos que têm em conta este factor. Verifica-se que pode ser perigoso para a estabilidade do robot a colocação simultânea deste em dois tipos de terreno ou o choque com a fronteira de um terreno. Estes e outros aspectos, bem como os custos ou a facilidade de navegação, podem ser introduzidos no 'meadow map' através da inclusão de zonas de transição e de zonas proíbidas.

Neste caso a criação do mapa é feita do mesmo modo. Quando duas zonas diferentes se intersectam são produzidas zonas rectangulares de transição e zonas proibidas. Deste modo obriga-se a que a passagem pela zona de transição seja o mais curta e rápida possível dando maior segurança ao robot.

Hu e Brady [25] debruçam-se sobre o planeamento sob condições de incerteza para um robot móvel que opera num ambiente do tipo fábrica ou armazém. O trabalho exposto por estes autores enquadra-se no seu projecto de construção de um sistema robótico móvel que possa operar numa ambiente industrial dinâmico, evitando obstáculos inesperados e reconhecendo artefactos industriais.

Assume-se que é fornecido ao robot um modelo geométrico 2-D, que é uma projecção do layout da fábrica, populada por máquinas ferramenta, estações de trabalho, zonas de armazenamento e espaço livre. Contudo, há objectos desconhecidos como peças de mobília, caixas, etc. cujas posições se podem alterar durante as missões do robot. Haverá ainda pessoas e outros robots também em deslocação sobre os quais não existe disponível qualquer informação a priori. Apenas os sensores "on-board" são usados para os detectar dentro de um determinado alcance, em tempo real.

Em [24] Arkin apresenta a arquitectura navegacional do AuRA (Autonomous Robot Architecture) que incorpora um planeador hierárquico, constituido por piloto, navegador, planeador e um gestor de esquemas motor.

Esta arquitectura, a que nos referiremos também na secção deste capítulo dedicada à navegação reflexiva, tem exactamente em conta a necessidade de lidar com obstáculos imprevistos. O navegador opera a partir de um mapa estático escolhendo um caminho que não tem em conta obstáculos não representados. Tal significa que terá de haver um módulo responsável pela detecção desses obstáculos e que interactue com o navegador requisitando uma via alternativa, módulo esse que é o piloto. O piloto funciona apenas em pequenas distâncias e da interacção entre os dois subsistemas vão resultando, se necessário, novos planeamentos da trajectória a seguir.

A principal representação usada pelo navegador é um mapa "meadow". Este modela o espaço livre numa colecção de polígonos convexos. Um mapeamento do tipo grafo de vértices do espaço livre permite obter uma representação fácil para a operação de um algoritmo como o A*.

Na fase de inicialização uma série de vértices descrevendo o alcance máximo do robot são calculados. Em seguida a área constituida por esses vértices é transformada até obtermos um conjunto de polígonos convexos. O algoritmo utilizado é recursivo, dividindo sistematicamente a área em duas, até que todas as regiões sejam convexas.

A última fase junta todas as regiões convexas, resultando uma representação do espaço visto de cima. Este tipo de mapa é muito flexível, permitindo fáceis alterações ou adições. Uma vez obtida a representação do espaço, inicia-se o planeamento.

O planeador é constituido pelo planeador da missão, pelo navegador e pelo piloto.

O planeador da missão é responsável por interpretar comandos de alto nível, fazer um planeamento das acções a tomar, decidir a natureza da missão e actuar em caso de falhas provocadas pelos outros dois subsistemas. Deste modo torna-se muito fácil a um operador externo comandar o robot, pois o modo de operação do robot é totalmente transparente para este. O operador só terá de lhe transmitir a missão a executar de uma forma natural.

O navegador aceita um ponto de partida e um ponto objectivo transmitidos pelo planeador da missão e determina, com base no mapa, o melhor percurso. O algoritmo

Planear uma trajectória livre de colisões neste ambiente parcialmente desconhecido é obviamente difícil. Dado que é desejável a optimalidade global da solução, torna-se necessário esclarecer determinadas questões:

. O que é uma trajectória óptima que satisfaça as restrições colocadas sobre o robot móvel e o seu ambiente?

. Como quantificar o custo da incerteza num ambiente industrial dinâmico?

. Se o robot encontrar um obstáculo inesperado durante uma porção da trajectória, que decisão deverá tomar posteriormente: tentar de novo ou desistir desta trajectória?

Para responder a estas questões, Hu e Brady investigaram uma abordagem probabilística do problema, na procura de soluções óptimas em termos de média.

É construido um mapa topológico a partir do mapa geométrico, contendo informação acerca da forma como se interligam as diferentes partes do ambiente. Esta informação é tipicamente invariante para o ambiente e permite ao robot planear uma trajectória viável e verificar a sua localização à medida que viaja. O mapa é um grafo simples não orientado constituido por uma rede de lugares (nós) classificados como salas, corredores, cantos e junções. A sua formulação toma em consideração o tamanho do robot, as características topológicas e geométricas do ambiente e o alcance dos sensores usados.

Na figura 5.37 podemos observar um mapa topológico do laboratório em que Hu e Brady realizaram as suas experiências, gerado a partir da sua descrição geométrica.

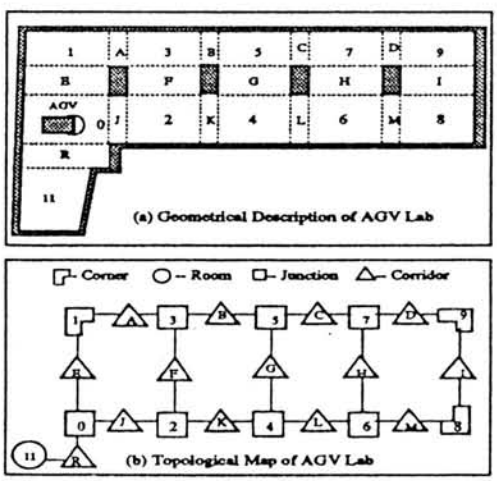


fig. 5.37 Mapa geométrico e mapa topológico

O mapa topológico permite conseguir de uma forma rápida e conveniente o armazenamento e abstracção da informação geométrica sobre o mundo real. É principalmente usado por um planeador global para determinar uma rota ou conjunto de pontos intermédios que o robot deverá seguir ao viajar do ponto de partida para o destino.

Para determinar a trajectória é usado um critério de tempo mínimo pois uma trajectória de distância mínima pode obrigar o robot a desacelerar bruscamente em cantos, parar para girar, ou ainda provocar situações de falta de segurança, ao passar muito perto de obstáculos.

O planeamento de uma trajectória global necessita de uma boa representação do espaço livre para decidir por onde efectuar os movimentos. Para uma fábrica ou armazém é possível usar uma representação directa do espaço de movimento livre como canais de movimento limitados por paredes, máquinas ferramenta, etc.. Deste modo reduz-se grandemente a complexidade de procurar uma trajectória óptima e viabiliza-se o planeamento de trajectórias em tempo real.

As "estradas" que representam o espaço livre (figura 5.38) consistem de linhas rectas e arcos que formarão as arestas do grafo de pesquisa. Este tipo de representação pode ser obtida com um algoritmo do tipo Voronoi [9].

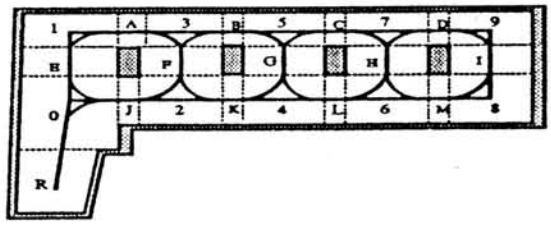


fig. 5.38 Uma representação de "estradas"

A função de custo escolhida para a atribuição de mérito a uma trajectória consiste de duas partes:

. o custo determinístico que depende de factores como comprimento, largura e "straightness",

. o custo de incerteza, estimado de acordo com uma distribuição a priori para parâmetros desconhecidos e que varia dinamicamente de acordo com a informação sensorial.

Seja t_{ij} o tempo gasto pelo robot a viajar a uma velocidade constante do nó i para o nó j sem obstáculos, e u_{ij} o custo de lidar com obstáculos inesperados entre i e j (0 para nenhum obstáculo, $+\infty$ para bloqueio completo e >0 para situações intermédias). Então, o custo total associado ao arco ou linha recta que une os nós i e j é definido como

$$C_{ij} = t_{ij} + u_{ij} .$$

Estes pesos de custo podem ser estimados inicialmente com base em informação prévia e posteriormente actualizados com base em informação sensorial. Cada peso de custo tende para infinito quando a largura do espaço livre é inferior à largura do robot e decresce à medida que a largura do espaço livre aumenta.

A função de custo definida é usada para atribuir um peso a cada arco do mapa topológico, formando-se um grafo de pesquisa não orientado. Com base neste grafo, o algoritmo de planeamento global gera uma trajectória óptima minimizando o custo total entre as posições de partida e objectivo. Como a incerteza de encontrar obstáculos inesperados é tomada em consideração, a trajectória resultante será a trajectória de custo mínimo e probabilidade mínima de colisão.

Num ambiente dinâmico, uma técnica útil é dotar o robot móvel da capacidade de estimar a probabilidade de sucesso de uma trajectória com base em travessias anteriores. A informação sobre obstáculos fornecida pelos sensores é usada não apenas para evitar os obstáculos e replanear a trajectória, mas também para as futuras tomadas de decisão. Deste modo a incerteza e risco associados a cada decisão podem ser reduzidos.

Hu e Brady apresentam um modelo estatístico construido para os obstáculos inesperados. Classificam os obstáculos em três categorias:

. O1 - um obstáculo estático posicionado aleatoriamente que bloqueia parcialmente o caminho do robot.

. O2 - um obstáculo estático posicionado aleatoriamente que bloqueia totalmente o caminho do robot.

. O3 - um obstáculo com movimento aleatório que atravessa o caminho do robot.

Estas diferentes situações têm efeitos diferentes no movimento do robot. O1 faz com que o robot se desvie, contornando o obstáculo detectado, mas o robot continua o seu

movimento em frente. O2 força o robot a parar e pagar o custo extra de retroceder. Quando O3 é detectado, o robot deve alterar a sua velocidade, acelerando ou reduzindo-a, de forma a evitar o obstáculo. Em [25], Hu e Brady apenas se debruçam sobre os dois primeiros casos.

O modelo estatístico derivado pelos dois autores permite actualizar a componente u_{ij} de C_{ij}, a partir quer da informação prévia, quer da informação das observações, para os tipos de obstáculos O1 e O2. Os detalhes deste modelo poem ser consultados em [25].

De forma a lidar com o problema da persistência e mudança, os autores introduzem ainda uma função do tipo $e^{-\mu t}$ para captar a tendência do custo de incerteza estabelecido se tornar falso. Assim, num certo intervalo de tempo, o custo de incerteza entre dois determinados nós irá decrescendo, o que significa que o robot poderá mais uma vez experimentar essa rota.

Uma vez gerada uma trajectória óptima, esta é passada ao controlador do robot. O robot mover-se-á ao longo dessa trajectória até que o conjunto de sensores de sonar com que está equipado detectem a presença de obstáculos inesperados. Se se tratar de um obstáculo do tipo O1 (figura 5.39) que bloqueia parcialmente o caminho, o planeador local (evitar de obstáculos) ajusta a velocidade e direcção do robot para o evitar em tempo real.

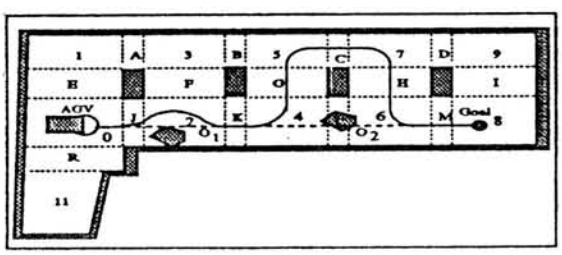


fig. 5.39 O robot tenta atingir um objectivo

Se o caminho estiver completamente bloqueado por um obtáculo do tipo O2, o planeador local reduz a velocidade do robot até este parar e pede uma trajectória alternativa ao planeador. Este aumenta o custo da porção actual da trajectória para infinito e replaneia uma trajectória alternativa. Quando o robot recuar para iniciar a nova trajectória, o custo de incerteza da trajectória bloqueada é alterado de infinito para o valor dado pelo modelo probabilístico.

Se forem encontrados mais obstáculos, a pesquisa de trajectórias continua do mesmo modo até todas as trajectórias terem sido tentadas. Se nenhuma tiver sucesso, o robot móvel retorna ao nó de partida e comunica a falha.

Com base no modelo estatístico proposto, o robot móvel é capaz de aprendizagem de conhecimento global a partir da execução de trajectórias. Assim, se o robot móvel encontrar frequentemente obstáculos que bloqueiem completamente uma porção do caminho, o custo desta trajectória parcial aumentará rapidamente, de forma que será removida do planeamento por algum tempo. À medida que a sua função de custo decai exponencialmente, esta porção de trajectória voltará a ser trajectória candidata e o robot tentá-la-á novamente.

Na figura 5.40 mostramos a escolha feita pelo robot de uma trajectória em arco, devido ao aparecimento frequente de obstáculos durante travessias anteriores. No entanto o robot tentará a trajectória em linha recta de tempos a tempos, dependendo da taxa de queda exponencial.

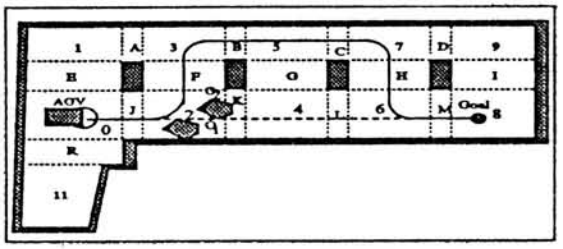


fig. 5.40 O robot escolhe uma trajectória em arco

5.5 AMBIENTES DESCONHECIDOS

Wallner, Lueth e Langinieux apresentam em [30] um algoritmo que permite a um robot efectuar movimentos rápidos e livres de colisões num ambiente desconhecido e dinâmico. A rotina foi concebida para controlar uma trajectória pré-planeada durante a sua execução, cobrindo o tradicional fosso entre a pesquisa de uma trajectória óptima, processo que consome muito tempo, e a necessidade de reacção rápida para evitar obstáculos.

O método desenvolvido por estes autores tem como motivação principal a necessidade de refinar as trajectórias dos planeadores locais e tornar as computações mais eficientes. O algoritmo foi desenvolvido para uma plataforma móvel com três graus de liberdade e equipada com um anel de sensores de alcance ultra-sónicos.

Duas representações são frequentemente usadas para planeamento de trajectórias:

1. Um modelo cartesiano do ambiente, que é uma projecção 2-D do ambiente do robot. O planeamento com tal modelo deve envolver não só a pesquisa de uma ligação entre uma posição de partida e uma posição objectivo, mas também um teste de toda a região ocupada pela geometria do robot de foma a verificar se haverá ou não intersecções com obstáculos.

2. Em contraste com este primeiro modelo, para um robot com n graus de liberdade, o espaço de configuração tem a dimensão n. Cada coordenada é testada para verificar se é ou não uma posição livre de colisões para o robot. O planeamento de trajectórias reduz-se então a encontrar uma série ligada de coordenadas livres. Contudo, a construção do espaço de configuração requer muito processamento de informação, consumindo demasiado tempo para aplicações em tempo real.

O desejo de combinar as vantagens de ambos os modelos, nomeadamente a simplicidade de construção e o planeamento não geométrico, levou os autores a formularem uma nova representação ambiental.

As medidas de alcance são expressas em coordenadas do mundo. As posições em que os obstáculos foram detectados durante os últimos períodos de observação estão disponíveis numa tabela de sensores. O planeamento local escolhe de cada vez um conjunto de medidas mais próximas do robot no momento actual.

Consideremos inicialmente o caso estático. Cada ponto medido é transformado numa linha, tal que esta é ortogonal ao raio de medida e tem comprimento p igual à projecção visível do robot. Quando o robot não está parado, esta tansformação não é suficiente para garantir movimentos livres de colisões. A distância e tempo de desaceleração devem também ser tomados em consideração. Por este motivo uma nova trasformação de linhas para triângulos é efectuada. A trajectória do robot, quando este se desvia dos obstáculos, não é rectilínea, mas curvilínea, como se mostra na figura 5.41. Contudo estas curvas podem ser simplificadas para linhas rectas. A altura h deste triângulo imaginário depende do módulo da projecção da velocidade do robot na direcção central.

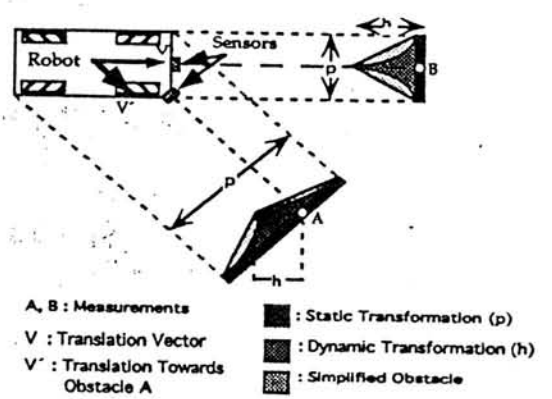


fig. 5.41 Transformação dos obstáculos

Uma outra simplificação adoptada é considerar para esta transformação apenas os obstáculos para os quais aponta o vector de translação. A altura de todos os outros triângulos é colocada a zero. Assim, qualquer que seja a velocidade de translação do robot, se este se estiver a deslocar para um lugar em que não haja obstáculos virtuais (obstáculos alargados), então o movimento é garantidamente livre de colisões.

O resultado desta transformação é semelhante a um espaço de configuração para uma velocidade e orientação específicas do robot, motivo pelo qual se chama "Espaço de Configuração Local".

Para o mapeamento da informação sensorial é utilizada uma grelha. A sua orientação é fixa em relação ao sistema de coordenadas absolutas. A grelha, com um lado de comprimento R e tamanho c das células, é localizada de forma a conter o robot e os obstáculos em consideração, ou seja, os obstáculos localizados num anel em torno do robot cujos raios interior e exterior correspondem respectivamente ao alcance mais curto e mais longo. Na figura 5.42 podemos observar um exemplo deste mapeamento:

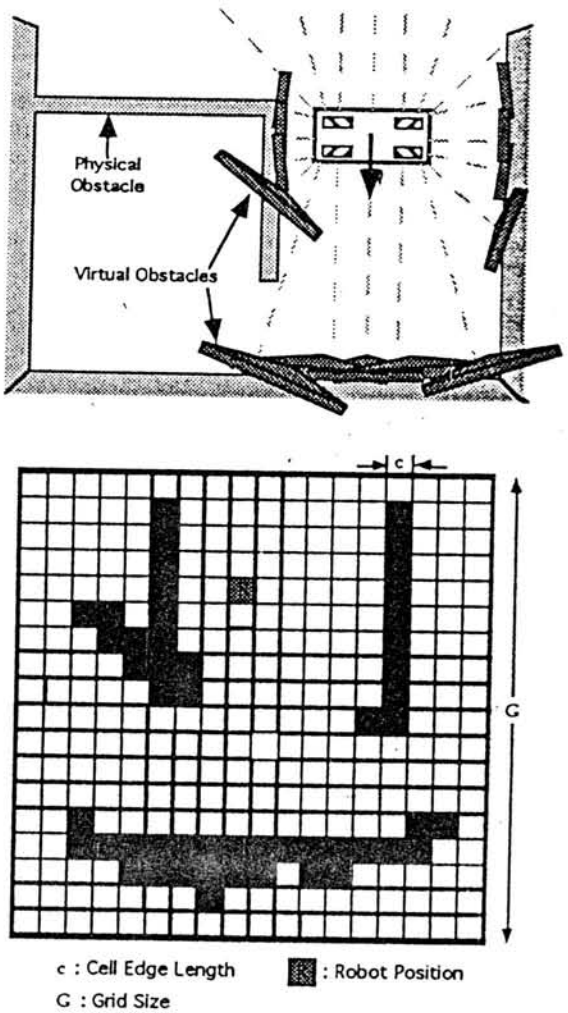


fig. 5.42 Mapeamento de obstáculos virtuais num modelo de grelha

Se tivermos uma discretização fixa da grelha ($n*n$), à medida que o robot se movimentar, R variará e, correspondentemente, c variará.

Esta definição da grelha apresenta diversas vantagens:

1. O número de células permanece constante, qualquer que seja a densidade dos obstáculos.

2. Com R e c variáveis, conseguimos obter alta precisão no caso de uma distribuição densa de obstáculos e

3. um aumento no alcance do planeamento e melhor comportamento preditivo no caso de maior dispersão dos obstáculos.

Na figura 5.43 podemos observar que uma grelha criada de forma a incluir todos os obstáculos sob consideração com o robot localizado no seu centro terá um tamanho maior que uma grelha em que o robot não esteja localizado no centro.

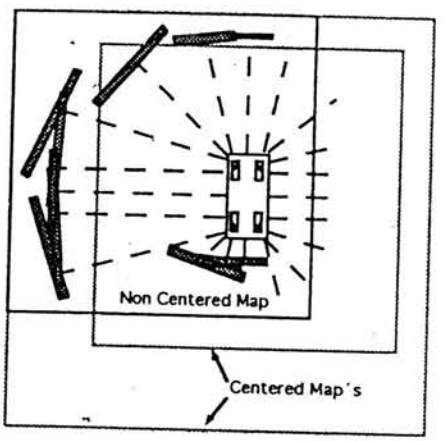


fig. 5.43 Mapeamento eficiente da informação

No entanto, o modelo apresentado apresenta alguns problemas:

1. Quando um robot rectangular está orientado com o seu lado maior para dois obstáculos, os obstáculos virtuais podem ser alargados de um forma que bloqueie o caminho. Contudo, se na altura em que o robot alcança os obstáculos, a sua orientação se alterasse de 90°, os obstáculos alargados poderiam permitir um espaço aberto através do qual o robot pudesse passar.

2. Quando o robot se move a grande velocidade em direcção a um obstáculo, o alargamento artificial na segunda dimensão (ao longo do vector da velocidade) poderá também bloquear o caminho. Contudo, se o robot desacelerar devido a tal, o bloqueio pode diminuir e a trajectória estaria novamente livre. No entanto esta desaceleração tem lugar demasiado cedo, o que resulta num movimento lento.

Para ultrapassar estes problemas, é sugerida uma modificação: para os obstáculos dentro de um alcance pré-determinado, ou seja, uma vizinhança imediata, a transformação mantém-se inalterada; para os obstáculos além desta vizinhança, o alargamento passa a depender apenas da sua distância ao robot.

Desprezando a aceleração, obtem-se a seguinte relação para a distância e a orientação em relação a um obstáculo:

$$s_{rot} = v \cdot \Delta\alpha / \omega$$

s_{rot} - distância

v - velocidade em relação ao obstáculo

w - velocidade de rotação

$\Delta\alpha$ - ângulo entre a orientação actual e a orientação ideal

Usando esta equação e a geometria do robot, é possível calcular o crescimento necessário de um obstáculo. Para uma distância superior à soma da distância de segurança e da distância de desaceleração, o crescimento de um obstáculo é desnecessário. Para obter um comportamento mais suave, *h* é reduzido de uma forma linear (figura 5.44).

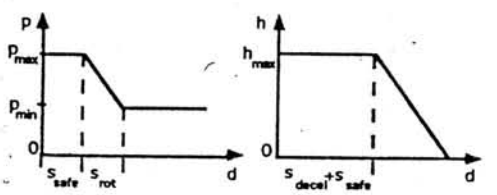


fig. 5.44 Dependência do alargamento dos obstáculos em relação à distância

Mesmo com algumas simplificações, o cálculo da geometria de um obstáculo é um procedimento com custos relativamente altos. Como a informação base (coordenadas de uma medida) é sempre a mesma, é possível calcular "off-line" formas para valores específicos de d, v, $\Delta\alpha$ e ω. A transformação pode ser acelerada significativamente quando a forma mais adequada é escolhida de uma tabela em vez de ser calculada de cada vez.

O processo de mapeamento descrito permite reduzir o problema do planeamento de trajectória ao planeamento para um ponto. Wallner *et al.* usam um método baseado no campo de distâncias.

Conhecida a posição objectivo do robot, os campos de distâncias são construidos de acordo com as seguintes regras:

1. A célula em que está localizado o objectivo é inicializada com um valor de defeito ("1" na figura 5.45).

2. Desde que não contenham um obstáculo, às quatros células vizinhas desta célula é atribuido um valor:

$$\text{valor da célula} = \text{valor inicial} + 1.$$

Estas células são guardadas numa lista OPEN.

3. O algoritmo continua, pesquisando a lista OPEN. Cada uma das quatro células vizinhas de uma célula, que não esteja já preenchida ou seja a fronteira da grelha obtem o valor:

$$\text{valor da célula} = \text{valor da célula central} + 1$$

até que a célula que contém a posição do robot seja atingida.

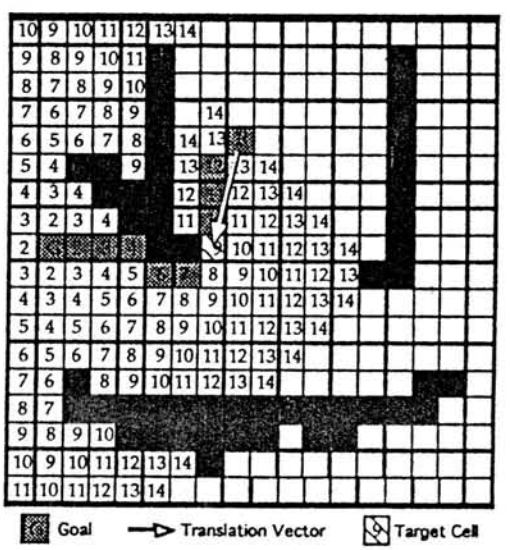


fig. 5.45 Campos de distâncias e pesquisa de gradiente

O algoritmo de planeamento é um método de gradiente simples. Começando na posição do robot, pesquisa ao longo das células de menor valor dos 8 vizinhos até atingir a posição objectivo. O planeamento local não produz uma trajectória completa, mas apenas o comando para o próximo movimento. O gradiente do campo na posição do robot indica já uma direcção livre de colisões para o objectivo, no entanto num único passo de pesquisa não são possíveis mais de 8 direcções. Como a trajectória planeada até passar o primeiro

obstáculo deve ser uma linha recta, é possível repetir a pesquisa de gradiente até tal acontecer. O vector entre a posição do robot e esta última posição indica um movimento livre de colisões e cada célula entre as duas aumenta de 8 o número de possíveis direcções.

O tamanho da grelha é limitado e em muitos casos poderá não conter a posição objectivo. Uma solução simples seria escolher uma célula de passagem na intersecção da linha recta entre a posição do robot e o objectivo com a fronteira da grelha. No entanto esta pode não ser a trajectória mais curta para o objectivo (figura 5.46).

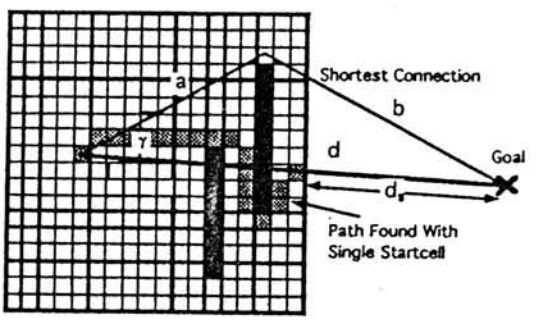


fig. 5.46 Planeamento não óptimo com um objectivo fora da grelha

Menos grave seria este efeito se diversos pontos de partida localizados na fronteira da grelha fossem inicializados com valores correspondentes à sua viabilidade. Em [30] podem ser encontrados detalhes sobre o método que Wallner *et al.* utilizam para obter estes pontos e respectivos valores.

Os autores apresentam ainda em [30] algumas simplificações que permitem aumentar a eficiência do algoritmo em certos casos:

1. O uso de uma grelha rectangular cujo comprimento seja dinamicamente adaptado às observações é mais apropriado a situações, como corredores, em que a grelha contém muitas células que não fornecem qualquer informação. Deste modo conseguiríamos uma redução no número de células a tomar em consideração.

2. Nos casos em que a trajectória pré-planeada está livre, o planeamento local apenas testa se tal é realmente verdade, o que pode ser conseguido de uma forma relativamente simples: as células localizadas na ligação directa entre a posição do robot e o objectivo são testadas para verificar se contêm algum obstáculo. Se este teste

resultar "falso", então o movimento continua inalterado; caso contrário faz-se o planeamento local que se tem vindo a descrever.

3. Na aproximação de um objectivo que se situe perto de um obstáculo, o mapeamento utilizado não garante um movimento livre de colisões. Para lidar com este problema, pode ser efectuado um teste simples: verificar se a geometria do robot projectada na posição objectivo intersecta qualquer célula ocupada (figura 5.47).

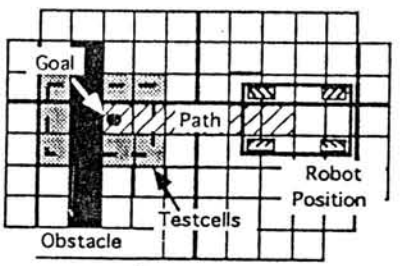


fig. 5.47 Região de teste adicional na proximidade do objectivo

Os autores fazem ainda algumas considerações sobre o planeamento de rotações, tendo em consideração que a qualquer rotação deve corresponder um teste na grelha que é efectuado de forma simples: as células aproximadamente ocupadas durante a rotação são testadas para verificar alguma intersecção com obstáculos. Se o teste for positivo, então o comando de rotação não é dado, enquanto o comando de translação permanece sem ser afectado.

Numa estação Sun 4 com 16 MIPS, a transformação dos obstáculos e a construção do campo de distâcias demora entre 10 e 15 mseg.

Numa experiência efectuada num corredor (figura 5.48) a velocidade média conseguida foi de 0.7 m/seg, tendo sido necessárias várias mudanças de orientação. Um tempo de ciclo de 0.3 seg para o planeamento local foi suficiente nessa experiência.

De um modo geral, o algoritmo proposto combina maior eficiência com diversas vantagens sobre outras soluções, não tendo por exemplo problemas com mínimos locais ou oscilações perto de obstáculos.

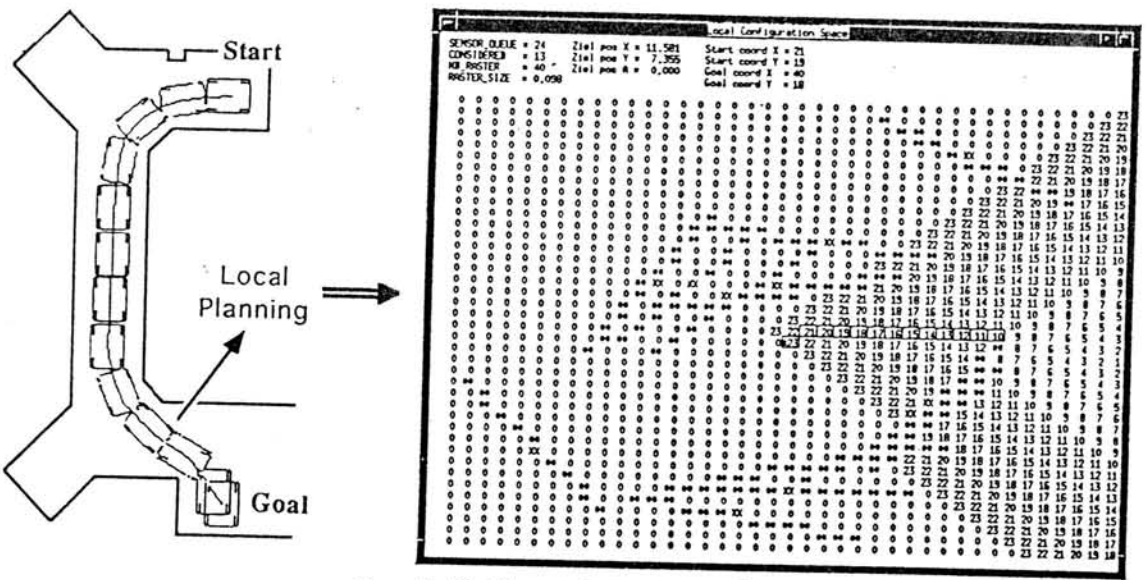


fig. 5.48 Exemplo em corredor

(* - posição do robot, XX - medidas, # - obstáculos virtuais)

5.6 NAVEGAÇÃO REFLEXIVA

Ronald Arkin tem vindo a desenvolver uma arquitectura robótica fortemente influenciada por considerações cibernéticas [31]. A AuRA (Autonomous Robot Architecture) tem tido como base para a sua concepção diversos estudos neuro-científicos e psicológicos. Um dos aspectos mais significativos desta influência envolve a aplicação da teoria dos esquemas a questões como o controlo motor, estratégias perceptuais e sobrevivência do robot.

O regime de controlo motor (esquemas motores) da AuRA está fortemente relacionado com um modelo apresentado por Arbib e House [38] para o desvio de obstáculos do sapo. O uso de comportamentos primitivos é caracterizado por uma utilização do conhecimento de alto nível do ambiente e da missão para a selecção de estratégias perceptuais e motoras, com uma rede dinâmica de comportamentos activos que se alteram à medida que o robot progride no mundo.

O sistema de execução de planos usa uma metodologia navegacional baseada em esquemas. A escolha de esquemas como unidade comportamental básica para o planeamento navegacional foi feita com base na relevância dos esquemas como explicação do comportamento motor em sistemas animais.

Um esquema motor é definido como a unidade básica de comportamento motor, da qual podem ser construidas acções complexas. Consiste do conhecimento de como agir, bem como do processo computacional pelo qual é estabelecido.

O termo esquema tem tido muita utilização, mas o seu uso em psicologia é anterior à sua aplicação em inteligência artificial e outras áreas das ciências de computadores. Os esquemas não são idênticos a "frames", são estruturas de conhecimento que armazenam informação sobre como reagir ao mundo, mas são também processos computacionais concorrentes que têm a capacidade de excutar tarefas perceptuais e motoras necessárias.

Esta abordagem à navegação de baixo nível baseada em esquemas é simples. Diferentes intenções motoras são especificadas pelo planeador de alto nível (figura 5.49).

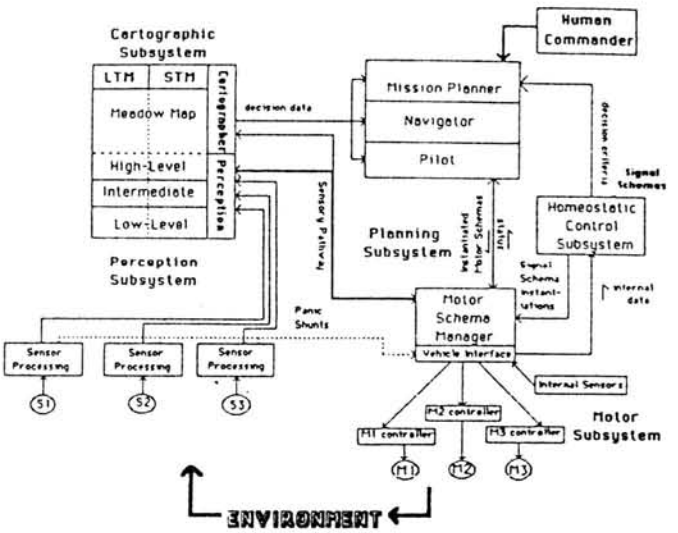


fig. 5.49 AuRA - arquitectura de um robot autónomo

Estas resultam num conjunto de esquemas motores separados e independentes, que incorporam a motivação global para o robot, reflectindo a sua missão e ambiente actuais.

Alguns dos esquemas motores primitivos desenvolvidos incluem:

. Mover-para-objectivo: movimentar-se em direcção a um objectivo percepcionado.

. Evitar-obstáculo-estático: afastar-se de um obstáculo detectado.

. Mover-em-frente: movimentar-se numa direcção geral.

. Ficar-no-caminho: permanecer numa estrada ou corredor detectados.

. "Dock": abordar um local de "docking" de modo seguro

. Ruído: movimentar-se de modo aleatório.

A saída de cada um destes esquemas é um vector que descreve a acção motora que o robot deve tomar em resposta a acontecimentos ambientais percepcionados específicos Uma metodologia de campos potenciais é empregue como base para o vector de saída. Dois campos de esquema representativos podem ser observados na figura 5.50.

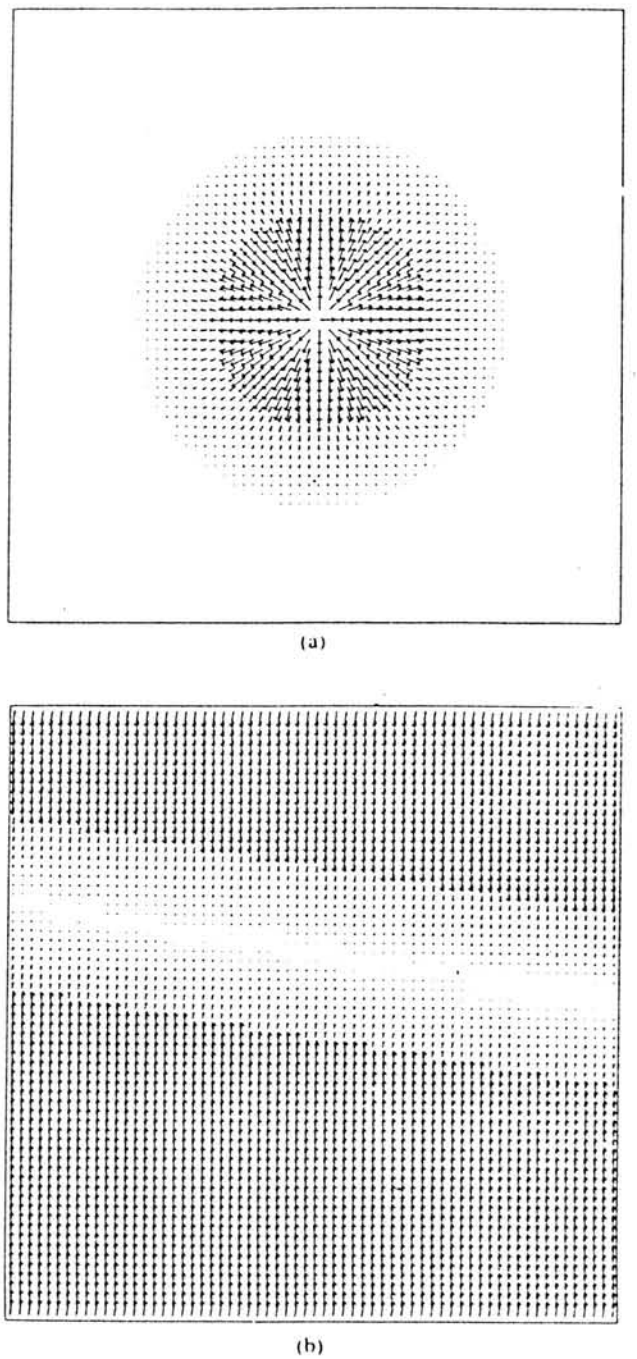


fig. 5.50 Dois campos de esquemas motores (a) Evitar-obstáculo-estático (b) Ficar-no-caminho

O robot não necessita de calcular a totalidade do campo (operação de elevados custos) para se mover. Apenas um vector baseado na posição actual do robot é calculado. O robot reage imediatamente de um modo reflexivo às suas percepções - nenhuma trajectória é

calculada através do campo global. Cada um dos vectores de saída para todos os esquemas motores activos é normalizado e somado, fornecendo a velocidade e direcção com que o robot se deve mover, dadas as prescrições comportamentais e as condições ambientais actuais.

Arbib e House demonstraram que um modelo relacionado com este, obtido para o comportamento de desvio em anfíbios, pode explicar certos tipos de comportamentos motores animais observados. No seu modelo, as acções motoras primitivas estão associadas com campos vectoriais que são mapeados no mundo presentemente percebido pelo sapo. Embora os autores não reivindiquem que este método é necessariamente a técnica realmente usada para navegação animal, ele pode ser usado para fornecer uma base cibernética para o desenvolvimento de sistemas reflexivos de execução de planos para controlo robótico.

O modelo Arbib-House para o planeamento de trajectórias encontrado no comportamento dos sapos inclui um campo de atracção para a presa radialmente simétrico, relacionado com o esquema mover-para-objectivo encontrado na AuRA. Um campo barreira repulsivo ovóide, cuja força decai com a distância à barreira, é semelhante ao campo repulsivo circular que rodeia os obstáculos que se encontra na AuRA. Na AuRA existem diversos campos adicionais não encontrados neste modelo (ficar-no-caminho, ruído, etc.). Em ambos os modelos é usada a soma dos vectores individuais.

Na figura 5.51 podemos ver resultados de uma simulação para a navegação baseada em esquemas. As experiências, efectuadas com o robot George, incluem a manifestação de comportamentos simples e complexos, como seguir paredes, navegar em ambientes com obstáculos, espera impaciente, "docking" e outros.

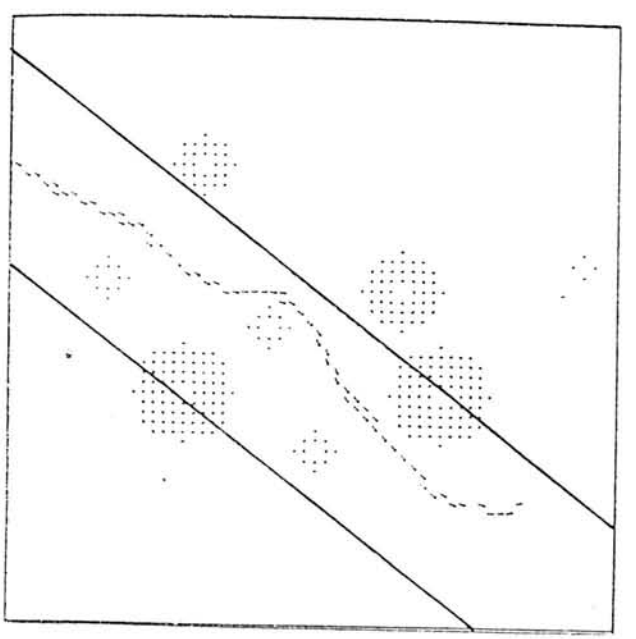


fig. 5.51 Corrida simulada na presença de oito obstáculos, um caminho e um objectivo

As vantagens da navegação baseada em esquemas são muitas:

. oferecem uma explicação plausível para a locomoção animal,

. respondem directamente à informação sensorial,

. são prontamente adaptáveis ao processamento distribuido,

. podem reflectir a incerteza na percepção,

. cada esquema tem uma carga computacional relativamente baixa, permitindo uma resposta rápida em tempo real.

As metodologias de campo potencial, usadas isoladamente, sofrem de alguns problemas. Em particular os máximos e mínimos locais e o comportamento cíclico podem provocar insucessos nas metodologias navegacionais baseadas em esquemas. A AuRA lida com estes problemas através do uso de um planeador hierárquico que escolhe inicialmente os comportamentos apropriados para minimizar a probabilidade de estes acontecimentos ocorrerem. Se estes problemas surgirem, o planeador global é reinvocado para tentar reconfigurar os esquemas de modo a corresponder à nova informação reunida sobre o mundo. Se mesmo assim ocorrerem falhas, o navegador pode ser reinvocado para replanear um nova rota global. Finalmente se ocorrerem novamente falhas, o planeador de missão pode ser questionado sobre a eventualidade de a missão de alto nível não ser cumprida.

Os restantes elementos da AuRA, bem como a sua relação com a arquitectura navegacional serão expostos no capítulo sobre arquitecturas.

BIBLIOGRAFIA

[1] T. Lozano-Pérez
"Spatial Planning - a configuration space approach"
IEEE Transactions on Computers, Feb 1983, pp. 108-120

[2] R. A. Brooks, T. Lozano-Pérez
"A Subdivision algorithm in configuration space for findpath with rotation"
IEEE Trans. on Systems, Man and Cybernetics,
vol. SMC-15, no. 2, Mar/Apr 1985, pp. 224-233

[3] R. H. Davis, M. Camacho
"The application of logic programming to the generation of paths for robots"
Robotica (1984), vol. 2, pp. 93-103

[4] R. A. Brooks
"Solving the find-path problem by good representation of free space"
IEEE Trans. on Systems, Man and Cybernetics,
vol. SMC-13, no. 3, Mar/Apr 1983, pp. 190-197

[5] K. W. Tang, R. Jarvis
"Collision-free path finding amongst polygonal obstacles using efficient free space triangulation"
ICARV '92, vol. 3, pp. RO 11.1.1-11.1.5

[6] A. Liegeois, C. Moignard
"Minimum-time motion planner for mobile robots on uneven terrains"
S. G. Tzafestas (ed.), Robotic Systems, pp. 391-398
(1992)

[7] W. S. Ko, L. D. Seneviratne, S. W. E. Earles, W. C. Ko
"Implementation of robot collision-free path planning in a maze using the triangulation algorithm"
ICARV '92, vol. 3, pp. RO 10.7.1-10.7.5

[8] A. M. Thompson
"The navigation system of the JPL robot"
Proc. 5th IJCAI, 1977

[9] O. Takahashi, R. J. Schiling
"Motion planning in a plane using generalized Voronoi diagram"
IEEE Trans. Robotics and Automation, vol. 5, no. 2, pp. 143-150
1989

[10] J. S. Singh, M. D. Wagh
"Robot path planning using intersecting convex shapes: anlysis and simulation"
IEEE J. Robotics and Automation, vol. RA-3, no. 2, Apr. 1987, pp. 101-108

[11] S. Khambhampati, L. S. Davis
"Multiresolution path planning for mobile robot"
IEEE J. Robotics and Automation, vol. RA-2, no. 3
1986

[12] H. Noborio, T. Naniwa, S. Arimoto
"A quadtree-based path-planning algorithm for a mobile robot"
Journal of Robotic Systems, vol. 7, no.4, pp. 55-574
1990

[13] E. Palma-Villalon, P. Dauchez
"World representation and path planning for a mobile robot"
Robotica, vol. 6, pp. 35-40
(1988)

[14] O. Khatib
"Real-time obstacle avoidance for manipulators and mobile robots"
Int. J. Robotics Research, pp 72-89, 1986

[15] W. S. Newman, N. Hogan
"High speed robot control and obstacle avoidance using dynamic potential functions"
Proc. IEEE Int. Conf. Robotics and Automation, pp. 116-121, 1985

[16] C. W. Warren
"Global path planning using artificial potential fields"
IEEE J. Robotics and Automation, pp. 316-, 1989

[17] Y. K. Hwang, N. Ajuha
"A potential field approach to path planning"
IEEE Transactions on Robotics and Automation,
vol. 8, no. 1, Feb. 1992, pp. 23-32

[18] T. Ikazami, S. Ozono
"A new path planning method for mobile robots applicable to an arbitrary environment"
IROS '92, vol. 1, pp. 453-460

[19] K. Fujimura, H. Samet
"A hierarchical strategy for path planning among moving objects"
IEEE Trans. Robotics and Automation, vol. 5, no. 1
1989

[20] K. Kant, S. W. Zucker
"Towards efficient planning: the path-velocity decomposition"
Int. J. Robotics Research, pp. 72-89
1986

[21] T. J. Pan, R. C. Luo
"Motion planning for mobile robots in a dynamic environment with moving obstacles"
Proc. IEEE Int. Conf. Robotics and Automation, pp. 578-583, 1990

[22] V. Petridis, T. D. Tsiboukis
"An optimal solution to the robot navigation planning problem based on an electromagnetic analogue"
S. G. Tzafestas (ed.), Robotic Systems, pp. 297-303
(1992)

[23] P. K. Sinha, A. Benmounah
"Mobile robot trajectory planning"
S. G. Tzafestas (ed.), Robotic Systems, pp. 279-285
(1992)

[24] R. C. Arkin
"Navigational path planning for a vision-based mobile robot"
Robotica, vol. 7, part 1, pp. 49-

[25] H. Hu, M. Brady
"Planning with uncertainty for a mobile robot"
ICARV '92, vol. 3, pp. RO 13.4.1-13.4.5

[26] H. P. Moravec, A. Elfes
"High resolution maps from wide angle sonar"
Proc. IEEE Int. Conf. Robotics and Automation, pp. 116-121, 1985

[27] A. Elfes
"Sonar-based real-world mapping and navigation"
IEEE Trans. Robotics and Automation, pp. 249-265, 1987

[28] J. Borenstein, Y. Koren
"The vector field histogram - fast obstacle avoidance for mobile robots"
IEEE Trans. Robotics and Automation, vol. 7, no. 3, 1991

[29] B. Faverjon, P. Tournassoud
"The mixed approach for motion planning: learning global strategies from a local planner"
Int. Joint Conf. Artificial Intelligence, pp. 1131-1135, 1987

[30] F. Wallner, T. C. Lueth, F. Langinieux
"Fast local path planning for a mobile robot"
ICARV '92, vol. 3, pp. RO 10.1.1-10.5.5

[31] R. C. Arkin
"The impact of cybernetics on the design of a mobile robot system: a case study"
IEEE Transactions on Systems, Man and Cybernetics,
vol. 20, no. 6, Nov/Dec 1990, pp. 1245-1257

[32] N. J. Nilsson
Principles of Artificial Intelligence
Springer-Verlag, 1982

[33] P. E. Hart, N. J. Nilsson, B. Raphael
"A formal basis for the heuristic determination of minimum cost paths"
IEEE Trans. on Systems, Science and Cybernetics, vol. SSC-4, no.2, Jul. 1968

[34] R. E. Tarjan, C. J. Van Wyk
"An O(nloglogn)-time algorithm for triangulating a simple polygon"
SIAM Journal of Computing, vol. 17, no. 1, Feb. 1988

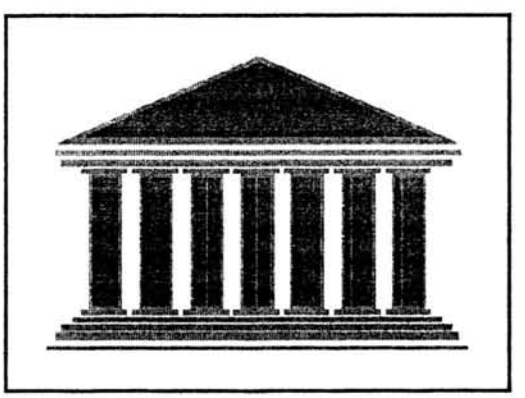
[35] H. Samet
"Neighbour finding techniques for images represented by quadtrees"
Computer Vision, Graphics and Image Processing, 18, pp. 37-57
(1982)

[36] H. Samet
"Region representation: boundary codes from quadtrees"
Communication ACM, 23, pp. 171-179
(1980)

[37] G. B. Dantzig
Linear Programming and Extensions
The Rand Corporation, Princeton University Press, 1963, Section 17-3

CAPÍTULO 6

ARQUITECTURAS



6.1 INTRODUÇÃO

Desde a década de 60 até ao presente a investigação em robótica móvel tem produzido arquitecturas que se encaixam em duas categorias principais: a decomposição funcional e a decomposição baseada em comportamentos.

Os robots com arquitecturas baseadas em decomposição funcional utilizam normalmente um enquadramento SMPA (Sensor - Model - Plan - Act). O mundo externo é observado através de sensores e é elaborada uma descrição. Um planeador de trajectórias procura um percurso óptimo através de uma correspondência entre a descrição actual e um modelo interno do ambiente. O robot move-se, então, ao longo da trajectória planeada.

Embora esta arquitectura tenha tido sucesso em várias aplicações [1] [2] [3], apresenta um ponto fraco no "sensing" e processamento do ponto de vista da robustez contra as falhas. Ocorrendo um erro num processo, o plano de movimento preparado deixa de poder ser executado. Tais falhas ocorrem frequentemente em ambientes complexos e dinâmicos.

A decomposição baseada em comportamentos (arquitectura "subsumption") foi inicialmente proposta por Brooks [4] como forma de ultrapassar esta dificuldade. No sistema por ele apresentado, o comportamento básico é estruturado em camadas na ordem *evitar objectos*, *vaguear*, *explorar*, *construir mapas*, etc.. A robustez é assegurada pela distribuição e activação paralelas destes modos comportamentais.

Embora esta decomposição melhore muito a robustez face a falhas no "sensing" e processamento, tende a ser ineficiente do ponto de vista do cumprimento da missão do robot. Num sistema "subsumption" apenas uma camada é reservada para o vaguear, ou seja, para controlar o robot se o comportamento em camadas superiores falhar, enquanto vários tipos de modos comportamentais adaptativos necessitam de ser seleccionados para recuperar de uma falha. Um sistema baseado em comportamentos com prioridades fixas tende a adoptar um comportamento de baixo nível num ambiente dinâmico, o que é ineficiente do ponto de vista do cumprimento da missão.

Recentemente têm sido propostos alguns sistemas baseados em comportamentos que procuram dar uma capacidade "intencional" à configuração robusta reflexiva/reactiva.

Arkin [5] propôs uma arquitectura baseada em esquemas. "Esquema" é uma metodologia usada para descrever a interacção entre percepção e acção. Cada esquema motor individual corresponde a um comportamento reflexivo. É utilizada uma rede dinâmica em vez de uma configuração em camadas.

Noreils [6] [9] apresentou uma configuração com três níveis (funcional, controlo, planeamento) para robots móveis em interiores. Este sistema é mais flexível que os sistemas "subsumption" tradicionais devido ao facto de o nível funcional ser programado pelo nível de controlo, dependendo da missão. Actualmente, a parte de planeamento envolve dois módulos baseados em regras: o planeador geral e o planeador de trajectórias. Neste sistema o planeador recupera de algumas falhas ocorridas na navegação através de regras preparadas.

Apesar de diversas melhorias, os sistemas que usam arquitecturas baseadas em comportamentos continuam a ter um ponto fraco na combinação dos comportamentos intencionais para cumprir uma missão com o comportamento reflexivo/reactivo, especialmente do ponto de vista da recuperação de falhas.

A recuperação de falhas apenas por planeamento ou a arbitragem de comportamentos através de uma lista de prioridades podem negar os méritos essenciais das arquitecturas baseadas em comportamentos, ou seja, a robustez e a flexibilidade. Watanabe *et al.* [7] procuram preencher o fosso entre o comportamento reflexivo/reactivo e o comportamento intencional de maneira a apoiar esses méritos.

Connell apresenta uma arquitectura denominada SSS [8], que combina uma camada de "servo-control", uma camada "subsumption" e uma camada simbólica, procurando explorar completamente as vantagens de cada uma delas.

Noreils [9] tem trabalhado também ao nível das arquitecturas para robots autónomos e cooperativos, com base (1) na colaboração para a decomposição de uma tarefa em sub-tarefas que são alocadas a um conjunto de robots e (2) na coordenação, em que os robots coordenam as suas actividades para levar a cabo a tarefa inicial usando protocolos de coordenação.

Neste capítulo focaremos em particular as arquitecturas propostas por Arkin [6], Watanabe *et al.* [7] e Noreils [9], que nos parecem evidenciar o estado actual da investigação em arquitecturas baseadas em comportamentos para robótica móvel. Sobre as arquitecturas baseadas em decomposição funcional, parece-nos que as referências e o material exposto nos capítulos dedicados à representação espacial, visão e navegação serão suficientemente ilustrativos dessa categoria de arquitecturas.

6.2 ALGUMAS ARQUITECTURAS

A arquitectura proposta por Arkin [5], a AuRA, que foi já referenciada no capítulo sobre navegação, baseia-se num conjunto de considerações cibernéticas e visa a produção de navegação inteligente para um robot móvel. A AuRA é constituida por cinco subsistemas principais (figura 6.1):

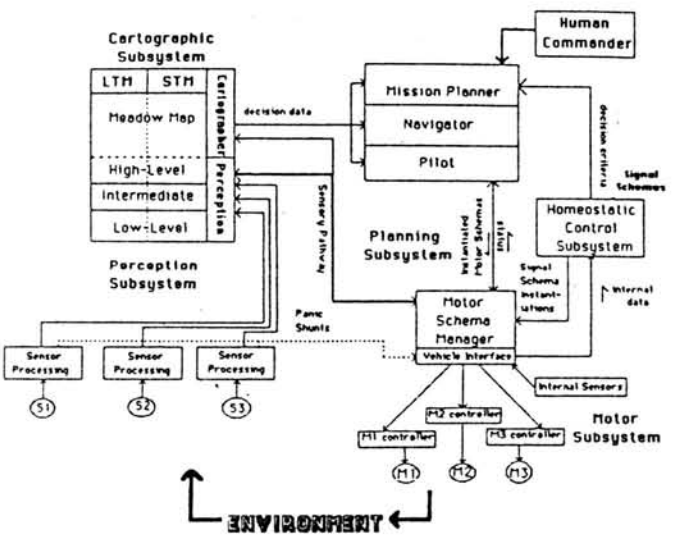


fig. 6.1 AuRA - arquitectura para robots autónomos

. Subsistema de percepção, em que a a informação perceptual entra no sistema, é filtrada e canalizada para as estratégias perceptuais apropriadas para a percepção orientada à acção ou construção da memória de curto prazo.

. Subsistema cartográfico, onde o conhecimento estático e dinâmico do mundo é mantido e oferecido para utilização ao subsistema de planeamento. A gestão de incerteza espacial ocorre também dentro do subsistema cartográfico.

. Subsistema de planeamento, que consiste de um planeador hierárquico e um sistema de execução de planos distribuido (gestor de esquemas motores). Assim, permite-se a

utilização de conhecimento de alto nível para a selecção e formulação de planos, bem como a navegação reflexiva/reactiva altamente acoplada à informação sensorial imediata.

. Controlo homeostático, que permite a sobrevivência do robot com base em "sensing" interno.

. Subsistema motor, que traduz os resultados da execução do plano em comandos específicos para o veículo robótico implementado.

O desenvolvimento da arquitectura AuRA foi motivado por vários objectivos conceptuais, entre os quais:

. Ser aplicável no maior leque possível de domínios navegacionais. A AuRA foi já testada em diversos domínios, como interiores e arredores de edifícios, ambientes fabris e em simulações de navegação 3-D no espaço.

. Permitir o crescimento e desenvolvimento incremental. O uso de comportamentos motores e estratégias perceptuais, codificados como esquemas permite acrescentar de pronto novos módulos comportamentais ou perceptuais sem refazer o design do sistema.

. Utilizar conhecimento de alto nível, quando disponível, para objectivos navegacionais. A inclusão de estruturas de memória de longo e curto prazo fornece os depósitos do conhecimento do mundo. A memória de longo prazo incorpora conhecimento relativamente estático do mundo do robot em termos de espaço livre (usando um "meadow map") e conhecimento semântico "implantado" de "landmarks", características do terreno, etc.. A memória de curto prazo mantém um modelo dinâmico do mundo construido a partir de observações sensoriais da vizinhança imediata do robot.

. Encorajar a independência em relação aos sensores e ao veículo. Utilizando estruturas esquema como unidade perceptual básica, novos sensores podem facilmente ser acrescentados. Este aspecto da independência relaciona-se de perto com a modularidade atrás referida. A independência em relação ao veículo é encorajada através da restrição da maioria das características dependentes do veículo ao subsistema motor da AuRA. Este subsistema é o único grande componente que necessitaria de alterações significativas se um novo tipo de veículo fosse usado.

. Permitir a sobrevivência dos robots em ambientes com riscos. A utilização do controlo homeostático é um dos pontos chave da AuRA.

. Explorar modelos cognitivos, neuro-científicos e etológicos sempre que apropriado para fins navegacionais.

No capítulo sobre navegação apresentámos já os conceitos principais sobre a navegação reflexiva na arquitectura AuRA. Recomendamos por isso a leitura dessa parte do nosso trabalho para complementar o estudo dos restantes aspectos que aqui serão focados.

Relativamente à percepção, levanta-se a questão de como canalizar a informação perceptual para os comportamentos motores individuais. Em vez de construir um modelo do mundo baseado em informação sensorial, Arkin propõe uma ligação entre as necessidades imediatas de um esquema motor e os processos perceptuais requeridos para o apoiar.

A premissa fundamental em que se baseia esta abordagem é a percepção orientada à acção (figura 6.2). Basicamente este princípio afirma que apenas a informação necessária para uma determinada acção motora é extraída do ambiente. O controlo e selecção dos algoritmos perceptuais é ditado pelas intenções motoras do robot.

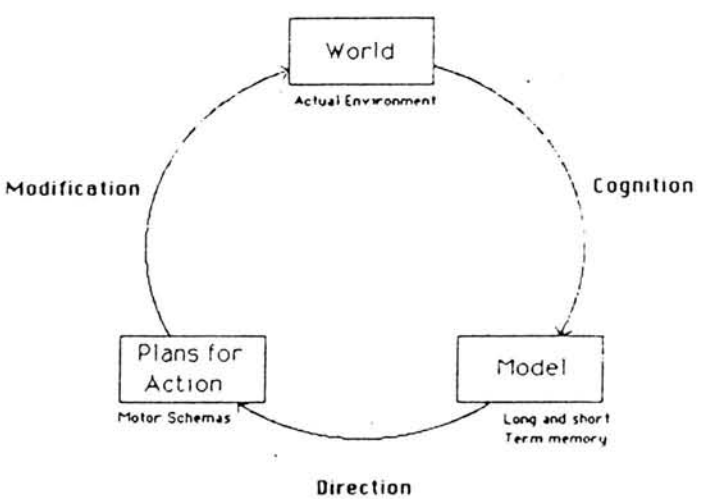


fig. 6.2 Ciclo percepção - acção

A distinção entre os comportamentos voluntários e automáticos que surgem nos humanos pode ser directamente mapeada nas duas escolas de pensamento navegacional prevalentes na robótica móvel: a abordagem hierárquica e a abordagem de controlo

reflexivo distribuido. A arquitectura AuRA explora ambas as abordagens através do uso de um sistema de execução de planos distribuido e de um planeador hierárquico de três níveis para a selecção, parametrização e instanciação de comportamentos motores relevantes.

Um conceito importante nesta abordagem da percepção é o conceito de "affordances", que são as potencialidades de acção inerentes a um objecto ou cena, ou seja, as actividades que podem ocorrer quando um organismo de certo tipo encontra uma entidade de certo tipo. "Affordances" podem ser identificadas com esquemas perceptuais, ou seja, estratégias perceptuais usadas para interactuar com o mundo, satisfazendo as necessidades de uma acção motora específica do organismo.

Para observarmos como a percepção pode ser ligada à acção motora, vejamos o exemplo para um esquema "evitar-obstáculos-estático". Se este esquema está activo (o que acontecerá normalmente no mundo real para que o robot não colida com obstáculos), precisa de algum meio de percepcionar obstáculos no mundo. A fonte algorítmica da percepção do obstáculo não é preocupação do comportamento motor, apenas lhe interessa informação sobre a localização dos obstáculos e, se disponível, uma medida da certeza dessa percepção. É irrelevante para o esquema motor se a informação sobre o obstáculo surge de visão, sensores ultra-sónicos ou qualquer outro sensor. Apenas é necessário escolher um ou mais destes algoritmos e implantá-los no próprio esquema motor.

Outro aspecto importante da percepção orientada à acção é a utilização de "expectations" para melhorar o processamento perceptual. Este aspecto tinha já sido referido a propósito dos sistemas de visão móvel, que, sendo computacionalmente muito pesados, beneficiam enormemente desta característica.

As "expectations" fornecem aos processos perceptuais conhecimento sobre onde procurar um determinado acontecimento perceptual e a forma como aparecerá esse acontecimento. A continuidade temporal e pressupostos de consistência podem permitir a um sistema restringir a possível localização de um objecto de um instante de tempo para o seguinte. O conhecimento interno sobre os mecanismos do mundo real e conhecimento de física "naive" permitem a colocação de restrições às possíveis localizações de um objecto em relação ao observador. Este facto, em conjunto com o conhecimento do movimento próprio do sistema, permite a aplicação de restrições significativas aos recursos computacionais utilizados. Estas estratégias básicas, colectivamente denominadas mecanismos de foco de atenção, permitem a sistemas computacionais lentos e limitados funcionar eficazmente num mundo dinâmico.

A procura de um máximo de autonomia do robot implica encontrar mecanismos pelos quais o robot possa com segurança gerir a sua condição interna, bem como o seu ambiente exterior. Esta característica, denominada controlo homeostático, pode ser definida como a capacidade de manter um estado de equilíbrio relativamente estável entre os diferentes mas interdependentes elementos ou grupos de elementos num organismo ou grupo.

A abordagem proposta por Arkin usa um modelo simplista de um dos paradigmas de controlo usados pelos mamíferos para a homeostasia, o sistema de controlo endócrino. A implementação desta estratégia envolve a geração de uma nova classe de esquemas designados esquemas de sinal. Dentro desta classe temos os esquemas emissores, associados aos sensores internos do robot e os esquemas receptores implantados nos esquemas motores à semelhança dos esquemas perceptuais. Os esquemas receptores podem mesmo ser vistos como esquemas perceptuais com uma visão do mundo interno, em vez das condições externas. A função do esquema receptor é modular dinamicamente os ganhos e parâmetros internos dos esquemas motores à medida que as condições internas do robot se alteram, produzindo alterações suaves e coerentes na resposta motora. O papel do esquema emissor é continuamente difundir globalmente informação respeitante à condição interna actual do robot.

Em [5], para ilustrar os diversos aspectos envolvidos, Arkin apresenta a aplicação desta metodologia ao comportamento específico do "docking".

A arquitectura apresentada por Watanabe *et al.* [7], alicerça-se num sistema baseado em comportamentos (figura 6.3).

A navegação básica é tratada por um módulo separado, que recebe os vários comportamentos a tomar, escolhe o de maior prioridade e emite-o para o executor de movimentos. O executor de movimentos transmite as ordens necessárias ao controlo do robot.

Os agentes que proporcionam os comportamentos estão divididos em três grupos: o nível reflexivo, o nível intencional e o nível adaptativo:

. O nível reflexivo tem como função garantir a segurança mínima do robot. Recebe informação de um conjunto de sensores, informação essa que é redundante, mas que possibilita deste modo uma maior garantia de segurança, dado que os erros de um sensor podem ser minimizados pela presença de informação de outros sensores. Este nível é responsável pela detecção de obstáculos e por evitar colisões.

. O nível intencional tem a seu cargo o movimento do robot, segundo o método mais adequado. Tem como função assistir o robot na navegação até ao objectivo de uma forma eficiente, podendo emitir comportamentos do tipo 'navegue até um sub-objectivo' ou 'active os sensores de modo a escolher um novo sub-objectivo'.

. O nível adaptativo tem como função a recuperação de falhas do nível intencional ou de entradas em 'deadlock'.

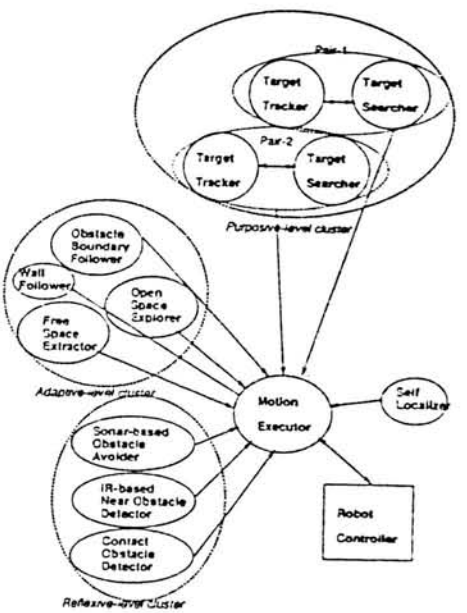


fig. 6.3 Configuração do sistema

A função desempenhada pelo executor de movimentos é criar um plano de acção que controle o "motor drive circuit", através da selecção de um comportamento apropriado, tendo em conta a missão global e a situação actual do robot. Portanto, de acordo com a prioridade de cada nível, e de acordo com a situação actual o executor terá de escolher o comportamento mais adequado. Assim, pode preferir movimentar-se escolhendo o comportamento decidido pelo nível intencional, resolver um erro recorrendo ao nível adaptativo ou observar o ambiente aceitando o comportamento do nível reflexivo.

Um comportamento do nível reflexivo é preferido aos dos outros níveis em situações de 'deadlock', pois a segurança do robot tem nessa altura a maior prioridade. Um comportamento do nível intencional tem preferência quando a procura de um objectivo teve sucesso. Na ocorrência de uma falha provocada por este nível é seleccionado imediatamente o nível adaptativo.

Uma falha local pode ser detectada e corrigida apenas no nível intencional. A redundância de sensores tenta minimizar a ocorrência destas falhas. Uma falha global ocorre quando o comportamento dominante não coincide com os objectivos do planeamento. Pode ocorrer se o robot intersectar um objecto desconhecido que lhe bloqueie continuamente o caminho, ou se o robot ficar preso entre dois obstáculos, não podendo sair. Tais problemas serão tratados pelo nível adaptativo.

Noreils apresenta em [9] uma arquitectura para a cooperação entre robots móveis autónomos.

A cooperação é composta de duas fases:

1. Colaboração, ao nível da divisão de uma tarefa em subtarefas e sua alocação a um conjunto de robots;

2. Coordenação, em que se pretende que os robots coordenem as suas tarefas de modo a completar a tarefa global através do uso de protocolos de coordenação.

Quando uma tarefa não pode ser realizada por um único robot, tendo de ser realizada em conjunto, surge a necessidade de estabelecer uma cooperação entre os diversos robots. Para tal é necessário conseguir a sincronização das acções entre sistemas, que pode ser conseguida através da troca de informações, sinais, etc.. A coordenação baseia-se num conjunto de sinais através dos quais uma entidade é capaz de influenciar o comportamento de outra.

Surge igualmente a necessidade de se incluir uma colaboração entre os diferentes sistemas. A fase de colaboração tenta agrupar os robots em conjuntos que constituirão uma equipa com uma função comum, a de realizar a tarefa original. Se ocorrer uma falha durante a execução esta fase deverá ser reinicializada.

De modo a simplificar a cooperação entre os robots móveis, Noreils propõe a seguinte divisão em quatro etapas;

1. Decomposição. Uma tarefa é dada a um robot, o qual a divide em subtarefas, faz o seu escalonamento e calcula o número de robots e as capacidades de cada robot necessários para o desempenho de cada tarefa.

2. Alocação. O robot tenta formar uma equipa com as capacidades necessárias, tendo em conta restrições de tempo e recursos.

3. Planeamento local. Cada robot já foi afectado à tarefa que terá de executar, sendo responsável pelo planeamento para a sua realização. É necessário ter em conta as restrições de recursos que podem a estar a ser utilizados por outros robots.

4. Execução.

A arquitectura proposta por Noreils (figura 6.4) apresenta uma decomposição em três níveis:

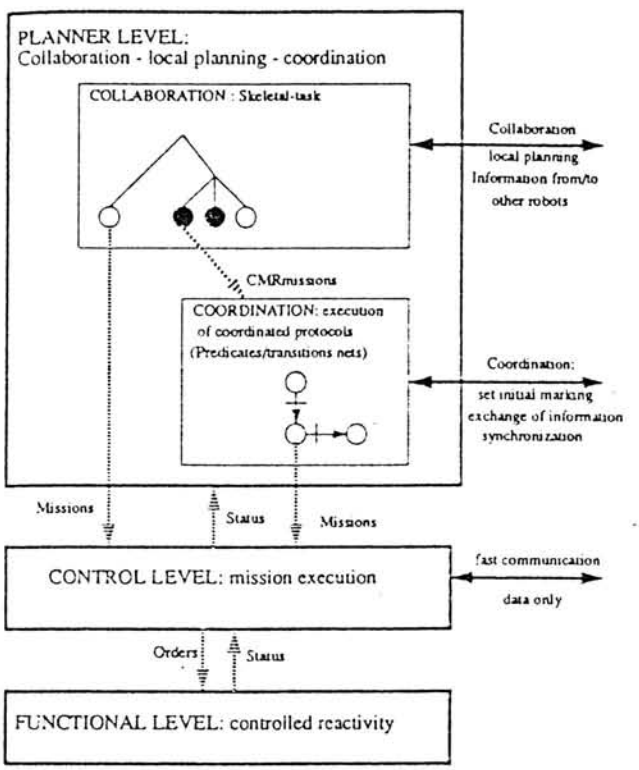


fig. 6.4 Arquitectura proposta por Noreils

- Nível funcional - composto por um conjunto de módulos responsáveis por funções básicas de processamento, detecção de eventos e programação reactiva. Estes módulos realizam a percepção e o processamento de informação, a computação do movimento e das acções e os processos "closed-loop".

- Nível de controlo - é responsável pela gestão das missões. Estas são fornecidas numa linguagem de alto nível, sendo o nível de controlo capaz de as interpretar, executar e recuperar, em caso de erro.

- Nível de planeamento - é responsável pela tomada de decisões e está dividido em dois sistemas:

- O sistema de coordenação que gere protocolos de coordenação, conhecidos por "CMRmissions".

- O sistema de colaboração que gere a colaboração entre robots e o planeamento local. Inclui o modo como se decompõe uma tarefa em subtarefa e a sua alocação a uma rede de robots. O planeador local gera planos compostos por uma sequência de operadores primitivos.

Fornece-se uma tarefa a um robot no nível de planeamento. O sistema de colaboração divide-a em subtarefas e faz a sua alocação a um conjunto de robots, criando assim uma equipa para a execução da tarefa inicial. Cada subtarefa passa a ser um objectivo do sistema de planeamento, que gera um plano local ou volta a dividir a tarefa em subtarefas.

O planeamento local é constituido por primitivas que realizam missões ou "CMRmissions" (protocolos). As missões são passadas ao nível de controlo, e as "CMRmissions" ao sistema de coordenação.

O desenvolvimento de um planeador local nestas condições depara com duas dificuldades:

- Detecção de conflitos. Tal acontece quando dois ou mais robots competem pelo mesmo recurso. Assim, terá de haver interacção entre os diferentes robots aquando da realização dos planeamentos locais.

- Resolver conflitos. Os conflitos podem ser resolvidos quer por negociação quer pela inserção de pontos de sincronização nos planeamentos. Além disso, cada robot tem uma prioridade, e é dada preferência aos robots com maior prioridade de modo a evitar ciclos ou 'deadlocks'.

A robustez do sistema é conseguida do seguinte modo: se um robot não puder completar a sua tarefa, esta pode ser alocada a outro robot, não comprometendo a equipa.

A alocação de tarefas a outros robots é feita através da troca de protocolos. O robot responsável pela alocação envia uma mensagem definindo a tarefa e as capacidades necessárias para a completar. Os robots interessados responderão à chamada, e receberão uma confirmação caso sejam escolhidos.

A cooperação é mantida até todos os robots realizarem as tarefas que lhes foram transmitidas, momento a partir do qual serão libertados para outras tarefas.

Entre as importantes vantagens desta arquitectura estão:

1. A modularidade: o robot pode trabalhar de uma forma autónoma ou dentro de uma equipa;

2. A robustez: se alguns módulos do robot falharem, este é ainda capaz de realizar tarefas importantes;

3. A programabilidade.

BIBLIOGRAFIA

[1] C. Thorpe, M. Herbert, T. Kanade, S. A- Shafer
"Vision and navigation for the Carnegie-Mellon Navlab"
IEEE PAMI, vol. 10, no. 3, pp. 362-373, 1988

[2] M. A. Turk, D. G. Morgenthaler, K. D. Gremban, M. Marra
"VITS - a vision system for autonomous land vehicle navigation"
IEEE PAMI, vol. 10, no. 3, pp. 342-361, 1988

[3] V. Graefe
"Dynamic vision systems dor autonomous mobile robots"
Proc. IEEE/RSJ IROS'89, pp. 12-23, 1989

[4] R. A. Brooks
"A robust layered control system for a mobile robot"
IEEE Journal of Robotics and Automation, vol. 2, no. 1, pp. 14-23, 1986

[5] R. C. Arkin
"The impact of cybernetics on the design of a mobile robot system: a case study"
IEEE Transactions on Systems, Man and Cybernetics,
vol. 20, no. 6, Nov/Dec 1990, pp. 1245-1257

[6] F. R. Noreils, R. Prajoux
"From planning to execution monitoring control for indoor mobile robots"
IEEE Robotics and Automation, pp. 1652-1658, 1991

[7] M . Watanabe, K. Onoguchi, I. Kweon, Y. Kuno
"Architecture of behavior-based mobile robot in dynamic environment"
Proc. 1992 IEEE Int. Conf. Robotics and Automation, vol. 3, pp. 2711-2718

[8] J. H. Connell
"SSS: a hybrid architecture applied to robot navigation"
Proc. 1992 IEEE Int. Conf. Robotics and Automation, vol. 3, pp. 2719-2724

[9] F. R. Noreils
"An architecture for cooperative and autonomous mobile robots"
Proc. 1992 IEEE Int. Conf. Robotics and Automation, vol. 3, pp. 2703-2710

ANEXOS

ANEXO A - USO DE FUZZY INFERENCING CHIPS

Baseado em:

F. G. Pin, H. Watanabe, J. Symon, and R. S. Pattay

"Using custom-designed VLSI fuzzy inferencing chips for the autonomous navigation of a mobile robot"

Proc. 1992 IEEE/RSJ Int. Conf. on Intell. Robots and Systems, vol. 2, pp. 790-795

A construção de módulos gestores em robots móveis autónomos que têm de funcionar em ambientes desconhecidos ou dinâmicos é um dos maiores desafios no campo do planeamento e controlo do movimento de robots. Isto deve-se ao facto de o sistema ter de cooperar com as inúmeras imprecisões e incertezas causadas tipicamente por: (1) erros na leitura dos sensores que levam a incorrecções na posição do robot e na descrição do ambiente, (2) imprecisões ou falta de conhecimento sobre o sistema e (3) aproximações e imprecisões nos esquemas de processamento de informação, como arredondamentos, truncagens e discretização, que são usados para gerar decisões ou controlar os sinais de saída.

O raciocínio qualitativo refere-se a um conjunto de metodologias que têm vindo a ser desenvolvidas para fornecer metodologias de gestão de decisões em sistemas onde as incertezas não podem ser eliminadas. Um dos factores que causaram a não expansão destas metodologias foi o facto de não existir hardware que permitisse o processamento e a inferência directamente em termos de variáveis 'fuzzy'.

No entanto, foram agora produzidos em cooperação com o MCNC, Inc., chips VLSI que podem ser programados para lidar com variáveis qualitativas e regras. Neste circuito implementa-se a teoria dos conjuntos Fuzzy, no qual as funções podem tomar qualquer valor no intervalo [0, 1] em vez de apenas os valores discretos {0, 1}.

Tal irá permitir o tratamento das incorrecções e imprecisões das entradas no sistema, como atrás referimos. Na vida real não podemos lidar com valores discretos, mas sim com intervalos de valores. É neste ponto que tem especial relevo a teoria dos conjuntos Fuzzy, que se baseiam precisamente nesta variação em torno de um ponto médio.

Esta teoria formula as operações a realizar entre variáveis fuzzy, como sejam as operações básicas, somas, multiplicações, ou outras compostas destas. Deste modo é possível realizar os mesmos cálculos realizados anteriormente para variáveis discretas e obter as saídas necessárias, como sejam a direcção e velocidade que o robot deve possuir.

Os sensores de alcance adaptam-se perfeitamente às variáveis fuzzy visto que nos dão um intervalo de valores entre os quais se situa um objecto.

Estamos assim aptos a tratar expressões como muito perto, perto, afastado, longe e muito longe, e não apenas os valores limites. É também possível tratar problemas como desacelerações e aproximações de uma forma mais suave do que era possível anteriormente.

Está assim aberto um grande campo de aplicações que permite adaptar o movimento dos objectos e o tratamento da informação a condições mais reais.