# Fixação Visual:
# Uma Abordagem Computacional

Carlos Alberto Esteves Rodrigues Paredes

7 de Novembro de 1996

# Fixação Visual:
# Uma Abordagem Computacional

Carlos Alberto Esteves Rodrigues Paredes

7 de Novembro de 1996

# Agradecimentos

O meu primeiro agradecimento é para o **Dr. Jorge Dias**, muito particularmente pela sua paciência e pelo empenho posto na tarefa de me orientar. Para o melhor e para o pior, esta tese é tanto minha como dele.

Obrigado também ao Dr. Helder Araújo pela disponibilidade para rever o trabalho.

Igualmente imprescindível foi o apoio monetário concedido pela Junta Nacional de Investigação Científica e Tecnológica (JNICT), através do programa PRAXIS XXI.

Ao longo de dois anos e de muitos altos e baixos, várias foram as pessoas que contribuíram para esta tese, de uma forma ou de outra. Entre elas, quero destacar:

- O Luis, por nunca se ter negado à exigente tarefa de *neuro-cirurgião.*
- O Júlio, pelo esforço literário.
- A malta do laboratório, em particular os viciados na *boa conselheira.*

Adicionalmente, os meus mais especiais agradecimentos são devidos:

- Aos meus pais. Entre muitas outras coisas, por me deixarem andar na *boa vida* até às tantas! ☺
- À Rita, por partilhar os sucessos e escutar os lamentos, etc... ☺

Coimbra, 18 de Setembro de 1996,

Carlos Alberto Cardeira

# Índice

# Parte I

# Dissertação

# Capítulo 1

# Introdução

A visão é, sem dúvida, o mais importante dos cinco sentidos humanos. A sua versatilidade é tal que se torna difícil imaginar uma situação em que a informação visual seja totalmente inútil. Tal justifica, em grande medida, o enorme interesse que desperta actualmente a visão por computador, partilhado por investigadores com as mais diversas formações académicas. Aliás, são inúmeros os trabalhos que surgem motivados por descobertas ou princípios puramente biológicos, quer nos humanos [LUD94], quer em outros animais [LET59]. A complexidade associada à maioria dos processos visuais é outra das fontes de interesse para quem sobre eles se debruça. De facto, a mais simples das tarefas desempenhadas pelos ser vivos pode, quando transposta para o universo das máquinas, tornar-se no mais complicado dos quebra-cabeças. O trabalho que aqui se descreve surgiu da necessidade de melhorar o comportamento de um sistema de seguimento, resultante da integração de uma plataforma móvel com um sistema de visão activa. Em termos gerais, o sistema de seguimento tem como objectivo perseguir um alvo em movimento. Vamos então começar, de forma muito sucinta, pela análise do funcionamento do sistema de seguimento.

## 1.1 O Passado

O sistema de seguimento nasceu de um projecto de licenciatura realizado no ano lectivo de 1993/94 [DIA95-2], integrado no projecto VARMA (Visão Activa para Robótica Móvel Autónoma), financiado pela JNICT (Junta Nacional de Investigação Científica e Tecnológica) e designado oficialmente por VARMA I. É composto por uma estrutura mecânica, sobre a qual foram colocadas duas câmaras de vídeo, que simula dentro de certos limites alguns dos movimentos de uma cabeça humana. Essa estrutura, a que chamámos sistema de visão activa, é suportada por uma plataforma móvel que se convencionou designar como o corpo do sistema (ver a figura 1.1). Globalmente, o VARMA I executa as seguintes acções:

1. Aguardar até que um alvo se movimente dentro do campo de visão das câmaras.

2. Movimentar o sistema de visão activa de forma a que o alvo se mantenha dentro do campo de visão.

3. Deslocar a plataforma móvel de forma a manter constante a distância ao alvo.

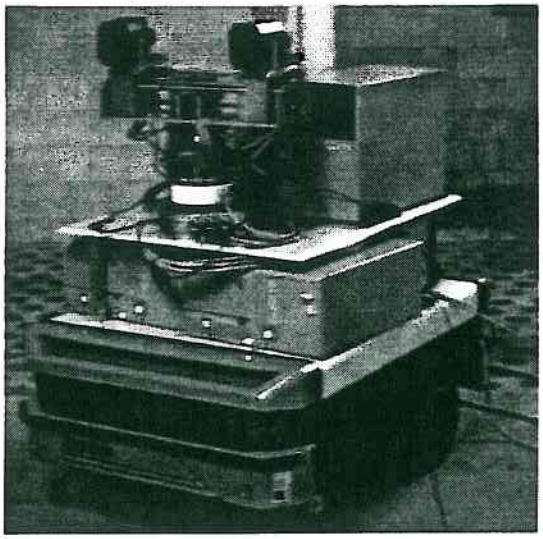


Figura 1.1: Imagem do sistema de seguimento. Para além da plataforma móvel e do sistema de visão activa, são visíveis os três computadores pessoais responsáveis pelo processamento. Atrás do sistema de visão activa, é parcialmente visível a caixa onde estão instalados os módulos de potência dos motores de passo.

Estas acções podem resumir-se numa única tarefa que, em termos humanos, se designaria por "seguir um alvo". No caso do VARMA I, os alvos seleccionados são, tendencialmente, pessoas.

Para processamento, o VARMA I dispõe de três computadores pessoais com CPUs 80486DX2 da Intel a 66MHz, estando dois deles destinados às duas placas de captação de imagem (uma para cada câmara), ambas sem capacidade de cálculo própria e com memórias relativamente lentas. Globalmente, a capacidade de cálculo revelou-se insuficiente, restringindo de forma apreciável a liberdade de acção face aos objectivos traçados. Os principais afectados pelo baixo poder de cálculo são os processos visuais que, obviamente, condicionam fortemente o funcionamento global do sistema, mas não o impedindo de funcionar em tempo real. Embora a velocidade seja um entrave considerável, a análise crítica do sistema não parou com a sugestão de adquirir *hardware* dedicado para o processamento de imagem. Ponderou-se igualmente a hipótese de melhorar e/ou substituir os algoritmos computacionais ligados à simulação dos processos visuais, bem como as técnicas de controlo da estrutura mecânica. Desta reflexão nasceu esta tese, em cujos objectivos nos vamos agora debruçar.

## 1.2 Objectivos

Com vista à concentração de esforços, optou-se por limitar o trabalho ao melhoramento dos processos visuais e do controlo do sistema de visão activa. Por outras palavras, estudaram-se os aspectos relacionados com a detecção e localização dos alvos e com a sua interligação com o sistema de visão activa, preterindo as questões associadas à plataforma móvel do VARMA I. A terceira das acções enumeradas atrás deixou assim de ser contemplada. Adicionalmente, foi decidido que os processos visuais desenvolvidos deveriam enquadrar-se numa teoria mais geral de fixação visual, capaz de reagir e/ou ser integrada

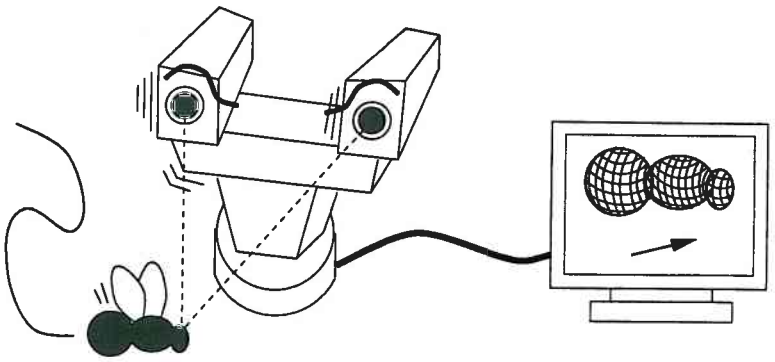


Figura 1.2: O objectivo é desenvolver um processo de fixação visual e executá-lo com base no sistema de visão activa. Pretende-se que o processo possa ser aplicável a tarefas como a recuperação da estrutura tridimensional e a detecção do movimento dos alvos.

em projectos com outro tipo de solicitações, como sejam a recuperação de estrutura, a vigilância, a detecção de movimento ou a extracção de contornos. A figura 1.2 é uma caricatura desse objectivo, representando o sistema a detectar o movimento de um alvo ao mesmo tempo que recupera a sua estrutura tridimensional. O aspecto crucial a reter é que, tal como sugere a figura, o sistema terá de funcionar em tempo real, reagindo em tempo útil às solicitações externas. A construção de sistemas capazes de reagir em tempo real é um dos objectivos mais vezes almejado, e também de mais difícil concretização, para os grupos de investigação que trabalham em visão por computador. Não é assim de estranhar que aqueles que o conseguem se tornem pontos de referência e comparação.

## 1.3 Trabalhos na Área

Nos últimos anos temos assistido à proliferação de sistemas de visão activa. Entre eles, podemos destacar o trabalho de Clark e Ferrier em Harvard [CLA88][CLA92][FER92], da equipa de Brown em Rochester [BRO88] [BRO92][BAL92], de Murray e outros, em Oxford [SHA93][MCL95][BRA94] e do grupo CVAP do KTH de Estocolmo [PAH92][EKL95]. Todos estes grupos desenvolveram e desenvolvem trabalhos de investigação em torno dos seus sistemas de visão activa, explorando áreas como a calibração, a recuperação de estrutura, a detecção de movimento, a vigilância, o seguimento, o cálculo de mapas de profundidade e a navegação, entre outras. Os processos visuais escolhidos para a abordagem a cada um destes temas variam consideravelmente, não só com os temas em si, mas também com os objectivos traçados, com os investigadores envolvidos e com o *hardware* disponível, entre outros factores. Muitos destes processos encerram problemas complicados, para os quais foram e continuarão a ser propostos algoritmos diversificados, com resultados diversos. De um modo geral, poucos são os métodos que se destacam, resumindo-se habitualmente a escolha a uma pesagem entre o desempenho revelado face ao problema e as exigências computacionais associadas à sua implementação.

No entanto, em alguns casos, há métodos que se evidenciaram na resolução de determinado tipo de problemas. A correlação, por exemplo, é uma das abordagens mais comuns ao problema do seguimento de alvos, ainda que podendo ser encarada de diferentes pontos de vista e a diversos níveis de complexidade [PAH93] [DICE92] [DICJ92]. Outra das aplicações da correlação é a disparidade binocular, tal como descreve D. Fleet [FLE94]

ao utilizar técnicas que ponderam a fase no cálculo da correlação [KUG75]. O interesse pela correlação deve-se à facilidade com que permite efectuar a correspondência de áreas de imagens. De facto, a generalidade dos sistemas de visão activa são multi-oculares (tipicamente binoculares), o que leva os investigadores a tentar desenvolver técnicas para associar características nas diferentes imagens obtidas por cada um dos sensores ópticos disponíveis. É claro, a correlação não é a única técnica de correspondência existente. Igualmente comum é a extracção de pontos de interesse na imagem (contornos, arestas, cantos, etc.) que são depois utilizados para efectuar o seguimento [KASS87] [CLA96], contrastando assim com o seguimento de áreas utilizado na correlação.

No sistema VARMA I, o controlo do sistema de visão activa e a navegação da plataforma móvel foram desenvolvidos com base no seguimento de alvos através da técnica da correlação. Daí que o sistema VARMA I seja controlado com informação posicional. Não é esse o caso do sistema desenvolvido em Oxford, por exemplo, que utiliza igualmente a informação sobre o movimento. O fluxo óptico é a técnica mais comum para a recuperação de informação sobre o movimento que ocorre em sequências de imagens [MUR95], tendo sido igualmente explorada a vertente do fluxo óptico normal [ALO94] [HUA94]. No nosso caso, o principal entrave à utilização directa de informação sobre o movimento são os controladores dos motores de passo utilizados no sistema, porque apenas permitem o controlo posicional. Assim, mesmo tirando partido da informação do movimento, o controlo final terá sempre de ser posicional.

Soluções como a correlação e o fluxo óptico representam óptimos pontos de partida para um trabalho sobre fixação visual. No entanto, estão longe de ser as únicas abordagens possíveis. Como exemplo, temos o trabalho realizado por Taalebinezhaad no MIT [TAA92], que pretende recuperar o movimento e a estrutura com base na fixação, mas explicitamente preterindo a correlação de imagens e o fluxo óptico em favor de *métodos directos*, investigados com o objectivo de extrair o movimento e a estrutura directamente da intensidade luminosa das imagens. A recuperação de estrutura é, de facto, uma das áreas de maior aplicabilidade da fixação. Há alguns anos, Ballard antecipou que a utilização do ponto de fixação como origem de um sistema de coordenadas facilitaria a recuperação da estrutura do alvo fixado [BAL91]. Posteriormente, Fermuller e Aloimonos demonstraram como tal poderia ser feito partindo de um ponto de fixação real [FER93]. Utilizando estes pressupostos, Fairley, Reid e Murray desenvolveram um sistema de seguimento através da recuperação da estrutura afim [FAI95], demonstrando a versatilidade das técnicas de fixação visual. É esta a principal conclusão a tirar desta panorâmica dos trabalhos mais recentes com sistemas de visão activa em geral e com a fixação visual em particular: nenhuma técnica é específica de uma determinada área e a experiência mostra que há vantagens na integração e interligação dos diferentes algoritmos.

## 1.4 Estrutura do Texto

Ao longo deste trabalho, foram analisados e testados inúmeros métodos, não condenando à partida a experiência adquirida com o VARMA I. Daí resultou uma imprescindível selecção, não só através da crítica das técnicas em si, mas principalmente em função da sua aplicabilidade às nossas condições específicas. Como foi já referido, um objectivo fundamental do trabalho é a sua realização prática e em tempo real. Por outras palavras, não se pretende obter métodos teoricamente muito elaborados, mas cuja realização prática está completamente fora da capacidade de processamento disponível. Essa capacidade

resume-se a um dos computadores pessoais do VARMA I, responsável pela sincronização e controlo de todos os módulos do sistema, ficando a captação e o processamento de imagem a cargo de dois processadores digitais de sinal TMS320C40 da *Texas Instruments*. Os TMS320C40 funcionam a 40MHz e são capazes de desempenhos na ordem dos 50 MFLOP e 275 MOPS. É utilizada uma única placa de aquisição de imagem, à qual são ligadas ambas as câmaras, não sendo possível captar imagens de ambas em simultâneo. No próximo capítulo iremos abordar o *hardware* disponível com mais pormenor, deixando-se a sua especificação completa para o apêndice A.

Estruturalmente, dedicaremos o capítulo que se segue à fixação visual, definindo e esquematizando o conceito com base numa análise puramente intuitiva das características biológicas que nos são reveladas pelo comportamento humano. Importa realçar que o objectivo deste trabalho não é a reprodução do sistema visual humano, mas sim o de inferir uma metodologia de abordagem aos problemas, suficientemente sistemática para poder ser realizada em termos computacionais. Por outro lado, apresenta também flexibilidade necessária para ser aplicável a problemas e situações com características distintas.

No capítulo 3 concentramo-nos em questões relacionadas com a implementação prática do processo de fixação visual desenvolvido. Veremos diversos métodos de análise e processamento de imagem, discutindo também o seu desempenho e aplicabilidade aos nossos objectivos. Munidos dos algoritmos, equacionaremos no capítulo 4 o processo de fixação visual numa perspectiva propositadamente indefinida e generalista no que respeita a possíveis aplicações. No entanto, tal é feito de forma a que, em face de um problema concreto, seja possível extrair um algoritmo com viabilidade prática para funcionar em tempo real.

Finalmente, nos capítulos 5 e 6 veremos duas aplicações distintas do processo: o seguimento de alvos e a recuperação de estrutura. No primeiro exemplo espera-se que o sistema reaja em tempo real a um alvo que se movimente no seu campo de visão. No segundo pretende-se recuperar a informação tridimensional do alvo. Em ambos os casos, constataremos a aplicabilidade do método e as adaptações que é necessário fazer, em face dos diferentes objectivos.

# Capítulo 2

# Fixação Visual

## 2.1 Introdução

Procurando no dicionário definições para "fixar", somos confrontados com expressões do tipo: "não esquecer", "segurar" e "tornar-se estável". É curioso como estes sinónimos, aparentemente aplicáveis apenas em situações distintas, descrevem vários passos da fixação visual, como veremos ao longo deste capítulo.

Para compreendermos o que é, afinal, a fixação visual, vamos começar por constatar que, ao ler este texto, estamos a utilizá-la. Cada nova palavra interpretada, implica que ambos os olhos se orientem na sua direcção, percorrendo-a da esquerda para a direita e concentrando-se, a cada momento, nas mesmas letras. Poderemos assim definir a fixação visual como o acto de concentrar a atenção de ambos os olhos num mesmo alvo. Do ponto de vista humano, orientar os olhos na direcção de alvos distintos, para além de difícil, é potencialmente nauseante, já que o cérebro não é capaz de interpretar simultaneamente cenas distintas. A fixação é, de facto, um processo tão natural que raramente nos damos conta de que o fazemos.

Indo um pouco mais longe, rapidamente descobrimos outras razões para fixarmos. Uma delas é a percepção da distância. Uma vez que os olhos se encontram separados, a perspectiva que têm de uma mesma cena é forçosamente diferente. Supondo que ambos os olhos se encontram concentrados num determinado alvo, podemos esquematizar a diferença de orientação existente entre eles da forma que vemos na figura 2.1. Analisando a figura, não é difícil imaginar um processo de triangulação através do qual a distância ao alvo possa ser detectada com precisão. Do ponto de vista meramente biológico, concluir se este é ou não um dos processos utilizados pelos humanos não é uma questão trivial. No entanto, a criação de um modelo baseado no processo da triangulação ajusta-se às características do *hardware* de que dispomos, mais precisamente ao sistema de visão activa.

Outra característica importante, associada à fixação, é a localização na retina das imagens resultantes da projecção do alvo. Quando os olhos se encontram fixos num alvo, a sua imagem é projectada no centro de ambas as retinas, isto é, na fóvea. A fóvea é a zona da retina onde a distribuição de cones (células fotorreceptoras) é mais densa, correspondendo à área onde as imagens são captadas com maior definição. A importância desta característica reflecte-se a vários níveis: para começar, ao fixarmos, estamos a posicionar a projecção do alvo na zona da retina onde podemos extrair a maior quantidade de informação possível sobre ele, o que deverá favorecer os resultados da sua análise; por outro lado, ao centrar o alvo na retina, minimizamos a possibilidade de que algum acontecimento na sua vizinhança física nos passe despercebido. Adicionalmente, a

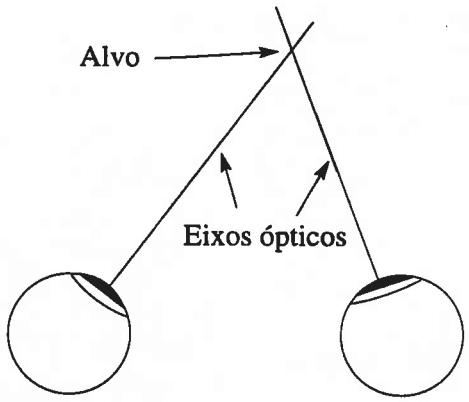



Figura 2.1: Modelo de percepção da distância. Cada olho sabe apenas que o alvo se encontra sobre o seu eixo óptico. Utilizando a informação de ambos os olhos, a localização do alvo revela-se um problema trivial, bastando para tal fazer a intersecção dos dois eixos, ou seja, a realização da fixação visual.

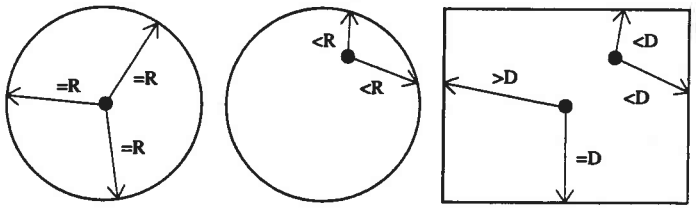



Figura 2.2: Supondo um campo visual circular, se partirmos da posição central, a distância a percorrer para atingir a periferia é sempre igual a $R$ (figura à esquerda). Partindo de qualquer outra posição, existem sempre várias direcções segundo as quais essa distância é inferior (figura ao centro). Esta característica pode ser generalizada para qualquer outro tipo de geometria de sensores que apresentem simetrias, incluindo o habitual formato rectangular das imagens sintéticas (figura à direita).

fixação permite-nos reduzir a probabilidade de o alvo abandonar o nosso campo de visão sem ser detectado. Isto porque, ao escolhermos o centro geométrico do campo de visão, estamos a maximizar a distância que nos separa de qualquer ponto da periferia (ver a figura 2.2). Em resumo, fixando conseguimos melhorar as condições de interpretação do alvo, facilitando a sua localização no caso de não permanecer estático e salvaguardando a detecção de ocorrências importantes (ou não) nas zonas da cena que não estão a ser analisadas em pormenor.

Mesmo que ambos os olhos estejam fixos num mesmo alvo, as suas projecções nas fóveas não são exactamente iguais. Na figura 2.3 podemos ver um par de imagens obtidas com as câmaras fixas no ponto central. Observando as imagens, é possível detectar diversas diferenças de perspectiva, algumas mais óbvias do que outras, mas todas resultantes da diferente localização das câmaras e das várias distâncias a que se encontram os objectos representados. A estas diferenças chamamos disparidade. A análise da profundidade estereoscópica com base na estimação da disparidade é um tópico clássico da visão por computador, tendo os trabalhos mais recentes concentrado a sua análise em técnicas baseadas no cálculo da fase [FLE94] [COZ97]. De um modo geral, os algoritmos desenvolvidos têm como objectivo a construção do mapa de profundidades associado a duas

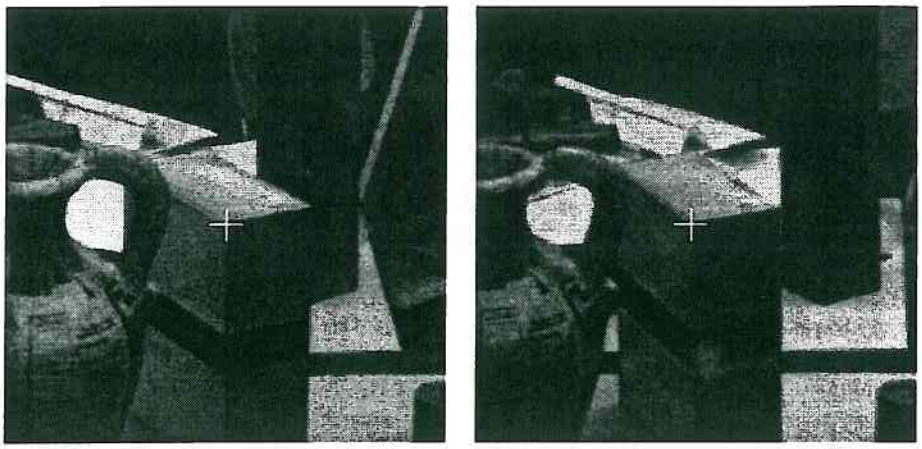


Figura 2.3: Par de imagens estéreo (esquerda e direita) obtidas com as câmaras fixas no ponto assinalado (centro da imagem).

imagens estéreo, como as que podemos ver na figura 2.3. Tal é conseguido percorrendo as imagens e associando, de forma rigorosa e sistemática, primeiro pontos, depois linhas, e assim sucessivamente até à construção de todo o mapa tridimensional. A reacção em tempo real raramente é um dos objectivos ponderados. No nosso caso, o rigor e o sistematismo são menos importantes do que a necessidade de detectar rapidamente um alvo em duas imagens distintas. Daí que os algoritmos de associação de características, embora úteis, terão de ser convenientemente enquadrados.

Até agora, debruçámo-nos essencialmente sobre os pormenores relacionados com a projecção dos alvos em cada uma das fóveas, a sua localização e as diferentes imagens resultantes. No entanto, a fixação não se resume à concentração de atenções num ponto fixo e determinado [YAR67], até porque, durante a fixação num alvo, mesmo que imóvel, ocorrem diversos movimentos dos olhos, alguns dos quais não são percepcionados pelo observador. Carpenter [CAR88] faz a seguinte classificação dos movimentos dos olhos:

- Movimentos rápidos de mudança da orientação: sacadas, micro-sacadas e fase rápida de nistagmo (oscilações do globo ocular em torno do seu eixo horizontal ou vertical).
- Movimentos lentos de mudança da orientação: perseguição lenta e vergência.
- Movimentos de manutenção da orientação: vestibulares e opto-cinéticos.
- Movimentos de fixação: tremores e desvios.

Os movimentos de sacada, geralmente de grandes dimensões, ocorrem quando os olhos são redireccionados para um local diferente da cena. A sua execução pode dever-se a inúmeras razões, desde a percepção de acontecimentos noutra zona da cena observada até ao simples desinteresse do observador. Quanto às micro-sacadas, interrompem usualmente movimentos de perseguição lenta. Isto porque, durante a perseguição lenta, a projecção do alvo na fóvea é mantida estável. No entanto, como a estabilidade não é total, os desvios conduzem a um erro que, ao atingir um certo limite, obriga à sua correcção. Daí as micro-sacadas. Adicionalmente, tanto a cabeça como o corpo poderão ser movimentados para ajudar à manutenção da estabilidade. Esta sequência de acontecimentos descreve aquilo que entendemos por fixação visual, e que se resume na seguinte frase: para haver

fixação, é necessário que seja seleccionado um alvo numa cena, que os olhos se orientem para ele e que a sua projecção seja mantida estável em ambas as fóveas. Excluindo os movimentos de vergência, que estão associados ao ajuste da zona focada em torno do ponto de fixação de disparidade nula, os restantes movimentos pautam-se por não serem directamente resultado da realimentação visual, o que é diferente de dizer que não participam activamente no processo de fixação. De um modo geral, os investigadores que trabalham com sistemas de visão activa concentram-se particularmente em algoritmos para executar os movimentos de sacada, micro-sacada, vergência e perseguição.

## 2.2 O Problema

Brown, Coombs, e Soong [BRO92] encontram-se entre os primeiros a relatar um sistema de visão activa com desempenho em tempo real (15Hz), baseando-se na extracção dos alvos através de filtragem de disparidades nulas. Embora com alguns problemas em cenas com alvos a diferentes disparidades, asseguram uma integração correcta da vergência e da perseguição, já que o alvo que origina a vergência é sempre aquele que será perseguido. Murray e outros [MUR93] desenvolveram um sistema de visão activa monocular que consegue executar a perseguição em tempo real (25Hz). Baseiam-se na extracção e seguimento de cantos, conseguindo detectar alvos com movimentos independentes, o que lhes permite executar a sacada inicial para o local e com a velocidade inicial de perseguição correctas. A detecção de movimentos assume que as deslocações do fundo são unicamente devidas à rotação da câmara, e que essa rotação é conhecida.

Do laboratório CVAP do Instituto Real de Tecnologia (KTH) de Estocolmo surge o relato de um sistema baseado na fixação visual, em tempo real (25Hz), que possui dois estados principais: a mudança de orientação e a perseguição [UHL96]. O primeiro é atingido a pedido, provocando uma resposta rápida do sistema que reage orientando as câmaras na direcção pretendida. Depois da sacada rápida inicial, seguem-se sacadas correctivas até que o ponto escolhido esteja adequadamente centrado. Uma vez centrado o ponto, inicia-se a fase de perseguição. Este tipo de abordagem pressupõe que o sistema consegue efectivamente centrar o alvo nas imagens, o que implica precisão na orientação e, se o alvo estiver em movimento, velocidade de reacção suficiente para manter o alvo centrado.

Uma vez que se baseiam igualmente na fixação visual, vamos tomar como referência o grupo do CVAP, e o seu sistema de visão activa, que apresenta um total de sete graus de liberdade mecânicos e 6 graus de liberdade ópticos [EKL95]. O sistema permite rodar e inclinar ambas as câmaras e todo o sistema de visão activa de forma independente, já que tanto as câmaras como o sistema têm unidades próprias de *pan* e *tilt*. Os graus de liberdade mecânicos são conseguidos com base em motores de passo, com resoluções de 50000 passos por volta, velocidades máximas na ordem dos 2 segundos por revolução e erros máximos de $0,05°$. Para processamento, o sistema dispõe de uma rede de 11 transputers, 2 dos quais com placas de digitalização integradas, mais uma placa com dois transputers para controlo e interface.

O nosso sistema de visão activa possui um total de 5 graus de liberdade mecânicos, diferenciando-se da cabeça do CVAP por não poder executar movimentos de *tilt* com as câmaras [DIA93]. Os motores de passo que conferem os graus de liberdade não permitem o controlo *on-the-fly* nem em velocidade. Os motores das câmaras têm velocidades típicas de 10 segundos por revolução com resoluções de 2000 passos por volta, não detectando

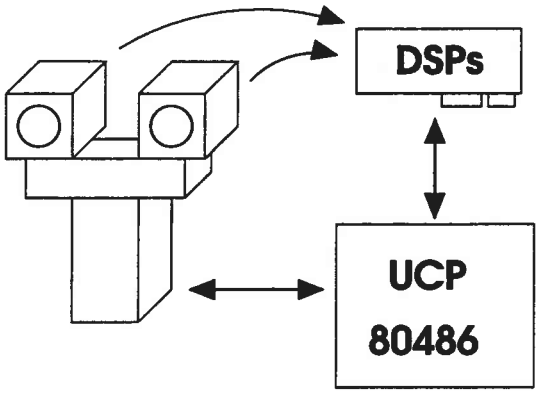



Figura 2.4: Módulos de *hardware*.

os descodificadores respectivos mais do que 1000 passos por volta. Os motores associados à rotação e inclinação do sistema têm resoluções de 36000 e 7200 passos por volta, com velocidades típicas de 99 e 34 segundos por revolução, respectivamente.

Em complemento ao sistema de visão activa, dispomos de dois módulos de processamento (ver a figura 2.4), compostos por dois processadores digitais de sinal (DSP) e uma unidade central de processamento (UCP). A sincronização dos vários módulos é efectuada pela UCP, baseada numa CPU 80486DX2 da Intel, funcionando a 66MHz. Os DSPs são dois TMS320C40 da *Texas Instruments*, sendo responsáveis unicamente pela tarefa de captar e processar as imagens. Adicionalmente, existe uma placa de aquisição de imagens, controlada por um dos DSPs. No apêndice A descreve-se em pormenor estas e outras características do *hardware*.

Embora baseada nos mesmos princípios gerais utilizados pelo CVAP, a nossa abordagem tem em consideração as limitações mecânicas e de processamento existentes. Em primeiro lugar, não iremos impor a condição de que o alvo deva estar centrado, ou mesmo numa vizinhança do centro, para que o processo se mantenha num determinado estado: será suficiente que se encontre no campo de visão das câmaras. Em segundo lugar, não distinguiremos os movimentos entre sacadas, sacadas correctivas e perseguição: todos os movimentos têm como finalidade centrar o alvo mas câmaras. Por último, teremos algum cuidado a seleccionar o alvo a fixar: um alvo facilmente identificável aumenta a probabilidade de vir a ser localizado no campo de visão das câmaras a cada iteração do processo. O principal objectivo destas considerações é conduzir a um processo que seja realizável no sistema de visão activa existente e funcione em tempo real.

## 2.3 O Processo

Na sequência da discussão anterior, e para concluir, vamos agora esquematizar o processo através do qual se simulará artificialmente a fixação visual. A figura 2.5 representa os três estados fundamentais desse processo: *Espera*, *Selecção* e *Fixação*. Antes de nos ocuparmos de cada um deles, importa comentar o facto de, no diagrama do processo de fixação, haver um estado com esse nome. Se, do ponto de vista biológico, a fixação é tida como uma reacção natural dos olhos, em termos computacionais, atingir uma situação de fixação não é uma tarefa célere nem simples. Daí que, para efeitos do processo de fixação visual, o sistema só será considerado como estando a fixar após serem reunidas determinadas

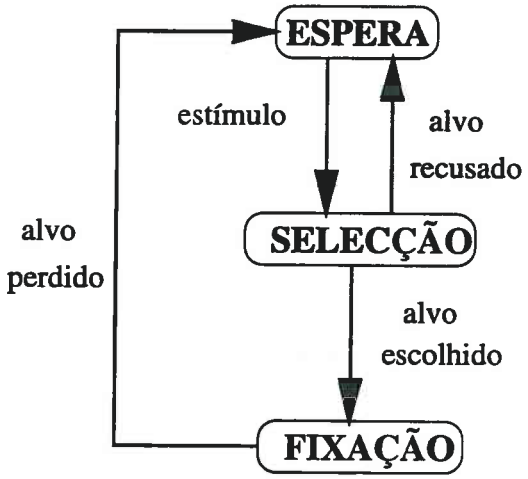



Figura 2.5: Diagrama de estados definido para a simulação do processo de fixação visual.

condições. Por outras palavras, os estados de *Espera* e *Selecção* não são mais do que fases de configuração do algoritmo de fixação visual, com o objectivo único de conduzir o processo ao estado de *Fixação* o mais rapidamente possível.

O processo de fixação inicia-se no estado de *Espera*. Neste estado, o observador não está concentrado em nenhum alvo específico, o que em termos computacionais significa que a fixação não está assegurada. Para evoluir deste estado, o observador necessita de ser visualmente estimulado. Os estímulos podem ser a movimentação de um alvo, ou a alteração das suas características (cor, forma, dimensão), entre outras. Generalizando, um estímulo é qualquer coisa que provoque uma alteração da cena. No nosso caso, vamos obrigar a que as alterações sejam detectadas por ambas as câmaras. Isto permitirá que, após a análise das mudanças ocorridas, se possam comparar os alvos que as provocaram e concluir se se trata ou não de um alvo único.

Surgido o estímulo, o processo passa ao estado de *Selecção*. Neste estado, como o nome indica, o observador terá de decidir se o alvo que gerou o estímulo lhe interessa. Por outras palavras, é necessário avaliar se deve ou não evoluir até ao estado de *Fixação*. Um observador humano pode tomar esta decisão ponderando inúmeros factores, com maior ou menor grau de subjectividade, como sejam o perigo e o interesse. No entanto, em termos computacionais, a decisão resume-se à apreciação de factores como a amplitude do movimento detectado, a dimensão do alvo e as características que facilitam a sua detecção em imagens futuras, por exemplo. Como seria de esperar, o observador é livre de rejeitar o alvo se o considerar desinteressante, o que no nosso caso corresponderá, usualmente, à constatação de haver uma fraca probabilidade de conseguir fixar o alvo em causa com algum sucesso. Em caso de recusa, o processo volta ao estado de *Espera*.

Por último, chegamos ao estado de *Fixação*. É importante recordar que, para aqui chegarmos, houve um alvo que provocou um estímulo em ambas as câmaras e passou nos critérios de selecção, levando-nos a decidir fixá-lo. Apresentam-se-nos agora duas tarefas fundamentais: o que fazer com o alvo fixado e como manter o estado de fixação. Ambas as tarefas dependem da aplicação em causa, estando de um modo geral intimamente ligadas. Por exemplo, se o objectivo for seguir o alvo, as tarefas resumem-se à resolução de um único problema. Deixamos por isso as considerações sobre essas tarefas para os capítulos 5 e 6, quando abordarmos os exemplos de aplicação do processo de fixação visual.

Do ponto de vista humano, o abandono do estado de *Fixação* pode ficar a dever-se a diversas razões, entre as quais figura o desinteresse. No nosso caso, há uma única razão para que se deixe o estado de *Fixação*, que é a não localização do alvo no campo de visão das câmaras. Note-se que este situação não tem de ser, forçosamente, o resultado de uma fuga, podendo dever-se simplesmente há incapacidade do algoritmo em reconhecer o alvo nas imagens. Quando se deixa escapar o alvo, volta-se ao estado de *Espera* e aguarda-se um novo estímulo.

## 2.4 Distribuição de Tarefas

Uma vez que o nosso sistema é composto por vários módulos distintos, terá de haver uma política de sincronismo entre eles, cuja responsabilidade é da unidade central de processamento. O facto de os módulos serem independentes facilita, em grande medida, a sua sincronização, permitindo à unidade central uma maior disponibilização de recursos. Independentemente da tarefa em causa, o ciclo global de execução do sistema é sempre igual, enumerando-se de seguida os vários passos que o constituem, bem como os módulos intervenientes:

1. **Interpretação dos comandos do utilizador pela unidade central.** Os comandos podem ser fornecidos pelo teclado, pelo rato ou por uma *trackball.* Se não houver qualquer comando disponível, a unidade central prossegue com a tarefa em curso.

2. **Captação de uma imagem com cada uma das câmaras, ao que se segue o seu processamento.** Esta tarefa é da responsabilidade dos DSPs, sendo o tipo de tratamento a dar às imagens determinado pela unidade central. Concluído o processamento, os resultados são enviados à unidade central.

3. **Alteração da posição dos motores do sistema de visão activa.** Esta tarefa é da responsabilidade dos módulos controladores dos motores, sendo as novas posições desejadas comunicadas pela unidade central, com base nos resultados obtidos pelos DSPs.

4. **Comunicação ao utilizador do estado actual do sistema.** A unidade central é responsável por esta tarefa, utilizando para tal o monitor do computador pessoal.

Este ciclo de execução é repetido indefinidamente. Em função dos comandos recolhidos no passo 1, executam-se diferentes tipos de tarefas nos restantes passos. Por exemplo, se o utilizador optar por um módulo de recolha de imagens em contínuo, o passo 3 não é necessário, saltando-se directamente da captação das imagens para a sua representação. Embora o ciclo anterior tenha sido apresentado segundo a sequência lógica de funcionamento, a independência dos vários módulos permite a paralelização das várias acções. Por exemplo, enquanto um novo par de imagens está a ser adquirido e processado, os controladores dos motores podem estar ocupados com as últimas posições solicitadas pela unidade central. O resultado é a redução do tempo de ciclo do sistema, tendo como único custo um atraso entre os instantes de acção e de reacção. Tudo isto pode ser facilmente observado analisando a figura 2.6. Na figura, podemos ver os três módulos do sistema, executando as tarefas que lhes competem (enumeradas atrás), com e sem paralelismo de acções. Quando as tarefas são executadas sequencialmente, o sistema demora

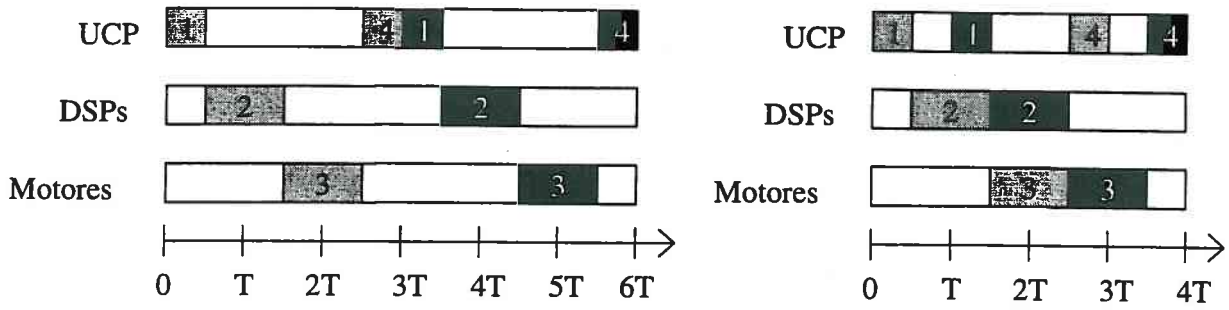



Figura 2.6: Diagramas temporais relativos ao ciclo global do sistema quando as tarefas são executadas sequencialmente (à esquerda) e quando há paralelismo (à direita). T representa um período temporal, com um limite mínimo de $40ms$, definido em função do tempo que demora a captar uma imagem de cada câmara, como veremos no capítulo 5.

três períodos de tempo desde o instante em que se inicia a interpretação das acções do utilizador, até que são apresentados os resultados, os quais são gerados de três em três períodos. Em contrapartida, se houver paralelismo, o sistema consegue gerar resultados com intervalos de apenas um período, triplicando a velocidade global. É claro, a resposta a uma determinada acção do utilizador continuará a ocorrer apenas três períodos após ter sido gerada, resultando num atraso ao qual não é possível fugir.

Estamos agora em condições de encarar os vários estados do processo de fixação. Até ao fim da secção, iremos abordar, em linhas gerais, as tarefas a desempenhar e os principais problemas que surgirão aquando da sua concepção. Começando pelo estado de *Espera* e recordando a secção anterior, temos como tarefa detectar movimentos que ocorram no campo de visão do sistema, isto é, das câmaras. Por outras palavras, é necessário analisar a sequência de imagens captada por cada uma das câmaras, procurando alterações nos níveis de intensidade luminosa dos pontos que as formam. Teoricamente, esta tarefa pode ser abordada com métodos bastante simples. No entanto, quaisquer que sejam os métodos, a sua realização prática reveste-se de diversas dificuldades, como sejam:

- A necessidade de distinguir entre os movimentos reais dos alvos e os movimentos aparentes que ocorrem quando o sistema de visão activa se movimenta. Este problema só se coloca, é claro, se optarmos por detectar movimentos de alvos quando as câmaras não estão em repouso.

- As alterações provocadas na imagem pelo ruído. Ao procurar alterações na imagem, é necessário levar em consideração que, mesmo numa sequência de imagens aparentemente idênticas, existem variações temporais nos valores das intensidades.

- As constantes variações na luminosidade. Como seria de esperar, a amplitude das intensidades dos pontos de uma imagem depende directamente da luminosidade da cena. Daqui resulta que, para uma mesma causa, as alterações provocadas numa imagem bem iluminada têm uma magnitude superior às provocadas numa imagem com fraca iluminação. Uma vez que as cenas reais raramente têm uma iluminação uniforme, é necessário antecipar que, um mesmo alvo, poderá provocar diferentes estímulos em locais diversos da cena.

Depois do estado de *Espera*, segue-se a *Selecção*. Esta inicia-se com a análise das características do movimento detectado, podendo incluir a recolha de características adi-

cionais. Dependendo dos métodos utilizados, é possível levar em conta características como:

- A velocidade, a direcção e o sentido do movimento detectado.
- A dimensão da área onde ocorreu o movimento e eventual segmentação da imagem.
- Dados sobre a *aparência* do alvo: intensidade, padrões, texturas, forma, etc.
- Características de invariância de escala e/ou rotação.
- Informação sobre a estrutura do alvo: Arestas, cantos, etc.
- O tempo até impacto.

De seguida, é necessário definir optimamente os limites e condições a impor, de forma a maximizar as hipóteses do sistema se manter, durante o maior intervalo de tempo possível, no estado de *Fixação*, já que é esse o nosso principal objectivo. Pelo que se acabou de dizer, a resposta parece ser imediata: definir condições rígidas e limites apertados, forçando a selecção de alvos óptimos. Infelizmente, o resultado prático desta solução é a rejeição da generalidade dos alvos, originando um sistema selectivista.

Um problema adicional está na necessidade de garantir que, nas sequências de imagens recolhidas com cada uma das câmaras, o alvo seleccionado seja o mesmo. Esta é uma tarefa complicada, por razões que incluem:

- As diferentes imagens que cada uma das câmaras capta do mesmo alvo, motivadas pelo afastamento existente entre elas (diferentes perspectivas).
- Parâmetros físicos das câmaras, que tendem a incrementar as diferenças existentes entre as duas imagens do mesmo alvo: abertura da íris, ampliação, focagem, sensibilidade dos sensores ópticos, etc.
- O desconhecimento, à partida, da distância a que o alvo se encontra, o que impede um alinhamento mais favorável das câmaras.

Estas características afectam também o desempenho dos métodos utilizados no estado de *Fixação*. Antes de passarmos a esse estado, importa salientar que, no sistema VARMA I, o tempo despendido nos dois estados descritos revelou-se um dos pontos mais críticos de todo o processo: de cada vez que o sistema era forçado a voltar ao estado de *Espera* e antes de ser novamente atingido o estado de *Fixação*, passavam alguns preciosos segundos que eram mais do que suficientes para que o alvo *escapasse*.

Atingido o estado de *Fixação*, a nossa tarefa parece simplificada: sabemos que ocorreu um movimento, o alvo causador foi detectado e isolado em ambas as sequências de imagens, e foram recolhidas informações sobre as características do alvo. Resta-nos controlar o sistema de visão activa, de forma a que a imagem do alvo se projecte no centro de ambas as câmaras, a exemplo do que já vimos quando analisámos o comportamento humano. Esta não é, no entanto, uma tarefa simples. A sua resolução passa por procurar o alvo em cada novo par de imagens captadas, levando em consideração que o alvo se movimenta de forma desconhecida e que a sua orientação espacial pode variar. Além disso, existem limites físicos para a velocidade e a excursão dos motores, bem como para o campo de visão das câmaras, o que dificulta a optimização da solução final.

Recordando os sinónimos de fixação, "não esquecer", "segurar" e "tornar-se estável", percebe-se agora a curiosa relação existente entre eles e o processo descrito atrás. Depois de seleccionado um alvo, é necessário memorizar dados sobre a sua estrutura e aparência, de modo a que a sua detecção futura seja possível. Pretende-se também que a cabeça se movimente de forma a manter o alvo dentro do campo de visão das câmaras, sem o perder. Finalmente, a movimentação da estrutura mecânica representa um problema típico de controlo, que se pretende seja o mais estável possível.

Damos assim por concluída a análise conceptual do processo de fixação visual. Antes de fazermos a sua concretização prática e darmos exemplos reais da aplicação do processo, vamos debruçar-nos sobre diversos métodos de análise e processamento de imagens, de entre os quais seleccionaremos os mais adequados em função dos objectivos pretendidos e das limitações existentes. Assim, no capítulo 3 abordaremos exclusivamente as vantagens e inconvenientes de algumas técnicas de tratamento de imagens, bem como a possibilidade de as interligar e os resultados assim obtidos. De seguida, no capítulo 4, tentaremos construir um algoritmo base para o processo de fixação visual, utilizando algumas das técnicas descritas, sem ter como objectivo construir uma aplicação ou resolver um problema concreto. A ideia é, tão somente, criar uma espinha dorsal para eventuais aplicações. Finalmente, nos capítulos 5 e 6 veremos duas aplicações concretas com objectivos bem definidos, construídas a partir do algoritmo base, mas com as imprescindíveis alterações e adaptações decorrentes das finalidades pretendidas.

# Capítulo 3

# Métodos e Técnicas de Análise e Processamento de Imagens

## 3.1 Introdução

Com vista à implementação prática do processo de fixação visual, seleccionámos alguns métodos e técnicas de análise e processamento de imagens bastante conhecidas, que integrámos e adaptámos de forma a que fossem aplicáveis em tempo real. A fundamentação teórica desses métodos não se enquadra nos nossos objectivos, podendo ser encontrada nas várias referências feitas ao longo do texto. Neste capítulo pretendemos mostrar o aproveitamento que fizemos das diversas técnicas, bem como as vantagens em associar diferentes métodos em busca da solução para um determinado problema. Importa referir que foram desenvolvidos algoritmos para todas as técnicas abordadas neste capítulo, e optimizados de forma a poderem ser executados nos *DSPs* em tempo real.

Os métodos que iremos analisar foram escolhidos tendo como objectivo a exploração de várias áreas da computação visual: as transformações, as pirâmides, o fluxo óptico, a extracção de contornos e a correspondência. Por transformação entende-se a aplicação às imagens de um operador, linear ou não, que tem como resultado a alteração da dimensão, da resolução e/ou de outras propriedades das imagens. Neste campo, veremos duas técnicas: o plano *log-polar* e as filtragens. A abordagem desta última será integrada na construção de pirâmides de imagens, que resulta directamente da aplicação de filtros e de reduções de escala. Continuaremos depois com a análise do resultado da aplicação às imagens do cálculo do fluxo óptico e do fluxo óptico normal, quando são utilizados diferentes tipos de transformações. De seguida, veremos métodos de extracção de contornos de imagens, bem como algoritmos que permitem melhorar o seu desempenho. Tal como no caso do fluxo óptico, iremos comparar os resultados que se obtêm com a detecção de contornos quando são utilizados diferentes tipos de transformações. Por último, abordaremos a correspondência que, como o nome indica, está relacionada com os algoritmos destinados à associação de determinadas características numa ou mais imagens. O método que veremos é a correlação, que é um dos processos mais conhecidos para efectuar a correspondência, sendo apresentados e comparados resultados com e sem a extracção prévia de contornos.

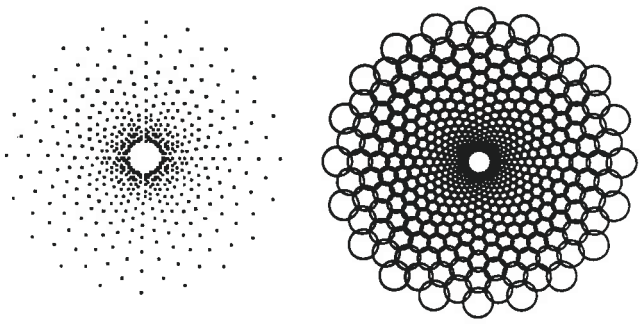


Figura 3.1: Estratégia de amostragem proposta por Sandini e Tagliasco para obtenção de imagens *log-polar*. Na imagem esquerda, podemos ver a grelha de amostragem. Na direita, delimitam-se as áreas (círculos) que cada amostra representa.

## 3.2 O Plano *Log-Polar*

### 3.2.1 Introdução

As imagens amostradas em coordenadas polares têm já uma história considerável no campo da visão por computador. A aplicação do logaritmo à componente radial criou um interesse particular na técnica, já que permite obter imagens com características semelhantes às captadas pelo olho humano, sendo designadas por imagens *log-polar*. De facto, a distribuição de foto-receptores na retina não é uniforme, atingindo a sua densidade máxima na fóvea e diminuindo na direcção da periferia, podendo ser descrita por uma lei logarítmica [SCH80]. Quando não se dispõe de dispositivos capazes de capturar imagens mapeadas directamente em formato *log-polar*, torna-se necessário utilizar câmaras com matrizes de sensores ópticos distribuídos uniformemente, fazendo de seguida a sua reamostragem para o plano *log-polar*. Existem diversos métodos e abordagens para efectuar esta transformação. Grosso modo, todos eles podem ser derivados do método proposto por Sandini e Tagliasco [SAN80], em 1980.

### 3.2.2 Método de Sandini e Tagliasco

Sandini e Tagliasco basearam-se na configuração do cortex do sistema de visão humano para conceber o método agora descrito. Em termos gerais, verifica-se que, no olho humano, o período de amostragem varia com a distância à fóvea, sendo aproximadamente linear dentro dela. Com base na formulação matemática destas características [SCH77], foi possível conceber uma estratégia de amostragem polar, espacialmente variante, caracterizada pela relação linear entre o período de amostragem $P$ e a distância ao centro ou excentricidade $E$. A grelha de amostragem é baseada em círculos concêntricos, sobre os quais é efectuado um número constante de amostras $N$, que corresponderá à resolução angular da imagem *log-polar*. A resolução radial é determinada pelo número $L$ de círculos consecutivos existentes na grelha de amostragem.

Passando à formulação analítica, para um dado número de amostras $N$, o raio do $i$-ésimo círculo da grelha, com $1 \leq i \leq L$, é dado por:

$$E_i = E_{min}\left(1 + \frac{\pi}{N\sqrt{3}}\right)^{i-1} \tag{3.1}$$

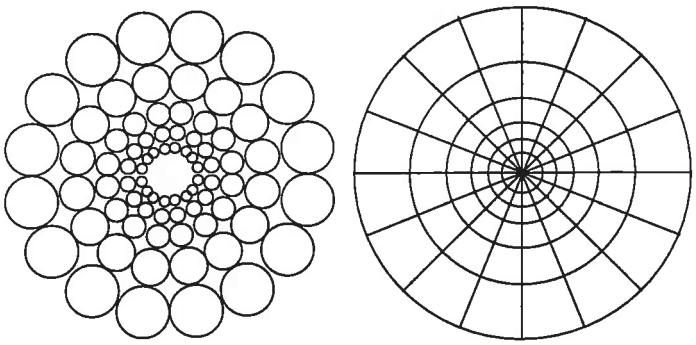


Figura 3.2: Estratégias de amostragem simplificadas, utilizando áreas circulares (à esquerda) e secções de arcos (à direita).

Na equação anterior, $E_{min}$ corresponde à menor excentricidade, isto é, ao círculo da grelha com menor raio. O seu valor deverá ser escolhido em função do mínimo período de amostragem, para evitar a sobre-amostragem da área central da imagem (voltaremos a este assunto mais tarde). A equação que nos permite determinar o período de amostragem é a seguinte:

$$P_i = \frac{2\pi E_i}{N\sqrt{3}} \tag{3.2}$$

A figura 3.1 ajuda a esclarecer a mecânica de amostragens proposta por este método. A cada amostra corresponde uma determinada área, devendo pesar-se todos os pontos da imagem nela contidos, para obter o valor final da amostra. Esta pesagem é particularmente importante nas amostras exteriores, já que a área da imagem por elas coberta é consideravelmente maior do que nas zonas centrais. A área associada a cada amostra é um círculo cujo raio é igual à diferença entre excentricidades consecutivas. Para centro do círculo escolhe-se a posição da amostra respectiva. Nestas condições, para as amostras situadas sobre o $i$-ésimo círculo da grelha, o raio $R_i$ será dado por:

$$R_i = E_{i+1} - E_i = \frac{\pi E_i}{N\sqrt{3}} \tag{3.3}$$

Como seria de esperar, este valor corresponde a metade do período de amostragem. Note-se ainda que, em cada círculo da grelha de amostragem, a posição das amostras sofre um deslocamento angular relativamente ao círculo precedente. Esse deslocamento é igual a $\frac{\pi}{N}$, permitindo uma máxima cobertura da imagem, sem aumentar o número de amostras ou a área associada a cada uma delas. Recordando que em cada círculo da grelha são efectuadas $N$ amostragens, não será de estranhar que as amostras apresentem as mesmas posições angulares de dois em dois círculos.

### 3.2.3 Simplificações e Considerações

A variação ao método descrito anteriormente que mais vezes é encontrada consiste na eliminação do deslocamento angular $\frac{\pi}{N}$, existente entre as amostras pertencentes a círculos consecutivos. Esta alteração permite simplificar consideravelmente a descrição analítica do método, que passa a ser:

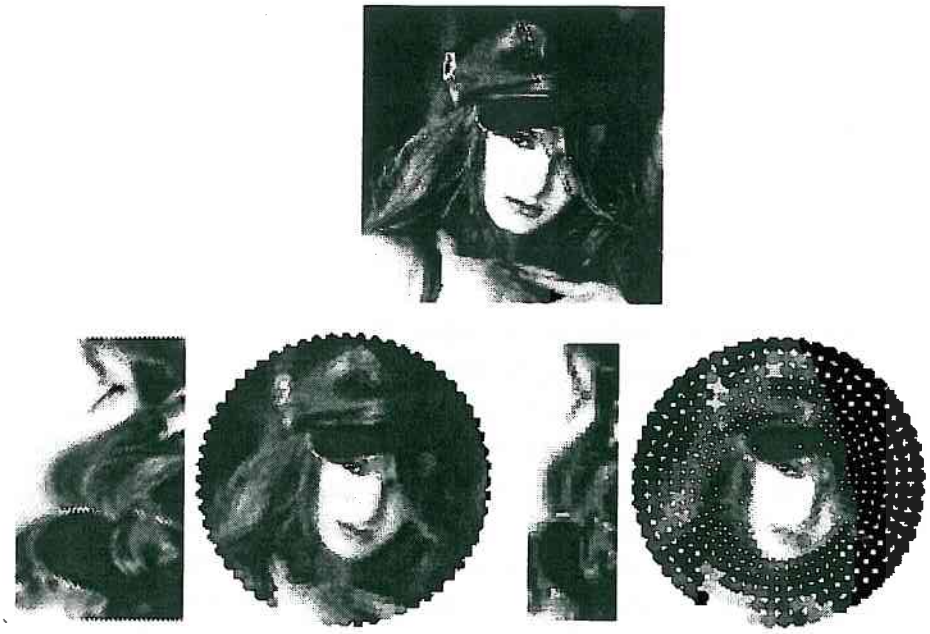


Figura 3.3: Exemplo da aplicação dos métodos de transformação *log-polar* a uma imagem. A imagem de teste (em cima) foi transformada utilizando o método de Sandini e Tagliasco (em baixo à esquerda) e o método simplificado com áreas circulares (em baixo à direita). Para cada um dos métodos, podemos ver a imagem resultante no plano *log-polar* e ainda a imagem que se obtém ao fazer a transformação inversa, isto é, do plano *log-polar* para o plano cartesiano.

$$\begin{aligned} E_i &= \left(\frac{N+\pi}{N-\pi}\right)^i \\ P_i &= \frac{2\pi E_i}{N} \\ R_i &= \frac{P_i}{2} \end{aligned} \tag{3.4}$$

No lado esquerdo da figura 3.2, podemos ver a estratégia de amostragem que daqui resulta. Note-se que as relações 3.4 dependem de um pressuposto relativamente ao valor de $N$: assumiu-se que $N$ é tal que $R_i$ pode ser aproximado por metade do comprimento do arco que separa duas amostras consecutivas (o valor exacto para $R_i$ seria igual a metade do comprimento da corda respectiva). Obviamente, o valor de $L$ também influi nesta restrição (recorde-se que $1 \leq i \leq L$).

O principal inconveniente da simplificação efectuada reside no facto de não cobrir completamente a área da imagem amostrada. Para colmatar estas falhas, podemos substituir as áreas circulares por secções de arcos, tal como se exemplifica no lado direito da figura 3.2. Ao contrário do que se poderia esperar, a utilização das áreas circulares não é preterida sistematicamente em favor da utilização de secções de arcos. Isto deve-se, fundamentalmente, à menor complexidade necessária para a sua execução, sem grande prejuízo nos resultados da sua aplicação, quando associada com outros processamentos. Na figura 3.3, podemos comparar os resultados que se obtêm ao aplicar o método de Sandini e Tagliasco e o método simplificado com áreas circulares. Relativamente às imagens *log-polar*,

importa realçar que a resolução radial, quando se opta pelo método simplificado, aparece reduzida para metade porque, na prática, o método simplificado resume-se à eliminação das amostras que, no método de Sandini e Tagliasco, servem para "tapar" as falhas de amostragem. Esta conclusão pode ser mais facilmente compreendida analisando as imagens resultantes da transformação inversa (incluídas na figura 3.3). Como podemos ver, a transformação inversa da imagem *log-polar* obtida através do método simplificado contém áreas em branco, correspondentes a pontos que não são contemplados por nenhuma das áreas circulares de amostragem.

Um aspecto importante em todos os métodos, deixado em aberto quando analisámos o método de Sandini e Tagliasco, é o do limite mínimo para a excentricidade. Esse valor é importante para evitar que, a duas amostras distintas, correspondam exactamente os mesmos pontos da imagem. Esse efeito é conhecido por sobre-amostragem, já que resulta da amostragem da imagem para lá dos seus limites de resolução. Facilmente se compreenderá que a excentricidade mínima terá sempre de ser maior ou igual ao mínimo período de amostragem. No caso do método de Sandini e Tagliasco, a aplicação do valor de $E_{min}$ está já contemplada nas equações. Tal não acontece se forem utilizadas as simplificações descritas pelas equações 3.4. Nesses casos, teremos de calcular um valor mínimo para o índice $i$, que será dado por:

$$i_{min} = \log_{\frac{N+\pi}{N-\pi}} E_{min} \tag{3.5}$$

Assim, os valores válidos para o índice passam a ser $i_{min} \leq i \leq L$, com $i$ inteiro. Note-se que, em qualquer dos casos, se a excentricidade mínima escolhida for muito maior do que o mínimo período de amostragem, não serão contemplados por nenhuma das áreas de amostragem muitos dos pontos da zona central da imagem.

A definição da excentricidade mínima não é a única abordagem possível para o problema. Alguns autores optam por fazer a pesagem de pontos na área de amostragem sempre que a resolução da imagem o permite, substituindo-a por uma sub-amostragem quando tal não é possível. A sub-amostragem resume-se à obtenção de uma estimativa para a intensidade de um ponto que, na realidade, não existe na imagem. Essa estimação é feita com base nos pontos reais da imagem, recolhidos na vizinhança hipotética do ponto pretendido, podendo ser mais ou menos complexa dependendo da pesagem efectuada e do número de pontos vizinhos considerados.

Para concluir a apresentação das estratégias de amostragem, importa referir que nem todos os métodos relacionam o número de amostras $N$ com a base logarítmica da grelha. Nos métodos apresentados, essa relação foi calculada tentando minimizar simultaneamente a área da imagem não contemplada pela amostragem e as intersecções que ocorrem entre as áreas de amostragem. Podemos observar facilmente estas restrições na figura 3.1 (à direita) e na figura 3.2 (à esquerda). Como se pode ver, no primeiro exemplo, todas as zonas da imagem são contempladas, verificando-se intersecções tangenciais das áreas de amostragem. Já no segundo caso, opta-se por uma distribuição compacta dos círculos, sem intersecções, deixando áreas da imagem a descoberto, que são tanto maiores quanto maior é a distância ao centro. Este tipo de considerações perde significado sempre que o objectivo não inclui fazer a pesagem de um conjunto de pontos por amostra. Nesses casos, o estabelecimento de uma relação entre o número de amostras $N$ e a base logarítmica não é necessária, podendo até não ser desejável, pois, para além da complexidade associada às equações, os objectivos em causa poderão ser melhor atingidos alvitrando valores e fazendo a sua correcção e optimização por tentativa e erro.

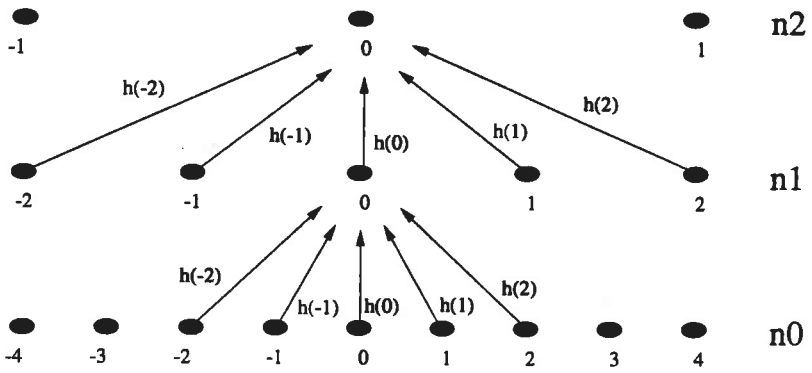



Figura 3.4: Pirâmide Laplaciana unidimensional. De cada vez que subimos um nível, filtramos o sinal e aproveitamos apenas uma em cada duas amostras, reduzindo para metade o seu número.

A transformação *log-polar* irá ser utilizada ao longo deste capítulo conjuntamente com os diferentes métodos estudados, voltando a aparecer novamente no capítulo 6 na base de um processo de seguimento de contornos. Nesse capítulo serão feitos alguns comentários sobre a aplicabilidade e o interesse da utilização das transformações para o plano *log-polar*.

## 3.3 Pirâmides

### 3.3.1 Introdução

A construção de pirâmides de imagens é, basicamente, uma técnica de compactação de imagens. A ideia é, dada uma determinada imagem, reduzir significativamente as suas dimensões, sem perder a informação nela contida, de forma a obter uma imagem equivalente em termos de conteúdo, mas de inferior tamanho. As vantagens são evidentes, não só ao nível dos recursos necessários para guardar as imagens, mas principalmente ao nível da velocidade que se ganha ao processar um menor volume de informação. Burt foi um dos percursores do trabalho que se faz actualmente ao nível do processamento com pirâmides [BUR84], tendo inclusivé desenvolvido *hardware* dedicado para a sua geração [BUR86]. É com um método de sua autoria que abordaremos a técnica das pirâmides.

### 3.3.2 Pirâmide Laplaciana

A pirâmide Laplaciana foi desenvolvida por Burt e Andelson, com o objectivo primário de comprimir imagens [BUR83]. Baseia-se numa filtragem, seguida da amostragem de um em cada dois pontos da imagem, reduzindo a sua dimensão para metade. Analisando o caso unidimensional, podemos ver na figura 3.4 a forma como evolui o processo de compactação. A sua formulação matemática é a seguinte:

$$n_{i+1}(k) = \sum_{p} h(p-2k)n_i(p) \tag{3.6}$$

Na fórmula, $n_i$ representa as amostras do nível $i$ da pirâmide e $h$ representa os pesos do filtro utilizado. Os valores de $h$ são cruciais em todo o processo, já que só a sua correcta definição permitirá a reconstrução de um nível da pirâmide (filtrado) a partir

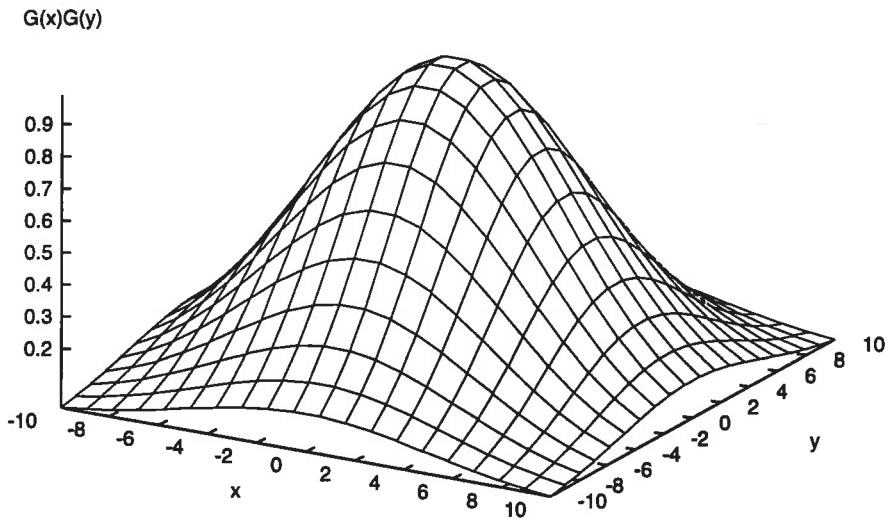



Figura 3.5: Superfície Gaussiana, com $\sigma$ igual a 5.

do nível imediatamente acima. Em termos gerais, os valores de $h$ reportam-se a um filtro passa baixo, devendo a sua largura de banda ser duas vezes inferior à frequência de amostragem, para que seja possível a recuperação do sinal filtrado. Embora o nosso objectivo não seja, à partida, recuperar as imagens dos níveis inferiores a partir das dos níveis acima, a correcta definição dos valores de $h$ não perde por isso importância, porque, quanto melhor for a resposta em frequência do filtro utilizado, melhor será a qualidade das imagens de menor resolução. Consequentemente, uma correcta definição do filtro permite-nos esperar melhores resultados dos algoritmos que viermos a aplicar às imagens, particularmente ao nível da eliminação do ruído e da remoção de distorções [MEE87].

### 3.3.3 Máscaras Gaussianas

#### Definição e Formulação

Um dos filtros mais habitualmente utilizados e com melhores resultados globais, tem como base a curva Gaussiana. Os filtros discretos bidimensionais Gaussianos (ou máscaras Gaussianas), são circularmente simétricos (ou isotrópicos), desde que ambos os eixos tenham o mesmo desvio padrão. Para gerar estas máscaras, parte-se da curva unidimensional, cuja expressão matemática é:

$$G(x) = e^{-\frac{x^2}{2\sigma^2}} \tag{3.7}$$

Na fórmula, o parâmetro $\sigma$ é utilizado para controlar a abertura da curva. Daqui chega-se rapidamente à equação bidimensional, cuja expressão é obtida da seguinte forma:

$$\begin{aligned} S(x,y) &= G(x)G(y) \\ &= e^{-\frac{x^2}{2\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} \\ &= e^{-\frac{x^2+y^2}{2\sigma^2}} \end{aligned} \tag{3.8}$$

Em termos geométricos, podemos imaginar a geração da superfície $S(x,y)$ (ver a figura 3.5) como resultado da rotação da curva unidimensional em torno da recta dada por

$x = 0$. Para chegarmos à máscara Gaussiana, teremos primeiro de discretizar $S(x, y)$. Por questões de simplicidade e clareza, iremos trabalhar somente numa dimensão, isto é, com a equação 3.7. Uma vez que a curva não tem zeros, o primeiro passo será decidir o valor de $G(x)$ abaixo do qual se considerará a função nula. A escolha desse valor está directamente relacionada com a dimensão da máscara que pretendemos obter, ou seja, com o número de amostragens de $G(x)$ a efectuar. Seja $N$ esse número, pretendemos que:

- A máscara Gaussiana resultante seja quadrada, isto é, a sua dimensão total seja $N \times N$. Esta restrição justifica-se por questões de simetria.
- $N$ seja ímpar, visto que a máscara deverá ser simétrica relativamente ao seu centro, tal como a função $G(x)$. Daqui resulta que a função $G(x)$ será amostrada no seu centro (para $x = 0$) e ainda $\frac{N-1}{2}$ vezes em cada um dos semieixos.

Definindo $G_{min}$ como o limiar abaixo do qual se desprezarão os valores de $G(x)$, podemos agora calcular a largura da curva Gaussiana pretendida, ou seja, o parâmetro $\sigma$, com:

$$\begin{aligned} G(N) &= G_{min} \\ <=> e^{-\frac{\left(\frac{N-1}{2}\right)^2}{2\sigma^2}} &= G_{min} \\ <=> 2\sigma^2 &= -\frac{\left(\frac{N-1}{2}\right)^2}{\ln G_{min}} \end{aligned} \tag{3.9}$$

Note-se que $G_{min}$ é igual à constante de Nepper elevada a um número negativo, pelo que será sempre menor do que a unidade e, consequentemente, o seu logaritmo natural será sempre negativo. Obtido o valor de $\sigma$, resta agora amostrar $G(x)$ para todos os valores de $x$ inteiros tais que $|x| \leq \frac{N-1}{2}$. Analogamente, obteremos a máscara Gaussiana amostrando $S(x, y)$ para todos os pares de valores $(x, y)$ inteiros tais que $|x| \leq \frac{N-1}{2}$ e $|y| \leq \frac{N-1}{2}$.

### Convolução

A aplicação de uma máscara a uma imagem designa-se por convolução. O resultado da convolução de uma máscara com uma imagem é ainda uma imagem, onde cada ponto resulta da pesagem dos pontos da imagem inicial pelos elementos da máscara. Com vista à tradução do processo matematicamente, vamos supor que $S(x, y)$ é uma máscara de dimensão $N \times N$ e $I(x, y)$ é a imagem com a qual pretendemos convolver a máscara. Nestas condições, o ponto $F(x, y)$ da nova imagem será dado por:

$$F(x,y) = \sum_{i=-\frac{N-1}{2}}^{\frac{N-1}{2}} \sum_{j=-\frac{N-1}{2}}^{\frac{N-1}{2}} I(x+i, y+j)S(i,j) \tag{3.10}$$

Para além da análise da equação anterior, é importante realçar os seguintes aspectos:

- Os limites de variação dos somatórios estão de acordo com a formulação das máscaras quadradas, o que não impede, obviamente, que possam ser adaptados a máscaras com características diferentes.

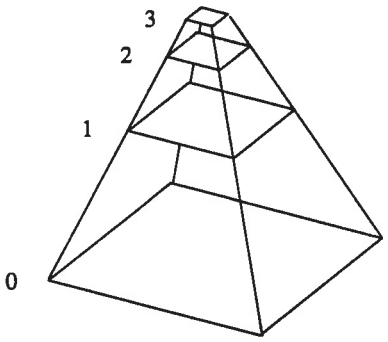



Figura 3.6: Representação esquemática de quatro níveis de uma pirâmide, considerando-se o nível 0 como o mais baixo, correspondente à imagem original.

- Uma vez que a convolução transforma imagens em imagens, é natural que se queiram obter valores para $F(x,y)$ dentro dos mesmos limites dos valores de $I(x,y)$. Para que tal se verifique, deve-se normalizar a máscara criada, antes de a convolver com as imagens. A normalização consiste na multiplicação de todos os pesos da máscara por um valor $K$, escolhido de forma a que os pesos resultantes respeitem a relação: $\sum S(i,j) = 1$.

Finalmente, refira-se que, no caso particular da máscara Gaussiana, é possível reduzir o número total de multiplicações necessárias para a convolver com uma imagem. De facto, recordando que $S(x,y) = G(x)G(y)$, podemos simplificar a equação 3.10 da seguinte forma (os limites dos somatórios foram omitidos para maior clareza):

$$\begin{aligned} F(x,y) &= \sum_i \sum_j I(x+i,y+j)S(i,j) \\ &= \sum_i \sum_j I(x+i,y+j)G(i)G(j) \\ &= \sum_i G(i) \sum_j I(x+i,y+j)G(j) \end{aligned} \tag{3.11}$$

A equação 3.11 demonstra que a aplicação da máscara Gaussiana é equivalente à convolução do filtro unidimensional com a imagem, primeiro na vertical e depois na horizontal, ou vice-versa. Esta conclusão é particularmente útil ao nível computacional, podendo ser generalizada para qualquer outra máscara que goze da propriedade da *separabilidade*.

### 3.3.4 Pirâmides Bidimensionais

A extensão ao caso bidimensional da formulação da pirâmide Laplaciana unidimensional, vista atrás, deverá agora ser trivial. Depois da filtragem de um nível com uma máscara Gaussiana ou outra qualquer, aproveitamos para o nível superior apenas um em cada conjunto de 4 pontos (agrupados num quadrado de $2 \times 2$). O resultado é uma imagem com metade da resolução segundo cada um dos eixos e com um quarto da área, tal como se vê na figura 3.6.

Isoladamente, as pirâmides têm pouca utilidade, para além da sua vertente como técnica de compactação de imagens. No entanto, associada a outras técnicas, a construção de pirâmides revela-se um método de processamento com inúmeras vantagens,

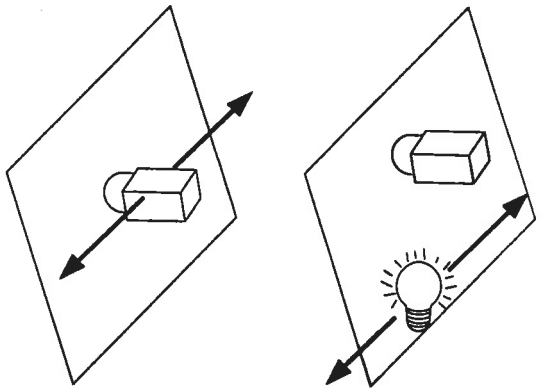


Figura 3.7: Duas situações em que o fluxo óptico não coincide com o movimento. À esquerda, temos uma câmara que se movimenta em frente a um plano unicolor uniformemente iluminado. O movimento não provoca alterações na imagem, pelo que o fluxo óptico é nulo. À direita, a câmara está estática, tendo-se introduzido na cena uma fonte de luz, que é movimentada aleatoriamente. As alterações de luminosidade no plano originam um fluxo óptico não nulo, que não corresponde a qualquer movimento do plano.

nomeadamente ao nível da velocidade de processamento e da redução do ruído das imagens. Nas próximas secções, veremos alguns exemplos da aplicação das pirâmides durante as fases de pré-processamento das imagens. Para concluir, refira-se que as pirâmides estão na base do estudo dos espaços de escalas, cuja principal particularidade é a introdução da resolução como variável adicional a utilizar na análise de imagens [LIN91].

## 3.4 Fluxo Óptico

### 3.4.1 Definição

Por definição, o fluxo óptico corresponde ao movimento aparente das intensidades dos pontos de uma imagem, quando a câmara se movimenta relativamente à cena visualizada. Equivalentemente, se a câmara estiver estática, pode-se aplicar a definição a um alvo ou alvos que se movimentem, relativamente à câmara, dentro do seu campo visual. Decorre da definição que, se um determinado movimento não provocar alterações nas intensidades dos pontos das imagens, o fluxo óptico será nulo. Por outro lado, se ocorrerem variações de luminosidade na cena, o fluxo óptico será não nulo, sem que tenham ocorrido movimentos. Resumindo, o fluxo óptico pode não ser igual ao movimento. Na figura 3.7 exemplificam-se dois casos em que ocorre esta situação.

Uma vez que o fluxo óptico é a única informação que temos disponível, teremos de nos contentar com o facto de as situações exemplificadas não serem os casos mais comuns.

### 3.4.2 Formulação Matemática

Nesta secção, vamos fazer uma rápida abordagem à formulação matemática do fluxo óptico, sugerindo-se a consulta do livro de Horn [HOR86] para uma análise mais detalhada. Seja $I(x, y, t)$ a intensidade do ponto localizado na posição $(x, y)$ da imagem obtida no instante $t$. Se $u(x, y)$ e $v(x, y)$ forem as componentes $x$ e $y$ do vector fluxo óptico naquele

ponto, é de esperar que a intensidade seja a mesma no ponto $(x + \delta x, y + \delta y)$ no instante $t + \delta t$, onde $\delta x = u\delta t$ e $\delta y = v\delta t$. Traduzindo esta relação matematicamente, temos que, para um pequeno intervalo de tempo $\delta t$:

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \tag{3.12}$$

Apenas com esta relação, não é possível definir $u$ e $v$ de forma única. Felizmente, podemos tirar partido de os movimentos serem, de um modo geral, contínuos. Daí que, se considerarmos que o fluxo óptico reflecte essa continuidade, podemos expandir a equação anterior numa série de Taylor, o que origina a seguinte relação:

$$I(x, y, t) + \delta x \frac{\partial I}{\partial x} + \delta y \frac{\partial I}{\partial y} + \delta t \frac{\partial I}{\partial t} + e = I(x, y, t) \tag{3.13}$$

Na equação, $e$ representa termos de ordem superior a 1 em $\delta x$, $\delta y$ e $\delta t$. Eliminando $I(x, y, t)$ em ambos os membros, dividindo por $\delta t$ e calculando o limite quando $\delta t$ tende para 0, obtém-se:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{3.14}$$

Esta relação não é mais do que a expansão da derivada de $I$ relativamente ao tempo, que deverá ser nula no limite, tal como era esperado. Recorrendo às abreviações:

$$I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}, I_t = \frac{\partial I}{\partial t}, u = \frac{dx}{dt}, v = \frac{dy}{dt}, \tag{3.15}$$

podemos escrever a seguinte relação simplificada:

$$I_x u + I_y v + I_t = 0 \tag{3.16}$$

### 3.4.3 Fluxo Óptico Normal

Consideremos um espaço bidimensional de eixos $u$ e $v$ a que chamamos espaço de velocidades. Os valores de $(u, v)$ que satisfazem a relação 3.16 situam-se sobre uma linha recta no espaço de velocidades. Daí que, depois de medidos os valores de $I_x$, $I_y$ e $I_t$ nas imagens, não é possível identificar mais do que essa linha (ver a figura 3.8). Reescrevendo a equação 3.16, temos:

$$(I_x, I_y) \cdot (u, v) = -I_t \tag{3.17}$$

Podemos assim obter a componente na direcção do vector gradiente de intensidades $(I_x, I_y)^T$ com:

$$\frac{I_t}{\sqrt{I_x^2 + I_y^2}} \tag{3.18}$$

No entanto, não é possível determinar directamente a componente perpendicular a esta direcção. Tal facto designa-se habitualmente como o problema da abertura, chamando-se ao fluxo assim calculado fluxo óptico normal. Esta aparente limitação não diminui o interesse pelo seu estudo nem o universo de situações em que se aplica, com alguns resultados de relevo [SIN94].

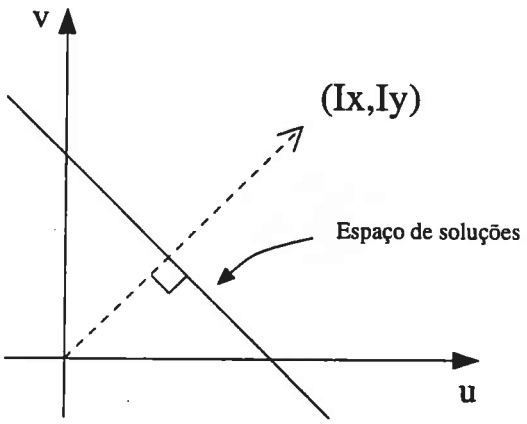



Figura 3.8: Espaço de velocidades. Depois de obtidos os valores das componentes $I_x$ e $I_y$, podemos afirmar que o vector velocidade do fluxo terá de se situar sobre uma linha recta, perpendicular à direcção do vector gradiente de intensidades. No entanto, a sua localização exacta não é conhecida, sendo apenas possível determinar a componente na direcção do gradiente.

Introduzindo restrições adicionais à equação 3.16, é possível chegar a métodos de cálculo que permitem estimar ambas as componentes do fluxo óptico, sendo o método de Horn & Schunk um dos mais conhecidos [HOR81]. De um modo geral, estes métodos recorrem a aproximações sucessivas, exigindo várias iterações até que as estimações possam ser consideradas aceitáveis. De seguida veremos alguns exemplos da aplicação do fluxo óptico normal e também do método de cálculo de Horn & Schunk.

### 3.4.4 Alguns Exemplos

As aplicações do fluxo óptico são muitas e diversas. Por exemplo, Papanikolopoulos descreve um processo de seguimento utilizando um câmara montada num manipulador, que consegue seguir alvos que se movimentem numa trajectória paralela ao plano imagem. O processo baseia-se num algoritmo de fluxo óptico suficientemente robusto para permitir o seguimento de alvos em tempo real [PAP93]. Noutro campo, Tistarelli e Sandini fazem a transposição do cálculo do fluxo óptico para o plano *log-polar*, descrevendo um método para cálculo do tempo até impacto [TIS93].

Para ficarmos com uma ideia das potencialidades práticas do fluxo óptico, vamos analisar alguns casos resultantes da sua aplicação. No primeiro exemplo, utilizamos uma sequência de imagens sintéticas, nas quais se simula a deslocação de um quadrado em direcção ao centro da imagem, partindo da margem esquerda. Apenas com este exemplo, analisaremos os resultados da aplicação do fluxo óptico em quatro situações:

1. Cálculo do fluxo óptico normal nas imagens originais.

2. Cálculo do fluxo óptico nas imagens originais, utilizando o método iterativo de Horn & Schunk.

3. Cálculo do fluxo óptico normal na sequência de imagens obtida depois de se aplicar a transformação *log-polar* às imagens originais.

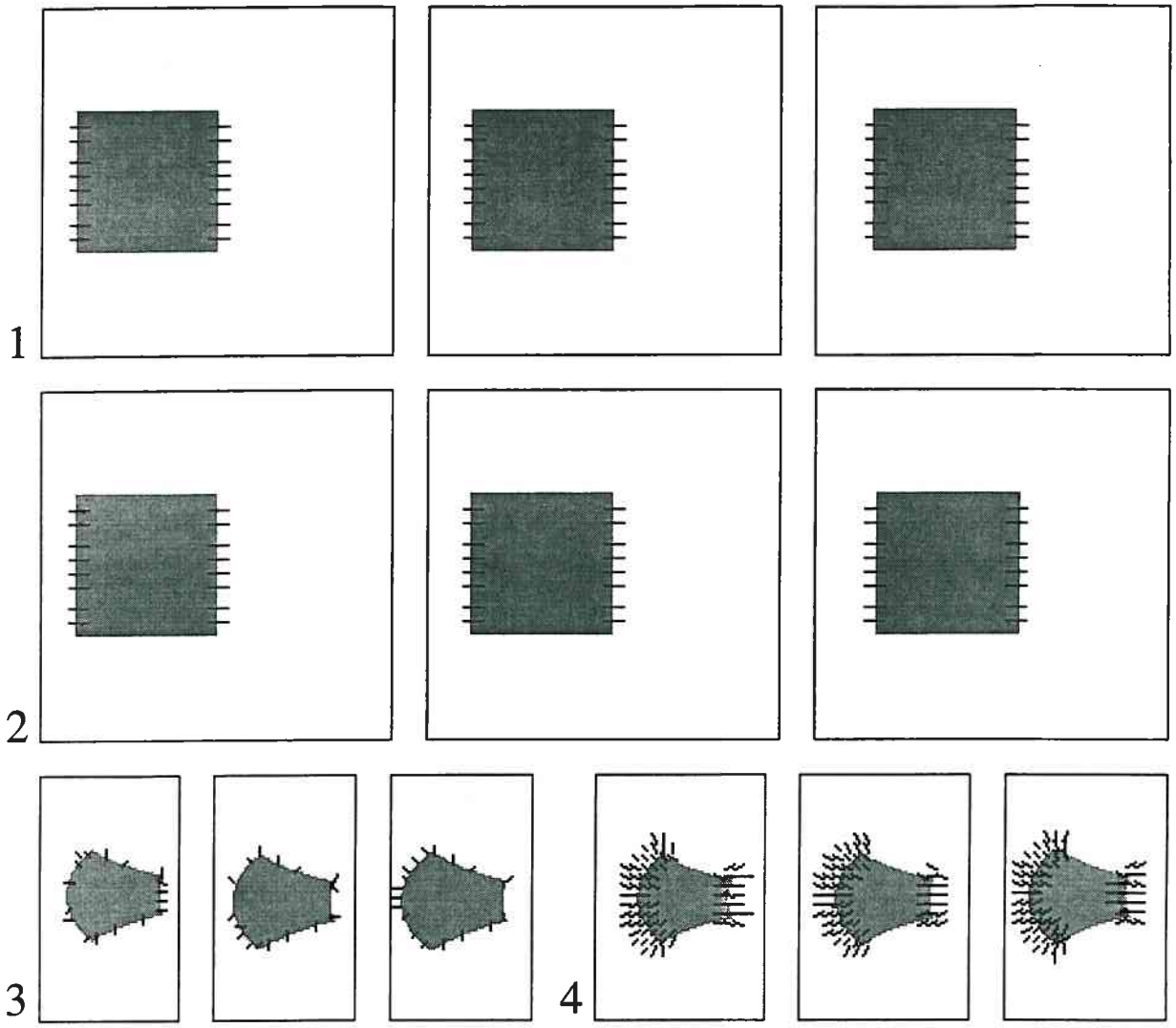



Figura 3.9: Resultado da aplicação do fluxo óptico à primeira sequência de imagens. Os resultados foram obtidos utilizando o fluxo óptico normal (1), o método de Horn & Schunk (2), o fluxo óptico normal em imagens *log-polar* (3) e o método de Horn & Schunk em imagens *log-polar* (4).

4. Cálculo do fluxo óptico utilizando o método iterativo de Horn & Schunk, utilizando a sequência de imagens obtida depois de se aplicar a transformação *log-polar* às imagens originais.

Os resultados podem ser observados conjuntamente na figura 3.9. Comparando as sequências 1 e 2, verifica-se que apenas existe fluxo óptico nas arestas verticais, o que se justifica em face do tipo de movimento simulado ser horizontal. Igualmente de esperar deveria ser o facto de o fluxo óptico normal ser igual ao fluxo óptico, já que o movimento é perpendicular às arestas verticais. O mesmo já não acontece nas imagens mapeadas para o plano *log-polar*. Note-se como, na sequência 3, os vectores são tendencialmente normais à superfície, enquanto que, na sequência 4, os vectores tendem a orientar-se mais ou menos no mesmo sentido. Relativamente a este último comentário, importa referir que, ao falarmos na orientação dos vectores no mesmo sentido, teremos de ter em conta

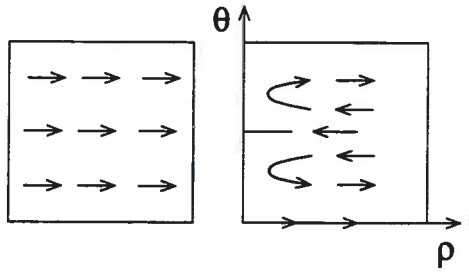



Figura 3.10: Se, num plano cartesiano, o fluxo óptico apontar para o lado direito, tal como se exemplifica na figura da esquerda, a distribuição equivalente no plano *log-polar* é semelhante à que se representa na figura da direita.

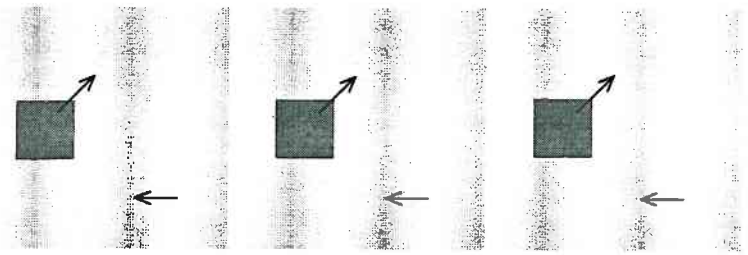


Figura 3.11: Sequência de imagens sintéticas simulando o movimento de um quadrado preto sobre um fundo com uma textura ondulada. O quadrado movimenta-se para cima e para a direita, partindo do lado esquerdo das imagens. O fundo movimenta-se para a esquerda.

que os vectores estão representados num plano de características polares. A figura 3.10 esclarece o porquê desta salvaguarda.

Com vista a uma análise mais rigorosa das orientações dos vectores de fluxo óptico, vamos avançar para outro exemplo, depois do qual calcularemos os histogramas de orientações dos vectores obtidos. A sequência que utilizaremos está representada na figura 3.11. Tal como no exemplo anterior, vamos combinar o fluxo óptico com o fluxo óptico normal, aplicado tanto às imagens originais como à sua equivalente *log-polar*. Os resultados obtidos podem ser vistos na figura 3.12.

Antes de mais, refira-se que, para construir os histogramas, foram calculados os fluxos em todos os pontos das imagens. Nas figuras apenas são representados alguns deles por questões de clareza. Adicionalmente, como o fluxo óptico normal não nos permite determinar o sentido do movimento detectado, a sua pesagem no histograma é feita tanto em $\theta$ como em $\theta+180^o$. Daí que, no histograma 1, apareçam dois picos de orientações, para $0^o$ e para $180^o$. O mesmo já não se passa com o método de Horn & Schunk, cujos vectores gerados têm um sentido bem definido, aparecendo por isso no histograma 2 apenas um pico em $180^o$. Em qualquer uma destas situações, apenas foi detectado o movimento do fundo (recorde-se que o fundo se movimenta para a esquerda, isto é, com uma orientação de $180^o$). No entanto, teremos de levar em conta que, uma vez que as imagens são sintéticas, as orientações são exactas, o que provoca picos muito elevados e anula eventuais tendências menos acentuadas. Isto torna-se evidente ao utilizar a transformação para o plano *log-polar*, onde as orientações dos vectores já são mais dispersas. Relativamente ao fluxo óptico normal, representado no histograma 3, o resultado é o que já tínhamos

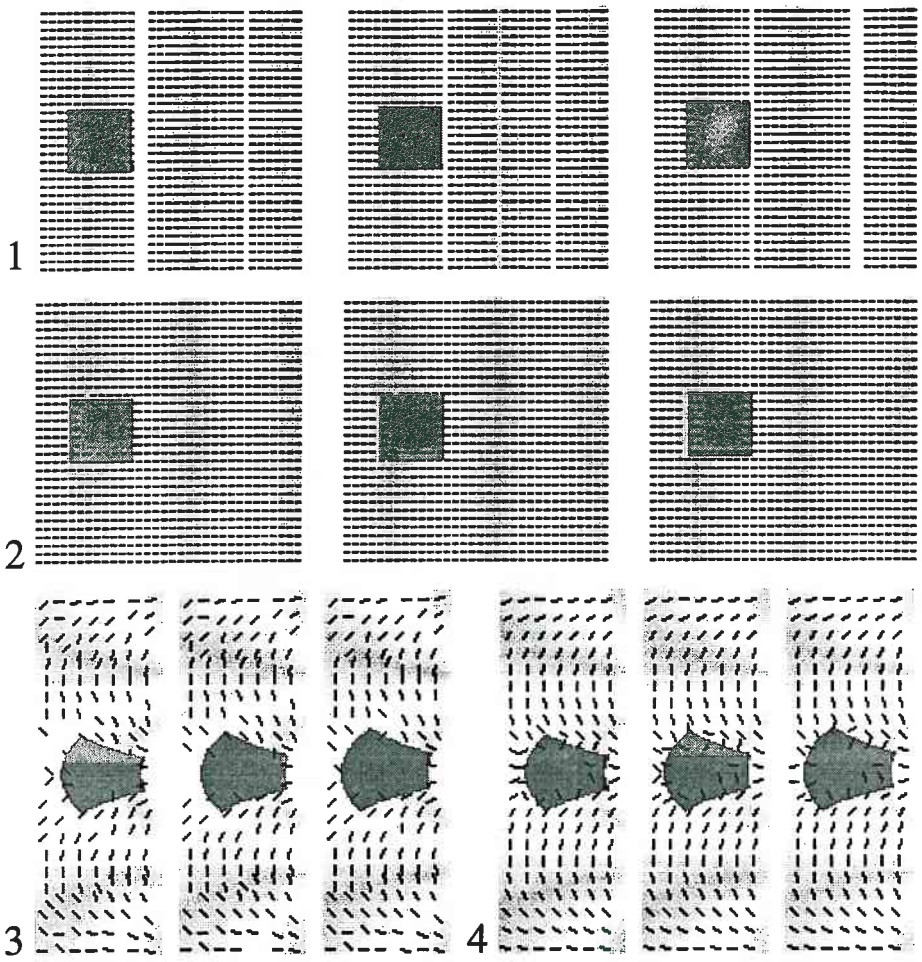



Figura 3.12: Resultado da aplicação do fluxo óptico à sequência de imagens da figura 3.11. Os resultados foram obtidos utilizando o fluxo óptico normal (1), o método de Horn & Schunk (2), o fluxo óptico normal em imagens *log-polar* (3) e o método de Horn & Schunk em imagens *log-polar* (4).

concluído para o caso das imagens originais, isto é, o movimento do fundo sobrepõe-se completamente ao movimento do quadrado, não sendo possível identificar este último. Finalmente, no histograma 4, temos mais uma vez uma supremacia dos vectores com orientações próximas dos 180°, mas desta vez com manifestações adicionais em torno dos 0° e dos 90°. Esses picos adicionais descrevem duas componentes de um movimento que, conjugado, corresponde a uma orientação de 45°, que é exactamente o sentido do movimento executado pelo quadrado. O facto desse movimento ter sido detectado em função de duas componentes fica a dever-se às características geométricas do quadrado.

Como foi já referido, os motores de passo que controlam o sistema de visão activa apenas podem ser controlados posicionalmente. No entanto, isso não impede que se tire partido das técnicas de cálculo do fluxo óptico para melhorar o desempenho do sistema. Uma forma de o fazer é, por exemplo, utilizar a informação de movimento para reduzir a área das imagens que iremos processar. Isto porque, em princípio, apenas nos interessa analisar as áreas para onde o alvo se movimentou. Se não conseguirmos obter, ainda que de forma pouco precisa, esta informação, resta-nos pesquisar o alvo a toda a extensão das imagens, com óbvios encargos computacionais. No capítulo 4, veremos como integrar a

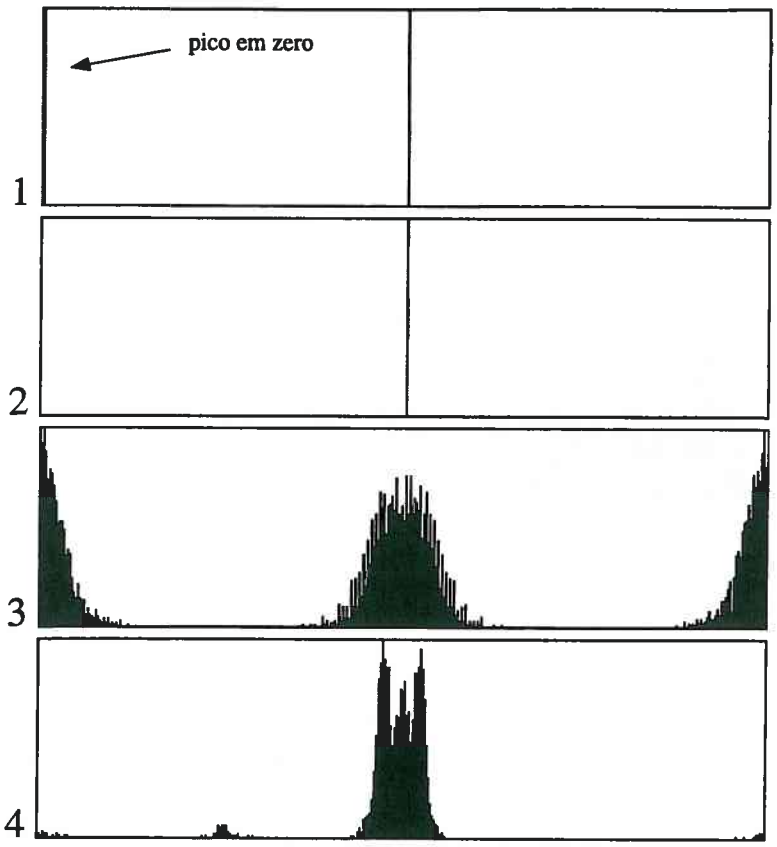



Figura 3.13: Histograma de orientações dos vectores de fluxo óptico representados na figura 3.12. O eixo horizontal dos histogramas corresponde ao ângulo que os vectores fazem com a horizontal, entre 0° até 360°. Note-se que, nos histogramas 3 e 4, o sentido dos vectores não corresponde directamente às imagens *log-polar*, mas sim ao ângulo que esses vectores fazem com a horizontal quando remapeados para um plano cartesiano.

informação de movimento no processo de fixação visual.

## 3.5 Detecção de Contornos

Entre as técnicas mais utilizadas no processamento de imagem, encontram-se as que permitem detectar os contornos das cenas. A sua identificação é útil à generalidade das áreas da visão por computador, desde o reconhecimento de padrões até à detecção de movimento. Também nós iremos fazer uso destas técnicas neste trabalho.

### 3.5.1 Detector de Roberts

Como regra, todas as técnicas de extracção de contornos estão sujeitas a um compromisso entre a qualidade dos resultados que geram e o peso computacional necessário para as realizar. O detector de Roberts é um dos mais antigos métodos conhecidos. Datando de 1965, é simples e rápido, calculando uma aproximação para o gradiente da intensidade luminosa em cada ponto da imagem. Uma vez que os contornos são zonas de acentuada variação da intensidade, uma pesquisa dos vectores gradiente de maior módulo equivale,

| +1 | 0 |
|---|---|
| 0 | -1 |

Gx

| 0 | +1 |
|---|---|
| -1 | 0 |

Gy

Figura 3.14: Máscaras que constituem o detector em cruz de Roberts.

Figura 3.15: As imagens central (módulos) e direita (ângulos) representam o resultado da aplicação do detector em cruz de Roberts à imagem de teste, à esquerda.

teoricamente, à pesquisa dos contornos existentes na imagem. Na prática, o detector limita-se à convolução de uma imagem com as duas máscaras representadas na figura 3.14.

A rapidez do detector de Roberts deve-se às reduzidas dimensões das máscaras propostas, também responsáveis pela sensibilidade ao ruído existente nas imagens. Na figura 3.15 podemos ver o resultado da sua aplicação. Note-se como as arestas menos nítidas não são detectadas em toda a sua extensão, ao contrário das mais visíveis, que deram origem a linhas muito espessas (de largura não unitária).

### 3.5.2 Selecção de Máximos Locais

No extremo oposto do universo de detectores de contornos, temos o detector de Canny [CAN83][CAN86]. Este método foi concebido e optimizado tendo em consideração alguns critérios, entre os quais se encontra a necessidade de obter uma única resposta a um único contorno. Por outras palavras, cada contorno deve ser detectado como uma única linha de largura unitária.

O algoritmo proposto por Canny é composto por várias fases, que serão abordadas aqui de forma muito sucinta. O processo começa com a aplicação de um filtro Gaussiano à imagem, seguido de um operador semelhante ao de Roberts para detectar o vector gradiente da intensidade em cada ponto. Calculado esse vector, é procurado um máximo local segundo a direcção do gradiente, uma vez que ela é, teoricamente, normal ao contorno em cada ponto (este processo é conhecido como *supressão de não-máximos*). Finalmente, os máximos locais são sujeitos a uma janela de aceitação, definida por dois limiares $L_1$ e $L_2$, com $L_1 > L_2$. Os que estiverem acima de $L_1$ pertencem definitivamente ao contorno. Os que, embora inferiores a $L_1$, sejam superiores a $L_2$ e estejam na vizinhança de um máximo *definitivo*, pertencem também ao contorno.

Como facilmente se conclui, a pesquisa de máximos locais e consequente processo de aceitação servem exactamente para estreitar a largura das curvas devolvidas pelas duas primeiras fases do algoritmo. Embora com um desempenho aceitável, o processo mostra-se demasiadamente pesado face ao nosso objectivo de celeridade. No entanto, o tratamento

dos máximos locais do gradiente é uma pista importante com vista à criação de um método alternativo. Veremos de seguida o funcionamento do método a que se chegou.

Uma abordagem simplista, mas imediata, seria a selecção pura e simples dos máximos locais do gradiente. Rapidamente se conclui que os resultados não seriam famosos, já que a magnitude do gradiente ao longo de um contorno só é constante em casos muito especiais. Os pontos que nós pretendemos não são apenas os máximos locais, mas sim todos os pontos ao longo do topo do que se pode imaginar como sendo uma cordilheira montanhosa. Adicionalmente, de entre todos os pontos que formam o topo, queremos apenas os que se encontram acima de um determinado limiar, para garantir a eliminação de contornos não existentes. Nestas condições, concebeu-se o seguinte algoritmo:

1. Para todos os pontos da imagem, repetir 2.
2. Se o módulo do gradiente estiver abaixo de um determinado limiar, o ponto não pertence ao contorno (voltar a 1).
3. Avaliar o módulo do gradiente nos oito pontos que rodeiam o ponto candidato.
4. Se há três ou mais vizinhos cujo módulo do gradiente é igual ou superior ao do ponto candidato, o ponto não pertence ao contorno (voltar a 1).
5. O ponto pertence ao contorno (voltar a 1).

O algoritmo anterior é uma simples regra para determinar se um ponto pertence ou não a uma curva de largura unitária. Tomemos, como exemplo, a curva da figura 3.16. Agora, seleccionemos aleatoriamente um grupo de nove pontos, formando um quadrado de $3 \times 3$. Note-se que, independentemente do grupo escolhido, obtemos no máximo três pontos da curva. Imaginando que a figura representa gradientes, estando os de maior módulo nas quadrículas pretas, torna-se evidente a razão de ser do passo 4 do algoritmo. De facto e de um modo geral, uma curva não possui mais do que três pontos em cada grupo de $3 \times 3$ pontos da imagem a que pertence.

O resultado da aplicação do algoritmo pode ser visto na figura 3.17. Os contornos estão agora definidos por curvas de largura quase unitária, se bem que à custa da sua contiguidade e de alguma irregularidade. No caso de a imagem ser ruidosa, uma simples alteração ao algoritmo melhorará significativamente os resultados. Basta modificar o passo 5 de forma a excluir pontos que não tenham um único vizinho com módulo superior. Note-se que, se por um lado podemos eliminar assim pontos esporádicos, por outro eliminamos também os máximos locais propriamente ditos.

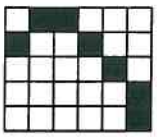


Figura 3.16: Representação de uma curva por pontos, tal como é definida numa imagem.

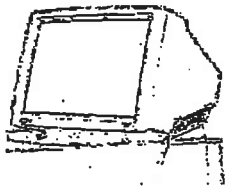


Figura 3.17: Resultado da selecção de máximos locais, após o uso do detector em cruz de Roberts.

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

Gx

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| +1 | +2 | +1 |

Gy

Figura 3.18: Máscaras que constituem o detector de Sobel.

### 3.5.3 Detector de Sobel

Para tentar reduzir a sensibilidade ao ruído e aumentar a precisão da detecção de contornos, fizeram-se testes adicionais com um operador bastante conhecido, designado por detector de Sobel. A aplicação deste detector é em tudo análoga à do detector em cruz de Roberts, com a particularidade de utilizar máscaras de $3 \times 3$. Na figura 3.18, podemos ver as duas máscaras utilizadas para calcular as componentes do vector gradiente, $G_x$ e $G_y$. Os resultados da utilização deste operador podem ser vistos na figura 3.19.

### 3.5.4 Outras Transformações

Não veremos agora os resultados que se obtêm quando se aplica a detecção de contornos às imagens transformadas para o plano *log-polar*. Esse estudo será feito mais tarde, no capítulo 6. Nessa altura, veremos como, da interligação das duas técnicas, resultam algumas propriedades interessantes. Faremos ainda algumas considerações sobre a forma de as explorar e veremos como a sua aplicação pode complementar o processo de fixação visual na tarefa de recuperação da estrutura tridimensional dos alvos.

Figura 3.19: As imagens esquerda (módulos) e central (ângulos) representam os resultados obtidos com a utilização do detector de Sobel. A imagem direita foi gerada após a selecção de máximos locais.

Figura 3.20: Aplicação do detector em cruz de Roberts, seguido do algoritmo de selecção de máximos locais, aos vários níveis de uma pirâmide de imagens reais (a imagem base da pirâmide original pode ser vista na figura 3.3, em cima).

Relativamente à utilização da detecção de contornos conjuntamente com a técnica das pirâmides, podemos ver na figura 3.20 o resultado da aplicação a uma pirâmide de imagens reais do detector de Roberts seguido da selecção de máximos locais. Como se pode observar, conforme os níveis da pirâmide vão aumentando, os pormenores realçados pela detecção de contornos são cada vez em menor número. Por outras palavras, ao subirmos na pirâmide estamos a detectar as variações mais acentuadas e de maiores dimensões existentes na imagem, ou seja, ao descermos na pirâmide, conseguimos visualizar mais facilmente os pequenos pormenores. Tal não é de estranhar se recordarmos que a construção de uma pirâmide é, por definição, a estruturação da imagem a vários níveis de resolução. No entanto, entre sabermos que determinada imagem é menos pormenorizada e conseguirmos efectivamente detectar os pormenores existentes, há uma grande distância. Associando as duas técnicas da forma que vimos, conseguimos não só anular essa distância, como também acelerar um eventual processo baseado na análise de imagens de contornos. Isto porque, no caso de os objectivos do processo estarem relacionados apenas com os contornos de maiores dimensões, as imagens dos níveis mais altos são suficientes, o que reduz o peso computacional da sua análise em face da menor resolução das imagens. Se pretendermos levar em conta características de menores dimensões, uma análise prévia a um nível mais alto permitirá sempre reduzir a área onde incidirá a análise de pormenor, resultando igualmente numa melhoria do desempenho dos algoritmos utilizados [LIN93]. Na secção seguinte veremos uma aplicação prática desta associação de técnicas.

## 3.6 Correlação de Imagens

### 3.6.1 Definição

No capítulo 2, definimos a fixação como o acto de concentrar a atenção de ambos os olhos no mesmo alvo. Depois, adaptámos a definição às duas câmaras existentes no sistema de visão activa, passando a fixação a ser a tarefa de movimentar as câmaras, de forma a que o alvo se projecte no centro das duas imagens. Uma vez que as câmaras não têm a mesma perspectiva da cena e o movimento do alvo não é conhecido, a localização da projecção do alvo nas imagens é uma incógnita. Daí que, com cada novo par de imagens captadas, se torna necessário procurar o alvo em cada uma delas, definindo assim a sua posição actual. O método tradicionalmente utilizado para o fazer é a correlação. Note-se que, além de ser usada para encontrar o alvo com o passar do tempo, a correlação é igualmente útil para garantir que o alvo escolhido pela câmara direita é igual ao escolhido pela esquerda. Esta situação é particularmente crítica aquando da selecção do alvo.

Em termos gerais, a correlação pode ser definida como um processo baseado na pesagem de intensidades. Dadas duas imagens de igual área, a função pesa entre si o valor dos pontos que formam cada uma das imagens, retornando uma medida da sua semelhança. A função matemática que descreve a correlação é relativamente complexa, levando em consideração características estatísticas das imagens que, simultaneamente, lhe conferem uma robustez ímpar e um peso computacional elevado. Sejam $D$ e $E$ duas imagens de igual área $A$, podemos definir analiticamente a correlação da seguinte forma:

$$C = \frac{\sum_{i,j \in A} D(i,j)E(i,j)}{\sqrt{\sum_{i,j \in A} D^2(i,j) \sum_{i,j \in A} E^2(i,j)}} \tag{3.19}$$

Os valores possíveis para $C$ encontram-se sempre dentro do intervalo de 0 a 1, sendo a máxima semelhança atingida para $C = 1$. Embora pouco sensível ao ruído e às diferenças de luminosidade das imagens, esta função tem a desvantagem de ser computacionalmente muito pesada.

### 3.6.2 Algumas Simplificações

Existem inúmeros métodos alternativos para calcular a correlação [PEA77]. Uma das funções mais simples e célere é a soma dos quadrados das diferenças (SQD). Uma aproximação comum a essa função é a soma dos módulos das diferenças (SMD), com desempenhos semelhantes, mas computacionalmente menos pesada. Sejam $D$ e $E$ duas imagens de igual área $A$, podemos definir analiticamente as duas funções da seguinte forma:

$$C_{sqd} = \sum_{i,j \in A} (D(i,j) - E(i,j))^2 \tag{3.20}$$

$$C_{smd} = \sum_{i,j \in A} |D(i,j) - E(i,j)| \tag{3.21}$$

Note-se que, nestes casos, quanto menor é o valor de $C$, maior é a semelhança das áreas, atingindo-se a máxima semelhança para $C = 0$. Como se poderá imaginar, sem processamentos adicionais, ambas as funções descritas atrás apresentam grande sensibilidade a factores como os valores médios da intensidade dos pontos de cada imagem. O problema reside, portanto, em compensar de alguma maneira estas desvantagens, evitando

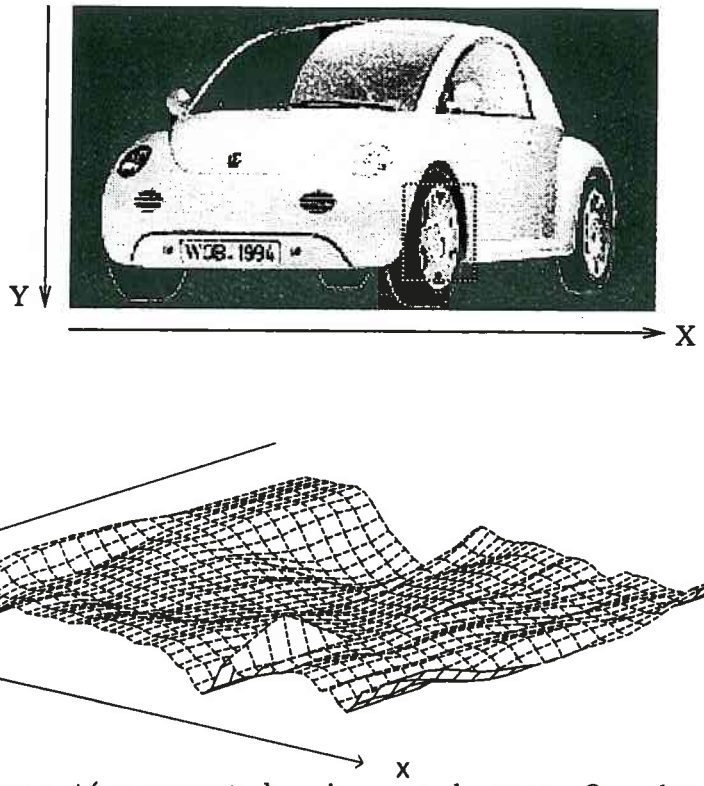



Figura 3.21: Em cima está representada a imagem de teste. O padrão seleccionado é a parte da roda dianteira limitada por uma caixa. Em baixo podemos ver os resultados do primeiro exemplo. A posição do ponto de máxima correlação coincide com o local de onde foi extraído o padrão.

a todo o custo ter de recorrer a funções mais complexas. Felizmente, o simples facto de tentarmos obter a maior frequência possível de captura de imagens é, por si só, um passo nesse sentido, pois as imagens adquiridas em instantes temporais próximos têm luminosidades semelhantes. Por outro lado, uma vez que as câmaras são iguais, é relativamente fácil regular as suas íris, de forma a obter aproximadamente a mesma luminosidade nas imagens capturadas por cada câmara. Até que ponto são estas condições suficientes para garantir bons resultados? Como já foi referido, o nosso objectivo não é obter métodos de desempenho irrepreensível, mas que acarretariam um considerável esforço computacional. Da forma como foi apresentada, a correlação é-nos útil, sem representar um encargo insuportável.

De seguida veremos três exemplos da aplicação da correlação e deles salientaremos algumas particularidades. Para a função de semelhança será utilizada a soma dos módulos das diferenças. Em princípio, esta deveria ser a função mais célere. Curiosamente, nos DSPs, calcular o módulo ou o quadrado demora sempre um ciclo de relógio, pelo que o uso da função soma dos quadrados das diferenças resulta na mesma velocidade de execução.

### 3.6.3 Exemplos e Testes

Na figura 3.21 (em cima), podemos ver a imagem de teste utilizada no primeiro exemplo. Para efectuar o teste, vamos escolher uma área dessa imagem, à qual chamaremos padrão. Neste caso, o padrão é uma parte da roda dianteira do veículo, limitada na figura por uma caixa pontilhada. De seguida, vamos correlacionar o padrão com zonas de igual

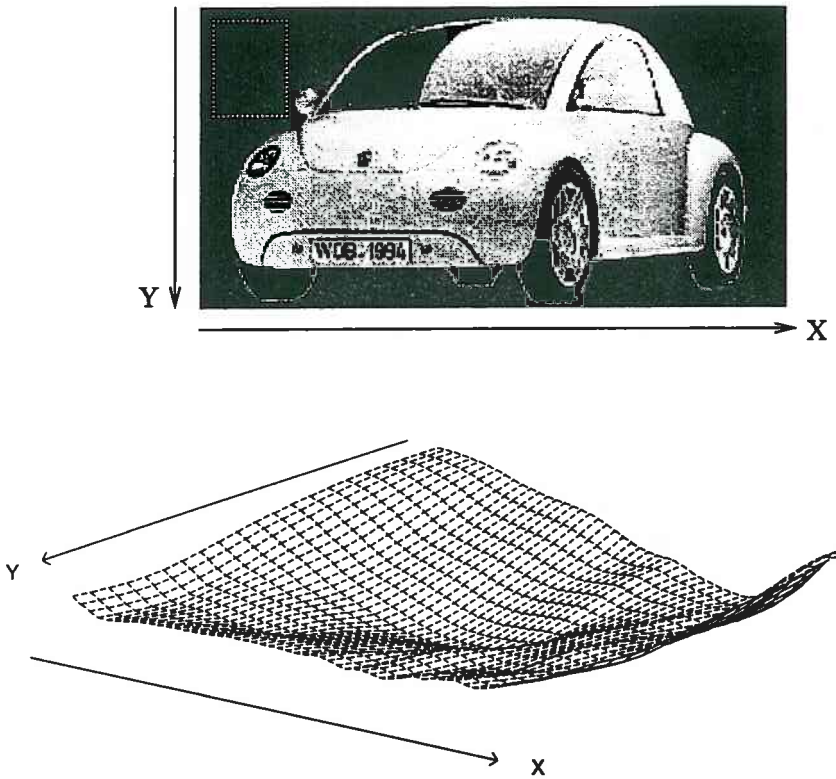



Figura 3.22: Em cima, está representada a imagem de teste. O padrão seleccionado é uma área da imagem sem textura. Em baixo, podemos ver os resultados do segundo exemplo e as várias áreas de máxima correlação obtidas.

área, em vários locais escolhidos da imagem e nela distribuídos uniforme e regularmente. Finalmente, utilizamos os valores obtidos para construir uma superfície. Porém, esta será representada e analisada invertida, uma vez que, com a função utilizada, quanto maior é a semelhança das áreas, menores são os valores obtidos.

Os resultados do primeiro exemplo são visíveis na figura 3.21 (em baixo). A conclusão mais imediata que podemos tirar, aliás já previsível, é que a posição do ponto de máxima correlação coincide com o local da imagem de onde o padrão foi extraído. Importa notar que, embora o máximo esteja destacado, na sua vizinhança foram obtidos valores igualmente altos. Por um lado, esta característica é uma vantagem, já que permite a localização do padrão mesmo quando as áreas correlacionadas não são exactamente iguais. No entanto, ela é também responsável pela possível obtenção de vários máximos locais, ou equivalentemente, pela detecção de mais do que uma zona semelhante ao padrão, como veremos de seguida.

No segundo exemplo, vamos utilizar a mesma imagem de teste, mas seleccionando um padrão diferente. Analisando a figura 3.22 (em cima), podemos ver que a área escolhida para padrão não possui características que a individualizem, ou seja, não tem textura. Repetindo o processo de teste utilizado no primeiro exemplo, chegamos aos resultados representados nas figura 3.22 (em baixo). Como se pode ver, na área da imagem onde se localiza o padrão temos várias zonas de máxima correlação, bem como na área oposta. Note-se que nem sequer é possível localizar o padrão extraindo o máximo absoluto. Este exemplo demonstra bem a importância da selecção de um padrão conveniente.

Para assegurarmos a existência de textura no padrão, podemos recorrer à detecção de contornos, tal como foi descrita na secção anterior. Utilizando a mesma imagem de

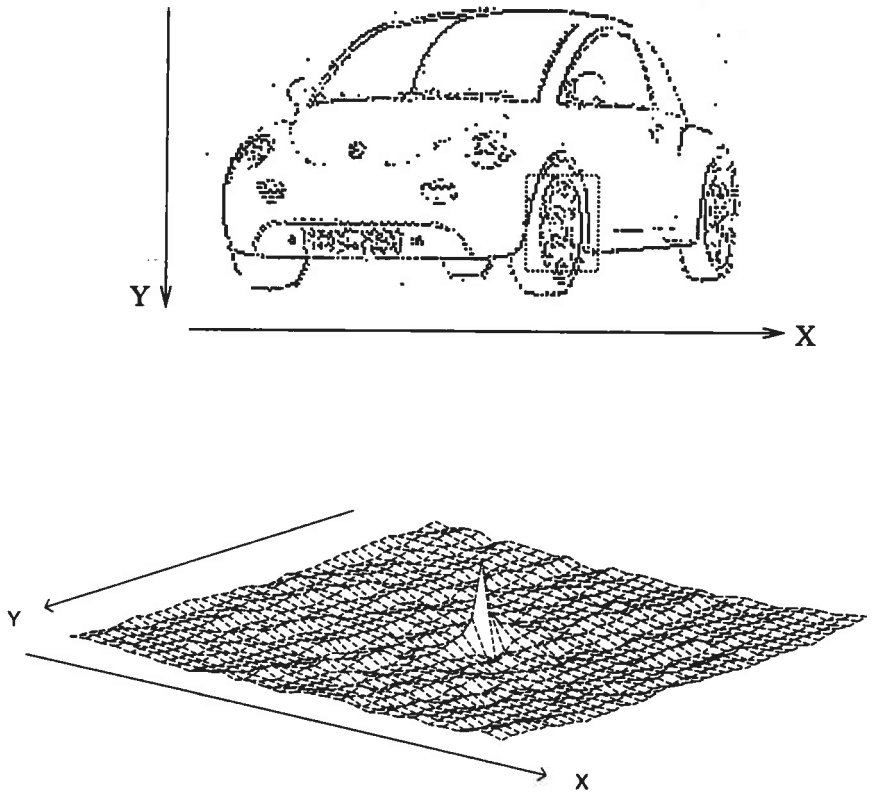



Figura 3.23: Em cima vemos a imagem de teste depois de sujeita ao detector em cruz de Roberts seguido da selecção de máximos locais. O padrão seleccionado é a parte da roda dianteira limitada por uma caixa. Em baixo representam-se os resultados obtidos, sendo visível o ponto de máxima correlação, que está destacado.

teste e aplicando o detector em cruz de Roberts e a selecção de máximos locais, obtém-se a imagem representada na figura 3.23 (em cima). Recordando a localização do padrão escolhido no segundo exemplo, é agora bem notória a ausência total de textura. Quanto ao padrão utilizado no primeiro exemplo, delimitado na figura por uma caixa pontilhada, temos exactamente o caso oposto.

Finalmente, vamos repetir o primeiro exemplo, utilizando agora a imagem de contornos da figura 3.23 (em cima). Tanto a localização do padrão como o processo de teste são os mesmos. A única diferença é que não será pesquisada toda a imagem, mas sim uma pequena área em torno do padrão. As posições das zonas correlacionadas são agora compactas, não havendo espaçamentos entre elas. Nestas condições, obtemos os resultados representados na figura 3.23 (em baixo). Como se pode ver, não há qualquer margem para ambiguidades. A aplicação da correlação a imagens de contornos é um processo bastante eficaz, com a contrapartida de exigir grande semelhança entre as imagens correlacionadas. O facto de o máximo estar isolado, havendo poucos vizinhos com um valor de correlação significativo, obriga a uma pesquisa densa da imagem.

### 3.6.4 Processo Híbrido

Nesta secção, vimos exemplos bem representativos de algumas das principais características da correlação, que iremos agora recapitular. A primeira conclusão a tirar é que, em imagens bem contrastadas e sem áreas semelhantes, a eficácia do método é incontestável, tal

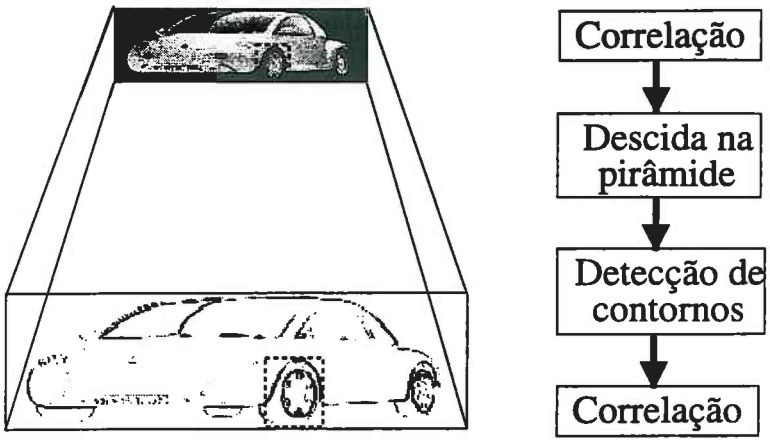



Figura 3.24: Processo cooperativo de correlação a vários níveis da pirâmide. Primeiro, pesquisamos a imagem a um nível mais alto e de menor resolução. Detectado um máximo, descemos na pirâmide, aplicamos a detecção de contornos à área em que foi encontrado o máximo e fazemos nova pesquisa.

como se pode ver pelos resultados obtidos no terceiro exemplo (figura 3.23). É claro que a obtenção de imagens com as características descritas não é trivial, embora a utilização de imagens de contornos pareça prometedora. Por outro lado, o facto de, na vizinhança do ponto de máxima correlação, não haver qualquer indicação de que se está na presença de um pico, faz temer os resultados que se obterão quando o algoritmo de detecção de contornos não gerar imagens aceitáveis. Deixando por momentos este problema, importa referir que, mesmo utilizando imagens não tratadas, a correlação permite detectar facilmente áreas texturadas, tal como vimos no primeiro exemplo (figura 3.21). Embora o máximo não esteja tão destacado como no terceiro exemplo, a sua identificação não oferece qualquer dificuldade, não só devido ao máximo em si, mas também a todos os valores na sua vizinhança, nitidamente mais elevados do que os restantes. Podemos assim pensar num processo de associar a robustez do primeiro exemplo com a precisão do terceiro, efectuando a pesquisa em duas fases: primeiro detectamos um pico aplicando a correlação sobre imagens não tratadas, afinando depois a sua localização aplicando a detecção de contornos à imagem, e repetindo a correlação a um nível mais baixo da pirâmide de resoluções. Aliás, foi exactamente isso que fizemos, já que o terceiro exemplo, como se referiu na altura, resultou da pesquisa de apenas uma pequena área da imagem em torno do padrão (ver a figura 3.24).

Finalmente, o ponto fraco do processo é, como vimos no segundo exemplo, a definição de padrões sem características identificativas. Tal não deveria ser de estranhar, no entanto, já que a correlação é, por definição, a localização de áreas constituídas por pontos com níveis semelhantes de intensidade luminosa. Daí que, se o padrão não possuir textura suficiente para o diferenciar, o resultado da sua pesquisa na imagem terá de ser a obtenção de vários máximos nas zonas pouco texturadas. Mais do que limitar a aplicação da correlação, este facto deverá ser interpretado apenas como um cuidado a ter sempre que é necessário extrair um padrão de uma imagem.

# Capítulo 4

# Simulação do Processo de Fixação

No capítulo anterior, vimos diversas técnicas que nos irão ser úteis para realizar o processo de fixação descrito no capítulo 2. Agora o objectivo é definir um conjunto de regras e algoritmos básicos, capazes de levar à prática os diversos estados que compõem o processo. Para além de identificar os métodos, importa determinar a forma como eles interagem, bem como a sequência com que devem ser aplicados. Ao longo deste capítulo, não se tem como meta uma eventual aplicação, mas sim a elaboração da espinha dorsal das possíveis aplicações do processo. A concretização dessas aplicações é o tema dos próximos dois capítulos, onde analisaremos exemplos concretos, bem como as necessárias alterações e adaptações em função dos casos. Deixaremos para esses capítulos algumas questões de ordem prática, directamente relacionadas com o *hardware* disponível e com os objectivos das aplicações em causa, como sejam o controlo do sistema, a definição do seu modelo matemático e o diagrama temporal do processo.

## 4.1 Resumo do Processo

Recapitulando: na nossa análise do processo de fixação definimos três estados fundamentais, que são o estado de *Espera*, o estado de *Selecção* e o estado de *Fixação*. No primeiro, o sistema deve aguardar que uma alteração na cena lhe desperte a atenção. Ao receber um estímulo, provavelmente resultante do movimento de um alvo, o processo avança para o estado de *Selecção*. Aqui deverá decidir se a causa do movimento é interessante.

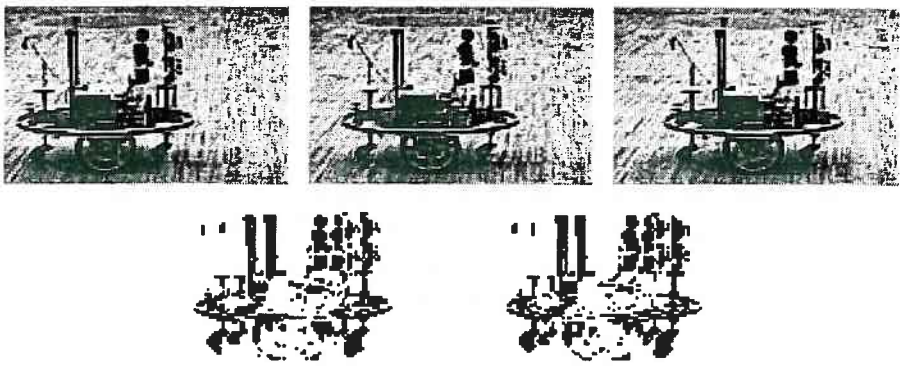


Figura 4.1: Sequência de imagens descrevendo o movimento de um *robot* e respectivas imagens de diferenças.

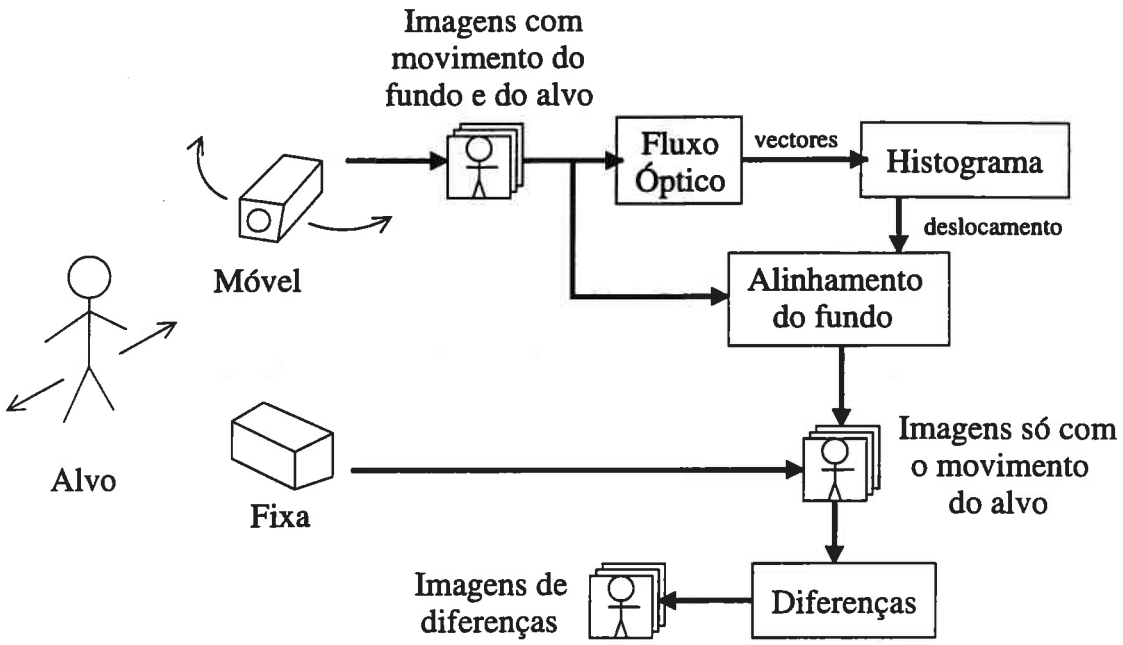



Figura 4.2: Diagrama de blocos para o estado de *Espera*. Se as câmaras estiverem imóveis, é suficiente o cálculo das imagens de diferenças. No caso de haver movimento, é necessário detectar a amplitude e a direcção do movimento do fundo, para que o seu deslocamento possa ser compensado antes de se efectuar a subtracção das imagens.

Obviamente, este conceito poderá ter diversas traduções, em função dos objectivos em causa. No entanto, há algumas características que poderão ser consideradas como comuns a todos os alvos "interessantes", como seja a necessidade de terem dimensões razoáveis relativamente à área da imagem. Tipicamente, o estado de *Espera* não fará qualquer tipo de análise dos alvos, limitando-se a reagir a um estímulo ocorrido em ambas as câmaras. Daí que seja fundamental verificar se o estímulo foi causado pelo mesmo alvo em cada uma das sequências de imagens. Esta tarefa é da responsabilidade do estado de *Selecção*. Ultrapassada a fase de selecção, atingimos o estado de *Fixação*, durante o qual o sistema deverá concentrar-se no alvo escolhido. Mais uma vez, o conceito tem diferentes significados consoante as situações. No entanto, em termos gerais, este estado implicará uma pesquisa das imagens obtidas pelas câmaras, em busca do alvo seleccionado. Nas próximas secções iremos analisar em pormenor cada um dos estados.

## 4.2 *Espera*

Recordando a descrição do estado de *Espera*, o objectivo é encontrar alterações nas sequências de imagens obtidas com cada uma das câmaras. O processo mais simples para fazê-lo é subtrair a imagem obtida no instante $T-1$, da imagem obtida no instante $T$. De um ponto de vista óptimo, se não houver alterações na cena, não há diferenças nos níveis de intensidade dos pontos de uma imagem para a outra, pelo que o resultado dessa subtracção é uma imagem nula. Isto, é claro, se o sistema de visão activa não estiver em movimento. É esse o caso que iremos analisar em primeiro lugar: sempre que o processo está no estado de *Espera*, as câmaras terão de estar em repouso relativamente à cena. Evita-se, assim, a necessidade de distinguir entre os movimentos reais dos alvos e os mo-

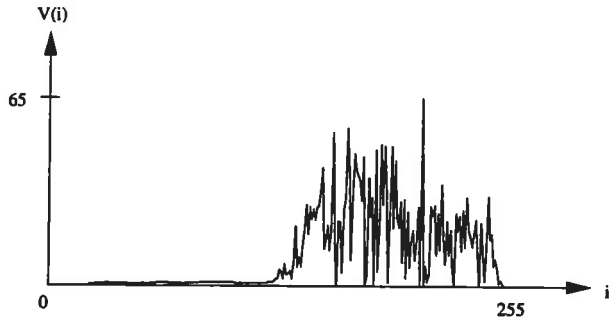



Figura 4.3: Variações médias nos diferentes níveis de intensidade luminosa de uma cena estática.

vimentos aparentes provocados pela deslocação das câmaras. A simplificação do problema originada por esta opção permite obter uma alta frequência de captação de imagens no estado de *Espera*, motivada pela menor complexidade das técnicas envolvidas. Podemos, assim, esperar que o tempo gasto no estado de *Espera* seja reduzido, o que permitirá atingir o estado de *Fixação* mais rapidamente.

Quando um alvo se movimenta, se as câmaras estiverem estáticas, as únicas alterações na sequência de imagens são as que resultam da mudança de posição do alvo. Se subtrairmos ponto a ponto duas imagens consecutivas dessa sequência, o efeito na imagem de diferenças equivale, em certa medida, à sobreposição de duas imagens do alvo. Essas imagens deverão estar ligeiramente destacadas em função da amplitude do movimento e posicionadas sobre um fundo uniforme (ou melhor dizendo, nulo), resultante da subtracção de pontos de igual intensidade. Na sequência representada na figura 4.1, podemos ver esse efeito. Como se pode concluir pelas imagens, o alvo em movimento está perfeitamente isolado do fundo, sendo trivial a tarefa de o localizar.

Menos simples é a análise dos casos em que as câmaras se movimentam. À partida, calcular a diferença das imagens nessas situações é inútil, já que, do ponto de vista das câmaras, toda a imagem se movimentou. Os métodos mais usuais para abordar este problema baseiam-se no cálculo do fluxo óptico. De um modo geral, parte-se do princípio que o movimento do fundo das imagens é uniforme. Adicionalmente, considera-se que a área da imagem ocupada pelo fundo é consideravelmente maior do que a área ocupada pelo alvo. No capítulo anterior, vimos uma situação em que estas condições foram reproduzidas com imagens sintéticas (ver a figura 3.11). Como se viu na altura, o movimento do fundo foi facilmente identificável depois de se construir um histograma de orientações dos vectores de fluxo óptico. Partindo dessa informação podemos, por exemplo, deslocar as imagens antes de efectuar a sua subtracção, conseguindo com isso o alinhamento dos fundos e a sua consequente anulação. É esta a ideia na base de um método bastante conhecido para determinar dois movimentos a partir de uma sequência de três imagens [BER90].

Na figura 4.2, esquematiza-se o diagrama de blocos do estado de *Espera* resultante da análise feita até aqui. Note-se que o diagrama descreve apenas o processo a aplicar para uma das câmaras, já que é equivalente para a outra.

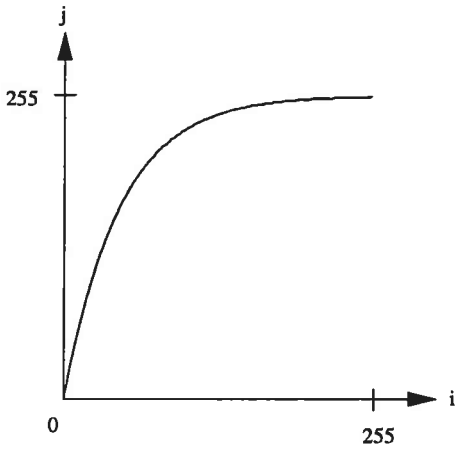



Figura 4.4: Curva de conversão de intensidades.

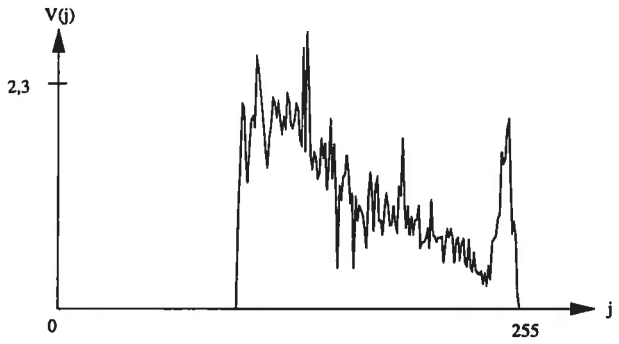



Figura 4.5: Variações médias nos diferentes níveis de intensidade luminosa de uma cena estática, após a conversão do intervalo de intensidades.

### 4.2.1 Luminosidade

Vamos abrir aqui um parêntesis para abordar um problema que se fica a dever ao ruído e às variações de luminosidade. Para se ter uma ideia da sua dimensão, atente-se ao gráfico da figura 4.3. No gráfico, $i$ representa os níveis de intensidade existentes numa imagem e $V(i)$ representa a variação média sofrida pelos pontos da imagem com esse nível de intensidade, ao longo de uma sequência de 20 imagens da mesma cena. Por outras palavras, adquiriram-se 20 imagens de uma mesma cena e contabilizaram-se as variações ocorridas nos vários pontos, em função das suas intensidades iniciais. Recorde-se que não há qualquer tipo de movimento na cena. Como se vê pelos resultados, os pontos de maior intensidade sofrem fortes variações ao longo do tempo. Uma vez que o nosso objectivo é detectar as variações ocorridas nas imagens, estas condições não nos permitem fazê-lo uniformemente em toda a gama de intensidades.

A cena que originou o gráfico da figura 4.3 continha algumas lâmpadas acesas, às quais se ficam a dever as variações na luminosidade. Para reduzir a influência das fontes de luz, vamos transformar os níveis de intensidades das imagens utilizando uma função de características logarítmicas (ver a figura 4.4). Conseguimos assim que o intervalo de intensidades $i$ original se torne menos sensível para valores elevados, amplificando ainda as variações na gama de valores mais baixos.

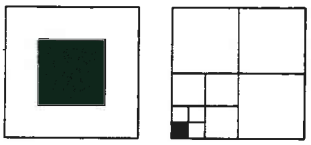


Figura 4.6: Representação das áreas limite para a aceitação de um alvo, relativamente à imagem total.

Repetindo a experiência com a mesma cena, mas com o intervalo de intensidades convertidas $j$, obtemos os resultados representados no gráfico da figura 4.5. Como se pode ver, as variações médias baixaram substancialmente. Os seus valores são agora equivalentes aos que se poderiam esperar devido ao ruído. Podemos, assim, filtrar tanto as variações de luminosidade como o ruído, definindo um limiar mínimo para as variações de intensidades, que poderá ser constante ao longo de toda a gama. Assim que surgir uma variação acima desse limiar, é entendida como um estímulo e passa-se ao estado de *Selecção*.

Para terminar, falta apenas referir a forma como o processo de conversão é executado nos DSPs. Na verdade, não podia ser mais simples. Ao serem digitalizados, os níveis de intensidades são convertidos automaticamente pelos DSPs, consultando uma tabela alterável em função das necessidades, não representando por isso qualquer peso ou atraso no processamento.

## 4.3 *Selecção*

Do estímulo que originou a passagem do estado de *Espera* para o estado de *Selecção*, sabemos apenas que se verificou em ambas as câmaras e que o alvo que o originou pode ser encontrado na sequência de imagens de diferenças. Como vimos anteriormente, a identificação do alvo numa imagem de diferenças é um processo trivial, desde que se tenha tido o cuidado de filtrar as imagens de diferenças, de forma a anular as alterações de intensidades provocadas pelo ruído. Se assim for, todos os pontos não nulos das imagens de diferenças pertencem ao alvo. Partindo deste ponto, a nossa primeira tarefa será identificar a área da imagem ocupada pelo alvo.

Por questões de simplicidade, apenas nos vamos preocupar em calcular áreas rectangulares. Isto é, para todos os efeitos, a área da imagem ocupada pelo alvo corresponde ao rectângulo que o circunscreve. Em termos práticos, aquela área é simplesmente o mais pequeno rectângulo que contém simultaneamente todos os pontos não nulos da imagem de diferenças (ver as figuras 4.7a e 4.7b).

Depois de definido o rectângulo, é imediato o cálculo da sua área, atingindo-se as primeiras condições de selecção a que o alvo é sujeito durante o estado de *Selecção* (ver a figura 4.6):

- A área não pode ser muito pequena. Tipicamente, áreas inferiores a $1/64$ (à direita na figura 4.6) da área total da imagem não são desejáveis. O alvo que causou um estímulo tão reduzido deverá estar muito longe, ou ser muito pequeno. Um alvo nestas condições é, geralmente, pouco interessante do ponto de vista computacional. Numa imagem com uma resolução de $256 \times 256$ pontos, a área mínima terá $32 \times 32$ pontos.

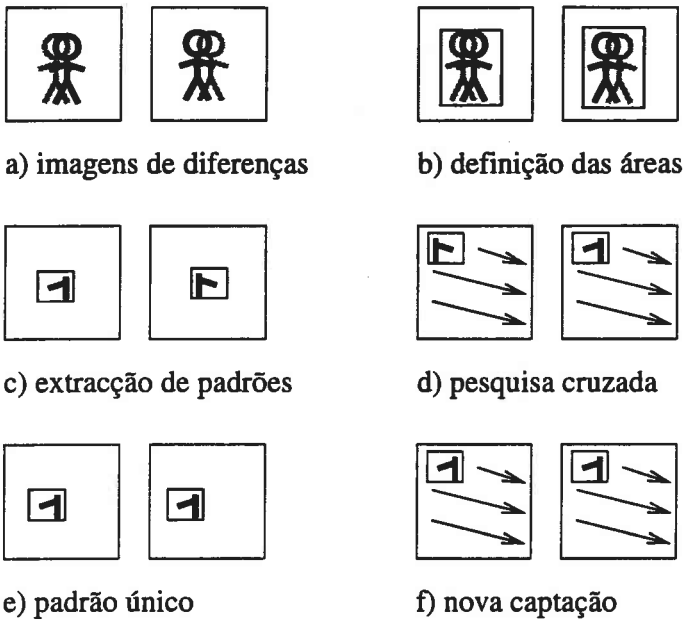



Figura 4.7: Esquematização das diferentes fases dos estados de *Selecção* (figuras de *a* a *d*) e de *Fixação* (figuras *e* e *f*). Partindo das imagens de diferenças (a), determina-se o menor rectângulo onde, em cada uma delas, se inscrevem todos os pontos dos alvos (b). De seguida, extraem-se áreas com dimensões pré-definidas de cada uma das imagens, centradas nos rectângulos respectivos, a que se chamará padrões (c). Neste caso, supondo que o alvo está de costas, foi extraído o braço esquerdo na imagem esquerda e o braço direito na imagem direita. Cada um dos padrões é depois pesquisado na imagem contrária, com vista a assegurar que o alvo que originou os estímulos é o mesmo (d). Por outras palavras, procura-se o braço direito na imagem esquerda e vice-versa. Caso ambos os padrões sejam encontrados, considera-se que o alvo é o mesmo. Finalmente, elimina-se o padrão que representa o braço direito (poderia ser o outro), substituindo-o pela área da imagem direita onde foi encontrado o braço esquerdo (e). Obtém-se assim um padrão de cada imagem representando a mesma zona do alvo. Estes padrões são usados a cada nova captação de imagens para localizar o alvo (f).

- A área não pode ser muito grande. Se a área for superior a 1/4 (à esquerda na figura 4.6) da área total da imagem, o alvo deverá estar muito perto, ou ser muito grande. Alternativamente, poderá ter ocorrido um movimento não esperado das câmaras, provocando a deslocação aparente de toda a cena. Em qualquer dos casos, iremos considerar que o alvo não é candidato à fixação. Numa imagem com uma resolução de 256 × 256 pontos, a área máxima terá 128 × 128 pontos.

Como foi referido anteriormente, a selecção dos alvos em função da sua dimensão, relativamente à área total da imagem, é um factor considerado como comum a qualquer aplicação que utilize o processo de fixação visual. Outro é a necessidade de ter ambas as câmaras orientadas para o mesmo alvo. De facto, embora o nosso sistema seja binocular, até agora o processo não implicou qualquer tipo de associação dos dados recolhidos com cada uma das câmaras. De seguida veremos como fazer essa interligação.

A determinação das áreas que circunscrevem o alvo em cada uma das imagens define, implicitamente, a localização do alvo nesses instantes. Por outras palavras, a posição dos

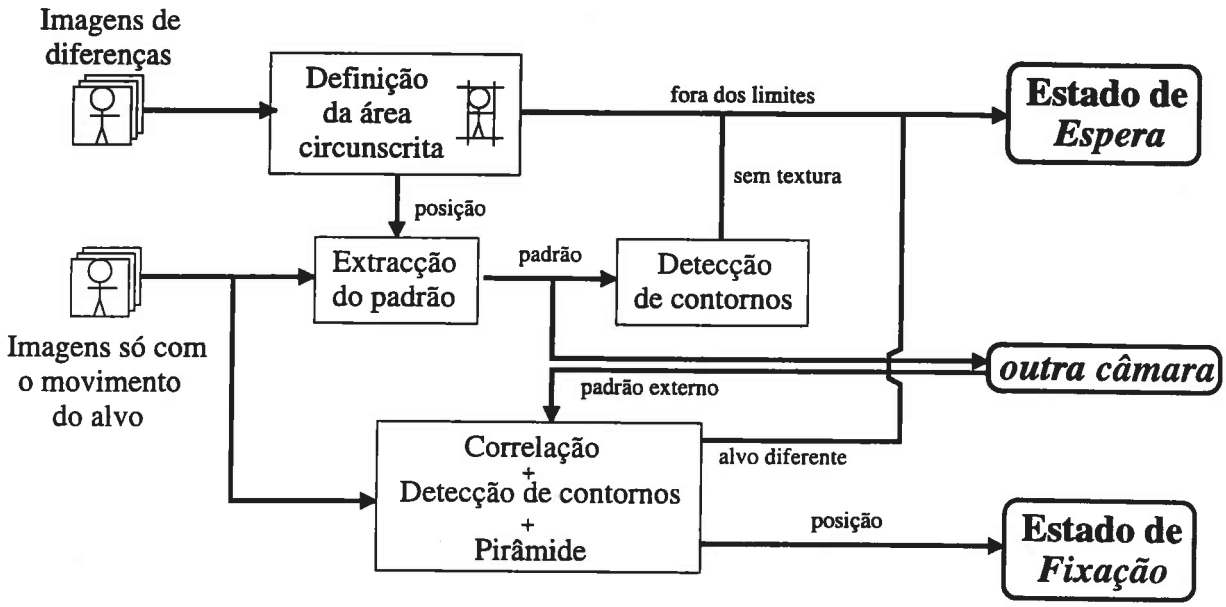



Figura 4.8: Diagrama de blocos para o estado de *Selecção*. O estado de *Selecção* herda do estado de *Espera* duas sequências de imagens: uma onde o fundo está alinhado, existindo apenas o movimento do alvo (sequência base) e outra que resulta da subtracção das imagens base. Utilizando as imagens de diferenças, é determinada a área dos alvos e a sua localização. Se a área estiver dentro de determinados limites, extrai-se um padrão de dimensões pré-definidas da sequência base, centrado na posição definida para o alvo. Se o padrão for texturado, é transferido para o bloco que processa as imagens captadas pela outra câmara, ao mesmo tempo que se recebe um padrão desse bloco. O padrão recebido é então pesquisado nas imagens base, recorrendo à correlação a vários níveis da pirâmide e à detecção de contornos, tal como se viu no capítulo anterior.

rectângulos calculados sobre as imagens de diferenças, pode ser utilizada para extrair, da sequência de imagens originais, as imagens do alvo. Obtidas essas imagens, podemos compará-las duas a duas, uma de cada câmara, utilizando a técnica da correlação de imagens que vimos no capítulo anterior. Optando por uma abordagem simplista, poderíamos limitar-nos a correlacionar as imagens do alvo, inferindo da sua semelhança com base no resultado da função de pesagem. O facto da dimensão das áreas ser, muito provavelmente, diferente, poderia ser resolvido definindo uma área de compromisso. No entanto, nunca seria possível garantir que essa área correspondia à mesma zona do alvo em ambas as imagens. Além disso, dependendo da forma do alvo, a área poderia conter uma grande parte da área do fundo que enquadra o alvo, influenciando o resultado da correlação. Fica, assim, claro que a aplicação da correlação terá de ser efectuada com base em áreas menores.

Em conformidade com as condições impostas no início da secção, resolvemos o problema começando por considerar que o alvo tem sempre uma área igual ou superior a 1/64 da área total da imagem. É esta a área das imagens que iremos correlacionar, independentemente do tamanho do rectângulo que circunscreve o alvo (que, por força dos limites impostos, terá sempre uma área maior). De seguida, vamos assumir que o centro da área detectada em cada imagem coincide com o centro do alvo nessa imagem. Nestas condições, extraímos de cada imagem uma área a que chamaremos padrão, cujo centro

deverá coincidir aproximadamente com o centro assumido para o alvo nessa imagem (ver a figura 4.7c). Ficamos assim na posse de dois padrões, que representam a zona central dos alvos detectados em cada uma das imagens. Bastará agora pesquisar, em cada uma das imagens, o padrão extraído da outra, utilizando, como já foi dito, a correlação (ver a figura 4.7d). Se a pesquisa for bem sucedida em ambos os casos, concluímos que o centro assumido para os alvos é visível em ambas as imagens, pelo que se poderá assumir que o alvo detectado é o mesmo.

Embora simples, este processo revela-se extremamente eficaz. É claro, tal como concluímos quando foi analisada a correlação, a definição de padrões texturados é crucial para o desempenho do método. Com vista ao aumento da robustez do estado de *Selecção*, impõe-se o estabelecimento de uma condição adicional, que é a de existência de textura nos padrões extraídos das imagens. Para tal, podemos recorrer à detecção de contornos que, como vimos no capítulo anterior, tem a vantagem adicional de nos permitir acelerar o processo global de pesquisa quando utilizada conjuntamente com as pirâmides de imagens. Na figura 4.8, podemos ver a diagrama de blocos correspondente ao estado de *Selecção*. Tal como anteriormente, apenas se esquematiza o diagrama de blocos para uma das câmaras. Note-se, no entanto, que neste caso há uma interferência dos dados recolhidos pela outra câmara, já que os padrões extraídos das imagens são trocados entre os dois blocos de processamento.

## 4.4 *Fixação*

Recapitulando, ao atingirmos o estado de *Fixação* temos, para cada uma das câmaras, a seguinte informação:

1. A localização e dimensão da área onde foi detectado um estímulo.
2. O deslocamento do fundo da imagem (fluxo óptico da cena que enquadra o alvo).
3. A sequência de imagens originais, a sequência de imagens alinhadas (iguais às originais quando não há movimento das câmaras) e a sequência de imagens de diferenças.
4. Um padrão representativo da área central do alvo.
5. Um padrão representativo da área central do alvo obtido com a outra câmara.
6. A localização, nas imagens alinhadas, do padrão obtido com outra câmara.

Por questões que se prendem com o conceito de fixação, é conveniente que os padrões extraídos de cada imagem pertençam à mesma zona do alvo. Para tal, vamos definir uma das câmaras como dominante, mantendo o padrão extraído na imagem captada por essa câmara. Quanto à outra câmara, o seu padrão é ignorado, sendo substituído pela zona onde foi encontrado o padrão dominante (ver a figura 4.7e). Nos humanos, o conceito de olho dominante está associado a factores que se prendem com a acuidade visual do indivíduo e com a forma com que o cérebro trata a informação. No nosso caso, a escolha da câmara dominante é aleatória, já que não introduz, à partida, qualquer tipo de alteração no funcionamento do processo.

Na sua forma mais simples, o estado de *Fixação* poderá resumir-se à pesquisa, em cada novo par de imagens captadas, dos padrões seleccionados, utilizando para tal o já familiar processo de correlação (ver a figura 4.7f). Conseguimos com isto detectar o

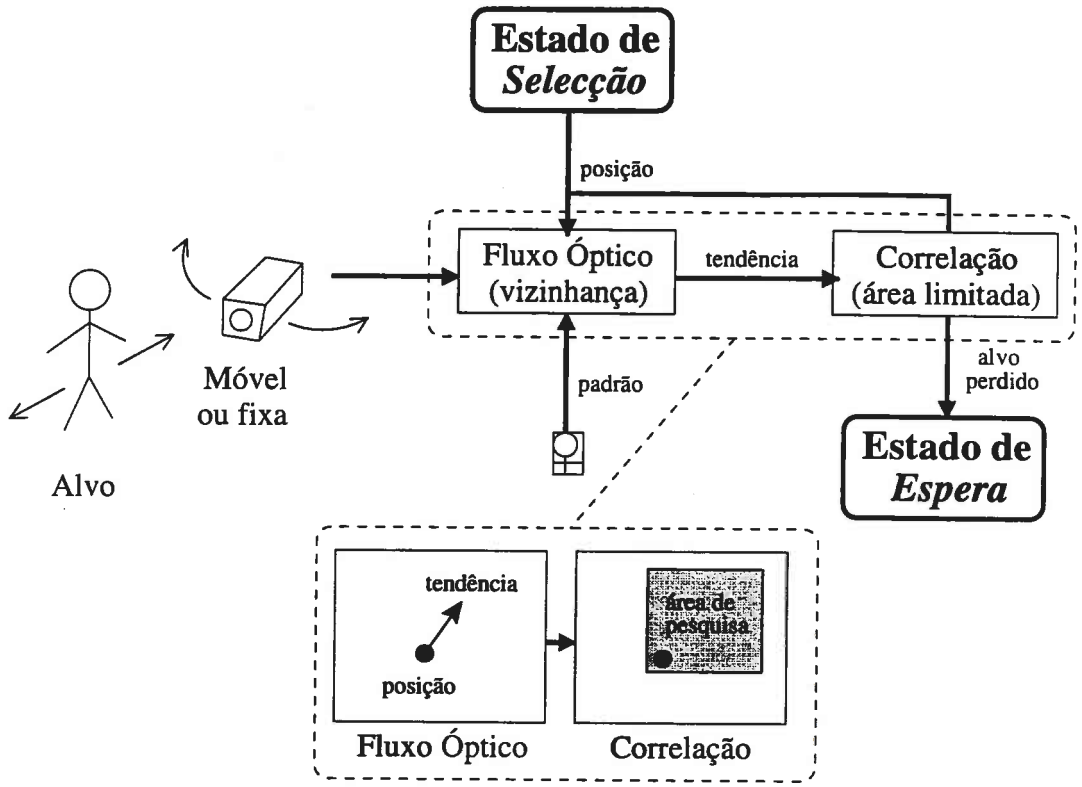



Figura 4.9: Diagrama de blocos para o estado de *Fixação*. Recorrendo inicialmente à posição detectada durante o estado de *Selecção*, calcula-se o fluxo óptico numa área vizinha do alvo de forma a detectar qual é a sua tendência de movimento. É assim possível limitar a área da nova imagem captada, na qual deverá ser efectuada a pesquisa do alvo. Desta pesquisa, efectuada com base na correlação, resultará a posição actual do alvo, que permitirá fechar o ciclo e manter a fixação.

alvo em todos os instantes futuros, cumprindo integralmente os objectivos definidos para a fixação visual. É claro que em função da aplicação em causa, levantam-se inúmeros problemas adicionais, como sejam o controlo do sistema de visão activa, a velocidade de reacção do sistema e a possibilidade do alvo deixar de ser identificável com os padrões extraídos. Nos capítulos seguintes veremos exemplos práticos da aplicação do processo de fixação visual, onde analisaremos estes e outros problemas. Por agora, vamos abordar uma forma de optimizar o processo de pesquisa dos padrões nas imagens, recorrendo ao fluxo óptico.

Como vimos, o fluxo óptico permite identificar, com alguma facilidade, movimentos que ocorram nas imagens, em particular nos casos em que há apenas um movimento. Recorrendo à sequência de imagens alinhadas e às posições onde foram localizados os padrões, é possível detectar a tendência de movimento do alvo, calculando o fluxo óptico numa vizinhança das posições encontradas. Note-se que a limitação do cálculo à área vizinha dos centros é fundamental, não só para reduzir o peso computacional do processo, mas também para assegurar que é calculado apenas o fluxo óptico gerado pelo alvo. Isto porque, por razões de velocidade e de robustez, vamos deixar de alinhar os fundos das imagens, trabalhando directamente sobre as imagens captadas. Calculada a tendência, podemos limitar a área das imagens com que iremos correlacionar os padrões, aumentando

assim a velocidade de execução. Na figura 4.9, está representado o diagrama de blocos que esquematiza o estado de *Fixação* do processo para uma das câmaras.

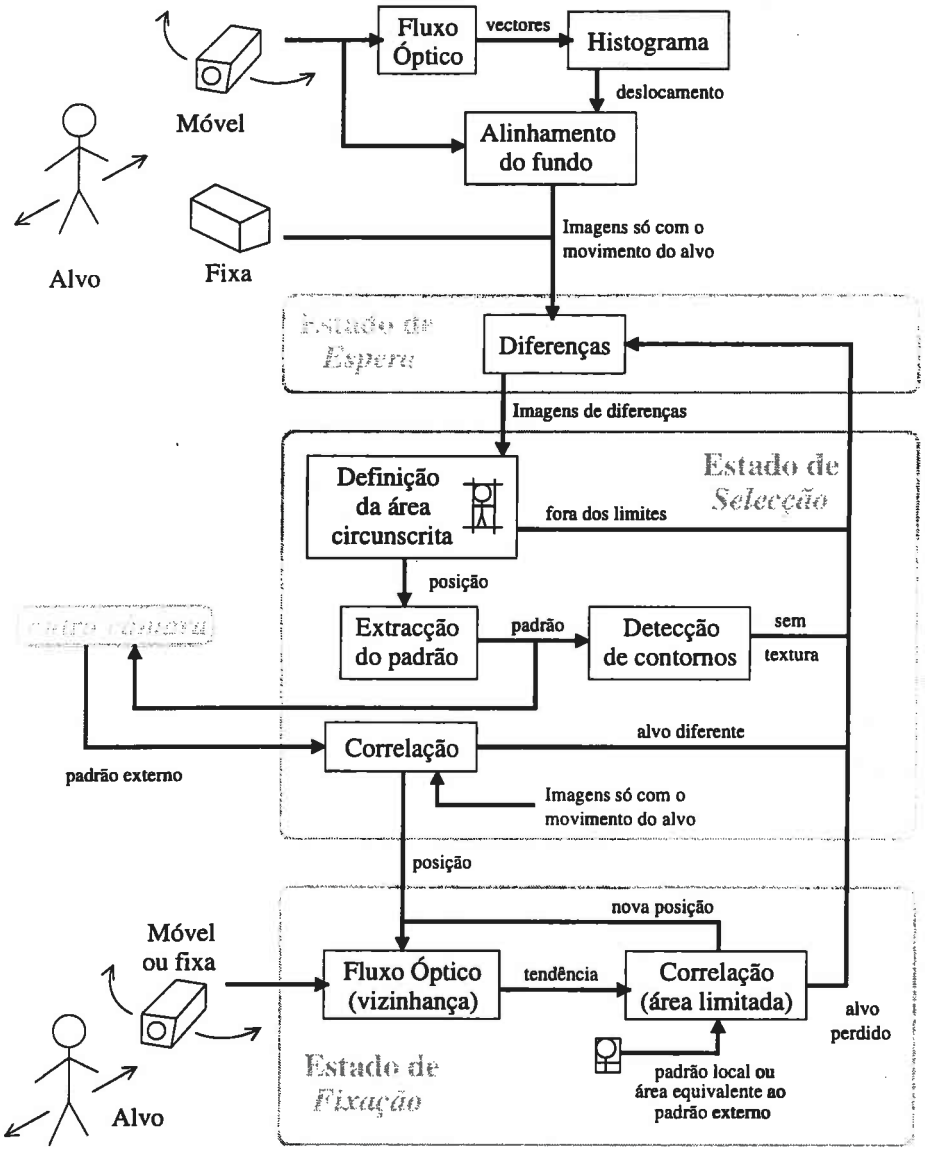



Figura 4.10: Diagrama de blocos global do processo de fixação.

## 4.5 Resumo

Na figura 4.10, podemos ver a esquematização global, por diagrama de blocos, do processo de fixação, tal como foi descrito ao longo deste capítulo. Como anteriormente, apenas se representa o bloco responsável pelo processamento das imagens captadas por uma das câmaras, já que o bloco associado à outra câmara é equivalente. Há apenas uma diferença, que consiste no padrão que é herdado pelo estado de *Fixação*. No caso da câmara dominante, o padrão corresponde à área extraída das imagens captadas pela própria câmara. No caso da outra câmara, o padrão corresponde à área das imagens onde foi encontrado o padrão dominante, durante o estado de *Selecção*.

Como vimos, no estado de *Espera*, o processo começa por alinhar as sequências de imagens captadas, de forma a que os movimentos dos alvos sejam detectáveis independentemente dos movimentos do fundo. As imagens resultantes são subtraídas, duas a duas, originando imagens de diferenças que, após serem filtradas, permitem identificar perfeitamente os alvos em movimento. Caso seja detectado um alvo em movimento, passa-se ao estado de *Selecção*, onde é calculada a área do alvo e a sua posição na imagem. Se a área estiver dentro dos limites pré-definidos, extrai-se das imagens originais um padrão representativo do alvo, obtido na posição em que este foi detectado. De seguida, utiliza-se a detecção de contornos para verificar se o padrão é texturado. Se assim for, troca-se o padrão extraído pelo padrão obtido com base nas imagens captadas pela outra câmara. O padrão da outra sequência é então pesquisado nas imagens originais, de forma a determinar se o alvo que gerou os estímulos em ambas as câmaras é o mesmo. Caso o padrão seja encontrado, chega-se ao estado de *Fixação*, escolhendo o padrão a utilizar com base na câmara em causa ser ou não a definida como dominante, tal como foi dito anteriormente. Neste estado, o primeiro passo consiste em detectar a tendência do movimento do alvo, calculando o fluxo óptico numa área em torno da última posição em que este foi encontrado. Com base nessa tendência, limita-se a área das imagens onde será pesquisado o padrão, acelerando o processo de determinar a nova posição do alvo. Se for possível determinar essa posição, volta-se ao primeiro passo, repetindo-se este ciclo até que o alvo deixe de poder ser encontrado com base no padrão. Quando o alvo não puder ser encontrado, volta-se ao estado de *Espera*, reiniciando-se todo o processo.

Está assim concluída a apresentação do processo de fixação visual. Nos próximos dois capítulos veremos aplicações concretas, incluindo a abordagem de alguns problemas de ordem prática e das adaptações necessárias em função dos objectivos pretendidos.

# Capítulo 5

# Perseguição por Fixação Visual

## 5.1 Introdução

Como poderemos constatar no final do próximo capítulo, as duas aplicações dadas como exemplo da utilização do processo de fixação visual são bastante distintas, tanto ao nível dos meios utilizados como dos fins almejados. O objectivo principal será obter a estrutura dos alvos, que se supõem estáticos. Com esse fim, iremos construir o modelo matemático do sistema de visão activa, e efectuar a calibração das câmaras. A velocidade dos algoritmos não é um objectivo prioritário. Em contrapartida, neste capítulo veremos uma aplicação com objectivos opostos, onde a resposta em tempo real é essencial, bem como a eficácia do processo, mesmo que à custa do rigor. A meta da aplicação resume-se a manter o sistema fixo num determinado alvo, independentemente dos movimentos que este possa executar. Por outras palavras, o sistema de visão activa deve movimentar-se de forma a que a projecção do alvo se faça no centro de ambas as câmaras. A isto chamamos perseguição. Uma solução possível deste processo pode ser realizada nos sistemas de visão activa associados a uma plataforma móvel. Nesses casos, a plataforma é controlada de forma a executar uma determinada trajectória relativamente ao alvo, como seja, por exemplo, o estabelecimento de uma distância constante entre ambos. Daí o nome perseguição, ou *pursuit* na versão anglo-saxónica.

A perseguição de alvos é uma das aplicações mais estudadas no campo da visão activa [BRA94] [COL94] [MUR95]. À primeira vista, poderá parecer que o algoritmo de fixação, descrito no capítulo 4, já desempenha esta tarefa, sobrando apenas o trabalho de controlo dos motores. De facto, se o alvo seguir o exemplo da Lua, mantendo-se a uma distância constante e mostrando sempre a mesma face, aquele algoritmo não necessita de qualquer alteração. Infelizmente, a generalidade dos alvos não segue esse padrão de acção. Adicionalmente, com vista à realização prática da aplicação, é preciso traduzir em segundos o peso computacional das várias técnicas propostas. Apesar das optimizações introduzidas na concepção dos métodos e na sua aplicação, existem limites físicos a respeitar. A próxima secção aborda todas estas questões.

## 5.2 Análise de Imagens

O primeiro desafio na realização das técnicas de análise de imagens é estabelecer um compromisso entre a sua celeridade e a precisão dos resultados obtidos. Uma vez que a velocidade de cada iteração do ciclo global de execução é crucial, os primeiros esforços têm

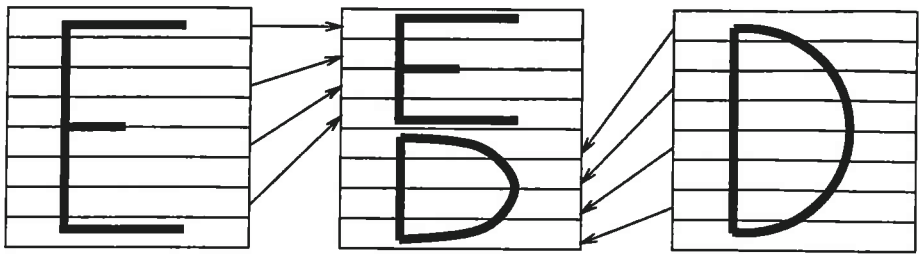


Figura 5.1: Aquando da captação, apenas é aproveitado um campo dos sinais PAL de cada uma das câmaras (**E**squerda e **D**ireita). Uma vez que as imagens são compostas por dois campos entrelaçados, o resultado são imagens com metade da resolução vertical original. Por outras palavras, há uma redução na resolução das imagens de 768 × 574 para 768 × 287 pontos. Adicionalmente, as imagens são captadas de forma a ficarem justapostas num dos *buffers* da placa de aquisição.

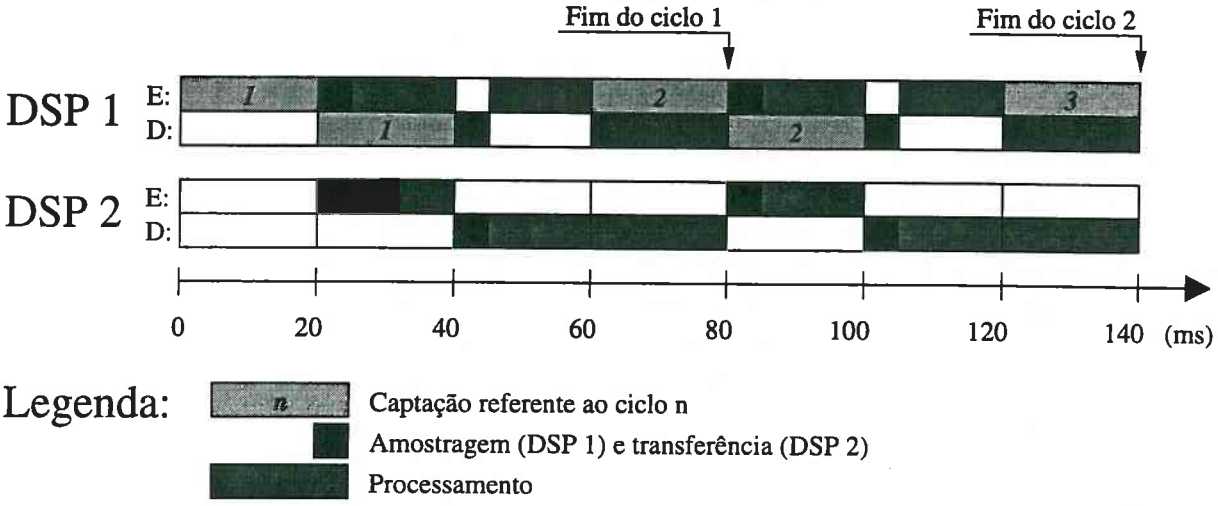



Figura 5.2: O diagrama temporal de processamento, individualizando os dois DSPs e o trabalho feito nas imagens (E)squerda e (D)ireita.

como objectivo a definição do número máximo de segundos gastos em cada período. O parâmetro principal no cálculo deste número é o tempo de captação de uma imagem. No nosso caso, esse tempo é de 20 milésimos de segundo, correspondente à duração de cada um dos dois campos que constituem um sinal PAL entrelaçado. A utilização de apenas um dos campos permite-nos, simultaneamente, obter uma maior rapidez de captação de imagens e reduzir a resolução das imagens resultantes. Adicionalmente, a redução da resolução possibilita a aquisição de ambas as imagens para o mesmo *buffer*, facilitando o seu tratamento (ver a figura 5.1).

Uma vez que é necessário captar duas imagens em cada ciclo, o mínimo tempo praticável é de 40 milésimos de segundo, tirando partindo do sincronismo dos sinais vídeo das câmaras. Depois de algumas experiências com os DSPs, optou-se pela definição de um período de 60 milésimos de segundo, o que equivale a cerca de 16,6 iterações por segundo, isto é, $16,6Hz$. Na figura 5.2, está representado o diagrama temporal de processamento dos DSPs em pormenor. Na realidade, como se pode ver, desde que se inicia a captação da primeira das duas imagens até que se obtêm os resultados, decorrem 80 milésimos de

segundo. É o paralelismo de processamento que permite a obtenção de um período 25% mais curto. O diagrama é relativo a uma situação de carga de processamento máxima, que ocorre quando se atinge o estado de *Fixação*. Durante os estados de *Espera* e *Selecção*, há uma menor carga computacional, não sendo crítico o desempenho dos algoritmos.

Definido o espaço temporal disponível, fizeram-se diversos testes com os vários algoritmos, tendo-se optado por executar o estado de *Fixação* recorrendo unicamente à correlação dos padrões com as imagens. Como se viu no capítulo anterior, a utilização do fluxo óptico permite reduzir a área de pesquisa, o que melhora o desempenho do bloco de correlação. No entanto, em face da falta de precisão e da baixa resolução dos motores de passo associados às câmaras, a delimitação da área de pesquisa, com base no cálculo do movimento do alvo, revela-se um processo ineficaz. Daí que se tenha optado por pesquisar o padrão na imagem em toda a sua área e a uma resolução inferior à das imagens captadas.

Nestas condições, o próximo passo é definir parâmetros como a resolução das imagens utilizadas, a dimensão dos padrões e o nível da pirâmide a que será feita a pesquisa. Relativamente às imagens, vamos subir dois níveis na pirâmide, a partir das imagens PAL originais ($768 \times 574$ pontos), resultando na utilização de imagens com $192 \times 143$ pontos. Em conformidade com o que se viu no capítulo 4, a área escolhida para os padrões é de $24 \times 16$ pontos, correspondendo aproximadamente a 1/64 das imagens. A pesquisa será feita três níveis acima na pirâmide, a uma resolução de $24 \times 17$ pontos, com a particularidade de não se efectuar a redução de resolução nem das imagens nem dos padrões. Isto porque, reduzindo os padrões em conformidade com o nível da pirâmide, ficaríamos com imagens de $3 \times 2$ pontos, manifestamente insuficientes. Por outras palavras, a pesquisa efectuada não é uma verdadeira pesquisa três níveis acima na pirâmide, mas sim uma pesquisa em que a correlação do padrão é efectuada sobre a imagem com intervalos de 8 pontos, tanto na vertical como na horizontal.

Para exemplificar o desempenho da aplicação, podemos ver na figura 5.3 uma sequência de imagens representando alguns décimos de segundo da perseguição de um pêndulo, a partir do instante em que o movimento foi detectado. O pêndulo encontra-se a cerca de um metro do sistema. Do lado esquerdo, temos a sequência correspondente à câmara esquerda e do lado direito a sequência correspondente à câmara direita. Como se pode ver, em ambas as sequências é seleccionada uma área bastante semelhante à inicial, correspondendo espacialmente à mesma zona do alvo. Este desempenho não se limita a alvos simétricos e contrastados. Na figura 5.4, podemos observar a mesma precisão quando o alvo perseguido é uma cara, igualmente afastada cerca de um metro do sistema. No entanto, ambos os alvos têm uma característica comum que justifica os bons resultados, que é o facto de manterem a mesma orientação relativamente às câmaras. Se, no caso do pêndulo, tal facto não é importante devido à sua simetria, o mesmo não acontece com a cara. De facto, se a cabeça rodar, o padrão deixa de ser visível, o que implica a impossibilidade de encontrar o alvo nos instantes futuros. O mesmo acontece se houver mudança de escala, isto é, uma aproximação ou afastamento do alvo (uma tentativa de resolução destes problemas pode ser encontrada em [SEE96]).

Quando um alvo deixa de ser identificável através dos padrões extraídos, poderemos optar por um de dois caminhos: voltamos ao estado de *Espera* (tal como foi apresentado no capítulo 4), ou continuamos indefinidamente à espera que o alvo retome a orientação original. Qualquer das soluções tem inconvenientes. A primeira implica um compasso de espera até que os novos padrões sejam adquiridos, já que todos os motores devem estar em repouso durante o estado de *Espera*. A segunda tem a desvantagem de implicar o blo-

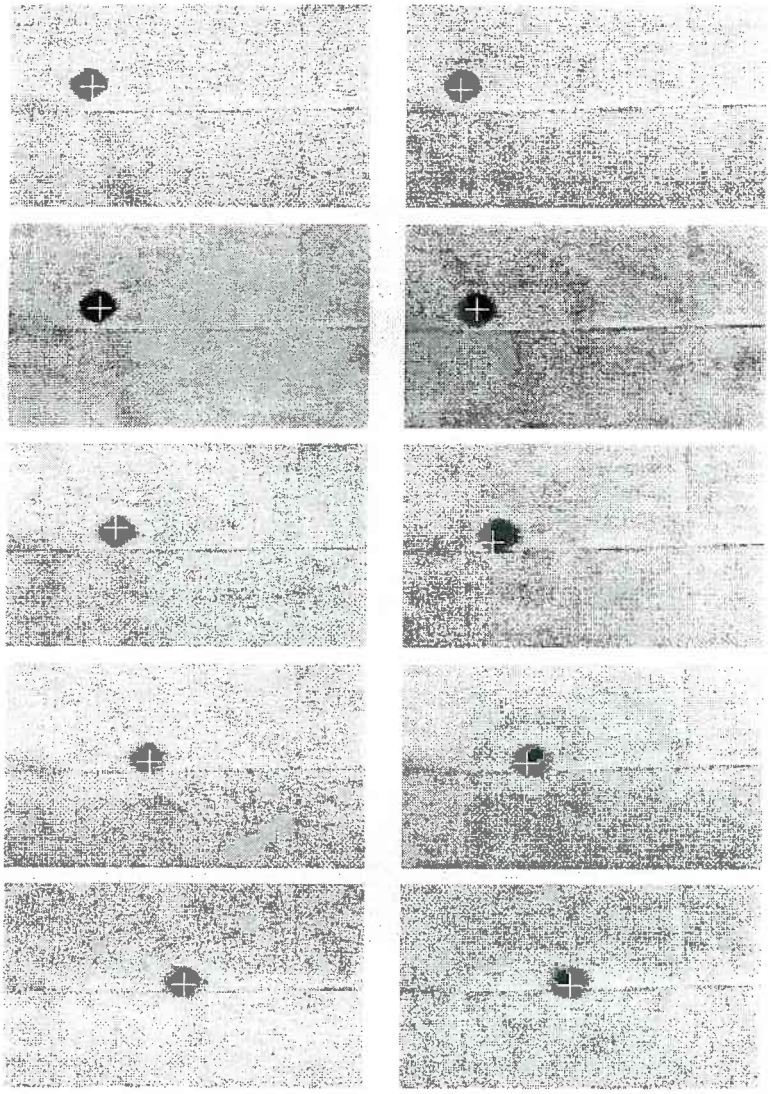


Figura 5.3: Sequência de imagens, obtidas com ambas as câmaras, durante a detecção e perseguição de um pêndulo.

queio do sistema durante um intervalo de tempo indeterminado, eventualmente infinito. Curiosamente, a solução ideal resulta da combinação das duas abordagens. Assim que os padrões deixam de ser detectáveis, os motores terão de parar, já que não há informação sobre a localização do alvo. Até estarem reunidas as condições para executar um ciclo do estado de *Espera*, o sistema deve continuar à procura dos padrões, retomando o funcionamento normal se tal acontecer. Caso contrário, a passagem ao estado de *Espera* é imediata. Adicionalmente, saliente-se que a transição para o estado de *Espera* implica a suspensão dos movimentos apenas por alguns décimos de segundo, aumentando a probabilidade de o alvo permanecer no campo de visão das câmaras. O diagrama da figura 5.5 ajuda a esclarecer todo o processo.

Uma forma de aumentar a robustez do algoritmo é tentar prevenir que os padrões deixem de ser válidos. Para o conseguirmos, vamos desenvolver um processo de actualização dos padrões, que deverá ocorrer sistematicamente durante o estado de *Fixação*. Na sua forma mais simples, o processo resume-se à substituição dos padrões, em cada ciclo, pela

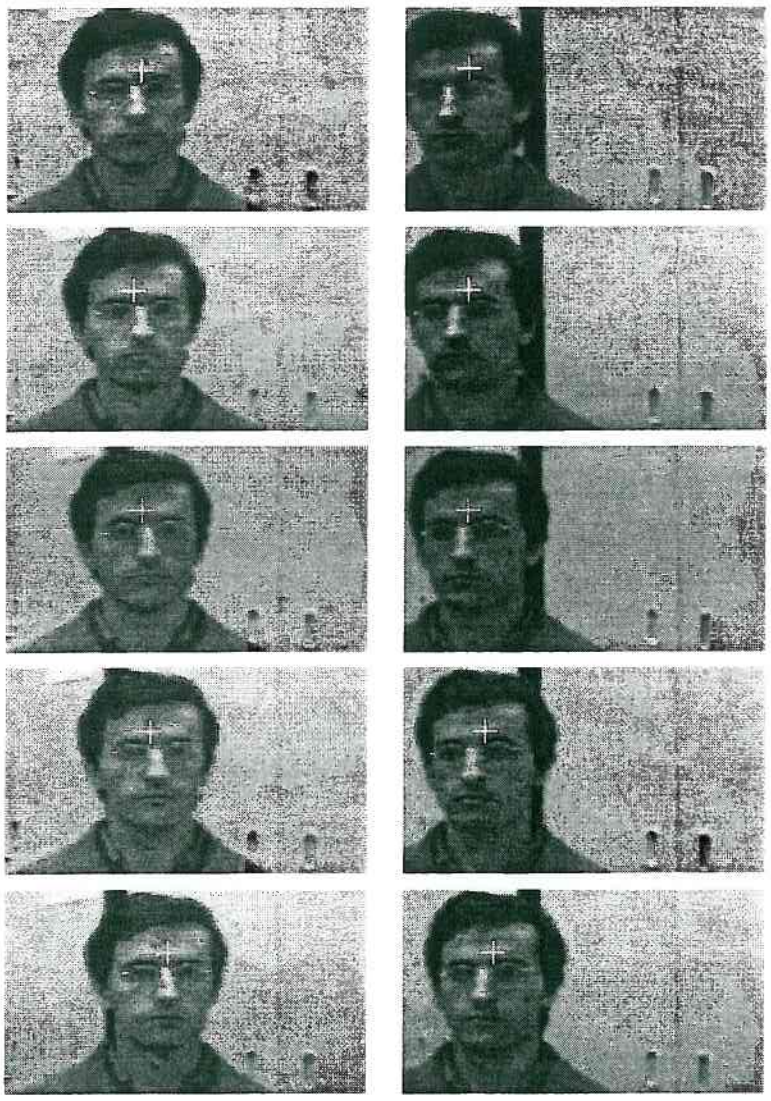


Figura 5.4: Sequência de imagens, obtidas com ambas as câmaras, durante a detecção e perseguição de uma cara.

área da imagem onde foram encontrados. O principal defeito desta solução resulta dos padrões poderem, se houver um erro de pesquisa, ser substituídos por zonas das imagens onde o alvo não se encontra. Convém, por isso, limitar a actualização dos padrões apenas aos casos em que a correlação revele a existência de uma grande semelhança. É claro que, se o alvo mudar de orientação de forma brusca, a actualização não se fará, sendo necessário voltar ao estado de *Espera*.

Em termos de resultados, a actualização dos padrões complementa o processo de retorno ao estado de *Espera* descrito anteriormente, aumentando a flexibilidade e a robustez do sistema. O desempenho global será o alvo da última secção deste capítulo, depois de abordarmos o problema do controlo dos motores.

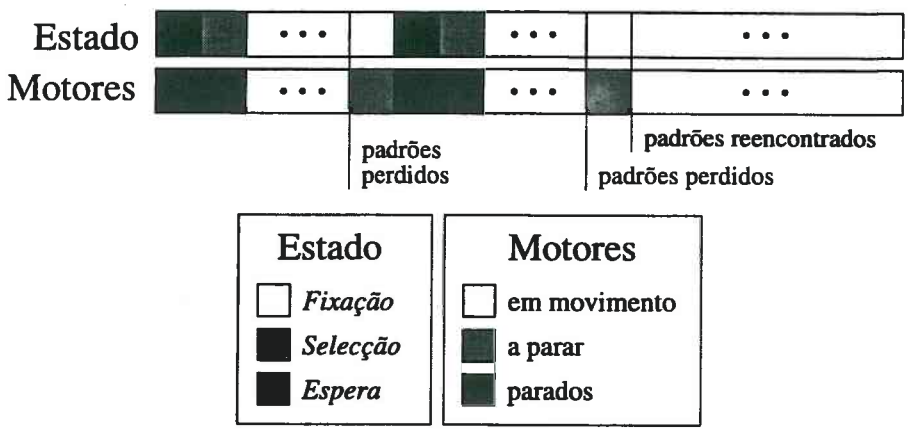



Figura 5.5: O diagrama mostra dois exemplos da reacção do sistema quando não é possível encontrar os padrões. No princípio, temos a evolução esperada do processo, do estado de *Espera* até ao estado de *Fixação*, iniciando-se a movimentação do sistema de visão activa apenas quando se atinge o estado de *Fixação*. Na primeira situação de perda de padrões, os motores param antes que, dentro do estado de *Fixação*, seja possível reencontrar os padrões. Daí resulta a passagem ao estado de *Espera* e consequente reinicialização do processo. Na segunda situação, os padrões são reencontrados antes dos motores pararem, permitindo a manutenção do processo no estado de *Fixação*.

## 5.3 Controlo

Como vimos, no processo de fixação os alvos são representados por padrões que, ao serem encontrados nas imagens, equivalem à identificação de numerosos pontos dos alvos. No entanto, para o controlo dos motores, apenas é necessário conhecer a localização de um único ponto. Isto porque, numa perspectiva óptima, o sistema de visão activa irá ser controlado de forma a que, no fim de cada ciclo de controlo, esse ponto se projecte no centro das imagens. Daí que, após cada ciclo do processo de pesquisa, apenas seja aproveitada a localização dos centros dos padrões encontrados. Para todos os efeitos, vamos considerar que esses centros correspondem à projecção do centro do alvo em cada uma das imagens.

No próximo capítulo, para tratarmos as localizações dos centros do alvo em cada câmara, iremos introduzir o modelo matemático do sistema de visão activa. Como veremos, ao aplicarmos as posições dos centros do alvo nas imagens ao modelo, conseguimos localizar tridimensionalmente a posição do centro do alvo. Neste caso, o nosso objectivo é controlar os motores em função das posições do alvo nas câmaras. Daí que, se recorrermos ao modelo matemático do sistema, teremos primeiro de calcular o ponto tridimensional correspondente ao centro do alvo, seguindo-se a sua propagação através de cada um dos referenciais do modelo (um por motor), calculando o movimento a executar pelo motor a que corresponde o referencial e propagando-o aos restantes de forma a que possa ser compensado. Esta abordagem tem duas desvantagens fundamentais, que são a complexidade de cálculos necessária e a interdependência que provoca nos diferentes referenciais ou, equivalentemente, nos vários motores. A primeira desvantagem afecta principalmente o desempenho do algoritmo, em face do peso computacional associado aos cálculos. A segunda desvantagem tende a acumular erros de cálculo resultantes das aproximações de valores e das incorrecções nas leituras dos descodificadores dos motores, propagando-os

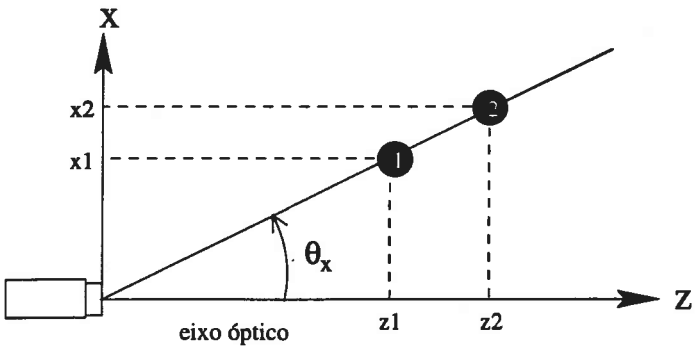



Figura 5.6: Dois alvos, a diferentes distâncias da câmara, mas fazendo o mesmo ângulo com o eixo óptico.

aos diferentes referenciais, o que provoca a introdução de erros derivados de um motor no cálculo dos parâmetros dos restantes.

A solução alternativa é controlar os motores individualmente, evitando sempre que possível a criação de dependências. Começando pelos motores das câmaras, vamos fazer o seu controlo com base nas projecções dos centros dos alvos. O movimento de cada motor dependerá unicamente da posição da projecção na imagem captada pela câmara respectiva. Tal é possível porque, como veremos, a relação entre a localização do alvo na imagem e o ângulo que faz com o eixo óptico da câmara é constante. Tomando como exemplo a figura 5.6, podemos ver dois alvos, a diferentes distâncias da câmara, mas fazendo o mesmo ângulo com o eixo óptico. Para calcular as suas projecções na imagem, vamos utilizar a expressão que as relaciona com a distância e o afastamento dos alvos relativamente à câmara:

$$x = (u - C_x)\frac{z}{K_x}$$

Esta expressão, retirada das relações 6.6, só será demonstrada no capítulo 6 quando abordarmos o problema da calibração das câmaras, na sequência da formulação do modelo matemático do sistema de visão activa. Sejam $z_1$ e $z_2$ as distâncias a que os alvos se encontram da câmara (medidas segundo o eixo óptico) e $x_1$ e $x_2$ os afastamentos perpendicularmente ao eixo óptico, podemos calcular a projecção horizontal dos alvos na imagem da seguinte forma:

$$u_1 = C_x + K_x\frac{x_1}{z_1}$$
$$u_2 = C_x + K_x\frac{x_2}{z_2}$$

Uma vez que $C_x$ e $K_x$ são constantes, respectivamente representando o centro e a relação *ponto/milímetro* da imagem (segundo a direcção horizontal), resulta de $\frac{x_1}{z_1} = \frac{x_2}{z_2}$ que $u_1 = u_2$. Por outras palavras, as projecções dos alvos são coincidentes. Manipulando as expressões e aplicando a função arco-tangente a ambos os membros, conclui-se que (ignorando os índices dos alvos):

$$\arctan\frac{u - C_x}{K_x} = \arctan\frac{x}{z} = \theta_x \tag{5.1}$$

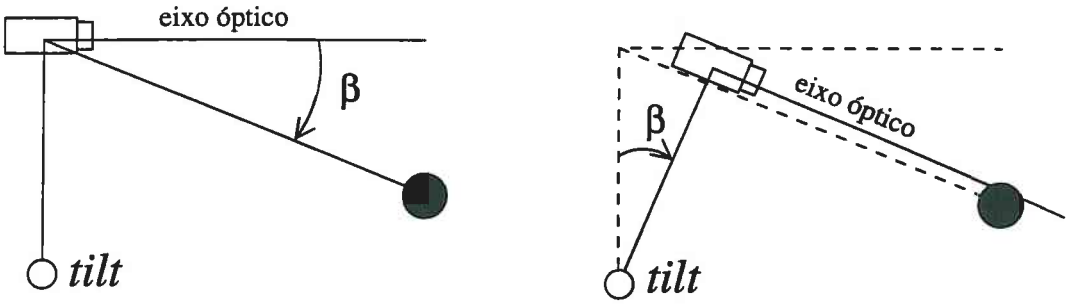



Figura 5.7: Incorrecção no cálculo do ângulo do *tilt*. As dimensões foram exageradas de forma a realçar a falta de coincidência dos eixos com os referenciais das câmaras.

Obtemos, assim, a relação que nos permite calcular o ângulo de rotação das câmaras, unicamente em função da localização do ponto nas imagens por elas captadas. O ângulo $\theta_x$ indica, no instante em causa, qual a amplitude da rotação a executar para que o alvo fique alinhado como o eixo óptico. No entanto, esta rotação pode não ser fisicamente realizável, quer por falta de velocidade dos motores, quer devido aos seus limites de excursão. Como veremos mais tarde, a rotação horizontal do sistema de visão activa, ou *pan*, será utilizada de forma a minorar estas limitações. Adicionalmente, através da relação 5.1 e da sua equivalente para os eixo vertical $y$, é possível estimar o campo de visão das câmaras. No capítulo 6 veremos que, quando a distância do alvo à câmara é superior a $1,5$ metros (ver a figura 6.6), $K_x$ e $K_y$ são aproximáveis a 2350 e 1750 respectivamente. Uma vez que as coordenadas dos pontos nas imagens podem variar entre $-383$ e 383 (na horizontal) e entre $-143$ e 143 (na vertical), obtemos os seguintes ângulos máximos (em valor absoluto):

$$\begin{aligned} \theta_x &= \arctan \frac{383}{2350} \approx 9,3^\circ \\ \theta_y &= \arctan \frac{143}{1750} \approx 4,7^\circ \end{aligned}$$

Por outras palavras, para que um alvo se projecte nas imagens, terá que fazer com os eixos ópticos das câmaras um ângulo horizontal inferior a $9,3^\circ$ e um ângulo vertical inferior a $4,7^\circ$.

Se considerarmos agora as projecções verticais dos pontos, chegamos, de forma análoga à anterior, à relação que nos permite fazer o controlo da inclinação frontal do sistema de visão activa, ou *tilt*. No entanto, surge-nos o problema do eixo de rotação do *tilt* não coincidir com a origem do referencial das câmaras. Daí que, se o motor executar o ângulo calculado, haverá um erro final que se fica a dever ao arco descrito pelas câmaras (ver a figura 5.7). Uma vez que o campo de visão das câmaras apenas permite a detecção de alvos até um ângulo vertical máximo de $4,7^\circ$, concluímos que a deslocação horizontal das câmaras nunca ultrapassa $1,7cm$, já que o braço que as separa do eixo de rotação do *tilt* tem um comprimento de $20,6cm$. Por outras palavras, o erro introduzido equivale à aproximação ou afastamento do alvo até um máximo de $1,7cm$, acrescido de um deslocamento vertical mínimo. Para além do erro ser insignificante, a sua eliminação implicaria o conhecimento da distância a que o alvo se encontra, criando exactamente o tipo de dependências que queremos evitar.

O controlo do motor que permite a rotação horizontal do sistema de visão activa, ou *pan*, é o único que depende das posições dos outros motores. Mais precisamente, depende das orientações dos eixos ópticos das câmaras. A ideia base no controlo do *pan* é manter o sistema de visão activa orientado na direcção do alvo, reduzindo a possibilidade de este sair do raio de acção dos motores das câmaras. Na figura 5.8 esquematiza-se este objectivo. O ângulo $\Delta_\alpha$ representa a rotação a efectuar, de forma a que o eixo do referencial *pan* intersecte o centro do alvo. Uma vez que o movimento do *pan* afecta as orientações das câmaras, estas terão de ser controladas de forma a compensar as rotações da estrutura. A exemplo do que fizemos para o *tilt*, vamos ignorar os efeitos resultantes dos arcos descritos pelas câmaras quando há rotações do *pan*. Podemos assim dizer que, se o *pan* roda um ângulo $\Delta_\alpha$, as câmaras deverão rodar o mesmo ângulo para manterem a sua orientação espacial. Por outro lado, para haver fixação quando o sistema de visão activa está orientado na direcção do alvo, isto é, quando $\alpha = \frac{\pi}{2}$, as câmaras deverão estar em posições simétricas relativamente ao eixo do referencial *pan*, tal como se indica na figura 5.8. Nestas condições, o cálculo do ângulo $\alpha$ é simplesmente:

$$\begin{aligned} & \tfrac{\pi}{2} - (\theta_E + \Delta_\alpha) && = (\theta_D + \Delta_\alpha) - \frac{\pi}{2} \\ <=> \quad & \tfrac{\pi}{2} - \theta_E - \tfrac{\pi}{2} + \alpha && = \theta_D + \frac{\pi}{2} - \alpha - \frac{\pi}{2} \\ <=> \quad & \alpha && = \frac{\theta_E + \theta_D}{2} \end{aligned} \tag{5.2}$$

Finalmente, importa ter em atenção que a velocidade de rotação do *pan* é baixa. Embora o ângulo $\alpha$ seja calculado como se mostra na equação 5.2, em cada ciclo de controlo apenas pode ser executada uma percentagem desse ângulo. Como veremos na secção seguinte, mesmo que a velocidade fosse suficiente, a estabilidade do sistema implicaria sempre uma resposta progressiva do *pan*. Convém reter que, independentemente da magnitude, as rotações do sistema de visão activa devem sempre ser compensadas pelos motores das câmaras.

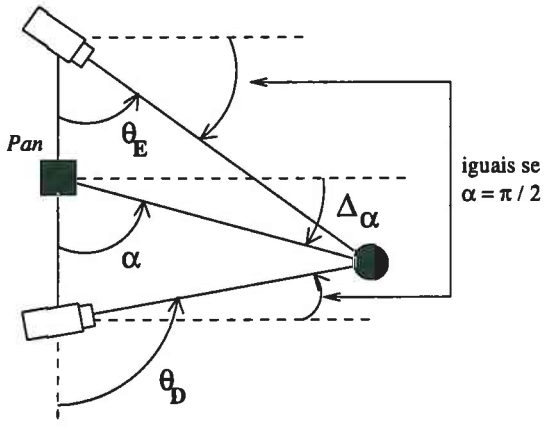



Figura 5.8: Cálculo do ângulo de *pan* (os ângulos são positivos nos sentidos indicados).

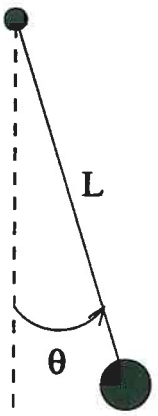



Figura 5.9: Pêndulo simples. O comprimento da corda é dado por $L$ e o ângulo com a vertical por $\theta$.

## 5.4 Desempenho

Independentemente de eventuais progressos conseguidos ao nível do processamento de imagens, a falta de velocidade, resolução, flexibilidade e fiabilidade dos motores que compõem o sistema de visão activa são, à partida, factores intransponíveis, determinantes no estabelecimento de limites para os objectivos da aplicação. Para se conseguir alguma rapidez, definiram-se parâmetros de controlo que permitem a execução de movimentos com a duração de cerca de $0,1$ segundos. Durante esse intervalo de tempo, os motores só conseguem percorrer um ângulo muito reduzido, particularmente crítico no caso das câmaras. Nestas condições, resta-nos explorar ao máximo os $16,6$ ciclos por segundo do módulo de análise de imagens, para optimizar o desempenho do sistema.

Começando pelos motores que controlam o *pan* e o *tilt*, a sua função resume-se ao apoio ao trabalho dos motores das câmaras, orientando a estrutura de forma a optimizar o campo de acção. Pretende-se, portanto, uma resposta estável. Para conseguir movimentos suaves, utilizamos um controlador equivalente a um PI (Proporcional Integral). A parte proporcional está assegurada, à partida, pelo ciclo de $0,1$ segundos, que limita as rotações de ambos os motores a um ângulo de $0,5°$. A parte integral é conseguida através de um processo de acumulação de amostras, no qual se faz a média das últimas $N$ posições obtidas, onde o $N$ deverá ser afinado em função das situações. No nosso caso, são incluídas no cálculo da média as últimas 5 posições obtidas, o que permite a suavização sem penalizar demasiadamente a velocidade de resposta.

No caso dos motores que controlam as câmaras, o ciclo de $0,1$ segundos limita a rotação dos motores a um ângulo de $1,8°$. Uma vez que se pretende uma resposta rápida das câmaras, as posições obtidas são enviadas aos motores sem nenhum outro tipo de processamento.

Como a análise das imagens é efectuada a uma velocidade superior à que se atinge no controlo dos motores, é possível tirar partido do excesso de informação para melhorar o desempenho. Em abono da verdade, na generalidade dos casos não há um excesso de dados, mas sim uma compensação de insuficiências, pois as iterações do ciclo de análise nem sempre resultam na detecção do alvo nas imagens. Surge assim uma incógnita sobre a sua posição naquele instante, impedindo que se prossiga a movimentação dos motores. Como o ciclo de controlo dos motores é mais demorado, o resultado global traduz-se por

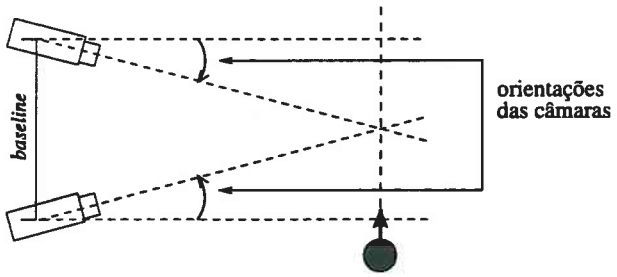



Figura 5.10: Esquematização do ambiente no qual foram recolhidos os valores experimentais. O pêndulo encontra-se a um metro das câmaras, descrevendo uma trajectória cujos extremos estão separados por cerca de 40 centímetros, que são percorridos em aproximadamente um segundo, isto é, com um período de dois segundos. Inicialmente, o pêndulo é largado do lado direito das câmaras, estando estas alinhadas com o centro da sua trajectória.

uma redução da probabilidade do próximo ciclo surgir sem que seja conhecida uma posição recente do alvo.

Para termos uma ideia mais exacta sobre o desempenho do sistema em condições reais, vamos analisar alguns exemplos em que o alvo a perseguir é um pêndulo. Recordando a equação de um pêndulo simples, temos que (ver a figura 5.9):

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\sin(\theta) \tag{5.3}$$

Na equação, $L$ é o comprimento da corda, $\theta$ indica o ângulo com a vertical e $g$ representa a constante de aceleração gravitacional. Quando a amplitude das oscilações é pequena, temos que $\sin(\theta) \approx \theta$, o que nos permite simplificar a equação 5.3 para:

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\theta \tag{5.4}$$

As soluções desta equação diferencial são do tipo:

$$\theta = A\sin(\sqrt{\frac{g}{L}}t + C) \tag{5.5}$$

onde $A$ e $C$ são constantes. Devido à aproximação $\sin(\theta) \approx \theta$, podemos dizer que 5.5 descreve a projecção horizontal do movimento do pêndulo. Numa situação ideal, seria essa a trajectória descrita pelos motores do sistema de visão activa ao perseguir o pêndulo, e será com base nela que iremos tirar conclusões sobre o seu desempenho.

Na figura 5.10 esquematiza-se o ambiente utilizado para as experiências. Em todos os casos, analisaremos os resultados recolhidos durante 10 segundos do movimento, a que correspondem sempre 100 amostras, considerando-se desprezável o efeito do amortecimento. Adicionalmente, vamos dividir as experiências em três partes: resposta a uma frequência constante, resposta a um degrau e resposta a um impulso. Em cada uma das partes, analisaremos em primeiro lugar o desempenho do sistema quando apenas são controladas as câmaras, comparando-o depois com os resultados obtidos se o motor associado ao *pan* for igualmente utilizado.

### 5.4.1 Ambiente Experimental

Antes de começarmos, importa salvaguardar o facto de que as condições em que as experiências foram realizadas levaram o sistema muito perto do seu limite de resposta, o que será particularmente visível nos gráficos das projecções do alvo nas imagens. Em termos gerais, o pêndulo revelou-se a melhor opção de entre os recursos disponíveis para efectuar os testes. No entanto, em face da dependência entre o comprimento da corda e a velocidade de oscilação do pêndulo, não foi possível efectuar experiências a frequências inferiores. Assim, esta configuração foi o melhor compromisso possível entre a estabilidade, a amplitude, a previsibilidade e a velocidade do sinal de entrada que se poderia utilizar nos testes.

### 5.4.2 Frequência Constante

Para obter uma frequência de oscilação constante, largámos o pêndulo do extremo direito da sua trajectória, tal como se descreve na figura 5.10. Numa primeira fase, o motor associado ao *pan* não é controlado, registando-se apenas as posições do pêndulo nas imagens e as posições dos motores das câmaras. Os resultados podem ser observados na figura 5.11. Como se pode ver pelos valores iniciais, o pêndulo parte do extremo direito dos campos de visão das câmaras que, recorde-se, estão alinhadas com o centro da trajectória do pêndulo. Ao longo dos 10 segundos amostrados, as posições equivalentes aos extremos da trajectória projectam-se em pontos mais próximos do centro das imagens, embora de forma irregular. Isto permite-nos concluir que o sistema está, de facto, a tentar manter o alvo centrado nas imagens, mas encontrando algumas dificuldades na execução dessa tarefa. A razão dessas dificuldades é visível nos gráficos das posições dos motores. Os gráficos representam praticamente uma onda triangular periódica, o que é indicativo de que os motores estão a funcionar perto do seu limite de velocidade. Adicionalmente, é de salientar a irregularidade da movimentação dos motores e o desfasamento existente entre as variações da posição do pêndulo e as variações das posições dos motores (note-se que as vertentes positivas dos motores estão relacionadas com as vertentes negativas das posições na imagem e vice-versa).

Vamos agora introduzir o controlo do motor associado ao *pan*. Na figura 5.12 podemos ver os valores obtidos neste caso. Para começar, verifica-se que as posições do pêndulo em ambas as imagens é aproximadamente igual. Além disso, as posições na imagem relacionadas com os extremos da trajectória estão mais próximas do centro e são mais regulares do que no exemplo anterior. Passando à análise das posições dos motores, não há grandes alterações nas posições das câmaras. A relativa melhoria nos resultados terá, portanto, de ser justificada pela inclusão do controlo do *pan*. Como podemos ver, o *pan* acompanha o movimento das câmaras, com um desfasamento que ultrapassa por vezes um quarto do período. Em princípio, este desfasamento deveria ser nocivo, já que contraria em certa medida o movimento das câmaras. No entanto, dadas as características do movimento do alvo, a movimentação do *pan* antecipa a inversão do movimento do pêndulo, resultando na melhoria global. É claro que idêntico resultado poderia ser obtido se se reduzisse a velocidade de resposta das câmaras. Ainda relativamente à movimentação do *pan*, importa recordar que a sua rotação é calculada com base na média dos ângulos das câmaras. Isto implica que o ângulo que será compensado em cada uma das câmaras devido à rotação do *pan* é simétrico. Daí que, uma vez que as câmaras rodam geralmente no mesmo sentido, a rotação do *pan* dificulta a tarefa de uma das câmaras, mas facilita

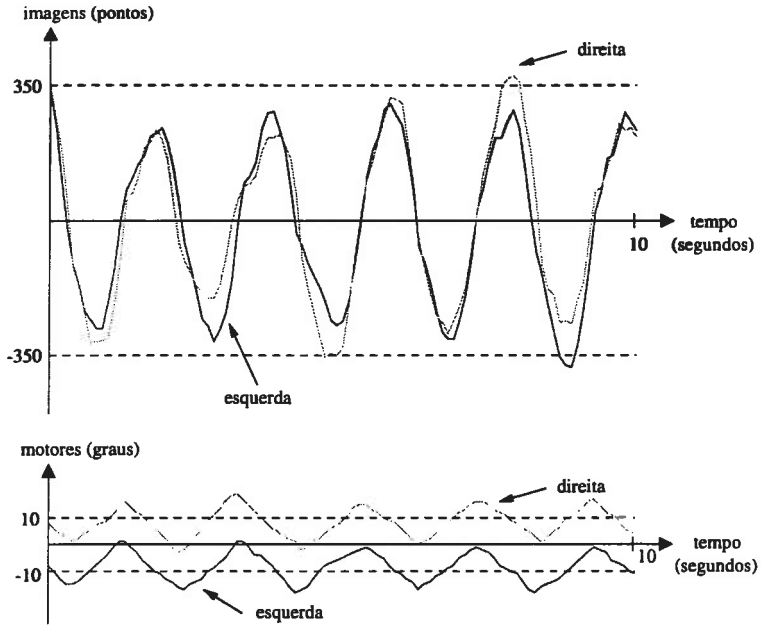



Figura 5.11: Resultados obtidos quando apenas são controlados os motores das câmaras e a entrada é um sinal oscilatório. Em cima, podemos ver a localização do pêndulo nas imagens. Em baixo, representam-se as orientações das câmaras, positivas no sentido anti-horário, relativamente a uma perpendicular à *baseline*, tal como se esquematizou na figura 5.10.

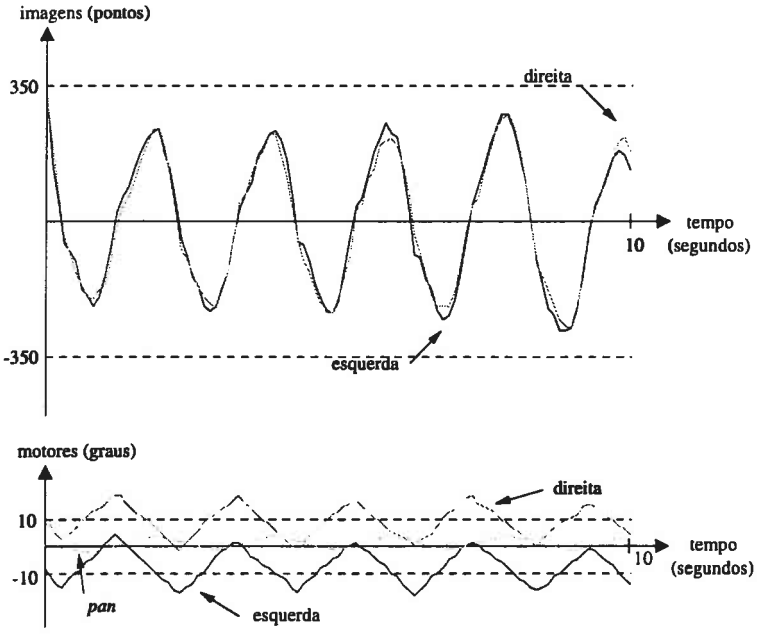



Figura 5.12: Resultados obtidos quando são controlados os motores das câmaras e do *pan* e a entrada é um sinal oscilatório. Em cima, podemos ver a localização do pêndulo nas imagens. Em baixo, representam-se as orientações das câmaras e do *pan*, positivas no sentido anti-horário, relativamente a uma perpendicular à *baseline*.

a tarefa da outra. É esta a justificação para a aproximação dos valores dos gráficos que representam a posição do pêndulo nas imagens. Finalmente, nos casos mais gerais, a rotação do *pan* tem a vantagem adicional de possibilitar ao sistema um maior campo de acção.

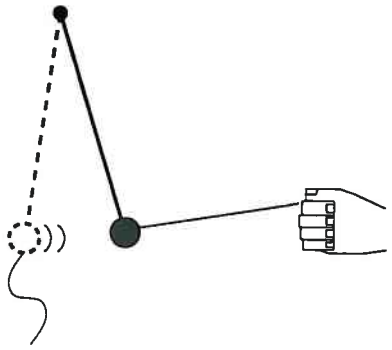


Figura 5.13: Recorrendo a uma linha fina e leve, suspendemos o movimento oscilatório do pêndulo, simulando sinais de entrada com as características de um degrau e de um pulso.

### 5.4.3 Resposta a um Degrau

Para analisar a resposta do sistema a um sinal de entrada em degrau, recorremos a um pequeno estratagema: utilizando uma linha fina e leve, alterámos o movimento natural do pêndulo, segurando-o no extremo direito da sua trajectória, numa altura em que era perseguido pelo sistema em condições análogas às descritas na secção anterior (ver a figura 5.13). A nossa análise inicia-se pouco depois da suspensão do movimento do pêndulo, assim que o sistema estabiliza. Na figura 5.14 podemos ver cerca de 3 segundos durante os quais o sistema está estabilizado, fixando o extremo direito da trajectória do pêndulo. De seguida, o pêndulo foi libertado, iniciando o seu movimento oscilatório normal. Como seria de esperar, o sistema reagiu movimentando-se de forma a continuar a fixar o pêndulo. Os últimos 7 segundos dos gráficos da figura 5.14 são relativos a essa fase.

Embora o sinal gerado desta forma não corresponda exactamente a uma entrada em degrau convencional, permite-nos mesmo assim tirar algumas conclusões sobre o desempenho do sistema, já que são contemplados instantes em que o pêndulo não se move, seguidos de um movimento que se assemelha à experiência analisada na secção anterior. De facto, são visíveis nos gráficos, durante os primeiros 3 segundos, as oscilações que ocorrem nas câmaras e nas imagens numa situação em que o sistema deveria estar em repouso. Por um lado, como a informação recolhida nas imagens não é filtrada, o sistema está constantemente a receber ordens de movimentação, justificando essas oscilações. No entanto, a principal razão prende-se com a imprecisão de posicionamento e de descodificação da localização dos motores das câmaras, para além da falta de velocidade e resolução.

Na figura 5.15 estão representados os gráficos relativos à inclusão do motor associado ao *pan*, numa situação em que o movimento do pêndulo foi suspenso e reiniciado em condições análogas às descritas atrás. Em termos comparativos, as conclusões são as mesmas a que já chegámos anteriormente, verificando-se alguma melhoria no que respeita à localização do alvo nas imagens. Relativamente à fase em que o sistema está estabilizado, continua a ser perceptível a falta de precisão do sistema, com constantes variações nas posições das câmaras e nas projecções do alvo.

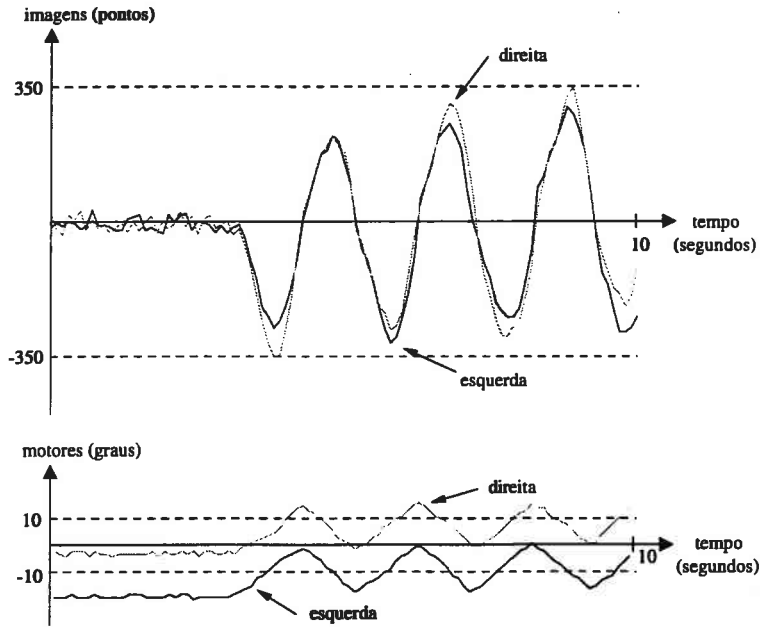



Figura 5.14: Resultados obtidos quando apenas são controlados os motores das câmaras e a entrada é um degrau. Em cima, podemos ver a localização do pêndulo nas imagens. Em baixo, representam-se as orientações das câmaras, positivas no sentido anti-horário, relativamente a uma perpendicular à *baseline*, tal como se esquematizou na figura 5.10.

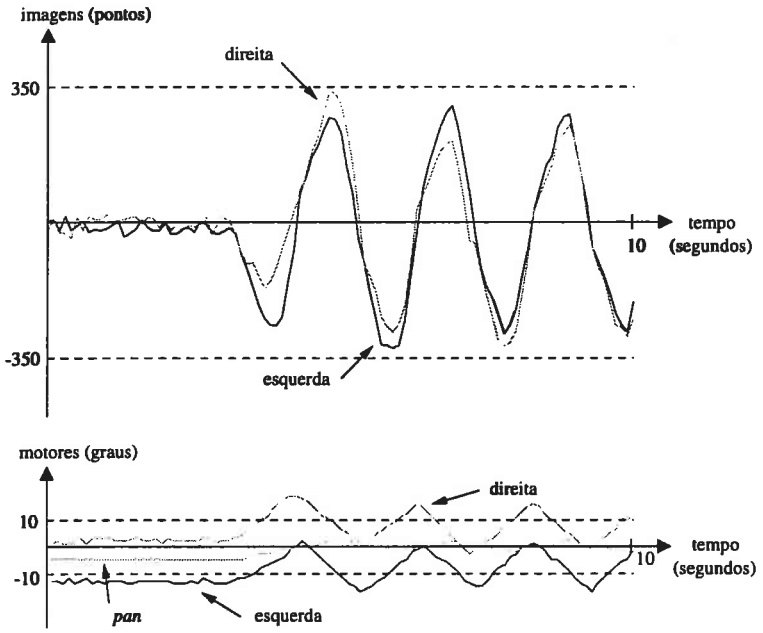



Figura 5.15: Resultados obtidos quando são controlados os motores das câmaras e do *pan* e a entrada é um degrau. Em cima, podemos ver a localização do pêndulo nas imagens. Em baixo, representam-se as orientações das câmaras e do *pan*, positivas no sentido anti-horário, relativamente a uma perpendicular à *baseline*.

### 5.4.4 Resposta ao Impulso

Para a análise da resposta do sistema a um impulso, vamos recorrer ao mesmo estratagema utilizado atrás. Tal como anteriormente, o movimento do pêndulo foi suspenso no extremo direito da sua trajectória, numa altura em que o sistema fixava o pêndulo. Assim que o sistema estabilizou, esperámos 5 segundos, ao fim dos quais deixámos oscilar o pêndulo até este ter atingido a posição mais baixa (i.e. a posição central) da sua trajectória. Nessa altura, bloqueámos novamente o seu movimento, resultando assim um sinal de entrada equivalente a um pulso com uma largura de cerca de $0,5$ segundos. Na figura 5.16 estão representados os resultados obtidos ao longo dos 10 segundos analisados. Observando os gráficos, podemos ver que o sistema reage de forma bastante rápida ao pulso, mantendo-se com a estabilidade possível durante o resto do tempo. Como seria de esperar, as posições das câmaras antes e depois do pulso não são as mesmas, já que a localização do pêndulo se alterou.

Novidades surgem apenas com a comparação destes resultados com os obtidos ao repetir a experiência, após a inclusão do motor associado ao *pan*. Na figura 5.17 estão representados esses valores. Como podemos ver, quando o pulso termina, há um claro *overshoot* do sistema, particularmente notório no gráfico de posições das câmaras. Tal fica a dever-se à falta de rapidez de reacção do *pan*, motivada não só pela sua baixa velocidade, mas também por se estar a utilizar as últimas 5 amostras para efectuar a parte integral do controlo.

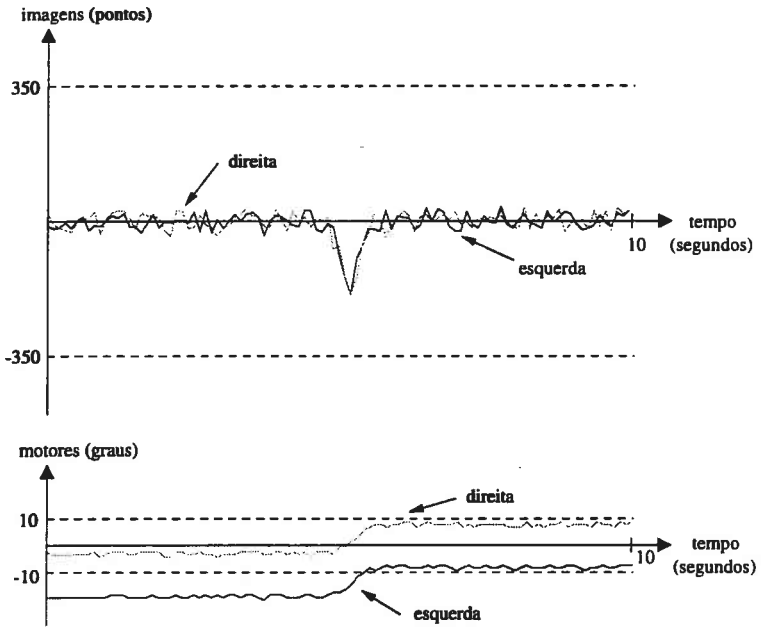



Figura 5.16: Resultados obtidos quando apenas são controlados os motores das câmaras e a entrada é um pulso. Em cima, podemos ver a localização do pêndulo nas imagens. Em baixo, representam-se as orientações das câmaras, positivas no sentido anti-horário, relativamente a uma perpendicular à *baseline*, tal como se esquematizou na figura 5.10.

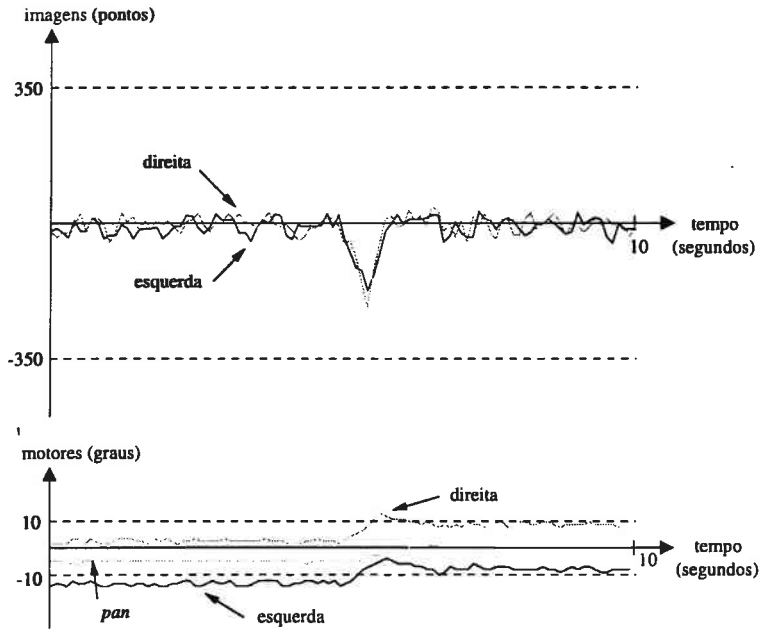



Figura 5.17: Resultados obtidos quando são controlados os motores das câmaras e do *pan* e a entrada é um pulso. Em cima, podemos ver a localização do pêndulo nas imagens. Em baixo, representam-se as orientações das câmaras e do *pan*, positivas no sentido anti-horário, relativamente a uma perpendicular à *baseline*.

# Capítulo 6

# Recuperação de Estrutura por Fixação Visual

## 6.1 Introdução

O segundo exemplo do processo de fixação que iremos analisar consiste na recuperação da informação tridimensional dos alvos. Algumas das técnicas mais comuns abordam essa tarefa como um problema de associação sistemática de arestas ou pontos característicos, o que é geralmente muito demorado, mesmo quando se tem grande capacidade de processamento. Neste capítulo, vamos ver um método dinâmico de resolver o problema, capaz de obter resultados bastante bons, com a desvantagem de ser semi-automático. Ao contrário da aplicação descrita no capítulo anterior, pretendemos recuperar a estrutura dos alvos com a maior precisão possível, pelo que veremos alguns exemplos da exactidão conseguida. Será ainda proposta uma linha de orientação para automatizar o processo, que não foi seguida no decurso do trabalho por se afastar demasiadamente dos objectivos iniciais, mas que se inclui aqui como sugestão para um trabalho futuro.

## 6.2 Descrição da Aplicação

Na base da aplicação de recuperação da estrutura tridimensional dos alvos, temos o princípio da estereovisão. Se ambas as câmaras estiverem fixas num mesmo ponto, é possível calcular a sua localização no espaço, relativamente ao sistema de visão activa. Em termos genéricos, a estrutura de um alvo não é mais do que um aglomerado de pontos tridimensionais, agrupados para formar arestas e planos. Utilizando o sistema de visão activa e o processo de fixação, pretendemos construir uma aplicação que permita a recolha de vários pontos de um alvo, que poderão depois ser utilizados para construir o seu modelo matemático. Refira-se que a estereovisão não se esgota com a reconstrução tridimensional, possibilitando outras abordagens com resultados não menos interessantes [GRI93].

Na figura 6.1 esquematiza-se o modo de funcionamento da aplicação. Para recolher cada um dos pontos, o utilizador humano controla o sistema de visão activa utilizando a *trackball*, posicionando as câmaras de forma a que o ponto que escolheu se projecte no centro das imagens. Atingir este objectivo com precisão em ambas as câmaras é praticamente impossível. Para melhorar a precisão dos resultados, vamos recorrer ao processo de fixação visual. Começamos por definir uma das câmaras como dominante. Isto significa

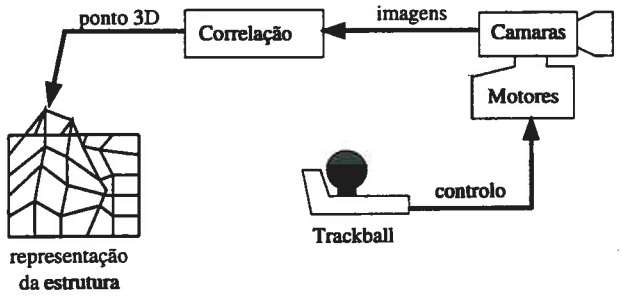



Figura 6.1: Diagrama esquemático do processo de recuperação de estrutura.

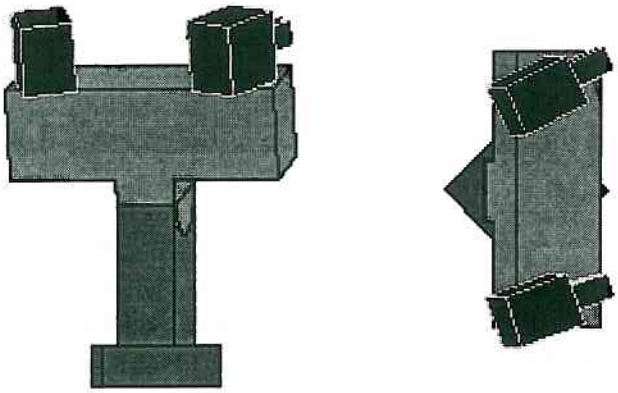


Figura 6.2: Duas perspectivas do modelo gráfico do sistema de visão activa.

que o utilizador humano deverá concentrar os seus esforços em posicionar correctamente o ponto escolhido apenas no centro da imagem correspondente à câmara dominante. Quanto à outra câmara, é suficiente que o ponto em causa se encontre dentro de uma área em torno do centro da imagem, porque, depois de definida uma das câmaras como dominante, vamos extrair um padrão do centro da imagem por ela captada. Utilizando o processo de fixação visual, procuramos na outra imagem esse padrão, limitando a área de pesquisa a uma vizinhança do centro. No final, temos a posição exacta do ponto nas duas imagens, o que nos permite calcular a sua localização espacial e acrescentá-la ao modelo do alvo. Note-se que a localização do ponto na imagem dominante não é necessariamente no seu centro. Isto porque, em função da fase de análise da textura dos padrões, o algoritmo poderá optar por extrair o padrão de uma área próxima do centro.

## 6.3 Modelo do Sistema

A obtenção do modelo matemático do sistema é uma tarefa fundamental para esta aplicação. O desempenho dos processos e a exactidão dos resultados está dependente de uma formulação correcta das várias características físicas do sistema, como sejam as dimensões, os graus de liberdade e a distância focal das câmaras, entre outras. Na figura 6.2, podemos ver uma representação gráfica do sistema de visão activa, obtida com base no modelo matemático que iremos descrever nesta secção.

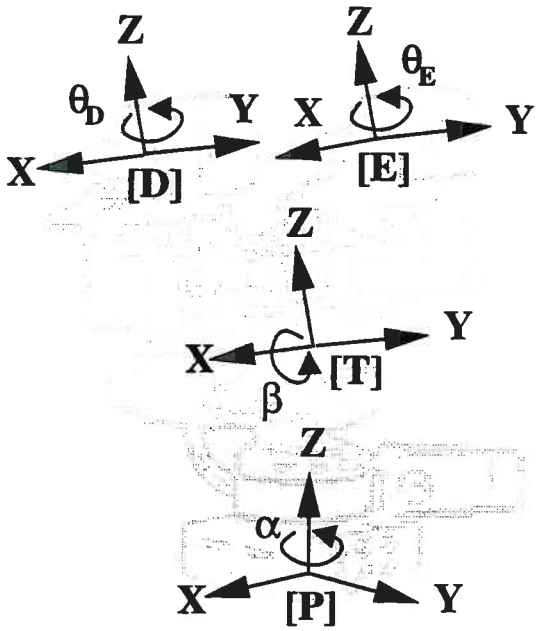



Figura 6.3: Os quatro referenciais do sistema de visão activa: $[P]an$, $[T]ilt$, câmara [D]ireita e câmara [E]squerda.

### 6.3.1 Referenciais

O primeiro passo para a descrição matemática do sistema é a definição da localização e orientação dos vários referenciais. Os referenciais estão directamente relacionados com os graus de liberdade do sistema. No nosso caso, tiramos partido de quatro dos cinco graus de liberdade existentes, já que a distância entre as câmaras, ou *baseline*, é mantida sempre no seu valor máximo. Vamos então definir referenciais para os restantes quatro graus de liberdade do sistema de visão activa (ver a figura 6.3), que são:

- *Pan* - Designação anglo-saxónica para o movimento de rotação sobre um plano horizontal. O seu ponto de aplicação é o centro da base do sistema. O eixo $Z$, em torno do qual está rodado de um ângulo $\alpha$ (relativamente ao mundo), contém o ponto de aplicação do referencial *tilt*.

- *Tilt* - Designação anglo-saxónica para o movimento de rotação sobre um plano vertical. Está rodado de um ângulo $\beta$ em torno do eixo $X$ e a sua origem pertence ao eixo $Z$ do referencial *pan*. O plano $XZ$ inclui os pontos de aplicação dos referenciais das câmaras.

- Câmaras Direita e Esquerda - Encontram-se rodadas de um ângulo $\theta_D$ e $\theta_E$ respectivamente, em torno do eixo $Z$. As suas origens pertencem ao plano $XZ$ do referencial *tilt*, com abcissas iguais segundo o eixo $Z$ e simétricas segundo o eixo $X$. Os centros dos referenciais coincidem, teoricamente, com o centro das matrizes de sensores ópticos das câmaras, designadas vulgarmente por imagens.

Por último, define-se o referencial do mundo, $[M]$, coincidente com o referencial *pan* quando o ângulo de rotação $\alpha$ é nulo.

### 6.3.2 Transformações

Definidos os referenciais, interessa obter as matrizes de rotação e de translação que os relacionam. Uma vez que cada referencial tem apenas um eixo de rotação, a obtenção das matrizes de rotação não apresenta qualquer dificuldade. O mesmo se passa com as matrizes de translação, dada a localização favorável dos referenciais. Sendo assim, as matrizes que nos permitem fazer a transformação de pontos $[X\ Y\ Z]^T$ de uns referenciais para os outros são as seguintes (as distâncias apresentadas, correspondentes às medidas do sistema, estão em centímetros):

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{[T]} = \begin{bmatrix} \cos(\theta_D) & -\sin(\theta_D) & 0 \\ \sin(\theta_D) & \cos(\theta_D) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{[D]} + \begin{bmatrix} 14,5 \\ 0 \\ 20,6 \end{bmatrix} \tag{6.1}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{[T]} = \begin{bmatrix} \cos(\theta_E) & -\sin(\theta_E) & 0 \\ \sin(\theta_E) & \cos(\theta_E) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{[E]} + \begin{bmatrix} -14,5 \\ 0 \\ 20,6 \end{bmatrix} \tag{6.2}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{[P]} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{[T]} + \begin{bmatrix} 0 \\ 0 \\ 25,7 \end{bmatrix} \tag{6.3}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{[M]} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{[P]} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.4}$$

Os valores apresentados para as abcissas $X$ das câmaras correspondem à situação em que elas se encontram o mais afastadas possível, isto é, quando a *baseline* tem um comprimento de 29,1 centímetros. As várias distâncias medidas são apresentadas com algumas reservas quanto à precisão, já que se torna difícil encontrar pontos de referência para definir, por exemplo, os centros das matrizes de sensores ópticos das câmaras, ou a altura do eixo de rotação do *tilt*.

### 6.3.3 Câmaras

Como foi já referido, as origens dos referenciais $[D]$ e $[E]$ coincidem com os centros das imagens das câmaras respectivas. Isto implica que os planos imagem, definidos pelos eixos $X$ e $Z$ dos referenciais, sejam igualmente coincidentes com as imagens (desprezam-se os defeitos de construção). A figura 6.4 esquematiza essa relação. Note-se que os referenciais das imagens estão rodados relativamente ao planos imagem, devido ao método de indexação utilizado computacionalmente (da direita para a esquerda e de cima para baixo). A rotação introduzida não é mais do que uma troca de nomenclaturas e sentidos de orientação dos eixos dos referenciais, tornando trivial a obtenção da matriz de transformação que os relaciona.

Infelizmente, o mesmo não se pode afirmar relativamente à obtenção das coordenadas de um ponto no referencial imagem, dada a sua localização em pontos. De facto, essa transformação implica o conhecimento de dois parâmetros das câmaras de difícil obtenção: a sua distância focal (relativa à lente utilizada) e a relação entre pontos e centímetros (associada às características físicas da matriz de sensores ópticos). Para os calcular, vamos começar por formular matematicamente o problema, concentrando-nos apenas numa das

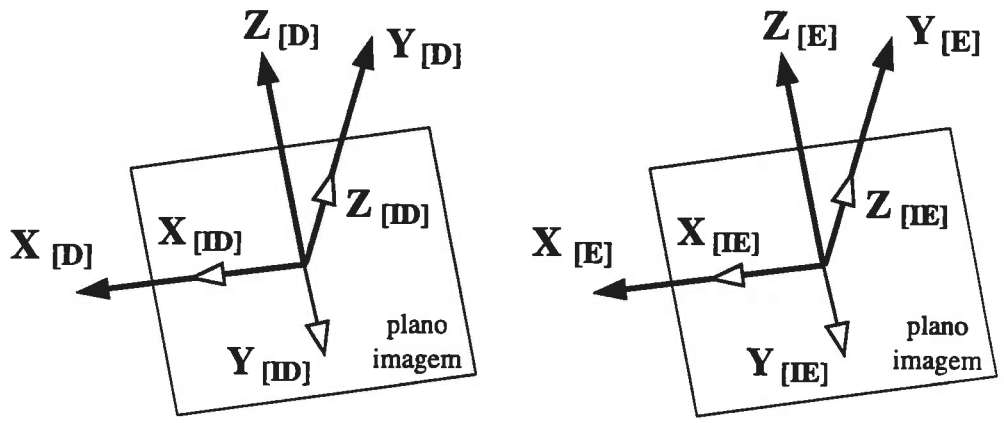



Figura 6.4: Referenciais da câmaras, planos imagem e referenciais das imagens.

câmaras (a dedução é análoga para a outra). Seja $(x, y, z)$ um ponto do referencial imagem, $(u, v)$ as suas coordenadas em pontos e $(C_x, C_y)$ o centro das imagens, podemos relacionar os vários parâmetros através das seguintes equações (utilizando o modelo de projecção em perspectiva):

$$\begin{aligned} x &= (u - C_x)\frac{z + F}{S_x F} \\ y &= (v - C_y)\frac{z + F}{S_y F} \end{aligned} \tag{6.5}$$

Nas equações anteriores, $F$ representa a distância focal da câmara e $S_x$ e $S_y$ representam a razão entre pontos e centímetros, respectivamente nas direcções horizontal e vertical da imagem. Embora as câmaras estejam equipadas com lentes motorizadas, o nível de ampliação das lentes nunca sofre alterações, pelo que a distância focal é constante. Nestas condições, o problema resume-se ao cálculo de três constantes: $F$, $S_x$ e $S_y$, efectuado na secção seguinte.

### 6.3.4 Calibração do Mínimo Número de Parâmetros

A obtenção de estimativas para os parâmetros $F$, $S_x$ e $S_y$ (entre outros) é conhecida por calibração. Existem diversos métodos de calibração de câmaras, com os quais é possível obter resultados com maior ou menor precisão, em função das situações em causa [BAT93] [MCL95]. Em face das características eminentemente práticas do projecto, prescindiu-se da utilização desses métodos em favor da estimação mais grosseira mas eficiente dos parâmetros experimentais. Uma vez que o nosso objectivo final não é a obtenção de estimativas para os parâmetros $F$, $S_x$ e $S_y$, mas sim a definição de um modelo que transforme pontos em centímetros, podemos fazer uma simplificação nas equações 6.5 que reduzirá para dois o número de parâmetros a calcular. Para tal, basta verificar que os valores possíveis para a distância focal $F$ nunca ultrapassarão algumas dezenas de milímetros, enquanto que qualquer alvo que se pretenda analisar se encontrará sempre a várias dezenas de centímetros. Podemos, portanto, desprezar $F$ no numerador das equações e fazer $K_x = S_x F$ e $K_y = S_y F$ no denominador, resultando as expressões 6.6 com apenas duas variáveis: $K_x$ e $K_y$.

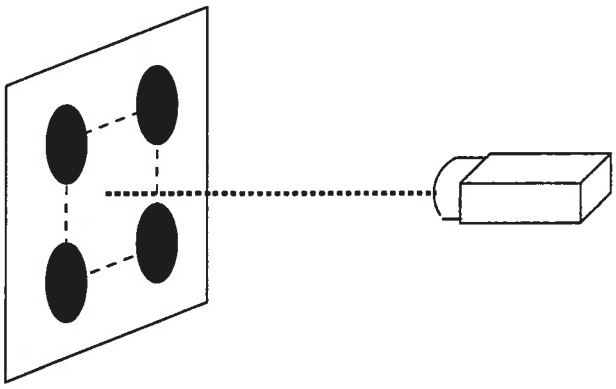


Figura 6.5: Ambiente experimental para calibração das câmaras.

$$\begin{aligned} x &= (u - C_x)\frac{z}{K_x} \\ y &= (v - C_y)\frac{z}{K_y} \end{aligned} \tag{6.6}$$

Para efectuar as medidas experimentais, usou-se um alvo contendo 4 círculos centrados nos vértices de um quadrado imaginário de dimensões conhecidas. De seguida, alinhou-se a câmara de forma a que o seu eixo óptico ficasse perpendicular ao alvo (ver a figura 6.5). Finalmente, calcularam-se os centróides dos 4 círculos e utilizaram-se esses valores na equação 6.6. Este processo foi depois repetido para várias distâncias da câmara ao alvo. Antes de abordarmos os resultados obtidos, importa ter em atenção os seguintes pontos:

- O cálculo dos centróides dos círculos revela-se um método bastante eficaz de determinar os vértices do quadrado, já que o ruído e as distorções fazem-se sentir de igual forma ao longo de toda a fronteira dos círculos, o que reduz em grande parte a sua influência no resultado final.

- Supondo uma orientação correcta do alvo, é possível compensar eventuais torções da câmara em torno do eixo óptico analisando as coordenadas dos centróides (os círculos localizados à esquerda devem ter a mesma abcissa, os localizados no topo devem ter a mesma ordenada, etc.).

- Como não se garante um alinhamento do eixo óptico da câmara com o centro do quadrado, apenas se utilizam no cálculo dos parâmetros as estimações, em pontos, das dimensões do quadrado. Se for respeitado o ponto anterior, o cálculo resume-se ao aproveitamento, por exemplo, da diferença de ordenadas dos círculos localizados à esquerda e da diferença de abcissas dos círculos localizados no topo.

A precisão de parâmetros como a distância da câmara ao alvo ou o centro das imagens não é crucial para o processo, já que a resolução das imagens e a precisão do sistema não exigem maiores cuidados, nem permitiriam tirar partido de medidas mais rigorosas. Nas condições descritas, obtiveram-se para $K_x$ e $K_y$ os valores representados na figura 6.6 (em função da distância $z$).

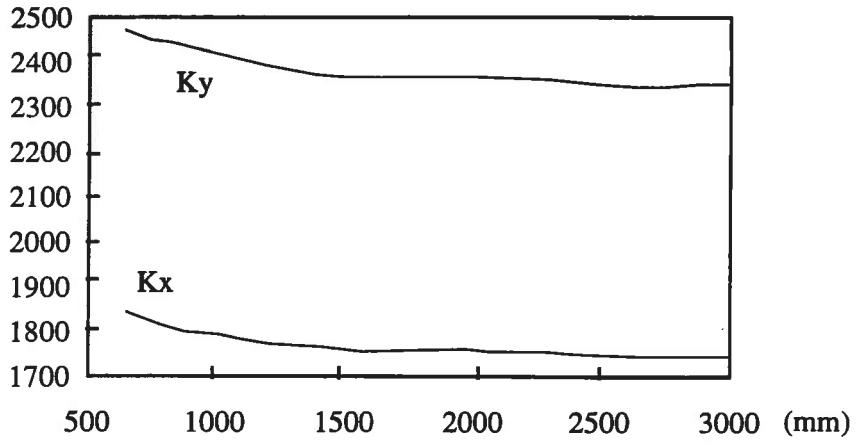



Figura 6.6: Valores de $K_x$ e $K_y$ obtidos em função da distância $z$.

Conclui-se que os valores são praticamente constantes, com uma ligeira elevação para distâncias mais próximas, que se ficam a dever às distorções causadas pela ausência de focagem (as lentes são mantidas focadas para o infinito). Note-se que $K_y$ apresenta valores superiores a $K_x$, devendo-se esperar imagens ligeiramente encolhidas segundo o eixo horizontal.

Com a estimação de $K_x$ e $K_y$ fica concluída a definição do modelo matemático global do sistema.

### 6.3.5 Reconstrução Tridimensional

A obtenção dos pontos tridimensionais a partir do modelo matemático descrito anteriormente reveste-se de algumas particularidades, que abordaremos agora. Para começar, vamos supor que foi detectado um alvo em cada uma das imagens, cujas coordenadas são $(u_E, v_E)$ e $(u_D, v_D)$, respectivamente na imagem esquerda e na imagem direita. Utilizando as equações 6.6, podemos relacionar estas coordenadas com os pontos dos planos imagem correspondentes. Adicionalmente, como se vê na figura 6.4, é trivial transformar os pontos dos planos imagem para os seus equivalentes $(x_E, y_E, z_E)$ e $(x_D, y_D, z_D)$ nos referenciais esquerdo e direito das câmaras, respectivamente. Resultam assim as relações:

$$\begin{aligned} x_D &= (u_D - C_x)\frac{y_D}{K_x} \\ z_D &= (C_y - v_D)\frac{y_D}{K_y} \\ x_E &= (u_E - C_x)\frac{y_E}{K_x} \\ z_E &= (C_y - v_E)\frac{y_E}{K_y} \end{aligned} \tag{6.7}$$

Mesmo considerando $K_x$ e $K_y$ como constantes, as quatro equações anteriores contêm seis variáveis, o que nos impede de resolver o sistema. Necessitamos, portanto, de mais duas equações. Para tal, vamos recorrer às transformações 6.1 e 6.2, que relacionam pontos dos referenciais das câmaras com pontos do referencial *tilt*. Uma vez que o ponto

em causa é o mesmo ponto de um determinado alvo, as suas coordenadas no referencial *tilt* terão de ser iguais. Podemos assim igualar as relações 6.1 e 6.2, obtendo:

$$\begin{aligned} x_D\cos(\theta_D) - y_D\sin(\theta_D) + 14,5 &= x_E\cos(\theta_E) - y_E\sin(\theta_E) - 14,5 \\ x_D\sin(\theta_D) + y_D\cos(\theta_D) &= x_E\sin(\theta_E) + y_E\cos(\theta_E) \\ z_D + 20,6 &= z_E + 20,6 \end{aligned} \tag{6.8}$$

Note-se que as componentes dos vectores são agora representadas por letras minúsculas. Uma vez que os ângulos $\theta_D$ e $\theta_E$ de que as câmaras estão rodadas são conhecidos, obtemos mais três equações sem aumentar o número de incógnitas. No entanto, as duas últimas equações são linearmente dependentes, como facilmente se conclui substituindo as relações 6.7 nessas equações:

$$\begin{aligned} (u_D - C_x)\frac{y_D}{K_x}\sin(\theta_D) + y_D\cos(\theta_D) &= (u_E - C_x)\frac{y_E}{K_x}\sin(\theta_E) + y_E\cos(\theta_E) \\ (C_y - v_D)\frac{y_D}{K_y} + 20,6 &= (C_y - v_E)\frac{y_E}{K_y} + 20,6 \end{aligned}$$

Ambas as equações estabelecem uma constante de proporcionalidade entre os valores de $y_D$ e $y_E$. Infelizmente, devido aos erros cometidos na obtenção prática dos diversos parâmetros, as constantes nunca são iguais, o que torna o sistema impossível. Adicionalmente, põe-se um problema que analisaremos com base na segunda equação. Simplificando-a obtemos:

$$\frac{y_D}{y_E} = \frac{C_y - v_E}{C_y - v_D}$$

Como se pode ver, a equação relaciona a distância do alvo a cada uma das câmaras com a projecção vertical nas imagens. Este facto é problemático, visto que as projecções verticais são valores inteiros e próximos de zero (recorde-se que o centro da imagem tem coordenadas nulas). Para constatarmos o problema, vamos imaginar que o alvo foi detectado, relativamente ao eixo horizontal, um ponto acima na imagem esquerda e um ponto abaixo na imagem direita. Pela equação anterior, deveria concluir-se que a razão entre $y_D$ e $y_E$ é igual a $-1$. Este exemplo permite-nos verificar quão críticos são os valores das projecções verticais do alvo. Vamos solucionar o problema começando por recordar que as projecções são obtidas com os alvos praticamente centrados nas imagens. Uma vez que as distorções provocadas pelas câmaras e pelas lentes são mínimas no centro, podemos considerar que as projecções verticais do alvo nas imagens são iguais. Na prática, tomamos o seu valor médio, isto é, $v = \frac{v_E + v_D}{2}$. Desta aproximação resulta de imediato que a distância do alvo a cada uma das câmaras é igual, o que só é exacto numa situação de fixação com as câmaras posicionadas de forma a formarem ângulos iguais com a *baseline*. Por outras palavras, ao recolher os pontos, quanto mais próximo estiver o sistema de visão activa desta configuração, maior é a precisão dos resultados.

Depois da análise anterior, o problema resume-se à resolução do seguinte sistema de equações, com $z = z_D = z_E$ e $y = y_D = y_E$:

$$x_D = (u_D - C_x)\frac{y}{K_x}$$

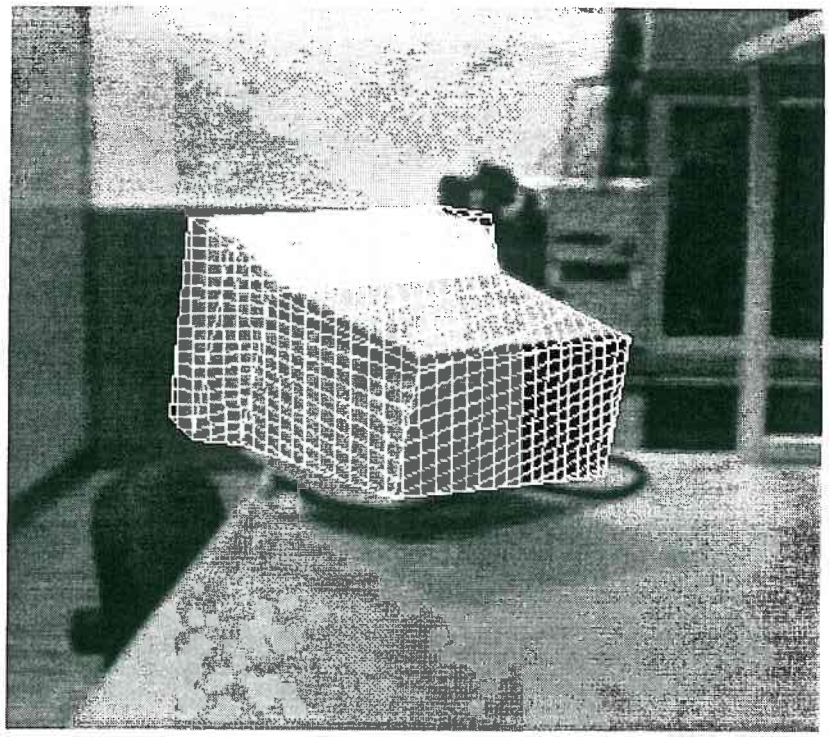


Figura 6.7: Resultados obtidos após a reconstrução 3D de um monitor.

$$
\begin{aligned}
z &= (C_y - v)\frac{y}{K_y} \\
x_E &= (u_E - C_x)\frac{y}{K_x} \\
x_D \cos(\theta_D) - y \sin(\theta_D) + 14,5 &= x_E \cos(\theta_E) - y \sin(\theta_E) - 14,5
\end{aligned} \tag{6.9}
$$

Obtemos assim as coordenadas tridimensionais dos pontos nos referenciais das câmaras. Qualquer um deles pode agora ser utilizado, conjuntamente com as transformações apresentadas na secção 6.3.2, para obter as coordenadas do alvo no referencial do mundo.

## 6.4 Resultados

Nas secções anteriores, descrevemos o processo mecânico através do qual um utilizador humano pode seleccionar um ponto de uma cena por ele escolhido. Vimos ainda a forma como podemos identificar com precisão esse ponto, recorrendo ao processo de fixação visual. Finalmente, apresentámos as equações matemáticas que nos permitem calcular a localização tridimensional do ponto relativamente ao sistema de visão activa. Nesta secção veremos os passos que restam para obter representações de estruturas como a que se mostra na figura 6.7.

Para obter os resultados da figura 6.7, foi necessário recolher, um a um, os 13 vértices visíveis do monitor. Na figura 6.8 podemos ver as coordenadas tridimensionais de todos esses pontos. É claro, para cada novo ponto recolhido, o sistema de visão activa teve de ser movimentado, de onde resultou a alteração da perspectiva que as câmaras têm do alvo. Para representar todos os pontos sobre uma mesma imagem, utilizou-se o seguinte método:

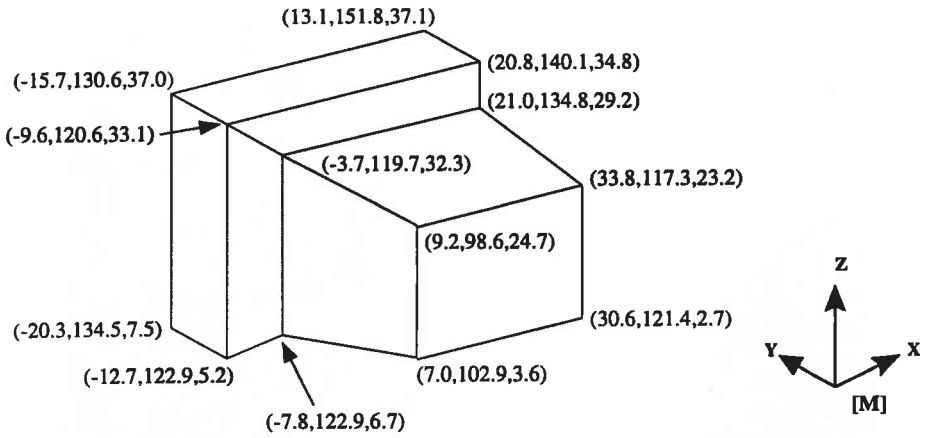



Figura 6.8: Coordenadas tridimensionais dos vértices do monitor representado na figura 6.7, relativamente ao referencial do mundo. O centro do monitor encontra-se aproximadamente a 1 metro do referencial do mundo e a cerca de 20 centímetros para a direita.

1. Ao ser seleccionado o primeiro ponto, é guardada a imagem visualizada pela câmara dominante, bem como a orientação do sistema de visão activa (designados de seguida como imagem base e orientação inicial, respectivamente).

2. Depois de obtidas as coordenadas tridimensionais do ponto, calcula-se a sua projecção na câmara dominante com base na orientação inicial. Utilizando os valores calculados, desenha-se uma cruz na imagem base.

3. Se houver mais pontos, selecciona-se um e volta-se ao ponto 2.

Depois de recolhidos os pontos e calculadas as suas coordenadas tridimensionais, o utilizador terá de indicar quais os pontos que formam arestas e quais as arestas que formam planos. Este processo é suportado por um interface gráfico, que facilita de forma significativa aquela tarefa. Uma vez definidos os planos, é calculada a sua intersecção com um conjunto de planos horizontais e verticais, paralelos e uniformemente espaçados, tal como se mostra na figura 6.9. A intersecção destes conjuntos de planos com os planos que definem o alvo originam rectas que, quando representadas na imagem base, formam uma malha que é responsável pela aparência tridimensional visível na figura 6.7 (recorde-se que a cena está representada do ponto de vista da câmara dominante, justificando a inclinação das rectas que resultam da intersecção dos vários planos).

Relativamente à precisão dos resultados obtidos, podemos considerá-la como bastante boa, embora tendo em consideração que o alvo escolhido não tem uma estrutura complexa. Para chegarmos a esta conclusão, calcularam-se as distâncias euclidianas entre os vários pontos recolhidos, que foram depois comparadas com as dimensões reais do monitor utilizado como alvo. O resultado dessa comparação pode ser analisado na figura 6.10. Como se pode ver, as diferenças são bastante reduzidas.

Nesta secção verificámos que, ao ser escolhido um monitor como alvo, apenas foi necessário localizar alguns pontos manualmente, para que a sua estrutura fosse completamente recuperada. No entanto, dependendo da complexidade do alvo, o número de pontos a localizar poderá aumentar demasiadamente, tornado impraticável a sua obtenção manual. Na secção seguinte aborda-se o problema da automatização do processo.

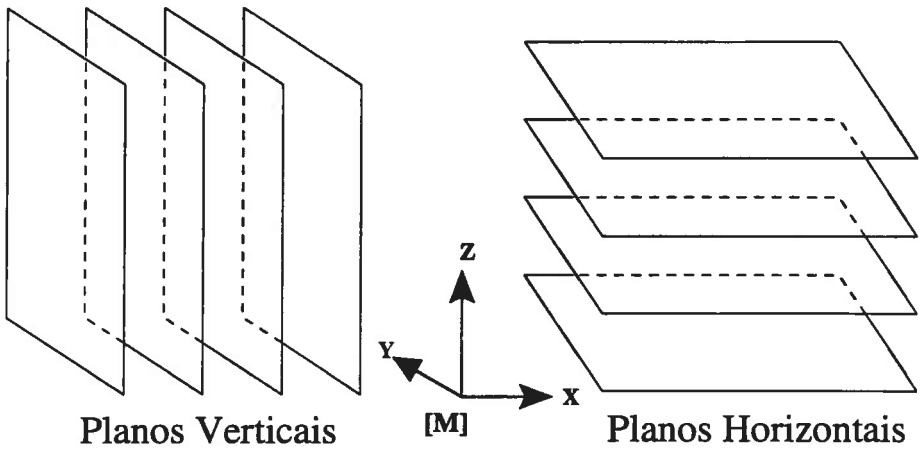


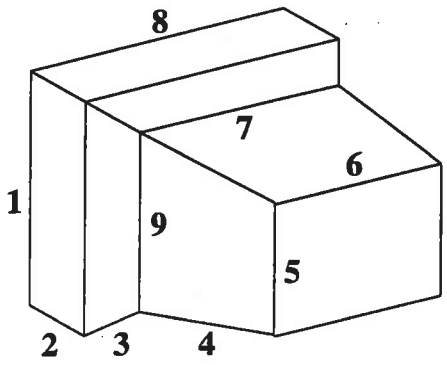

Figura 6.9: Conjunto de planos verticais e horizontais, paralelos e uniformemente espaçados, utilizados para construir a malha que dá o aspecto tridimensional à representação gráfica da estrutura dos alvos. A distância entre os planos pode ser variada de forma a melhorar o efeito visual, sendo aproximadamente igual a $2cm$ na figura 6.7.

## 6.5 Seguimento de Contornos

Uma maneira de automatizarmos o processo de recuperação de estrutura consiste em, directamente da análise das imagens, detectar quais os pontos do alvo que interessaria obter para construir o seu modelo tridimensional. Isto implica uma definição de políticas de observação e análise de cenas, tarefa que, à partida, não se afigura trivial. Uma maneira mais simples, eventualmente servindo como base para uma abordagem mais complexa, é fazer o controlo do sistema de visão activa de forma a que esta siga os contornos dos alvos, memorizando os vértices conforme estes vão sendo encontrados. Uma vez que o assunto foge demasiadamente dos objectivos deste trabalho, apenas abordaremos aqui os aspectos mais directamente relacionados com a fixação. A solução proposta resume-se à utilização da transformação *log-polar*, abordada no capítulo 3, como ponto de partida para um processo de seguimento de contornos.

A transformação para o plano *log-polar* possui características que podem ser exploradas num processo de seguimento de contornos, como veremos de seguida. Associando-lhe a detecção de contornos, podemos esquematizar o novo processo como se vê na figura 6.11. De um modo geral, a aplicação funcionará de forma análoga à descrita na secção anterior. As alterações surgem apenas quando se define um ponto do alvo. Nessa altura, o processo automático é activado, tentando o sistema detectar outros vértices. Para tal, deverá percorrer sistematicamente os contornos do alvo, até que seja atingida uma situação de bloqueio.

Para simplificar a nossa análise, vamos supor que os contornos são aproximadamente rectilíneos. Passamos assim a ter apenas um problema de seguimento de rectas no plano *log-polar*. Nestas condições, o primeiro passo terá de ser a obtenção da descrição matemática de uma recta no plano *log-polar*. Comecemos então por definir uma recta no plano cartesiano (ver a figura 6.12). A equação matemática mais comum para representar uma recta é $y = mx + b$, que é geral para rectas não verticais. Não é, no entanto, a expressão mais favorável. Uma representação alternativa e geral para todas as rectas é a seguinte:




| aresta | real | medida |
|---|---|---|
| 1 | 28 | 30.1 |
| 2 | 13 | 14.1 |
| 3 | 5 | 5.1 |
| 4 | 24 | 25.1 |
| 5 | 21 | 21.6 |
| 6 | 29 | 31.0 |
| 7 | 31 | 29.1 |
| 8 | 36 | 35.7 |
| 9 | 24 | 26.1 |

Figura 6.10: Comparação entre as dimensões reais do monitor utilizado como alvo e as distâncias euclidianas entre os vários pontos recolhidos (em centímetros).

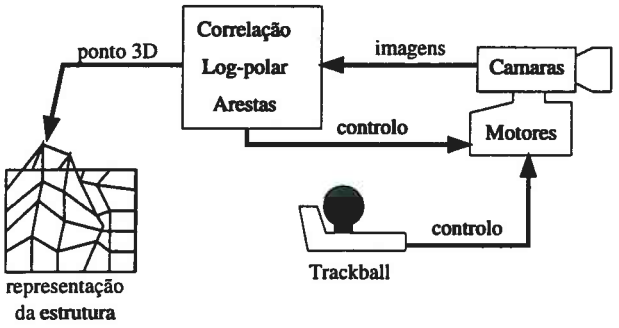



Figura 6.11: Automatização do processo de recuperação de estrutura.

$$r = x\cos(\alpha) + y\sin(\alpha) \tag{6.10}$$

Definida a recta no plano cartesiano, segue-se a sua transformação para o plano *log-polar*, definido pelos eixos $\rho$ e $\theta$. As expressões que nos permitem fazer a transformação são as seguintes:

$$\begin{cases} x = base^{\rho}\cos(\theta) \\ y = base^{\rho}\sin(\theta) \end{cases} \tag{6.11}$$

Estas relações não são mais do que as conhecidas equações de transformação para coordenadas polares. A única diferença está no facto de se ter substituído o raio por $base^{\rho}$, com vista à obtenção de um eixo radial logarítmico. Substituindo as relações 6.11 em 6.10 e simplificando através da utilização das vulgares relações trigonométricas, chega-se a:

$$r = base^{\rho}\cos(\theta - \alpha) \tag{6.12}$$

Analisando a equação 6.12, revelam-se-nos alguns aspectos interessantes. Um deles prende-se com as rectas que passam na origem, isto é, aquelas que têm um valor de $r$ nulo. De facto, substituindo $r$ por 0 em 6.12, temos:

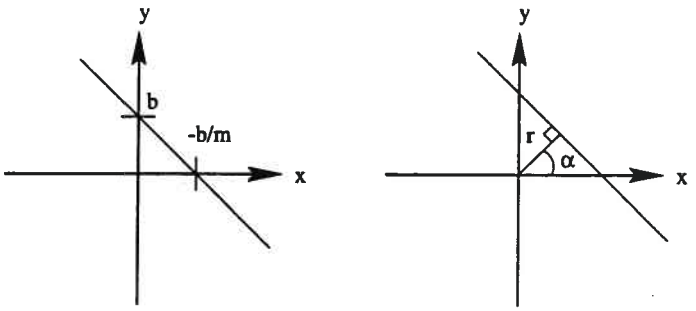



Figura 6.12: Duas formas matemáticas de representar uma recta. A da esquerda corresponde à relação $y = mx + b$. A da direita corresponde à relação $r = x\cos(\alpha) + y\sin(\alpha)$.

$$\begin{aligned} 0 &= base^{\rho}\cos(\theta - \alpha) \\ \Rightarrow\ 0 &= \cos(\theta - \alpha) \\ \Rightarrow\ \theta &= \alpha + \frac{\pi}{2} + n\pi \end{aligned} \tag{6.13}$$

Limitando $\theta$ ao intervalo de 0 a $2\pi$ conclui-se que, a transformação de uma recta que passe na origem, resulta em dois segmentos de recta, paralelos ao eixo $\rho$ e separados por uma distância $\pi$. Como exemplo, atente-se à recta da esquerda na figura 6.13, com $\alpha = -\frac{\pi}{4}$. Substituindo em 6.13 e atendendo ao intervalo de 0 a $2\pi$, obtêm-se dois segmentos de recta: $\theta = \frac{\pi}{4}$ e $\theta = \frac{5\pi}{4}$.

Quando $r$ não é nulo, terá um valor positivo, já que se trata de uma distância. Isso implica que o cosseno tenha de ser positivo:

$$\begin{aligned} r &\geq 0 \\ \Rightarrow\ \cos(\theta - \alpha) &\geq 0 \\ \Rightarrow\ |\theta - \alpha + 2n\pi| &\leq \frac{\pi}{2} \end{aligned} \tag{6.14}$$

Uma vez que o cosseno é uma função par, a cada valor de $\rho$ correspondem sempre dois valores de $\theta$ (no intervalo de 0 a $2\pi$). Como exemplo, temos a recta da direita na figura 6.13, onde $\alpha$ é nulo e $r$ é igual a 10. A curva resultante da transformação tem um valor mínimo para $\rho$, que se obtém exactamente quando $\theta = \alpha$, donde $\rho_{min} = \log_{base} r$.

Damos assim por terminada a análise matemática. Adaptando-a agora ao problema do seguimento de contornos no plano *log-polar*, podemos inferir algumas regras e características úteis. Note-se que as conclusões apresentadas assumem a utilização de imagens *log-polar* às quais foi aplicado um detector de contornos. Nestas condições, podemos concluir que:

- Se as câmaras estão fixas numa aresta, isto é, se a sua projecção passar no centro das imagens, a aresta será representada no plano *log-polar* por segmentos de recta horizontais, independentemente da sua orientação. Uma vez que a detecção de linhas horizontais numa imagem de contornos é um processo relativamente simples e célere, esta característica permite concluir facilmente se as câmaras estão ou não fixas numa aresta.

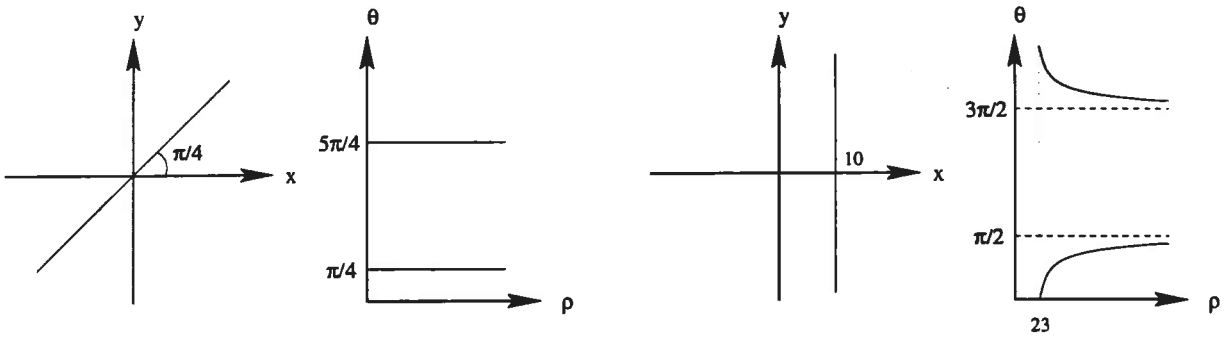



Figura 6.13: Duas rectas e respectiva representação no plano *log-polar*. A da esquerda passa pela origem ($r = 0$) e tem uma inclinação de $45^o$ relativamente ao eixo horizontal ($\alpha = -\frac{\pi}{4}$). A da direita é uma recta vertical, cujos parâmetros são $r = 10$ e $\alpha = 0$.

- Se, no plano *log-polar*, existirem vários segmentos de recta, a distância que os separa na vertical é suficiente para nos dizer se são representações da mesma aresta, caso em que a distância é igual a $\pi$, ou representam arestas diferentes, caso em que a distância é diferente de $\pi$, podendo concluir-se que nos encontramos sobre um vértice. Estas conclusões derivam do facto de, qualquer semi-recta que parta da origem, ser representada por um segmento de recta no plano *log-polar*. Se duas semi-rectas, aplicadas na origem, formam um ângulo $\alpha$, a sua distância vertical será exactamente $\alpha$.

- Caso as câmaras não estejam fixas sobre um aresta ou um vértice, as curvas representadas no plano *log-polar*são descritas pela expressão $r = base^{\rho} \cos(\theta - \alpha)$, como se viu atrás. Não precisaremos, no entanto, de trabalhar com aquela expressão, para concluir alguma coisa sobre a aresta. De facto, devido às características do plano *log-polar*, podemos saber rapidamente qual a menor distância a percorrer para atingir a aresta, bem como a direcção a tomar. Para tal, basta descobrir qual é o ponto da curva com menor valor de $\rho$. A distância a que se encontra do centro da imagem e a direcção a seguir para o atingir são dadas pelas suas coordenadas.

Finalmente, refira-se que é possível generalizar as conclusões anteriores para contornos não rectilíneos. Para tal, basta que a análise dessas curvas seja feita em arcos suficientemente pequenos, a ponto de poderem ser aproximados a uma recta. Note-se que, se decidirmos optar por esta abordagem, o plano *log-polar* facilita-nos o trabalho. Para isso, temos simplesmente de limitar a nossa análise a uma área próxima do eixo $\theta$. Dependendo da grandeza das curvaturas das arestas a seguir, teremos como único entrave a esta solução, a resolução e qualidade das imagens *log-polar* nessa área.

# Capítulo 7

# Discussão e Conclusões

No ano lectivo de 1993/94 teve início um projecto de simulação de processos visuais, com base numa plataforma móvel e num sistema de visão activa já existentes. Ao longo destes anos, foi sucessivamente atingindo as diversas metas traçadas no percurso para a obtenção de um sistema funcional com capacidade para reagir em tempo real. Integrado no projecto VARMA (Visão Activa para Robótica Móvel Autónoma), financiado pela JNICT (Junta Nacional de Investigação Científica e Tecnológica), gerou alguns demonstradores que foram sendo documentados em diversas publicações [FON94] [DIA95-1] [PAR96], e ainda na edição de 1995 da *International Conference on Robotics and Automation* do IEEE [DIA95-2].

## 7.1 Resumo do Trabalho Realizado

Esta tese integra-se no projecto VARMA na área relativa à simulação artificial de processos de realização de fixação visual. Com vista à evolução do trabalho e dos processos utilizados e aproveitando a experiência e os conhecimentos acumulados, propusemo-nos melhorar o desempenho do módulo de visão do sistema. Logo à partida, traçaram-se os seguintes objectivos:

- Concentrar esforços no desenvolvimento de processos visuais aplicáveis não só ao seguimento de alvos, mas também à recuperação de estrutura, vigilância, detecção de movimento, extracção de contornos, ou outros.

- Originar um sistema capaz de funcionar em tempo real.

Começámos por definir uma linha de orientação, a que chamámos fixação visual e na qual baseámos todo o trabalho. Modelizámos a fixação visual tendo como ponto de referência os exemplos biológicos, cuja análise foi feita de forma meramente intuitiva. Depois de ponderadas as soluções mais bem sucedidas de outros investigadores na área, definimos a fixação visual como (1) o acto de seleccionar um alvo numa cena, (2) orientar o sistema na sua direcção e (3) manter a sua projecção estável no centro de ambas as câmaras. Para tal, considerando as limitações práticas do nosso sistema, decidiu-se que:

- O alvo está fixo se for visível em ambas as câmaras.

- Todos os movimentos têm como finalidade centrar nas imagens as projecções do alvo.

- Os alvos a fixar devem reunir características que permitam facilitar a sua detecção nas imagens.

Assim, foi possível esquematizar a espinha dorsal do processo de fixação visual, que dividimos em três fases: Espera, Selecção e Fixação. De seguida, com vista à realização prática do processo, optámos por investigar técnicas de análise e tratamento de imagens, que analisámos pesando factores como a qualidade dos resultados, a complexidade computacional e a velocidade de execução. Daqui resultou a selecção de um leque de métodos com origens em diferentes áreas:

- O plano *log-polar* e as filtragens da área das transformações.
- As pirâmides da área da compactação de dados e reduções de escala.
- O fluxo óptico e o fluxo óptico normal da área da detecção de movimento.
- A extracção de contornos da área da recuperação de características das cenas.
- A correlação da área da correspondência de imagens.

Além do desempenho individual, ponderámos a forma como essas técnicas se poderiam interligar e quais as vantagens que daí resultariam. O trabalho dedicado às diversas técnicas foi efectuado mantendo sempre em perspectiva as disponibilidades e limitações do *hardware* existente, que explorámos em pormenor, tendo desenvolvido algoritmos com funcionamentos ao nível do tempo real para todas elas.

Munidos dos algoritmos, esquematizámos os três estados do processo de fixação visual em termos das suas componentes de análise e tratamento de imagens, bem como do fluxo de dados que ocorre entre os diversos blocos. O diagrama resultante pretende ser uma solução integrada de resolução dos diversos problemas que compõem cada um dos estados do processo, não necessariamente aplicável como solução optimizada (ou não) de qualquer problema concreto. A intenção é, unicamente, possuir abordagens práticas e céleres à generalidade das situações particulares que surgem com qualquer problema, sendo sempre necessário fazer uma avaliação do interesse ou não em incluir cada um dos módulos na solução final.

## 7.2 Resultados Obtidos e Trabalho Futuro

Concluímos com a descrição de duas aplicações, perseguição e recuperação de estrutura, marcadamente de características e objectivos distintos, mas com uma espinha dorsal comum, que é o processo de fixação proposto. Os resultados preliminares do trabalho foram apresentados na $2^{nd}$ *Portuguese Conference on Automatic Control*, sob o título *Recovering Shapes and Detecting Movements using Fixation*, que teve lugar no Porto durante o mês de Setembro de 1996 [PAR96]. Nos capítulos 5 e 6 mostrámos análises mais cuidadas do desempenho do processo nas duas situações, embora longe de poderem ser consideradas exaustivas. A velocidade dos algoritmos mostrou-se crucial no caso da perseguição, permitindo compensar as limitações dos motores de passo do sistema de visão activa. Já no caso da recuperação de estrutura, os resultados ficam a dever-se, principalmente, à facilidade com que foi possível integrar o processo de fixação visual com a fase não automática da solução proposta.

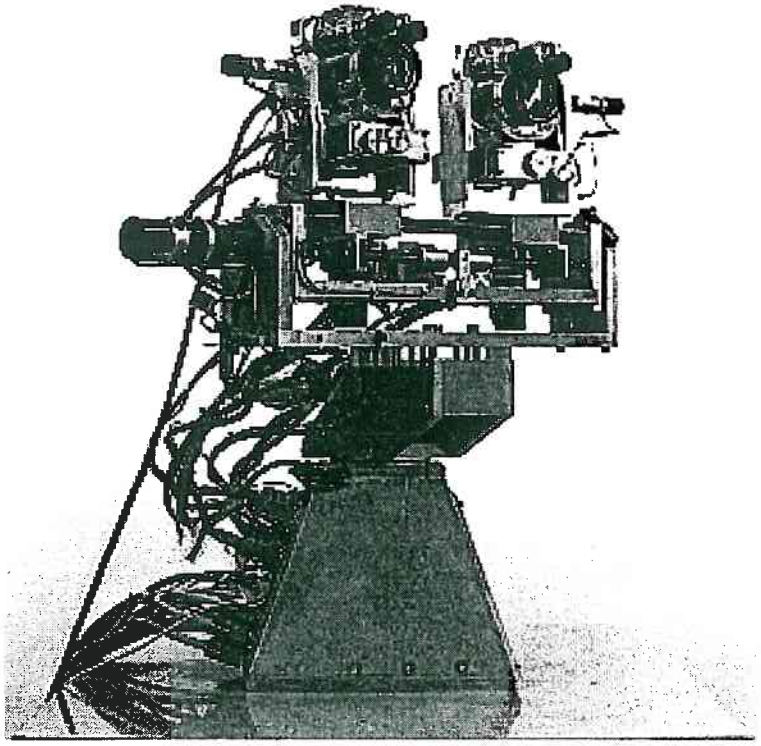


Figura 7.1: Novo sistema de visão activa. É baseado em motores de corrente contínua, possuindo maior precisão e velocidade do que a disponível no sistema de visão activa utilizado neste trabalho.

Em aberto ficam algumas linhas de acção, tanto nas aplicações apresentadas, como em outras potenciais aplicações. A automatização do processo de recuperação da estrutura tridimensional é um bom exemplo, sendo a utilização da detecção de contornos no plano *log-polar* uma ideia a explorar. Fica também a possibilidade de utilizar *hardware* mais flexível, nomeadamente fazendo a transferência do processo para o novo sistema de visão activa, já disponível no Pólo de Coimbra do Instituto de Sistemas e Robótica (ver a figura 7.1).

# Parte II

# Apêndices

# Apêndice A

# *Hardware*

Neste apêndice, vamos conhecer em pormenor cada um dos módulos físicos que constituem o sistema, esquematizados no diagrama de blocos da figura A.1: os dois processadores digitais de sinal (vulgos DSPs), o processador do computador pessoal e o sistema de visão activa. Qualquer deles pode funcionar independentemente dos outros, permitindo a paralelização das tarefas e o aumento da velocidade global do sistema. A sincronização dos vários módulos é efectuada pelo processador do computador pessoal, sendo por isso apelidado de unidade central de processamento. Os DSPs são responsáveis unicamente pela tarefa de captar e processar as imagens. De entre os periféricos do computador pessoal, destaca-se a existência de uma *trackball*, utilizada no controlo manual do sistema de visão activa e descrita no fim do apêndice.

## A.1 Sistema de Visão Activa

O sistema de visão activa é uma estrutura metálica basculante, controlada por motores de passo, com duas câmaras do tipo CCD no topo [DIA93]. Possui 5 graus de liberdade no total: inclinação frontal e rotação horizontal da estrutura, rotação horizontal independente das câmaras e variação da distância horizontal entre elas. Na posição vertical, o centro das câmaras encontra-se a $46,3cm$ da base e a $20,6cm$ do eixo de inclinação da estrutura. O máximo afastamento dos centros das câmaras é de $29,1cm$.

Os motores de passo disponíveis são operados por controlo posicional, não sendo possível o controlo *on-the-fly*. Nas tabelas A.1 e A.2 podemos ver os parâmetros dos motores responsáveis pela rotação das câmaras e pela inclinação e rotação da estrutura. Não se inclui o motor que controla a distância entre as câmaras, também conhecida por *baseline*, visto que se pretende que as câmaras estejam sempre no seu máximo afastamento. Quanto aos valores fornecidos, refira-se que as velocidades apresentadas são valores típicos e não máximos. Além disso, os descodificadores das câmaras têm metade da resolução dos motores respectivos o que, na prática, limita a resolução a $0,36°/passo$ ou $\approx 2,7passos/°$.

## A.2 DSPs

A captura e processamento de imagens é da responsabilidade de uma placa da *Transtech Parallel Systems*, com a designação TDMB412, na qual acentam dois módulos TIM-40: o TDM435 (*framegrabber*) e o TDM407 (ver a figura A.2). Ambos os módulos são baseados nos DSPs TMS320C40 [TI92] (vulgos C40) da *Texas Instruments*, funcionando a 40MHz

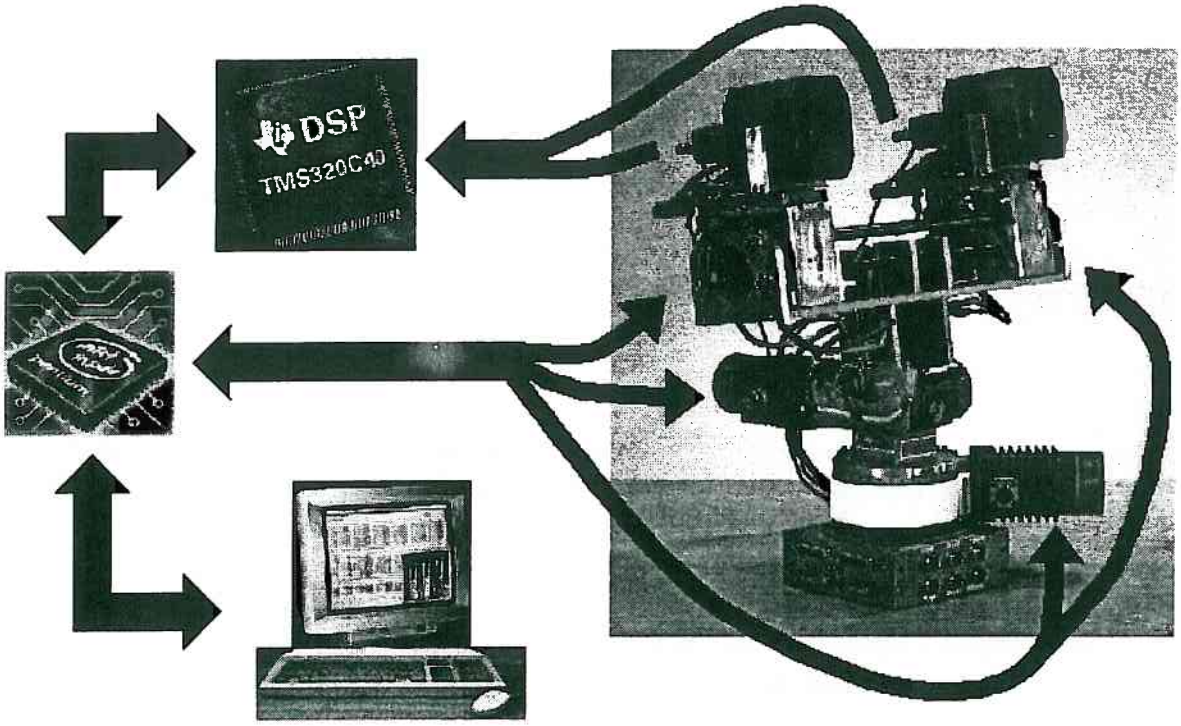



Figura A.1: Diagrama de blocos do *hardware* utilizado.

Tabela A.1: Parâmetros dos motores, em passos.

| **Motor** | **Excursão** (*passos*) | **Velocidade** (*passos*/*s*) | **Resolução** (*passos*/°) |
|---|---|---|---|
| Câmaras | $-185$ a 185 | $\approx 188$ | $\approx 5,6$ |
| Inclinação | $-300$ a 300 | $\approx 207$ | 20 |
| Rotação | $-10^4$ a $10^4$ | $\approx 363$ | 100 |

Tabela A.2: Parâmetros dos motores, em graus.

| **Motor** | **Excursão** (°) | **Velocidade** (°/*s*) | **Resolução** (°/*passo*) |
|---|---|---|---|
| Câmaras | $-33,3$ a $33,3$ | $\approx 33,84$ | $0,18$ |
| Inclinação | $-15$ a 15 | $\approx 10,35$ | $0,05$ |
| Rotação | $-100$ a 100 | $\approx 3,63$ | $0,01$ |

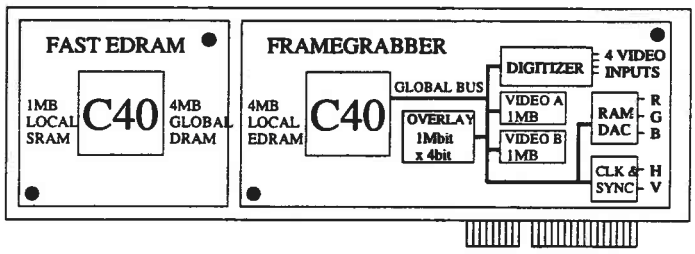



Figura A.2: Placa de expansão constituída pelos processadores digitais de sinal e pela unidade de aquisição de imagens.

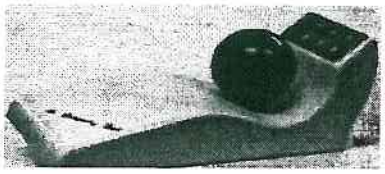


Figura A.3: *Trackball* utilizada para controlar manualmente o sistema de visão activa.

e capazes de atingir 50 MFLOP e 275 MOPS. O TDM407 combina a velocidade de processamento com a capacidade de memória, explorando um sistema denominado *Enhanced DRAM* (EDRAM), capaz de zero estados de espera. O *framegrabber* possui 4 canais de entradas vídeo independentes, com resolução programável, fazendo uma discretização monocromática de 8 bits e permitindo a sua representação RGB à saída. Inclui ainda 2 bancos de memória de $1024 \times 1024 \times 8bits$ e um banco adicional de sobreposição de 4 bits. A comunicação entre os DSPs é feita por um anel de comportas bidireccionais, capazes de transferir 20MBytes por segundo e cujo controlo, por DMA, não implica o bloqueio do processamento dos DSPs. A comunicação com a unidade central de processamento é feita através do BUS do computador pessoal, permitindo velocidades de transferência até 2MBytes por segundo.

## A.3 Unidade Central de Processamento

Como foi já referido, o controlo do sistema é da responsabilidade do processador do computador pessoal. O processador é um 80486DX2 da Intel, funcionando a 66MHz, com 8MBytes de memória RAM e 256KBytes de memória *cache*. Em face da autonomia dos controladores dos motores e dos DSPs, a unidade central tem apenas que os supervisionar e coordenar, podendo dedicar-se ao controlo dos periféricos, à gestão do interface gráfico e, principalmente, à execução do processo de fixação visual.

## A.4 *Trackball*

Em virtude do número de graus de liberdade do sistema de visão activa, torna-se complicado fazer o seu controlo manual usando periféricos comuns. A solução encontrada foi a de usar uma *trackball*, visto possuir flexibilidade necessária para o desempenho daquela tarefa, sem acarretar custos de processamento. A *trackball* representada na figura A.3

permite a detecção da magnitude das forças e torques aplicados à bola de controlo, num total de 6 graus de liberdade: deslocamento segundo X,Y e Z e rotação segundo os mesmos eixo. Os resultados detectados são comunicados à unidade central de processamento através do interface série RS-232. O formato e a política de envio são programáveis, assim como a velocidade da linha. Adicionalmente, no painel frontal da *trackball*, encontramos 8 botões de pressão com *leds* sinalizadores, cujo funcionamento é controlável por *software*.

# Apêndice B

# Relatório Prático

## B.1 Introdução

Numa perspectiva de continuidade do trabalho realizado, é imprescindível que se prevejam futuras alterações ou reutilizações dos programas concebidos, parcialmente ou na totalidade. É com esse objectivo que iniciamos este apêndice. Mais do que descrever o *software* desenvolvido, vamos isolar as funções e algoritmos em pacotes que possibilitarão uma fácil adaptação a trabalhos distintos, generalizando assim a sua utilidade. O código que compõe esses pacotes pode ser encontrado numa disquete disponível no arquivo do grupo de Robótica Móvel do Departamento de Engenharia Electrotécnica da Universidade de Coimbra.

No total, existem 5 pacotes, tratados separadamente nas próximas secções: Interface Gráfico, Memória Estendida, *Trackball*, Motores e Processadores Digitais de Sinal (ou DSPs). O Interface Gráfico e o controlo da Memória Estendida são os pacotes mais simples, constituindo um bom ponto de partida para quem pretende tirar o máximo partido dos programas. A complexidade dos pacotes que controlam a *Trackball* e os Motores reside mais na conveniência de se dominar as características do *hardware*, do que nos programas propriamente ditos. Quanto ao pacote dos DSPs, embora contendo apenas os ficheiros e as rotinas básicas é, ainda assim, o maior e o mais complexo, exigindo um estudo profundo até que dele se tire o máximo proveito.

Para terminar, veremos na secção B.7 uma forma de interligar todos os pacotes, usando como exemplo a aplicação global desenvolvida durante a parte prática do trabalho. Esta aplicação integra, entre outros, os módulos descritos nos capítulos 5 e 6.

## B.2 Interface Gráfico

O pacote do Interface Gráfico, ou GUI (do inglês *Graphical User Interface*), foi baseado numa biblioteca de funções gráficas para o compilador de C da *Microsoft*. Essa biblioteca disponibiliza um número alargado de funções, incluindo o controlo do rato, funcionando com a generalidade dos modos gráficos permitidos actualmente pela placa gráfica Super VGA. Os ficheiros encontram-se na directoria **GUI**, a partir da raiz da disquete, resumindo-se na tabela B.1 a sua finalidade.

Existe ainda uma directoria de nome **B**, a partir da directoria **GUI**, dedicada unicamente à criação dos botões utilizados no GUI. Os ficheiros lá existentes estão listados na tabela B.2.

| | |
|---|---|
| convpal.asm | - Rotina de conversão de paletas de cores |
| gui.c | - Ficheiro principal |
| make.bat | - Compilação dos ficheiros *convpal.asm* e *gui.c* |
| svgacc.h | - Definições da biblioteca gráfica |
| svgacc.lib | - Biblioteca gráfica |
| svgacc.txt | - Manual da biblioteca gráfica |

Tabela B.1: Ficheiros que constituem a biblioteca de funções gráficas.

| | |
|---|---|
| bgiobj.exe | - Converte formatos binários para objectos |
| clear.btn | - Botão de apagar (usado pelo *vgaedit*) |
| conv.c | - Converte os botões e gera o *header-file buttons.h* |
| conv.exe | - Versão executável do ficheiro *conv.c* |
| exit.btn | - Botão de sair (usado pelo *vgaedit*) |
| getter.btn | - Botão de ler (usado pelo *vgaedit*) |
| saver.btn | - Botão de salvar (usado pelo *vgaedit*) |
| vgaedit.exe | - Editor de botões |

Tabela B.2: Ficheiros dedicados à criação dos botões.

Ao utilizar este pacote, é possível construir interfaces de características semelhantes ao representado na figura B.1, obtido com a função *Snapshot* (ver a secção B.2.4) depois de ter sido seleccionada uma tarefa de visualização de imagens estéreo.

### B.2.1 Botões

Os botões são a única forma de interacção com o GUI actualmente disponível. Tal acontece apenas por uma questão de opção, já que, como veremos na secção B.2.3, a introdução do controlo do teclado é trivial. Para criar os botões, basta ir para a directoria **GUI\B** e executar o ficheiro *vgaedit.exe*. Este programa não é mais do que um rudimentar editor gráfico, que permite a criação, ponto por ponto, de desenhos de pequenas dimensões, ideais para servirem como botões. O controlo do programa baseia-se unicamente no rato, sendo bastante fácil de dominar. As cores disponíveis aparecem na parte superior do ecrã, num total de 256. No entanto, por questões de paleta, a utilização de cores com códigos superiores a 15 não será bem interpretada pelo GUI.

Existem apenas quatro opção no programa: salvar (botão com uma disquete), ler (botão com um livro), apagar (botão com uma rede) e sair (botão com uma porta). Uma vez concluída a criação do botão, deverá salvar-se o *bitmap* premindo o botão com a disquete e escolher um nome. Para que o botão seja facilmente integrado no GUI, o nome deverá respeitar duas condições: começar por uma letra e ter a extensão *.BTN*

Depois de criados os botões, deverá executar-se o ficheiro *conv.exe*. Este programa converte todos os botões criados para o formato objecto, acrescentando uma entrada por botão ao ficheiro *buttons.h*. O programa verifica quais os botões que já foram convertidos anteriormente e detecta se o ficheiro *buttons.h* já existe, pelo que não é necessário fazer alterações ao processo descrito, quer se estejam a criar botões pela primeira vez, ou simplesmente a acrescentar.

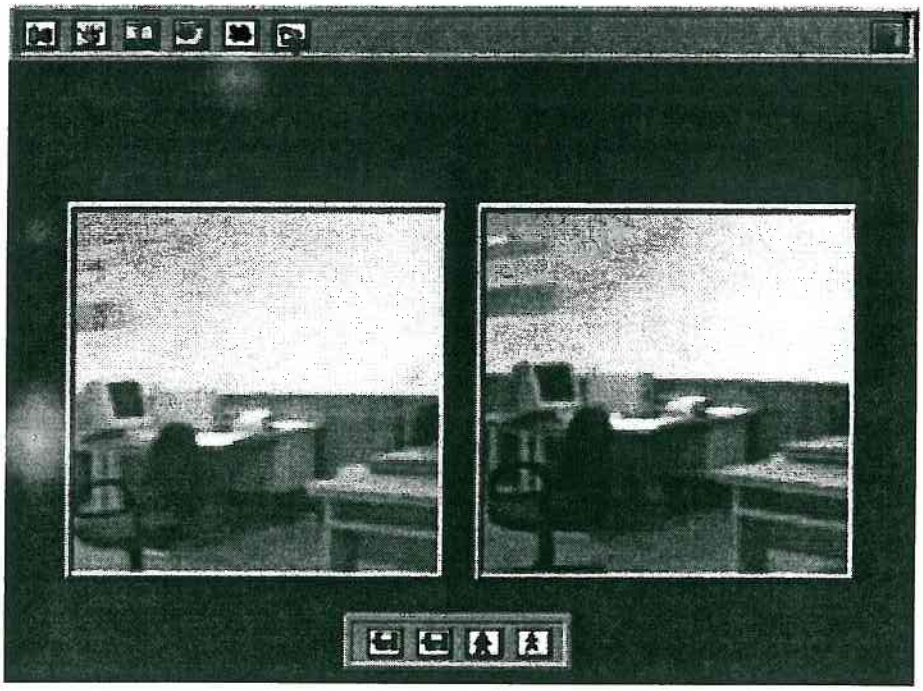


Figura B.1: Imagem global do interface gráfico.

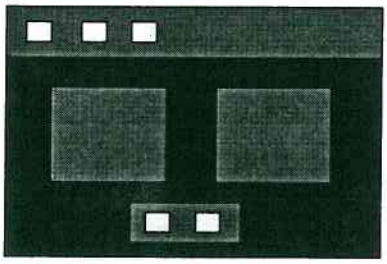


Figura B.2: Exemplo de um *desktop*.

### B.2.2 *Desktop*

Actualmente, a construção do *desktop* baseia-se unicamente em janelas fixas e em botões. Na figura B.2 está representado um exemplo de uma configuração possível do *desktop*. No topo do ecrã podemos ver a barra com os botões que definem as tarefas (três neste caso). Esta barra é mantida inalterada durante toda a sessão. Por baixo da barra de tarefas vemos três janelas e dois botões. Esta área corresponde à zona utilizável do ecrã, sendo automaticamente apagada e redesenhada de cada vez que se muda de tarefa. Como veremos de seguida, a definição das tarefas pretendidas está bastante simplificada, bem como a construção dos *desktops* associados a cada uma delas.

Editando o ficheiro *gui.c*, vamos começar por procurar a palavra **TASKS**. A primeira ocorrência verifica-se na função *InitGUI* junto da definição das variáveis *nms* e *pos*. Estas variáveis definem, respectivamente, o nome dos botões que representam as tarefas existentes e as posições em que eles devem aparecer na barra de tarefas. Como exemplo, vamos supor que o nosso GUI permite as três tarefas seguintes: cálculo do fluxo óptico, detecção de contornos e abandono do programa (esta opção não existe por defeito, devendo ser

sempre definida). Com vista à representação destas tarefas, criaram-se três botões com os seguintes nomes: *FLUXO.BTN DETECT.BTN* e *SAIR.BTN*. De seguida, foi executado o ficheiro *conv.exe*, que acrescentou ao *header-file buttons.h* uma entrada por cada um dos botões, atribuindo-lhes os nomes dos ficheiros, sem a extensão *.BTN*. Nestas condições, as variáveis *nms* e *pos* deveriam ser iguais a:

```
RasterBlock far *nms[]={ &FLUXO, &DETECT, &SAIR ,0 };
int pos[]={ 10,45,606 };
```

Note-se que, na variável *nms*, o *zero* existente no final é imprescindível. Relativamente às posições dos botões, apenas é fornecida a localização horizontal, sendo a única restrição o facto de as posições não deverem ultrapassar o valor 639, já que o modo gráfico escolhido tem uma resolução de 640 × 480 pontos. Continuando a procurar a palavra **TASKS**, encontramos a segunda ocorrência no fim da mesma função, integrada na expressão **INACTIVE TASKS**. Como o nome indica, esta zona do programa é dedicada às tarefas inactivas. Se, após a inicialização do programa, pretendermos que alguma tarefa não possa ser acedida, basta chamar a função *Inactive* passando o número da tarefa como parâmetro. O número da tarefa corresponde à sua posição na variável *nms*, começando a contagem em 0. Por exemplo, para desactivar a tarefa de cálculo do fluxo óptico, utiliza-se:

```
if (Condição Inicial) { Inactive(0); }
```

A última ocorrência da palavra **TASKS** leva-nos à definição dos ecrãs para cada tarefas, que é feita através das variáveis $DNum$ e $DPos$, relativas às janelas fixas, e das variáveis $BNum$, $Bpos$ e $BNm$, associadas aos botões. Começando pelos botões, temos de decidir quantos terá cada uma das nossas tarefas, bem como as posições em que devem aparecer. No nosso exemplo, a primeira tarefa terá dois botões: um para iniciar o cálculo e outro para terminar, necessitando a segunda tarefa de apenas um: o de executar. A terceira tarefa não precisa de qualquer botão. Assim, vamos começar por definir todos os botões com as variáveis *Bpos* e $BNm$, de forma análoga à que usámos para os botões das tarefas:

```
int BPos[][2] = { { 200,400 }, { 400,400 }, { 300,400 } };
RasterBlock far *BNm[] = { &INICIAR, &TERMINAR, &EXECUTAR };
```

Como se pode ver, há algumas diferenças que importa realçar:

- As posições dos botões incluem agora o deslocamento vertical. Recorde-se que o modo gráfico utilizado tem uma resolução de 640 × 480 pontos, não podendo os botões ultrapassar esses limites.
- Não é necessário colocar um *zero* no fim variável $BNm$, se bem que a sua inclusão não afecte o programa.
- A ordem com que os botões são definidos é relevante, devendo ser respeitada a sequência com que as tarefas foram definidas. Adicionalmente, os botões relativos a uma mesma tarefa devem aparecer na lista em posições consecutivas.

A definição das janelas fixas é em tudo semelhante à definição dos botões, usando-se para tal a variável *DPos*. Prosseguindo com o exemplo, queremos uma única janela em cada uma das tarefas (excluindo novamente a opção de abandonar o programa), sendo maior a janela da primeira tarefa. Teremos então algo parecido com:

```
int DPos[][5] = {   { 100,100,440,280, THICK},
                    { 200,200,240,80, THIN}      };
```

A variável anterior define duas janelas, com os cantos superiores esquerdos posicionados em 100, 100 e 200, 200 respectivamente. A primeira janela terá uma dimensão de $440 \times 280$ pontos, com uma moldura grossa. A segunda janela será mais pequena, ocupando $240 \times 80$ pontos e terá uma moldura fina. Tal como para o caso dos botões, também nas janelas é importante a ordem com que é feita a definição.

Resta definir quais as janelas e/ou botões pertencentes a cada uma das tarefas. Para tal, usam-se as variáveis *DNum* e *BNum*, respectivamente para as janelas e para os botões. Estas variáveis terão tantas entradas como o número de tarefas definidas, sendo cada entrada composta por dois números: um índice e um contador. O índice é utilizado para indicar, para a tarefa a que corresponde, qual é a posição da primeira janela e/ou botão nas listas que os definem, isto é, em *Bpos*, *BNm* e *DPos*. Quanto ao contador, serve para indicar quantas janelas e/ou botões pertencem à tarefa. Caso sejam 2 ou mais, as suas definições podem ser encontradas nas listas a partir da posição indicada pelo índice (daí a necessidade de serem definidas consecutivamente). Chegamos assim, no nosso exemplo, às seguintes definições:

```
int DNum[][2] = { { 0,1 }, { 1,1 }, { 0,0 } };
int BNum[][2] = { { 0,2 }, { 2,1 }, { 0,0 } };
```

Como se pode ver, os índices começam em *zero*. No caso de o contador ser nulo, o valor do índice deve ser igualmente nulo. De um modo geral, para verificar se não há erros na definição das variáveis, basta verificar as seguintes regras (os pares nulos, isto é, com índice e contador igual a zero, devem ser ignorados):

- O índice da primeira tarefa terá sempre de ser nulo.
- A soma do índice com o valor do contador é igual ao índice da próxima tarefa.
- A soma do índice da última tarefa com o valor do contador respectivo deverá ser igual ao número total de janelas ou botões definidos, conforme o caso.

Finalmente, note-se que, se houver sobreposição, os botões aparecem sempre por cima das janelas.

### B.2.3 Eventos

Definindo que está o *desktop*, não é necessária mais nenhuma alteração para se começar a utilizar o GUI. Para além das funções de inicialização (*InitGUI*) e fecho (*CloseGUI*), a utilização do GUI resume-se a chamar uma única função: *GUIEvent*. Esta função retorna muito rapidamente, libertando o processador para outras tarefas mais de acordo com a

aplicação em causa. Consegue-se assim, de uma forma simples, obter a mesma funcionalidade dos sistemas controlados por eventos. Os valores devolvidos pela função *GUIEvent* podem ser positivos, negativos ou nulos, em função do tipo de acção empreendida pelo utilizador:

- Se os valores são nulos, não foi detectada qualquer acção válida por parte do utilizador. As únicas acções válidas neste momento são a pressão de um dos botões. No entanto, se o botão premido corresponder à tarefa actualmente em curso, a acção é ignorada.

- Os valores negativos indicam uma mudança de tarefa, sendo a tarefa identificada pelo simétrico do valor retornado. Por exemplo, recorrendo ao caso utilizado na secção anterior, a selecção do botão *SAIR.BTN* resultaria no retorno do valor $-3$. Note-se que, assim que o utilizador escolhe outra tarefa, o ecrã é apagado e é desenhado automaticamente o novo *desktop*.

- Por último, os valores positivos correspondem à pressão dos botões que não representam tarefas. Assim, qualquer que seja a tarefa, ao primeiro botão a ela associado corresponderá o valor 1, ao segundo o valor 2, etc.

Tal como se acabou de referir, assim que o utilizador opta por mudar de tarefa, o *desktop* é imediatamente substituído, estando já alterado quando a função retorna. A partir daí, a única interferência do GUI no ecrã resume-se ao deslocamento do rato. Isto implica que, sempre que se pretenda alterar o ecrã, é necessário esconder o rato, fazendo-o aparecer depois das alterações. As funções de escrita no ecrã e de controlo do rato, bem como muitas outras com finalidades diversas, estão descritas e explicadas no manual da biblioteca gráfica, incluído na directoria **GUI** sob o nome *svgacc.txt*

Caso se pretenda aumentar a versatilidade do GUI, nomeadamente acrescentando o controlo do teclado, basta analisar a função *GUIEvent* e alterá-la em conformidade. Para além de ter um tamanho reduzido, a função é modular, podendo acrescentar-se qualquer tipo de controlo antes da última linha, isto é, do *return(0)*, correspondente à indicação da inexistência de eventos.

### B.2.4 *Snapshot* e *CONVPAL*

Para terminar o estudo do pacote GUI, vamos ver duas funções não incluídas na biblioteca, e que são de grande utilidade: a função *Snapshot* e a função *CONVPAL*. A função *Snapshot*, definida no ficheiro *gui.c*, efectua a captura do ecrã gráfico, gravando-o depois no formato *raster*, colorido, para um ficheiro de nome *screen.ras*. No ficheiro apenas são incluídas as primeiras 80 entradas da paleta de cores, correspondendo às 16 cores habituais seguidas de 64 tonalidades de cinzento. Isto porque o GUI foi construído tendo em mente a representação de imagens monocromáticas, habitualmente possuindo 256 níveis de cinzento (recorde-se que a VGA não permite a definição de mais do que 64 tonalidades da mesma cor).

A função *CONVPAL*, definida no ficheiro *convpal.asm*, efectua a conversão de imagens com 256 níveis de cinzento, gerando imagens que podem ser representadas com a paleta de cores descrita no parágrafo anterior. Esta função foi optimizada, podendo ser utilizada apenas em processadores a partir do 80386, e sempre trabalhando sobre imagens com tamanhos múltiplos de 4.

## B.3 Memória Estendida

As rotinas de acesso à memória estendida são bastante simples. Para funcionarem, necessitam que tenha sido instalado no PC um gestor de memória alta. A sua utilização deverá revestir-se de alguns cuidados, já que exploram um recurso que é partilhado pela generalidade dos programas. A melhor maneira de perceber a forma como funcionam é com exemplos, pelo que, após uma breve descrição, terminaremos a secção com um programa exemplo onde constataremos a simplicidade da sua aplicação.

### B.3.1 Descrição

O pacote de gestão da Memória Estendida encontra-se na directoria **XMS**, a partir da raiz da disquete. É composto apenas por dois ficheiros: *xms.asm*, que contém as rotinas em Assembly e *make.bat* que executa a compilação das rotinas. O quadro seguinte enumera as rotinas incluídas no ficheiro *xms.asm*, descrevendo sumariamente a sua finalidade.

| | |
|---|---|
| XMSExists | - Verifica a existência de um gestor de memória alta |
| RequestHMA | - Requisita memória alta |
| ReleaseHMA | - Liberta memória alta |
| EnableA20 | - Activa a linha de endereçamento número 20 |
| DisableA20 | - Desactiva a linha de endereçamento número 20 |
| GetXM | - Requisita memória estendida |
| FreeXM | - Liberta memória estendida |
| ToXM | - Copia para a memória estendida |
| FromXM | - Copia da memória estendida |

A descrição pormenorizada das características envolvidas na gestão da memória estendida está fora dos objectivos deste relatório. Algumas das funções anteriores interferem de forma radical com o funcionamento do computador, devendo ser utilizadas com precaução e apenas quando há um conhecimento concreto das acções executadas.

### B.3.2 Exemplo

Nesta secção vamos analisar, passo a passo, as funções mais importantes do pacote e a forma de as utilizar. Para começar, temos de testar se existe ou não um gestor de memória alta instalado no PC:

```
int version;
printf("XMS driver: "); version=XMSExists();
if (version==0) { printf("not loaded!\n"); return; }
else printf("version %x\n",version);
```

Como se viu, a função *XMSExists* não se limita a detectar a existência do gestor, retornando também a sua versão. Note-se que o valor retornado vem em código BCD,

devendo ser representado como se de um hexadecimal se tratasse. No entanto, deve ser lido como se fosse um inteiro.

Caso exista um gestor de memória estendida, o próximo passo será reservar alguma dessa memória. Para tal, usamos a função *GetXM*, que tenta reservar toda a memória estendida disponível.

```
unsigned EMBSize;
if ((EMBSize=GetXM()) < 1024)
        { printf("Not enough extended memory space\n"); return; }
else printf("EMB: %uK allocated.\n",EMBSize);
```

Quando retorna, a função *GetXM* devolve o número de *KBytes* reservados. Se esse valor for superior ou igual a 1024, a memória foi reservada, podendo ser utilizada pelas funções *ToXM* e *FromXM*. Caso contrário, a função não reserva a memória. A necessidade da existência de, pelo menos, *1 MByte* de memória livre para que a função a reserve, pode facilmente ser alterada no ficheiro *xms.asm*. Para tal, bastará procurar a função *GetXM* e modificar o teste à memória estendida livre, no início da função. Actualmente, depois de obtida a memória livre existente, é feita uma comparação com o valor 1024. É este o valor que deve ser alterado.

As funções que copiam de e para a memória estendida, respectivamente *FromXM* e *ToXM*, recebem três parâmetros de entrada, enumerados de seguida: o número de *bytes* que pretendemos transferir, o ponteiro para a área da memória de onde ou para onde queremos copiar dados e o *offset* da área de memória alta para onde ou de onde queremos copiar dados. Note-se que o número de *bytes* a copiar mais o *offset* da área de memória alta nunca poderá ser superior ao número de *bytes* de memória estendida reservados. De seguida poderemos ver um exemplo da aplicação destas funções:

```
extern ToXM(long count,unsigned char far *block,long XMoffset);
extern FromXM(long count,unsigned char far *block,long XMoffset);
...
ToXM(1024, buf1, 0);
ToXM(1024, buf2, 1024);
...
FromXM(1024, buf1, 1024);
FromXM(1024, buf2, 0);
```

A acção executada no exemplo anterior resume-se à troca dos dados contidos num dos *buffers* pelos dados do outro. Note-se que, neste caso, optou-se por pré-definir as funções, visto que os parâmetros utilizados são do tipo *long*, o que poderia originar problemas de execução não detectáveis aquando da compilação.

Antes de terminar o programa, é imprescindível que se invoque a função *FreeXM*, para que a memória reservada seja libertada. Se tal não acontecer, nenhum outro programa que recorra ao gestor de memória alta poderá aceder a essa área da memória.

## B.4 *Trackball*

A *trackball* para a qual foi concebido o pacote que agora se descreve, é uma *Geoball*, modelo *Geometry Ball*, da CIS Graphics, Inc. Antes de se ler esta secção, é aconselhável a leitura do manual do utilizador [CIS89], particularmente a parte respeitante à comunicação com a *trackball*. O pacote é constituído apenas por três ficheiros, que podem ser encontrados na directoria **TBALL** a partir da raiz da disquete. São eles:

| | |
|---|---|
| lowlevel.asm | - Rotinas de baixo nível |
| make.bat | - Compilação dos ficheiros *lowlevel.asm* e *rs232.c* |
| rs232.c | - Rotinas de controlo da *trackball* |

### B.4.1 Funções

Tal como já deu para desconfiar pelo nome do ficheiro, o pacote de controlo da *trackball* não é mais do que a adaptação de rotinas de controlo do porto série do PC, cujo protocolo utilizado é o *RS-232*. É claro, as rotinas não se limitam a fazer o controlo das comunicações, tratando também as mensagens enviadas pela *trackball*, o que poupa o trabalho de extrair os parâmetros de força e torque lá incluídos. Para utilizar o pacote, bastará executar a sequência seguinte:

```
init_serial(); receive_ON(); /* Inicialização da comunicação série */
...
/* detecção de forças e torques */
while ((func=getFandT(&Xt,&Yt,&Zt,&Xr,&Yr,&Zr))!=0) { ... }
...
close_serial(); /* Finalização da comunicação série */
```

Estas e outras funções incluídas no ficheiro *rs232.c* serão agora descritas em pormenor.

*init_serial*

Esta função instala o serviço à interrupção que controla a comunicação série. Note-se que não é feito qualquer tipo de teste relativamente ao serviço actualmente instalado. A função limpa o *buffer* de recepção e deixa as comunicações desactivadas (para as activar, usa-se a função *receive_ON*). Na configuração actual, é utilizado o primeiro porto série ou COM1. Para utilizar o COM2, basta incluir a linha *#define COM2* no início do programa.

*close_serial*

Esta função desinstala o serviço à interrupção instalado pela função *init_serial*. Note-se que não é feito qualquer tipo de teste relativamente ao serviço actualmente instalado. Ao sair, a função deixa as comunicações desactivadas.

*receive_ON*

Para activar a comunicação no porto série, deve-se chamar esta função. Caso não tenha sido instalado o serviço à interrupção com a função *init_serial*, o resultado será imprevisto.

*receive_OFF*

Para desactivar a comunicação no porto série, deve-se chamar esta função. Importa não confundir a desactivação com a desinstalação, já que a primeira corresponde unicamente a uma interrupção na recepção de dados, enquanto que a outra implica a alteração do serviço de resposta à interrupção.

*Warmstart*

Ao ser invocada, esta função executa o *warmstart* da *trackball* (ver o manual). De seguida, é programado o modo *Standard Dialog* e o protocolo *Binary*. Em caso de sucesso, a função retorna o valor 1. Esta função só deverá ser utilizada depois de instalado o serviço à interrupção e de se ter activado a comunicação. Se a *trackball* estiver já configurada com o modo e o protocolo indicados (através dos *dip switches*), não é necessário executar esta função aquando da inicialização das comunicações.

*getFandT*

Com o modo e o protocolo indicados anteriormente, a *trackball* envia automaticamente as rotações e translações efectuadas pelo utilizador, sempre que elas acontecem. Utilizando esta função, poderemos verificar se há algum comando da *trackball* no *buffer* de recepção. Se houver, a função retorna os parâmetros de força e torque relativos a esse comando. Caso contrário, retorna valores nulos. As forças e torques são retornados em 6 variáveis do tipo *char*, cujos ponteiros devem ser fornecidos à função, podendo os valores variar dentro do intervalo $-128$ a 127. Adicionalmente, a função retorna um valor do tipo *char*, cujos bits determinam o estado actual dos botões de pressão existentes no painel da *trackball*.

Note-se que esta função retira do *buffer* de recepção apenas um dos comandos enviados pela *trackball*. Os restantes, caso existam, permanecem no *buffer*. Se a frequência com que esta função é invocada não for superior à frequência de envio da *trackball*, será impossível fazer reagir o programa em sincronia com os comandos do utilizador.

### B.4.2 Controlo

De um modo geral, a *trackball* permite detectar claramente a ocorrência de movimentos de translação e rotação em qualquer dos sentidos e segundo qualquer dos três eixo de um referencial tridimensional aplicado à bola. No entanto, os valores das forças e torques recolhidos dependem bastante do tipo de movimento e do eixo em causa. Daí que a sua análise deva ser feita em função da aplicação pretendida e da sensibilidade do utilizador, sendo difícil definir uma estratégia global de acção. Na secção B.7 veremos uma abordagem possível.

## B.5 Motores

As rotinas de controlo dos motores podem ser encontradas no ficheiro *mt.c*, dentro da directoria **MOTORS** a partir da raiz da disquete. Além daquele ficheiro, apenas existe na directoria o ficheiro *make.bat*, que é utilizado na compilação das rotinas. Nesta secção abordaremos as funções mais importantes, do ponto de vista de quem quer apenas compilar e correr os programas.

### B.5.1 Funções

De entre todas as funções, apenas são necessárias 3 para fazer funcionar o sistema com um mínimo de controlo. São elas a função de inicialização, a que executa os movimentos e a que lê os descodificadores. De seguida descreve-se o seu funcionamento.

*home*

A função *home* tenta inicializar o sistema, movendo depois cada um dos motores para a posição definida como origem (vulgarmente designada por *home*). Por razões práticas, o motor número 2, responsável pelo controlo da *baseline*, apenas é iniciado, não sendo alterada a sua posição. Caso se pretenda afastar as câmaras o mais possível (máxima *baseline*), pode usar-se a função *maxBaseline*. Quando retorna, a função *home* devolve o valor 1 caso tenha completado a inicialização com sucesso, e 0 no caso contrário.

*descread*

Esta função lê o descodificador do motor cujo número lhe é fornecido como parâmetro. Os valores retornados indicam o número de passos a que o motor se encontra da posição definida como origem. O sentido de rotação é determinado pelo sinal desse valor, podendo as relações entre graus e passos, bem como outras características, ser consultadas no apêndice *Hardware* da tese. Os números associados aos motores estão indicados no início do ficheiro. A importância desta função reside no facto de que os motores, ao terminarem o seu movimento, poderão ou não ter atingido as posições pretendidas. Torna-se assim fundamental saber a que posição chegaram, antes de se efectuarem cálculos com base na sua localização.

*go_to*

A função *go_to* baseia-se na função básica de movimentação dos motores, *move*, permitindo a movimentação e controlo simultâneos de todos os motores. Como parâmetros, a função recebe 4 variáveis do tipo *long*, que determinam as posições, em passos, para onde pretendemos mover os motores (ver o cabeçalho da função para saber quais os motores suportados).

Além daquelas variáveis, a função recebe ainda duas *flags*: a de *feedback* e a de compensação. Quando não se quer *feedback*, a função lança os comandos de movimentação a todos os motores, respeitando os limites máximos (ver a secção B.5.2), retornando antes de os movimentos estarem terminados. Caso se opte pelo *feedback*, a função entra em ciclo, tentando aproximar os motores das posições especificadas. O ciclo termina quando todos os motores estão nas posições pretendidas, ou então depois de um número máximo de iterações.

Finalmente, a *flag* de compensação. Se estiver activa, as rotações enviadas ao *pan* (motor que controla a rotação da estrutura segundo o plano horizontal) são subtraídas às rotações enviadas às câmaras. Em termos práticos, isto significa que o *pan* não interfere com as posições espaciais das câmaras. Para mais informações, sugere-se a consulta do capítulo 5.

### B.5.2 Parâmetros

Actualmente, os parâmetros de movimentação e de limitação dos motores foram definidos com vista à execução de movimentos elementares que não excedam a duração de 1 décimo de segundo. Os parâmetros de limitação dos motores estão definidos no início do ficheiro, com os nomes $CTV$, $NPTV$ e $NTTV$, respectivamente para as câmaras, para o *pan* e para o *tilt*. Embora o seu nome implique velocidades máximas (TV significa *Top Velocity*), não é de velocidades que se trata, mas sim do número máximo de passos percorridos pelos motores em cada ciclo. Aqui, ciclo tem o mesmo significado utilizado na descrição da função *go_to*, definindo a movimentação elementar dos motores.

Os parâmetros de limitação foram determinados em consonância com os parâmetros de movimentação, definidos pela variável *params*, na zona inicial do ficheiro. Há 3 parâmetros por motor, F, R e S, que definem respectivamente a velocidade de arranque, a velocidade máxima e a aceleração/desaceleração do motor. A relação entre estes parâmetros e os valores reais respectivos não é linear, nem sequer trivial, devendo ser estudada directamente a partir das especificações dos controladores utilizados. Neste caso, a sua obtenção foi feita recorrendo ao método da tentativa e erro, tendo como objectivo principal a máxima rotação num intervalo de tempo determinado.

## B.6 Processadores Digitais de Sinal

Para que os ficheiros incluídos neste pacote funcionem sem problemas, é necessário que o *software* da TPS (*Transtech Parallel Systems*) tenha sido correctamente instalado, preferencialmente nas directorias definidas por defeito. Adicionalmente, as ligações das comportas não deverão ter sido alteradas, nem o endereço base dos portos da placa, que é 200h. Na estrutura actual, o pacote representa uma forma simples de iniciar uma aplicação baseada nos DSPs, possuindo apenas algumas rotinas optimizadas de transferência de dados entre o *host* (CPU do computador pessoal) e o *master* (C40 responsável pelo acesso ao BUS). Para além dessas rotinas, tudo o resto é apenas o estritamente necessário para que a aplicação possa correr.

### B.6.1 Ficheiros

Os ficheiros que compõem o pacote podem ser encontrados na directoria **DSPS**, a partir da raiz da disquete. A tabela B.3 descreve sumariamente esses ficheiros.

Nas secções seguintes vamos abordar separadamente os programas que irão correr em cada um dos 3 processadores envolvidos. Embora não seja necessário um conhecimento real sobre o funcionamento do processo para conseguir tirar dele algum proveito, aconselha-se a leitura dos manuais fornecidos pela TPS e a experimentação com os programas exemplo, antes de se utilizar este pacote. Uma das razões é que, aquando da eliminação de todas as rotinas não estritamente necessárias, foram removidas as rotinas que permitem ao C40 *master* utilizar funções como o *read* ou o *printf*. Essas funções, na realidade, eram executadas pelo PC, já que o C40 não comunica com os periféricos. Aliás, nos exemplos da TPS, o PC não é mais do que um servidor, o que justifica os nomes de *master* e *slave* para os C40, e *server* para o computador pessoal (substituído por *host* no nosso caso).

| | |
|---|---|
| disp435.h | - *Header-file* do *master.c* |
| error.h | - *Header-file* do *tops.c* |
| fifo.obj | - Rotinas de comunicação (usadas pelo *tops.c*) |
| go.bat | - Chama o programa *host* com a configuração da rede de DSPs |
| host.h | - *Header-file* do *tops.c* |
| load.nd | - Configuração da rede de DSPs |
| makem.bat | - Compilação do *master.c* |
| makes.bat | - Compilação do *slave.c* |
| maket.bat | - Compilação do *tops.c* |
| master.c | - Rotinas destinadas ao C40 *master* |
| master.cmd | - Parâmetros de compilação do *master.c* |
| paceload.h | - *Header-file* do *tops.c* |
| pack.obj | - Rotinas de comunicação (usadas pelo *tops.c*) |
| pipe.h | - *Header-file* partilhado pelo *master.c* e pelo *slave.c* |
| prts.h | - *Header-file* do *master.c* |
| slave.c | - Rotinas destinadas ao C40 *slave* |
| slave.cmd | - Parâmetros de compilação do *slave* |
| topenlnk.obj | - Rotinas de comunicação (usadas pelo *tops.c*) |
| tops.c | - Rotinas destinadas ao *host* (CPU do PC) |
| tops.h | - *Header-file* partilhado pelo *master.c* e pelo *tops.c* |
| tproto.h | - *Header-file* do *tops.c* |
| unpack.obj | - Rotinas de comunicação (usadas pelo *tops.c*) |

Tabela B.3: Ficheiros que compõem o pacote.

### B.6.2 *tops.c* e *fifoload*

O programa que corre no *host* é responsável pelo carregamento da rede, ou mais exactamente, pela comunicação do código do *master* e do *slave* aos C40 respectivos. De um modo geral, este módulo é igual ao fornecido nos exemplos da TPS, pelo que não nos oferece grandes comentários. Depois de iniciada a rede, o *host* poderá fazer o que quiser, nomeadamente comunicar com o C40 *master*. Para tal, basta escrever o código por baixo da linha onde se pode ler:

```
/* DO WHAT YOU WANT FROM NOW ON... */
```

Quando chega a esta linha, o programa já obteve uma indicação do C40 *master*, confirmando a inicialização. A troca de informações entre o *host* e o *master*, quando em pequeno volume, pode ser assegurada pelas funções *pack_** e *unpack_**. No entanto, no caso de se pretender transferir uma bloco de dados de grandes dimensões, essas funções revelam-se demasiadamente lentas. Por isso, foi desenvolvida a função *fifoload*, que executa a transmissão com base numa *FIFO*, acelerando consideravelmente a transferência. Esta função recebe dois parâmetros: o ponteiro para o *buffer* onde serão carregados os dados e o número de *KBytes* que serão recebidos. A particularidade desta função é poder receber qualquer tipo de dados, desde *chars* até *floats*. Atente-se, no entanto, aos seguintes pormenores:

- No caso de se querer transferir *floats*, é necessário utilizar, do lado do *master*, a

função *fifodumpfloat*. Isto porque o formato dos *floats* utilizado pelos C40 é diferente do formato IEEE, necessitando de ser convertido. No entanto, graças às funções paralelas dos C40, a velocidade de transmissão é igual quer se usem *floats* ou *chars*.

- Nos C40, a unidade mínima de armazenamento são $32bits$. Isto implica que, por exemplo, um *array* de 1000 *chars* ocupe 4000 bytes, isto é, o mesmo espaço ocupado por um *array* de 1000 *floats*. Para evitar problemas, aconselha-se o uso unicamente de *ints* e *floats* nos DSPs, e de *longs* e *floats* no PC. Desta forma, a *arrays* de iguais dimensões corresponderão áreas de memória de igual tamanho.

No ficheiro *tops.c* incluído no pacote, a área da função *main* dedicada à aplicação, isto é, abaixo da linha " DO WHAT YOU WANT FROM NOW ON...", foi preenchida com um ciclo muito simples. Esse ciclo permite definir as linhas gerais de como se devem processar as comunicações entre o *host* e o *master*, já que se baseia numa política de sincronização dos processadores, desejável a todos os níveis. O código do ciclo é o seguinte:

```
printf("Do something...");
operation=TASK_SOMETHING; pack_int(&operation,1,0);
tops_flush(0);
unpack_int(&operation,1,0); printf("done.\n");
```

Descrevendo o ciclo por palavras, temos que:

- O *host* decide mandar o *master* executar a tarefa TASK_SOMETHING, definida no *header-file* partilhado por ambos (*tops.h*).
- Uma vez que foi usada uma função *pack_**, é necessário fazer o *flush* do *buffer* para que o comando seja enviado. Isto não acontece com as funções *fifodump* e *fifodumpfloat*, utilizadas pelo *master*.
- Finalmente, o *host* espera até que a tarefa esteja concluída, altura em que o *master* lhe deve enviar um valor de retorno.

Caso o *host* pretenda fazer uma espera activa, dispõe das macros *FIFONotEmpty* e *FIFOEmpty*, que detectam respectivamente se não há e se há dados que podem ser lidos. Estas macros podem ser usadas tanto com as funções *pack_** e *unpack_** como com as funções *fifo**.

### B.6.3 *master.c* e *fifo**

Ao ser activado, o *master* começa por verificar se o *slave* está activo, confirmando depois a inicialização bem sucedida ao *host*. Posto isto, entra no seguinte ciclo de espera e execução de comandos. Para executar uma hipotética tarefa TASK_SOMETHING, vamos supor que o *master* necessita do apoio do *slave*, mandando-o executar a tarefa TAG_SOMETHING. As mensagens enviadas entre os DSPs são definidas no *header-file* partilhado por ambos (*pipe.h*), estando a responsabilidade da comunicação a cargo das funções *in_word*, *out_word*, *receive_msg* e *send_msg*. A sintaxe das rotinas difere do *master* para o *slave*, devendo consultar-se o cabeçalho de cada um dos ficheiros para se saber a forma correcta de as utilizar. Voltando ao ciclo de espera e execução, o seu funcionamento é o seguinte:

```
while(1) {
        unpack_int(&cmd,1,0);
        switch(cmd) {
        case TASK_SOMETHING:
                out_word(TAG_SOMETHING,0); in_word(5);
                pack_int(&cmd,1,0); tops_flush(0);
                break; } }
```

Depois de enviar o comando ao *slave*, o *master* aguarda até que o processamento termine, confirmando então ao *host* que a tarefa está concluída. Se for necessário enviar blocos de dados para o *host*, o *master* deverá recorrer às funções *fifodumpload* ou *fifodump*, respectivamente para transmitir *floats*, ou para transferir qualquer outro tipo de dados. Ambas as funções recebem dois parâmetros: o ponteiro para o *buffer* que contém os dados a enviar e o número de unidades de $32bits$ a enviar, devendo este número ser múltiplo de 256. Por exemplo, se queremos enviar um *array* de 1000 *floats*, devemos transferir 1024 unidades de $32bits$, o que implicará a recepção pelo *host* de $4KBytes$, correspondentes a um *array* de 1024 *floats*. Isto porque, por questões de velocidade, os menores blocos transmitidos pelas funções *fifo** são de $1KByte$.

### B.6.4 *slave.c* e DMA

O programa que corre no *slave* é em tudo idêntico ao que corre no *master*, fruto de uma dependência do *master* semelhante à que este tem relativamente ao *host*. Por essa razão, o ciclo de espera e execução do *slave* percorre os mesmos passos descritos anteriormente para o *master* enquanto processa os comandos do *host*. No entanto, o processo de comunicação através das comportas que ligam os DSPs difere grandemente do que se utiliza na comunicação via BUS entre o DSP *master* e a CPU do computador. Uma vez que o *slave* apenas pode comunicar através de comportas e somente com o *master*, esta é uma boa altura para nos concentrarmos no seu funcionamento e nas suas características, de entre as quais se destaca a velocidade, que poderá atingir os $40MBytes$ por segundo.

Tal como foi referido anteriormente, a forma como os DSPs comunicam implica a utilização de funções cuja sintaxe está descrita no cabeçalho dos ficheiros respectivos. De seguida reproduz-se o cabeçalho do ficheiro *slave.c*:

```
DOWNLOAD PROCEDURE
                        tags:   in_word(3);

                     buffers:   receive_msg(3, buffer, 1);
                                while(chk_dma(3));

UPLOAD PROCEDURE
                        tags:   out_word(tag,0);

                     buffers:   send_msg(0, buffer, size, 1);
                                while(chk_dma(0));
```

Como se pode ver, quer se envie ou se receba, existem sempre duas formas possíveis de o fazer. A exemplo do que se passa entre as funções *pack_** e as funções *fifo**, também para a transmissão entre os DSPs é melhor utilizar rotinas optimizadas quando existe um grande volume de dados para transferir. No entanto, ao contrário das funções *fifo**, as funções **_msg* apenas indicam ao controlador de DMA (*Direct Memory Access*) que pretendem fazer uma transferência, não necessitando de esperar pela sua conclusão. Sempre que se quiser verificar o estado da transferência, basta utilizar a função *chk_dma* para testar se o controlador de DMA está activo. Esta particularidade permite paralelizar o processamento com a transmissão de dados, aumentando significativamente a velocidade total das tarefas executadas.

## B.7 Pacote Global

Tal como foi referido anteriormente, vamos utilizar a aplicação prática desenvolvida como apoio à tese para exemplificar a aplicação dos diversos pacotes de *software* descritos. Como se poderá imaginar, a complexidade desta aplicação impede a sua abordagem em pormenor num trabalho com estas características. Nesse sentido, a finalidade desta secção é, tão somente, abordar algumas das rotinas e técnicas desenvolvidas, eminentemente relacionadas com a área da visão por computador, que poderão vir a ser úteis em trabalhos futuros.

Na raiz da disquete encontra-se um arquivo comprimido, de nome *tps.zip*, que contém todos os ficheiros criados durante a parte prática do trabalho. Para descomprimir esse arquivo, deverá usar-se o ficheiro *pkunzip.exe*, também incluído na raiz da disquete, executando o seguinte comando:

- pkunzip -d tps.zip

A lista de ficheiros arquivados é demasiado extensa para ser aqui enumerada, podendo ser consultada no ficheiro *readme* existente no arquivo. Em termos gerais, trabalhar com este pacote global equivale a gerir simultaneamente todos os pacotes descritos anteriormente. Nessa gestão inclui-se a necessidade de compilar individualmente todos os ficheiros principais, o que, a exemplo dos pacotes individuais, está facilitado com a existência de *batch-files*. Depois de serem gerados os ficheiros executáveis, pode-se iniciar a aplicação executando o ficheiro *go.bat*. Na secção seguinte veremos o que acontece depois de se iniciar a aplicação, bem como as tarefas nela incluídas. Esperamos assim facilitar a absorção das restantes secções, bastante técnicas, mas fundamentais para se tirar o melhor partido do trabalho realizado.

### B.7.1 *Go...*

Ao ser executado, o ficheiro *go.bat* carrega o programa do *host* que, por sua vez, carrega os programas dos DSPs. Se as inicializações forem concluídas sem problemas, deverá aparecer de seguida o interface gráfico (na figura B.1 podemos ver o seu aspecto depois de seleccionada a primeira tarefa). As tarefas disponíveis, seleccionáveis através dos botões existentes na barra superior do ecrã, são as seguintes:

Captação e representação contínua das imagens obtidas pelas duas câmaras. Esta tarefa inclui as opções de ler/gravar as imagens e de bloquear/desbloquear a aquisição.

Detecção dos contornos no último par de imagens captado com a primeira tarefa. É utilizado o operador de Sobel, seguindo da selecção de máximos locais, descrita na secção 3.5. Esta tarefa inclui a possibilidade de variar o limiar de aceitação dos módulos dos gradientes de intensidades.

Esta tarefa corresponde à aplicação descrita no capítulo 5.

Controlo da cabeça mecânica utilizando a *trackball*. No ecrã aparece um modelo gráfico, representado na figura 6.2 de duas perspectivas diferentes. O modelo gráfico acompanha os movimentos da cabeça mecânica, sendo possível escolher entre vários pontos de vista do observador.


Esta tarefa corresponde à aplicação descrita no capítulo 6, podendo ver-se na figura 6.7 um exemplo do *desktop* a ela associado.


Ao premir este botão, o ecrã actual é gravado num ficheiro de nome *screen.ras*, no formato *raster-file*. A tarefa que se encontrava activa continua a funcionar sem alterações (ver a secção B.2.4).


Termina o programa.

### B.7.2 Amostragem

Embora dispondo de autonomia de funcionamento, a placa de aquisição de imagens está associada ao DSP *master*, sendo este responsável pelo seu controlo. Depois de adquirida uma imagem, o *master* deverá copiá-la dos *buffers* da placa para uma área de memória local, de forma a que não se perca a imagem na próxima aquisição. A isto vamos chamar "amostragem". No ficheiro *master.c* existem várias funções de amostragem, optimizadas em função das diferentes resoluções das imagens resultantes.

Antes de descrevermos essas funções, importa realçar que, por razões de velocidade e facilidade de análise, a placa de aquisição foi configurada de forma a que as imagens de ambas as câmaras fossem captadas para o mesmo *buffer*. Na figura B.3 é possível ver a forma como isto foi conseguido. Como se pode ver, esta opção implica a perca de metade da resolução vertical das imagens. Uma vez que a resolução original das imagens é de $768 \times 574$ pontos, aquela redução não representa qualquer tipo de problema, já que, à partida, é demasiadamente elevada para permitir análises em tempo real.

As funções de amostragem disponíveis, incluídas no ficheiro *master.c*, serão enumeradas já de seguida. Como veremos, nas descrições utilizam-se os conceitos de "empacotado" e de "compacto". O conceito de empacotado está relacionado com o tamanho do mais pequeno bloco de memória disponível, vulgarmente um *byte*, mas que nos DSPs corresponde a 32 *bits* (isto é, 4 *bytes*). Uma vez que cada ponto das imagens ocupa um *byte*, se optarmos por guardar 4 pontos por bloco, dizemos que as imagens estão "empacotadas", visto

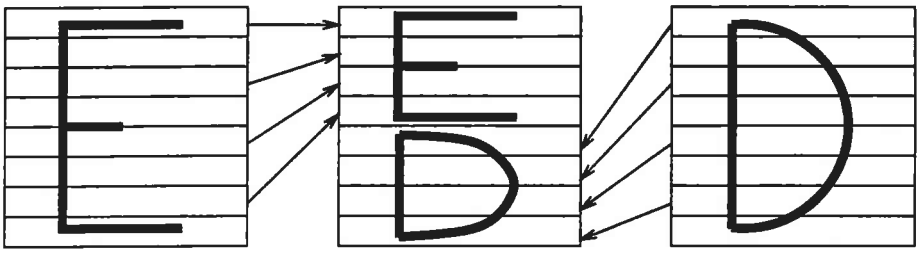


Figura B.3: Aquando da captação, apenas é aproveitado um campo das imagens de cada uma das câmaras (**E**squerda e **D**ireita). Uma vez que as imagens são compostas por dois campos entrelaçados, o resultado são imagens com metade da resolução vertical original. Adicionalmente, as imagens são captadas de forma a ficarem justapostas num dos *buffers* da placa de aquisição.

que estamos a reduzir para 25% o espaço de memória necessário. O conceito de compacto está relacionado com a perca de resolução vertical das imagens. Se, ao amostrarmos as imagens, não tivermos o cuidado ou não quisermos efectuar uma redução para metade da resolução horizontal, dizemos que as imagens resultantes são "compactas". As funções de amostragem são:

| | |
|---|---|
| sample256 | Amostra uma imagem de $256 \times 256$ pontos, empacotada. Esta função foi criada por razões históricas, já que a generalidade dos *framegrabbers* capta imagens com estas dimensões. |
| sampleC256 | Amostra uma imagem de $256 \times 256$ pontos, compacta e empacotada. Esta função acumula a vantagem de ser mais rápida do que a sua equivalente não compacta, com a de ter uma maior definição horizontal (útil quando a deformação não é problemática). |
| sample384 | Amostra uma imagem de $384 \times 287$ pontos, empacotada. A imagem resultante é a maior imagem não compacta que se pode obter com o esquema de aquisição utilizado. |
| sample192 | Amostra uma imagem de $192 \times 143$ pontos, empacotada. A imagem resultante equivale a reduzir 2 vezes a imagem gerada pela função *sample384*. |

Qualquer das funções recebe dois parâmetros de entrada: o ponteiro para o *buffer* da placa de aquisição e a área de memória onde se pretende guardar a imagem amostrada. Relativamente ao ponteiro para o *buffer* da placa de aquisição, importa salientar que, com a excepção da função *sample192*, o acesso à memória é sempre efectuado em modo *WORD*. Para evitar erros, estão definidas globalmente as constantes *WordA* e *ByteA*, destinadas ao acesso ao *buffer A* em modo *WORD* e em modo *BYTE* respectivamente. Ao chamar as funções de amostragem, sugere-se a utilização daquelas constantes, acrescidas de um *offset* quando necessário.

### B.7.3 Aquisição

A forma mais fácil de adquirir uma imagem, é utilizar a função *capture*. Esta função recebe um único parâmetro, 0 ou 1, respectivamente indicando se se pretende adquirir a imagem inferior ou superior. A designação das imagens, superior e inferior, é relativa ao seu posicionamento no *buffer* da placa. O funcionamento da função resume-se a configurar a placa de aquisição, seleccionar uma das entradas vídeo e esperar até que seja captado um campo completo. Estas tarefas poderão demorar entre 0.02 e 0.04 segundos, dependendo do tempo que a função tem de esperar até se iniciar um novo campo (cada campo tem uma duração de 0.02 segundos).

Quando o objectivo é o funcionamento em tempo real, esta função revela-se bastante lenta, já que não tira partido da autonomia da placa de aquisição. Nesse sentido, foram criadas funções que, usando o mesmo princípio, adquirem e amostram duas imagens, uma de cada câmara, de forma optimizada. A optimização é conseguida fazendo a amostragem de uma das imagens enquanto a outra está a ser adquirida. É claro, isto só é possível porque as rotinas de amostragem são executadas num intervalo de tempo inferior a 0.02 segundos. As funções de aquisição optimizadas são descritas de seguida. Note-se que nenhuma destas funções recebe parâmetros, já que as áreas de memória utilizadas para guardar as imagens amostradas estão definidas globalmente.

| | |
|---|---|
| regularCapture | Adquire uma imagem com cada câmara e amostra cada uma delas usando a função *sample256*. |
| compactCapture | Adquire uma imagem com cada câmara e amostra cada uma delas usando a função *sampleC256*. |

No caso de se pretender criar funções semelhantes a estas, mas utilizando outras funções de amostragem, não basta copiar o código e alterar o nome das funções. Isto porque as funções possuem código *Assembly* que inclui identificadores (vulgos *labels*). Se os *labels* não forem alterados, é gerado um erro na altura da compilação. Para terminar, refira-se que as funções *rec3DCapture* e *globalCapture*, responsáveis no *master* pelas aplicações descritas nos capítulos 6 e 5 respectivamente, utilizam este princípio de funcionamento. É claro, a complexidade das operações envolvidas é bastante superior.

### B.7.4 Correlação, *Log-Polar* e Detecção de Contornos

Nas duas últimas secções vimos algumas das funções incluídas no ficheiro *master.c*. As tarefas por elas executadas estão associadas à placa de aquisição, pelo que apenas o DSP *master* poderá utilizá-las. Nesta secção vamos ver algumas funções que podem ser utilizadas por qualquer um dos DSPs. Essas funções estão definidas nos ficheiros *match.c*, *dolog.c* e *edges.c*. Para as utilizar, basta incluí-las nos ficheiros *master.c* ou *slave.c*, recorrendo ao comando *#include "ficheiro.c"*.

A particularidade de todas estas funções é estarem integralmente escritas em *Assembly*. Por questões de velocidade, as funções foram concebidas com base nos problemas em causa, sendo pouco ou nada genéricas. Como pontos comuns têm o facto de trabalharem sobre imagens empacotadas e de estarem razoavelmente bem explicadas nos ficheiros fonte. Sempre que tal é viável, indica-se nos ficheiros a forma de as alterar para permitir a adaptação a diferentes parâmetros. São estas algumas das funções disponíveis:

| | |
|---|---|
| sobel | Aplica o operador de Sobel a uma imagem (o operador foi descrito na secção 3.5). |
| maximum | Aplica o algoritmo de selecção de máximos locais a uma imagem (o algoritmo foi descrito na secção 3.5). |
| dolog | Amostra uma imagem utilizando a transformação *log-polar*. Para tal recorre a uma tabela que deverá ter sido previamente gerada. Actualmente, essa tabela é enviada pelo *host* aquando das inicializações (ver a secção B.7.1). |
| FastMatch | Correlaciona um padrão com uma determinada área de uma imagem, utilizando a soma dos quadrados das diferenças e retornando a medida da sua semelhança. O facto de serem usadas imagens empacotadas levou a que tivessem sido criadas mais 3 versões desta função: FastMatch1, FastMatch2 e FastMatch3. Estas funções fazem o mesmo que a função FastMatch, assumindo um deslocamento da imagem de 1, 2 e 3 *bytes* respectivamente. |
| CopyPattern | Faz a extracção de uma área de uma imagem, com vista à sua utilização como padrão. |

As explicações fornecidas para as funções são meramente descritivas. A sua utilização e aproveitamento implicam a análise cuidada da forma como são usadas, acrescido de um estudo dos comentários incluídos no código, para além do próprio código sempre que for necessário.

### B.7.5 *Host*

As funções executadas no *host* estão divididas por 3 ficheiros: *tops.c*, *3drec.c* e *follow.c*. Estes ficheiros podem ser encontrados na directoria **SERVER**, depois de descomprimido o arquivo. Nos ficheiros *3drec.c* e *follow.c* podemos encontrar as funções responsáveis no *host* pelas aplicações descritas nos capítulos 6 e 5 respectivamente. No ficheiro *tops.c* encontram-se a generalidade das rotinas que tratam as restantes tarefas, incluindo o ciclo principal de execução (associado ao controlo do *master* e do interface gráfico).

Em face do número excessivo de rotinas, não as iremos enumerar aqui, remetendo-se os interessados para as explicações incluídas nos ficheiros fonte. A necessidade de sincronização e coordenação dos diversos módulos (GUI, motores, *trackball* e DSPs) originou uma aplicação bastante modular. Daí que a detecção do código responsável por uma determinada tarefa não deverá constituir uma missão impossível.

# Parte III

# Bibliografia

# Bibliografia


[ALO94] Y. Aloimonos, Z. Duric, *Estimating the Heading Direction Using Normal Flow*, International Journal of Computer Vision, vol. 13, no. 1, 33-56, 1994.

[BAL91] D. H. Ballard, *Animate vision*, Artificial Intelligence, no. 48, pp. 57-86, 1991.

[BAL92] D. H. Ballard, C. M. Brown, *Principles of animate vision*, CVGIP: Image Understanding, vol. 56, no. 1, pp. 3-21, 1992.

[BAT93] J. Batista, J. Dias, H. Araújo, A. de Almeida, *Monoplanar Camera Calibration*, British Machine Vision Conference, vol. 2, pp. 476-488, Surrey, UK, September 1993.

[BER90] J. Bergen, P. Burt, R. Hingorani, S. Peleg, *Computing Two Motions from Three Frames*, Technical Report from David Sarnoff Research Center, Subsidiary of SRI International, Princeton NJ 08543-5300, April 1990.

[BRA94] K. Bradshaw, P. McLauchlan, I. Reid, D. Murray, *Saccade and pursuit on an active head/eye platform*, Image and Vision Computing, vol.12, no. 3, April 1994.

[BRO88] C. M. Brown, D. H. Ballard, T. G. Becker, R. F. Gans, N. G. Martin, T. J. Ohlson, R. D. Potter, R. D. Rimey, D. G. Tilley, S. D. Whitehead, *The rochester robot*, Technical Report TR 257, Computer Science Department, University of Rochester, Rochester, NY, 1988.

[BRO92] C. M. Brown, D. Coombs, J. Soong, *Real-time smooth pursuit tracking*, A. Blake and A. Yuille, editors, Active Vision, chapter 8, MIT Press, Cambridge, MA, 1992.

[BUR83] P. Burt, E. Adelson, *The Laplacian Pyramid as a Compact Image Code*, IEEE Transactions on Communications, vol. 9, no. 4, pp. 532-540, 1983.

[BUR84] P. Burt, E. Adelson, *The pyramid as a structure for efficient computation*, in Multiresolution image processing and analysis, edited by A. Rosenfeld, Spring Series in Information Sciences, vol. 12, Springer, New York, 1984.

[BUR86] P. Burt, C. Anderson, J. Sinniger, G. van der Wal, *A Pipelined Pyramid Machine*, Pyramid Systems for Computer Vision, NATO ASI Series, vol. F 25, pp. 133-152, 1986.

[CAN83] J. Canny, *Finding Edges and Lines in Images*, Massachusetts Institute of Technology, AI Laboratory Technical Report 720, June 1983.

[CAN86] J. Canny, *A computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 6, pp. 679-698, 1986.

[CAR88] R.H.S. Carpenter, *Movement of the Eyes*, Pion Press, London, 1988.

[CIS89] *GEOBALL User's Manual*, CIS Graphics Inc, December 1989.

[CLA88] J. J. Clark, N. J. Ferrier, *Modal Control of an Attentive Vision System*, Proceedings of the 2nd International Conference on Computer Vision, Tampa FL, pp. 514-523, IEEE Computer Society Press, 1988.

[CLA92] J. J. Clark, N. J. Ferrier, *Attentive visual servoing*, A. Blake and A. Yuille, editors, Active Vision, chapter 9, MIT Press, Cambridge, MA, 1992.

[CLA96] J. C. Clarke, S. Carlsson, A. Zisserman, *Detecting and tracking linear features efficiently*, Proceedings of the British Machine Vision Conference, 1996.

[COL94] C. Colombo, M. Rucci, P. Dario, *Attentive behavior in an anthropomorphic robot vision system*, Robotics and Autonomous Systems, 12, pp. 121-131, 1994.

[COZ97] A. Cozzi, B. Crespi, F. Valentinotti, F. Wörgötter, *Performance of Phase-based Algorithms for Disparity Estimation*, Machine Vision and Applications, 9, pp. 334-340, 1997.

[DIA93] J. Dias, J. Batista, C. Simplício, H. Araújo, A. T. de Almeida, *Implementation of an Active Vision System*, Proceedings of the International Workshop on Mechatronical Computer Systems for Perception and Action, Halmstad University, Sweden, June 1-3, 1993.

[DIA95-1] J. Dias, C. Paredes, I. Fonseca, J. Batista, H. Araújo, A. de Almeida, *Pursuit: Experiments With Robots and Artificial Vision*, SMART - Semi-Autonomous Monitoring and Robotics Technologies, Workshop Notes, Lisbon, Portugal, April 1995.

[DIA95-2] J. Dias, C. Paredes, I. Fonseca, J. Batista, H. Araújo, A. T. de Almeida, *Simulating Pursuit with Machines - Experiments With Robots and Artificial Vision*, Proceedings of the 1995 IEEE International Conference on Robotics and Automation, Nagoya, Aichi, Japan, May 1995.

[DICE92] E. Dickmanns, *Expectation-based dynamic scene understanding*, A. Blake and A. Yuille, editors, Active Vision, pp. 303-335, MIT Press, Cambridge, MA, 1992.

[DICJ92] J. Dickmanns, *Color region tracking for vehicle guidance*, A. Blake and A. Yuille, editors, Active Vision, pp. 107-121, MIT Press, Cambridge, MA, 1992.

[EKL95] J.-O. Eklundh, K. Pahlavan, T. Uhlin, *The KTH-head-eye-system*, Vision as Process (J. L. Crowley and H. I. Christensen, editors), Basic Research Series, pp. 237-259, Springer Verlag, Berlin, 1995.

[FAI95] S. M. Fairley, I. D. Reid, D. W. Murray, *Transfer of Fixation for an Active Stereo Platform via Affine Structure Recovery*, Proceedings of the 5th International Conference on Computer Vision, 1995.

[FER93] C. Fermuller, J. Aloimonos, *The role of fixation in visual motion analysis*, International Journal of Computer Vision, vol. 11, no. 2, pp. 165-186, October 1993.

[FER92] N. J. Ferrier, *Trajectory Control of active vision systemns*, PhD thesis, Division of Applied Sciences, Harvard University, 1992.

[FLE94] D. J. Fleet, *Disparity from local weighted phase-correlation*, IEEE International Conference on SMC, pp. 48-56, San Antonio, October, 1994.

[FON94] I. Fonseca, C. Paredes, J. Dias, A. T. de Almeida, *Tracking Moving Objects with an Active Vision System and a Mobile Robot*, TELEMAN - Robotics and Remote Systems in Hazardous or Disordered Nuclear Environments, Student Research Projects Congress, Noordwijkerhout, Netherlands, June 1994.

[GRI93] W. Grimson, *Why Stereo Vision is Not Always About 3D Reconstruction*, A.I. Memo No. 1435, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, July 1993.

[HOR81] B. K. P. Horn, B. G. Schunk, *Determining optical flow*, Artificial Intelligence, vol. 17, pp. 185-203, 1981.

[HOR86] Berthold Klaus Paul Horn, *Robot Vision*, The MIT Electrical Engineering and Computer Science Series, The MIT Press, 1986.

[HUA94] L. Huang, Y. Aloimonos, *How normal flow constrains relative depth for an active observer*, Image and Vision Computing, vol. 12, no. 7, September 1994.

[KASS87] M. Kass, A. Witkin, D. Terzopoulos, *Snakes: Active contour models*, Proceedings of the 1st International Conference on Computer Vision, pp. 259-268, 1987.

[KUG75] C. Kuglin, D. Hines, *The Phase Correlation Image Alignment Method*, Proceedings of the IEEE 1975 International Conference on Cybernetics and Society, pp. 163-165, September 1975.

[LET59] J. Lettvin, H. Maturana, W. McCulloch, W. Pitts, *What the frog's eye tells the frog's brain*, in Proceedings I.R.E, vol. 47, pp. 1940-1951, 1959.

[LIN91] T. Lindeberg, *Discrete Scale-Space Theory and the Scale-Space Primal Sketch*, Ph.D. Thesis, Royal Institute of Technology, S-100 44, Stockholm, Sweden, 1991.

[LIN93] T. Lindeberg, *Discrete Derivative Approximations with Scale-Space Properties: A Basis for Low-level Feature Extraction*, Journal of Mathematical Imaging and Vision, vol. 3, no. 4, pp. 349-376, 1993.

[LUD94] K. Ludwig, H. Neumann, B. Neumann, *Local stereoscopic depth estimation*, Image and Vision Computing, vol. 12, no. 1, January/February 1994.

[MCL95] Philip F. McLauchlan, David W. Murray, *Active camera calibration for a head-eye platform using the variable state-dimension filter*, in press, PAMI, 1995.

[MEE87] P. Meer, E. Baugher, A. Rosenfeld, *Frequency Domain Analysis and Synthesis of Image Pyramid Generating Kernels*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-9, no. 4, pp. 512-522, July 1987.

[MUR93] D. Murray, P. McLauchlan, I. Reid, P. Sharkey, *Reactions to peripherial image motion using a head/eye platform*, Proceedings of $4^{th}$ International Conference on Computer Vision, pp. 403-411, 1993.

[MUR95] D. Murray, K. Bradshaw, P. McLauchlan, I. Reid, P. Sharkey, *Driving Saccade to Pursuit using Image Motion*, International Journal of Computer Vision, no. 16, pp. 205-228, 1995.

[PAH92] K. Pahlavan, J.-O. Eklundh, *A head-eye system - analysis and design*, Computer Vision, Graphics and Image Processing : Image Understanding, vol. 56, no. 1, pp. 41-56, 1992.

[PAH93] K. Pahlavan, T. Uhlin, J.-O. Eklundh, *Dynamic fixation*, Proceedings of the 4th International Conference on Computer Vision, Berlin, IEEE Computer Society Press, pp. 412-419, Los Alamitos, CA, 1993.

[PAP93] N. Papanikopoulos, P. Khosla, T. Kanade, *Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision*, IEEE Transactions on Robotics and Automation, vol. 9, no. 1, pp. 14-34, February 1993.

[PAR96] C. Paredes, J. Dias, A. de Almeida, *Recovering Shapes and Detecting Movements using Fixation*, Proceedings of the $2^{nd}$ Portuguese Conference on Automatic Control, Porto, Portugal, September 1996.

[PEA77] J. Pearson, D. Hines Jr., S. Golosman, C. Kuglin, *Video-rate Image Correlation Processor*, Application of Digital Image Processing, SPIE vol. 119, pp. 197-205, 1977.

[REE94] John Reekie, *Realtime DSP: The TMS320C30 Course*, School of Electrical Engineering, University of Technology, Sydney, Australia, February 1994.

[SAN80] G. Sandini and V. Tagliasco, *An Anthropomorphic Retina-like Structure for Scene Analysis*, CGIP, vol. 14, no. 3, pp. 365-372, 1980.

[SCH77] E. L. Schwartz, *Spatial mapping in the primate sensory projection: Analytic structure and relevance to perception.*, Biological Cybernetics, no. 25, pp. 181-194, 1977.

[SCH80] E. L. Schwartz, *A Quantitative Model of the Functional Architecture of Human Striate Cortex with Application to Visual Illusion and Cortical Texture Analysis*, Biological Cybernetics, no. 37, pp. 63-76, 1980.

[SEE96] Cahn von Seelen, Ulf M. Bajcsy, *Adaptive correlation tracking of targets with changing scale*, Ruzena GRASP Laboratory, Department of Computer and Information Science, University of Pennsylvania, June 1996.

[SHA93] P. M. Sharkey, D. W. Murray, S. Vandevelde, I. D. Reid, P. F. McLauchlan, *A modular head/eye platform for real-time reactive vision*, Mechatronics Journal, vol. 3, no. 4, pp. 517-535, 1993.

[SIN94] D. Sinclair, A. Blake, D. Murray *Robust Estimation of Egomotion from Normal Flow*, International Journal of Computer Vision, vol. 13, no. 1, pp. 57-59, 1994.

[TAA92] M. Taalebinezhaad, *Towards Autonomous Motion Vision*, AI memo 1334, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, April 1992.

[TI92] *TMS320C4x User's Guide*, Digital Signal Processing Products, Texas Instruments Inc, 1992.

[TIS93] M. Tistarelli, G. Sandini, *On the Advantages of Polar and Log-Polar Mapping for Direct Estimation of Time-to-impact from Optical Flow*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, no. 4, April 1993.

[UHL96] T. Uhlin, *Fixation and Seeing Systems*, Dissertation, CVAP Laboratory, Royal Institute of Technology, Stockholm University, May 1996.

[YAR67] A. L. Yarbus *Eye Movements and Vision*, Plenum Press, New York, 1967.