Universidade de Lisboa

Faculdade de Ciências

Departamento de Física

Monitorização automática de movimento: Caso de aplicação a modelos animais

Hugo Miguel Reis Trindade

Mestrado em Engenharia Física

2008

Universidade de Lisboa

Faculdade de Ciências

Departamento de Física

Monitorização automática de movimento: Caso de aplicação a modelos animais

Hugo Miguel Reis Trindade

Mestrado em Engenharia Física

Tese orientada pelos Prof. Doutor José Augusto e Paulo Fonseca

2008

Agradecimentos

Ao professor José Augusto por me ter dado a oportunidade de realizar este trabalho e por me ter ensinado que por mais díficeis que sejam os problemas ou os métodos a aplicar existe sempre uma solução simples. Queria agradecer ao professor Paulo Fonseca todo o apoio prestado durante este projecto. Queria também agradecer à professora Guiomar Evans a constante disponibilidade. E obrigado à Faculdade de Ciências da Universidade de Lisboa pelas condições de trabalho e pelo acolhimento prestados.

À Helena pela paciência e compreensão. À minha família por todo o apoio.

Resumo

Processamento de imagem digital; LabVIEW; Seguimento de animais; Aquisição de imagens; Filtro de Kalman.

Este trabalho apresenta um sistema de monitorização e seguimento ("tracking"), para ser utilizado tanto em imagens adquiridas em tempo real como em imagens pré-gravadas, com a finalidade de ser utilizado na investigação em comportamento animal. O sistema consiste de uma câmara de filmar, de uma placa de aquisição de imagem (*National Instruments*) e de um sistema de desenvolvimento (*LabVIEW*, complementado com o sub-sistema *NI Vision*, ambos *National Instruments*), dedicado a processamento de imagem, que inclui implementações de uma vasta gama de algoritmos apropriados para variadas tarefas de processamento de imagem.

O conjunto de aplicações foram implementados em *LabVIEW*, um ambiente visual de desenvolvimento de aplicações, com o recurso a variados algoritmos e técnicas de processamento digital de imagem e de monitorização automática.

Para a monitorização de animais é importante o registo das trajectórias, das distâncias percorridas e das respectivas velocidades instantâneas. Após variados testes, verificou-se que as aplicações de reconhecimento de um alvo e de seguimento do seu movimento baseadas apenas nos blocos de *LabVIEW-NI Vision* pré-existentes (ou pré-programados) no sistema não apresentavam a robustez necessária às aplicações pretendidas. Por esta razão houve a necessidade de desenvolver componentes de processamento específicos escritos em *M* (uma linguagem de programação disponibilizada no *LabVIEW* que apresenta muitas semelhanças ao *Matlab*) que permitiram melhorar significativamente a robustez do seguimento. O seguimento de alvos múltiplos foi também considerado.

O mais notável destes componentes consistiu na implementação de um *Filtro de Kalman* que, na sua forma mais simples, é um filtro-estimador óptimo, em sentido estatístico, apropriado para efectuar o seguimento de alvos cujo movimento é descrito por um sistema de equações determinísticas corrompidas por ruído Gaussiano e cuja observação está, também, afectada por ruído Gaussiano. O sistema foi testado para animais de dimensões e com velocidades de deslocação bastante diferentes, em imagens de vários tipos (em termos de cenários e de definição).

Abstract

Digital imaging processing; LabVIEW; Animal tracking; Image acquisition; Kalman filter.

This work presents a system for monitoring and tracking the movements of animals, which can be used both in real time and with pre-recorded images, to be used in the study of animal behavior. The system integrates a video camera, a data acquisition card (*National Instruments*) connected to a PC workstation and the *LabVIEW-NI Vision* software development system, dedicated to image processing, also from *National Instruments*, which offers a vast array of algorithms and techniques to be used in image processing.

The monitoring and tracking system was implemented in *LabVIEW*, a visual development environment, and in that system several digital image processing and automated monitoring techniques and algorithms were used.

It was necessary to record both the traveled distances and the instant velocities of the targets when doing animal tracking. It was concluded, after doing several tests with the system, that the application of monitoring and tracking based only on the pre-programmed *LabVIEW-NI Vision* blocks was not robust for the intended applications (and type of images). Thus there was the need to develop specific processing components, written in *M* (a programming language, similar to *Matlab*, present in LabVIEW) which enhanced the robustness of animal tracking. It was also tackled the tracking of multiple targets.

The main component developed in this work consists in the implementation of a *Kalman Filter*. In its simpler form it is an optimal filter/estimator, in a statistical sense, suited to be applied in the tracking of targets whose movement is described by a deterministic system of equations corrupted by Gaussian noise and where the observation of the target is also corrupted with Gaussian noise. The system was tested with animals of several dimensions and velocities and with images of several types (both in terms of scenarios and definition).

Índex

Capítulo 1

Introdução

Compreender as propriedades fundamentais dos organismos vivos passa por elucidar os mecanismos subjacentes ao seu comportamento. Esta tarefa representa um dos maiores desafios da biologia actual e comunica com outras áreas do conhecimento como a etologia, a ecologia comportamental, a neurociência, a fisiologia, a antropologia, podendo fazer parte de outras ciências como a psicologia ou a sociologia.

O comportamento animal pode ser uma janela de estudo do funcionamento de sistemas neuronais, pois a análise das capacidades cognitivas revela como diferentes espécies compreendem espaço e tempo ou reconhecem outros indivíduos. Estes estudos permitem ainda, por exemplo, uma compreensão das estratégias de acasalamento (que podem ajudar a explicar modelos evolutivos), compreender a comunicação animal ou mesmo contribuir para a definição de programas de conservação de espécies ameaçadas. Para além disso, os estudos em comportamento animal inspiram regularmente modelos computacionais e de redes neuronais utilizados em sistemas de inteligência artificial que aprendem com as suas acções.

A compreensão do comportamento animal é, assim, de grande importância para diversos ramos científicos e o auxílio das novas técnicas computacionais pode provar-se um auxílio precioso ao desenvolvimento deste campo.

1.1 Formulação do problema

O objectivo deste trabalho consiste no desenvolvimento de um sistema automático que permita monitorizar o movimento de animais. Este sistema pode ser utilizado, po exemplo, no seguimento de animais sujeitos a estímulos externos, para obter informação sobre mecanismos neuronais.

Para atingir este objectivo, propõe-se a construção de um programa de monitorização da posição de animais num espaço limitado para, através de um "back-end" de software, realizar a extracção de informação sintética e relevante sobre a sua movimentação. O sistema será testado através da realização de experiências com modelos animais.

Concretamente, pretende-se obter representações gráficas de trajectórias a duas dimensões, calcular velocidades de deslocamento médias e instantâneas e distâncias totais percorridas,

primeiro com um só animal e, depois, com dois animais a interagirem. Estas medições podem constituir uma ferramenta de grande utilidade na avaliação de interacções de corte ou agonísticas que, pela sua natureza, envolvem pelo menos dois animais.

A realização do projecto implica:

- construção de um sistema de aquisição automática de imagens, através da interligação de uma estação de trabalho, de uma placa de "hardware" de aquisição de imagem inserida no barramento PCI de um PC e de uma câmara digital, a cores, de alta resolução. Este sistema estará integrado com a ferramenta *LabVIEW* (LV) e a respectiva *Image Processing Toolbox*.
- Estudo dos algoritmos de processamento de imagem apropriados para a detecção e reconhecimento de formas.
- Estudo dos algoritmos de seguimento (*tracking*) de objectos móveis em filmes.
- Utilização privilegiada dos algoritmos disponíveis no *LabVIEW/Vision Development Module* para efectuar aquelas tarefas, e desenvolvimento de outros algoritmos, ou refinamento daqueles, caso venha a ser necessário.

1.2 Estado da arte

Tradicionalmente, o estudo do comportamento animal[1] é efectuado de forma directa, ou seja, o investigador observa o animal e anota todos os dados relevantes, com o objectivo de identificar padrões nos seus comportamentos. Este método é implementado com baixo custo e, em alguns casos, é mesmo a única forma de fazer observações em animais. Tem a desvantagem óbvia de ser exigente em recursos humanos (consome muito tempo) e de estar dependente da atenção do investigador, pois qualquer pequena distracção pode significar a perda de um evento digno de interesse. Outra desvantagem da observação directa é a dificuldade na obtenção de medições quantitativas (por exemplo, um observador não consegue medir distâncias ou velocidades de forma precisa).

Por este motivo, a observação automática é de grande utilidade científica. Nela, os erros de observação são diminuídos em grande escala, dado que o sistema não sofre de fadiga nem está sujeito a distracções e permite efectuar cálculos ou registar dados complexos em tempo real. Os sistemas automáticos permitem também estudar comportamentos que ocorrem apenas durante breves instantes e que são seguidos por grandes períodos de inactividade [1].

[1] Todas as referências a animais visam indivíduos pequenos (ratos, insectos, peixes ou aves) que podem ser estudados em ambientes controlados. Para animais de maior porte, e que se movimentam em grandes áreas, os métodos de monitorização são obviamente diferentes.

1.2.1 Evolução da monitorização automática de animais

A tecnologia para gravação automática de comportamentos animais tem evoluído drasticamente na última década. Os primeiros sistemas, baseados em electrónica analógica, eram capazes de seguir um só animal em ambientes artificiais (i.e, em áreas de teste completamente limpas de qualquer objecto na vizinhança do animal), podendo, por exemplo, a área ser examinada apenas através de uma grelha de sensores de infravermelhos (servindo como detectores únicos [2] ou combinados com uma série de strain gauges, *sensores de pressão* [3] sob a superfície), para estimar a posição do animal. Também foram utilizados métodos onde os movimentos eram medidos com o auxílio de sensores sensíveis ao toque: por exemplo, uma estimativa do movimento pode ser feita colocando o animal sobre um altifalante e monitorizando o sinal eléctrico que resulta da deslocação do animal sobre a sua superfície [4]. Outros estudos basearam-se no registo da alteração da capacitância de uma placa devida à aproximação de um animal [5], na utilização de ultra-sons [6] ou de radares Doppler de feixes de microondas [7].

O método de gravação de imagens em vídeo começou a ser largamente divulgado no princípio dos anos noventa e oferecia claras vantagens em termos de flexibilidade e precisão em relação aos anteriores. Contudo, os primeiros sistemas envolviam ainda um observador humano, que acompanhava manualmente o animal num visor com, por exemplo, um rato de computador [8]. Outro método utilizado nos primórdios consistia na detecção de picos nos sinais de vídeo analógico (picos esses que indicavam uma região de contraste entre o animal e o fundo da imagem) e a partir deles retirar as coordenadas da posição do alvo. Estes sistemas analógicos têm a desvantagem de só poderem ser utilizados numa montagem/cenário construído especificamente para a experiência em questão e de estarem limitados ao seguimento de um único animal [9].

A possibilidade da digitalização das imagens trouxe grande evolução (ou, mesmo, revolução) à monitorização animal devido ao facto de efectuarem a transposição, em tempo real, da imagem para uma grelha de alta resolução de pixels, o que permite que muitas operações de processamento possam ser efectuadas sem ser necessário proceder à gravação das imagens (esta possibilidade existe, mas os ficheiros resultantes rapidamente se tornam demasiado grandes e, por isso, a sua utilização é limitada e evitada em observações de longa duração). Caso se pretenda efectuar o processamento com um ritmo de imagens elevado, a complexidade (em termos de gasto de tempo) das operações envolvidas terá de ser limitada.

Em conclusão, pode afirmar-se que a disponibilidade de sistemas de processamento de imagem muito poderosos e de custo razoável permite a automatização das tarefas inerentes à monitorização do comportamento de animais. A rapidez de processamento de um PC comum, associada à funcionalidade inerente às placas de aquisição e processamento de imagem utilizadas hoje em dia, permite aplicar algoritmos complexos aos dados (imagens) adquiridos e deles extrair informação em quantidade e qualidade.

A utilização de software de monitorização de movimentos tem, por isso, aumentado, havendo disponíveis no mercado bastantes programas específicos para o efeito. Entre os

programas/companhias existentes destacam-se a *Noldus Ethovision*, a *ANY maze*, a *Med Associates Inc.* e a *Columbus Instruments*.

Vários estudos têm sido realizados com o objectivo de aperfeiçoar sistemas que possam seguir o comportamento de animais com maior rigor, ou em condições mais próximas das reais (do respectivo *habitat natural*). A aplicação biológica destes sistemas de monitorização está constantemente a alargar-se.

É importante referir também os métodos e técnicas de reconhecimento visual. Conceber sistemas que consigam reconhecer objectos definidos quando imersos em imagens com elevado grau de complexidade (isto é, sistemas que apresentem semelhança com o sistema olho/cérebro humano) é um problema clássico na área da Ciência dos Computadores, mais especificamente da Inteligência Artificial e da Aprendizagem Automática.

Nos sistemas de reconhecimento visual o objectivo é o de identificar um amplo número de objectos, com dimensões variáveis, e aqueles sistemas devem ser suficientemente robustos para contornar dificuldades como a oclusão do objecto ou a existência de alterações no fundo da imagem e ser receptivos ao aumento do número de objectos com resposta temporal constante. No caso da monitorização animal, o interesse estará em reconhecer os animais movimentando-se em ambientes complexos e/ou em reconhecer simultaneamente diversos animais individualmente.

Os primeiros estudos de reconhecimento visual envolviam a extracção das arestas de um objecto numa imagem e sua comparação com as arestas de todos os objectos registados numa base de dados. Para acelerar e melhorar o processo foram utilizados vários métodos matemáticos, em particular o *método de Newton*, em que as equações de projecção de um modelo tridimensional numa imagem bidimensional são por ele resolvidas assumindo como solução inicial as arestas do objecto a reconhecer [10].

Os problemas inerentes ao reconhecimento visual (rotação, translacção, mudanças de perspectiva e oclusões) podem ser ultrapassados com o auxílio do método de *geometric hashing*. Esta técnica permite reconhecer objectos comparando-os com aqueles guardados numa base de dados, mas é muito mais rápida que a baseada no método de Newton, pois o acesso à memória é baseado em informação geométrica que é invariante à posição do objecto. São retirados dos objectos as arestas, cantos ou curvas que são então representados como pontos num sistema de coordenadas. São aleatoriamente extraídos conjuntos de pontos do objecto na imagem e são comparados com os pontos retirados do modelo. O objecto é aceite como válido (reconhecimento positivo) se tiver um número suficientemente grande de sucessos [12].

Os avanços tecnológicos permitiram a aquisição e o processamento de imagens a cores. Relativamente a uma imagem colorida, um histograma de cores indica quantos valores de determinada cor existem na imagem, o que significa que o histograma é um elemento sintético invariante para translações, rotações, oclusões e mudanças de direcção de um objecto, ou seja, invariante a quaisquer alterações na imagem que preservem (aproximadamente) o seu conteúdo de cor. Os sistemas de reconhecimento por comparação de histogramas relacionam o histograma de cor do alvo com os histogramas de outros objectos guardados em bases de

dados [11].

Os sistemas de reconhecimento visual têm evoluído muito nos últimos anos, sobretudo na área do reconhecimento de faces humanas (reconhecimento de pessoas ou de emoções) [13],[14],[15]. Estes sistemas são bastante complexos e baseados em técnicas de processamento específicas para o objecto "face humana".

Existem também métodos de reconhecimento baseados no movimento, ou seja, é possível fazer a distinção entre objectos analisando as suas trajectórias. Estes são os sistemas mais utilizados em monitorização animal, devido ao facto de os animais terem poucas características que os permitam distinguir entre si e terem, muitas vezes, um comportamento considerado errático.

1.2.2 *Hidden Markov Models*

Sendo a detecção de objectos em imagens sequenciais uma tarefa modelada por um processo estocástico, grande parte dos trabalhos feitos nesta área envolvem o desenvolvimento de algoritmos com base na *propriedade de Markov*, onde se admite que o futuro dos processos em questão depende somente do estado presente e não dos estados passados, ou seja, o estado actual determina a distribuição de probabilidade do próximo estado. Em monitorização, o modelo estatístico mais utilizado é o denominado *Hidden Markov Model* (HMM) assim denominado porque as propriedades dos estados não são directamente observáveis, mas algumas variáveis influenciadas pelos estados são [16]. Mais especificamente, em monitorização automática podemos conhecer a posição de vários animais num determinado tempo t, mas desconhecemos a qual dos animais corresponde essa posição, sendo então a identidade dos animais o estado desconhecido.

Um dos métodos utilizados para estimar as propriedades de variáveis escondidas, a partir das observações num HMM, tem por base o algoritmo de *Monte Carlo* (Markov Chain Monte Carlo) [17], [18]. Este método é utilizado sobretudo no desenvolvimento dos sistemas de *Inteligência Artificial* (IA) (um aspecto menos evidente da importância da compreensão dos comportamentos sociais dos animais é a influência que tem tido no desenvolvimento de sistemas de IA). Esses estudos implicam muitas vezes a monitorização de múltiplos animais em simultâneo, como no caso das formigas (que apresentam como dificuldade o facto de serem animais muito pequenos, rápidos e praticamente idênticos entre si e de viverem em grandes colónias)[21] ou dos macacos (pela sua semelhança com o ser humano) [22].

Outro método utilizado em conjunção com HMMs é o *BraMBLe* (Bayesian Multiple Blob), um método estatístico que segmenta as partículas de uma imagem e reconhece o movimento de cada objecto com base em correlações Bayesianas das elipsóides gaussianas que melhor se adaptam ao objecto [19]. O *BraMBLe* é utilizado, por exemplo, na monitorização de roedores, um alvo de observação importante que tem diversas aplicações nos estudos dos efeitos de drogas em determinadas aplicações terapêuticas ou no aparecimento de mutações genéticas [20].

1.2.3 Uso do *Filtro de Kalman*

Um dos algoritmos mais utilizado no reconhecimento de trajectórias é o denominado *Filtro de Kalman* (FK), que pode ser considerado como operando sobre um análogo do HMM com variáveis de estado contínuas eventualmente não vísiveis (embora os sistemas contínuos sejam, na prática, discretizados, como é o caso da observação de imagens separadas entre si no tempo). O algoritmo correspondente ao Filtro de Kalman permite incorporar os dados já existentes e a actual observação do sistema e dar a melhor estimativa (num sentido estatístico de minimização da variância) do estado do sistema (por exemplo, a posição e a velocidade do objecto), desde que o sistema seja linear e o ruído que corrompe o sistema e as observações (ou medidas) seja do tipo Gaussiano. O Filtro de Kalman baseia-se na utilização da fórmula de Bayes para fazer previsões sobre o estado do sistema tendo em conta a observação actual e as passadas.

No quadro do seguimento de animais (conforme foi implementado neste trabalho), o filtro estima a localização de um objecto e mede a confiança desta localização, permitindo que a janela de busca do objecto no quadro seguinte seja colocada numa posição baseada nas posições obtidas nos quadros anteriores, melhorando o processo de monitorização [23]. O FK é utilizado na monitorização de ratos em túneis ou em ambientes escuros utilizando câmaras de infravermelhos (o que não poderia ser feito normalmente por observadores humanos) aumentando assim a gama de utilizações possíveis para os sistemas automáticos de monitorização [24]. Foi também utilizado para extrair informação relevante em filmagens aquáticas, após ser subtraído o fundo aquático e serem corrigidas as refracções provocadas pela água [25].

O Filtro de Kalman tem inúmeras aplicações, para além do reconhecimento de objectos, nas áreas de Processamento de Sinais, Controlo, Telecomunicações, Indústria Aeroespacial e Aviónica (seguimento de objectos voadores), Economia (análise e previsão com séries temporais de indicadores económicos), e outras.

1.2.4 Aplicação de redes neuronais

O problema da monitorização em tempo real, com reconhecimento de objectos móveis complexos, tem também sido resolvido com recurso à implementação de *Redes Neuronais Artificiais*. Uma rede neuronal é constituída por neurónios artificiais que aceitam vários sinais de entrada e geram um sinal de saída que depende da "função de ganho" (que é habitualmente não linear) associada a cada neurónio. Os parâmetros (ou "pesos") associados àquela função representam a memória do sistema. Uma rede neuronal aprende (isto é, aqueles "pesos" são definidos) através de um algoritmo de treino, que depende da aplicação da rede, e vai fazer com que os "pesos" de cada neurónio se ajustem ao problema que a rede visa resolver. Em reconhecimento visual, os modelos de redes neuronais são habitualmente combinados com diversos algoritmos de processamento de imagem que segmentam o objecto [26], [27], [28]. A fase de treino da rede é habitualmente demorada neste tipo de aplicações.

1.2.5 Recurso ao LabVIEW

Sistemas baseados no LabVIEW [38] começam a ser cada vez mais utilizados em monitorização automática de animais. O LabVIEW é, por exemplo, utilizado em estudos celulares, em análise dos agregados proteicos e de vírus [29] e em estudos de animais de maior porte tais como ratos [24].

Existem diversos conceitos e metodologias relacionados com o processamento de imagem digital. Todas as noções estão desenvolvidas no livro [30]. Os algoritmos existentes e a sua aplicação estão explicados em [31] e em [32].

O sistema de desenvolvimento utilizado neste trabalho, consistindo no *LabVIEW*, no *IMAQ* e no *NI Vision*, são bem descritos na documentação da *National Instruments* (a companhia que os vende), quer nos livros fornecidos com aqueles produtos, quer através da informação que se encontra numa boa quantidade de livros "independentes". Por exemplo, [33] é um livro dedicado especificamente ao processamento digital com o LabVIEW ([34] é também um bom auxiliar sobre este tema) e [35] é uma tese de mestrado que descreve, e analisa, a construção de Redes Neuronais Artificiais com programação em LV. Para a programação em LV [36] é um grande auxílio para principiantes na linguagem. Aspectos mais avançados estão contemplados em [37].

No site da National Instruments [38] está acessível diversa documentação adicional e existe, também, um fórum de discussão muito útil, ao qual recorremos várias vezes.

1.2.6 Dificuldades Associadas ao Problema

Um sistema de monitorização automática tem sempre dificuldades adjacentes. Se, por um lado, a gravação em filme dos acontecimentos permite a utilização de algoritmos mais complexos num processamento "off-line", por outro os filmes rapidamente podem atingir dimensões demasiado grandes para a capacidade dos computadores. A análise das imagens em tempo real implica utilizar sistemas algoritmicamente bastante mais simples que os utilizados no processamento "off-line" e que, por isso, mais facilmente originam erros. Existe também a dificuldade da pouca rapidez no processamento de imagens, pois um fenómeno de interesse para registo pode acontecer entre dois quadros consecutivos se o sistema demorar demasiado tempo a processar as imagens captadas (e se só capta a próxima após ter processado a actual).

O problema inerente a todos os programas de monitorização de objectos deve-se à distinção que tem de ser feita entre o objecto sob análise e o fundo em que se move e, caso existam vários objectos, distinguir também uns dos outros. Se os alvos a seguir forem semelhantes (no caso de animais são praticamente iguais) a dificuldade aumenta ainda mais. Existem outros problemas, tais como a oclusão dos objectos (pelo fundo ou por outro objecto), o aumento do número de objectos a seguir (e.g: animais que se reproduzem, ou que entram e saiem de tocas) ou alterações na iluminação (que podem provocar sombras e induzir o sistema em erro).

Capítulo 2

Processamento Digital de Imagem

2.1 Introdução

O Processamento Digital de Imagem (PDI) é uma área que agrupa um conjunto de técnicas e de tecnologias com imensas aplicações práticas no mundo de hoje. A partir do momento em que foi possível digitalizar os sinais analógicos de imagem (de filmes) em tempo real, armazená-los em computador, e devolver os sinais, já processados, também em tempo real, foi aberto o caminho para a televisão e cinemas digitais tão comuns na actualidade: esta aplicação é, porventura, das mais avançadas a utilizar os conceitos de PDI.

O conceito de imagem é generalizado em PDI. Por isso, dedica-se uma boa parte deste capítulo à definição de imagem e à introdução de uma série de conceitos a ela associados, a maioria dos quais são posteriormente aplicados no presente trabalho.

De seguida, apresentamos e discutimos técnicas de PDI aplicáveis a fins específicos: remoção de ruído, detecção de contornos, melhoria do contraste, por exemplo.

2.2 Imagem digital

O termo imagem, em geral, está associado a uma função bidimensional cujos valores representam intensidades luminosas. Pode-se definir como uma função $f(x, y)$ onde o valor da amplitude de f nas coordenadas espaciais (x, y) indica a intensidade (ou brilho) da imagem naquele ponto. Sendo a luz uma forma de energia, $f(x, y)$ tem de ser positiva e finita em todo o domínio da imagem, ou seja:

$$0 < f(x, y) < \infty \tag{2.1}$$

Em geral, x está associado à coordenada horizontal e y à coordenada vertical da imagem. Note-se que a anterior definição se aplica quer a imagens analógicas quer a imagens digitais.

Uma imagem digital é definida por uma função de valores discretos, sendo que x e y representam agora as coordenadas discretas de um elemento da imagem – isto é, um *pixel* ('picture element'). Uma imagem digital é pois um sinal discreto.

2.2.1 O Pixel

Um *pixel*[1] é o mais pequeno elemento individual de uma imagem digital. Ao ser criada uma imagem digital atribui-se a cada pixel um nível de cinza[2] que especifica o seu brilho ou um nível de cor que especifica a sua cor (Fig. 2.1).

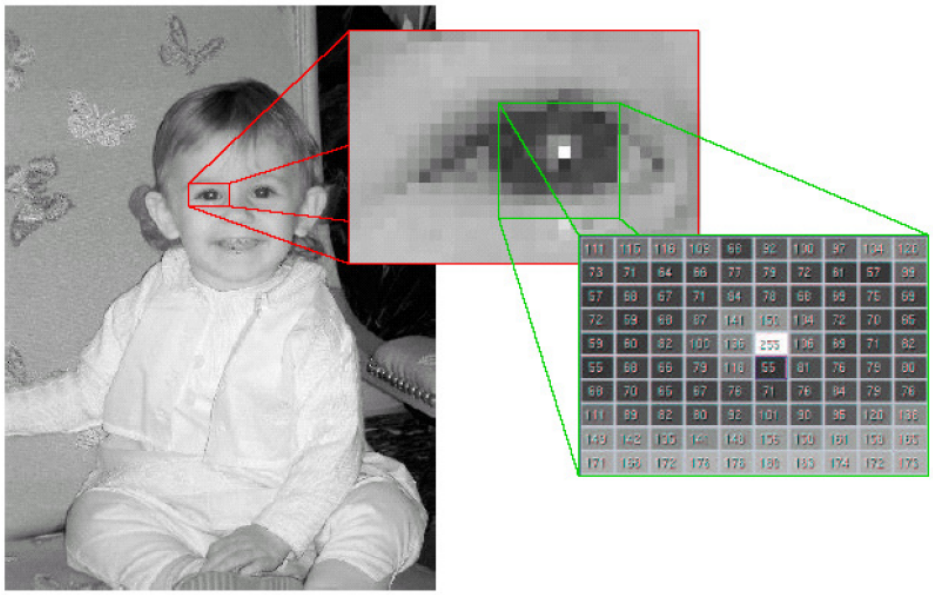


Figura 2.1: Ilustração do significado do pixel de uma imagem a preto e branco (ou com níveis de cinzento) [41].

Por convenção generalizada em processamento de imagens, as coordenadas $(0, 0)$ referem-se ao pixel localizado no topo superior esquerdo da imagem. O valor de x aumenta da esquerda para a direita e o de y de cima para baixo (Fig. 2.2).

Na medida em que uma imagem digital pode ser encarada como um vector de dados, podem ser executadas diversas operações considerando a imagem como uma *matriz* de dimensões $M \times N$ e cada pixel como um elemento dessa matriz, o que se representa de seguida.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & ... & f(0,N-1) \\ f(1,0) & f(1,1) & ... & f(1,N-1) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ f(M-1,0) & f(M-1,1) & ... & f(M-1,N-1) \end{bmatrix} \tag{2.2}$$

As operações algébricas habituais (soma, subtracção, multiplicação, divisão, logaritmo, etc...) podem ser realizadas pixel a pixel, e podem servir, por exemplo, para somar duas imagens. Também podem ser realizadas operações geométricas globais sobre a imagem,

[1]Há alguma controvérsia sobre a tradução (e escrita) de 'pixel' e do seu plural em Português. Grande parte das opiniões vai no sentido de manter as palavras em Inglês, ou seja, escrever 'pixel' e 'pixels': é esta a opção que vai ser seguida aqui.

[2]Os vários tipos de imagem, entre os quais a imagem cinzenta, serão definidos mais adiante.

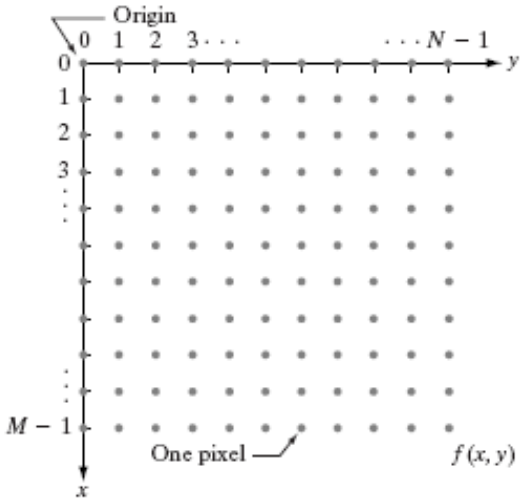


Figura 2.2: Representação da distribuição dos pixels de uma imagem [30].

tais como a translação e/ou a rotação, aplicar transformadas ou transformada inversa, etc... Estas operações não são aplicadas pixel a pixel, mas sim a um conjunto de pixels.

Conectividade entre pixels

A *conectividade* entre pixels é um importante conceito usado para estabelecer fronteiras entre objectos numa imagem, ou componentes de uma imagem. Nessa medida, é um conceito importante neste trabalho. Para a definir é necessária a noção de *vizinhança*, que vamos passar a expôr.

Cada pixel, p, no ponto (x, y), tem quatro vizinhos horizontais e verticais cujas coordenadas são

$$(x+1, y),\ (x-1, y),\ (x, y+1) \text{ e } (x, y-1).$$

Este conjunto de vizinhos é denotado por $\mathrm{N}_4(p)$. O conjunto dos quatro vizinhos diagonais, de coordenadas

$$(x+1, y+1), (x+1, y-1), (x-1, y+1) \text{ e } (x-1, y-1)$$

é, por sua vez, denotado por $\mathrm{N}_D(p)$. Estes oito pontos vizinhos de p são referidos globalmente por $\mathrm{N}_8(p)$. Note que alguns dos vizinhos de p estão fora da imagem digital (ou, mesmo, não existem) se (x, y) for uma coordenada situada na sua borda (ou fronteira).

Dois pixels são conectados[3] se forem adjacentes (por exemplo, se forem vizinhos um do outro segundo a definição $\mathrm{N}_4(p)$) e se os seus valores satisfizerem um determinado critério. Numa imagem binária, em que os pixels têm apenas valor 0 ou 1, dois pixels podem ser vizinhos mas não são considerados conectados se não tiverem o mesmo valor.

[3]Optou-se por este termo em vez de 'ligados', que também seria um termo legítimo.

Seja $\mathcal{V}$ um conjunto de níveis de cinza utilizado para definir a conectividade numa imagem cinzenta (aqui usada como exemplo):

- *conectividade-4*: dois pixels p e q, com valores de intensidade dentro de $\mathcal{V}$, têm conectividade-4 se q pertencer a $\mathrm{N}_4(p)$;
- *conectividade-8*: dois pixels p e q, com valores dentro de $\mathcal{V}$, têm conectividade-8 se q pertencer a $\mathrm{N}_8(p)$.

Seja $\mathcal{R}$ um subconjunto de pixels numa imagem. Caso $\mathcal{R}$ seja um conjunto conectado, segundo um determinado critério, é denominado de *região* de uma imagem.

Um *caminho (digital)* (ou *curva*) definido desde o pixel p, de coordenadas (x, y), até ao pixel q, de coordenadas (s, t), é uma sequência de pixels distintos de coordenadas

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$$

onde $(x_0, y_0) = (x, y)$, $(x_n, y_n) = (s, t)$ e os pixels (x_i, y_i) e (x_{i-1}, y_{i-1}) são conectados para $1 \leq i \leq n$. Neste caso, n é denominado de *comprimento* do caminho.

2.2.2 Propriedades de uma imagem digital

Profundidade de pixel

A *profundidade de pixel* de uma imagem é o número de bits usados para codificar o valor de intensidade associado a um pixel. Uma profundidade de pixel de valor n significa que o pixel pode ter 2^n valores diferentes. Por exemplo, se $n = 8$ bits, o pixel poderá apresentar 256 valores diferentes de intensidade que vão de 0 até 255.

Resolução

A resolução é uma medida do grau de detalhe discernível numa imagem e depende do número de pixels que a formam, e da sua profundidade. Quanto mais estes parâmetros forem aumentados, mais próximos estaremos da imagem original e, logo, melhor será a resolução da imagem. Convenciona-se que uma imagem composta por M linhas e N colunas tem uma resolução dada por $M \times N$.

Tipos de imagem

O tipo de imagem depende da natureza da informação associada aos seus pixels, $u_{i,j}$. Para $0 \leq i < N$ e $0 \leq j < M$ existem as seguintes possibilidades:

- imagem *binária:* $u_{i,j} \in \{0, 1\}$ (e.g. uma imagem a preto e branco, com os pixels pretos a valer 0 e os brancos a valer 1);

- imagem *inteira:* $u_{i,j} \in \{0, \ldots, 2^{N_b} - 1\}$ (e.g. uma imagem cinzenta de N_b =8 bits, com os pixels a poder tomar 256 valores diferentes);
- imagem *real:* $u_{i,j} \in \mathcal{R}$ (e.g., uma imagem analógica descrita pela função $f(x, y)$);
- imagem *complexa:* $u_{i,j} \in \mathcal{C}$ (e.g. a DFT bidimensional de uma imagem calculada, eventualmente, com a FFT[4]).

Representação de imagens a cores

Existem diversos modelos de representação de imagens a cores. Os mais utilizados são o RGB ("Red-Green-Blue") que é encontrado em equipamentos de visualização de imagens electrónicos, o CMY ("Cyan-Magenta-Yellow") que é muito utilizado nas impressões a cores e o HSI ("Hue-Saturation-Intensity") que tenta modelar, de uma forma fidedigna, o processo pelo qual o (o olho do) ser humano descreve e interpreta a cor.

O número de planos de uma imagem corresponde ao número de matrizes de pixels que compõem a imagem. Uma imagem em escala de cinzentos é composta apenas por um plano. Uma imagem a cores é composta por três planos, cada um deles contendo a informação sobre uma cor, que são combinados na altura da visualização da imagem (Fig. 2.3.)

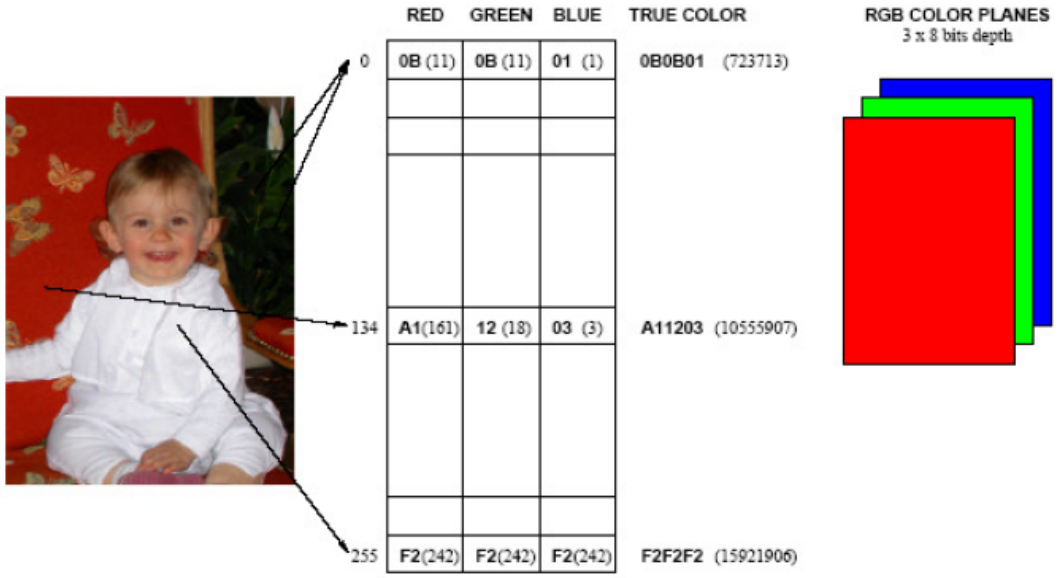


Figura 2.3: Exemplo que ilustra a combinação dos três planos de cor [41].

Formatos de armazenamento de imagem

O armazenamento de imagens aborda a respectiva gravação, leitura e compressão. Existem vários formatos de armazenamento de imagem de uso generalizado, muitos deles su-

[4]DFT e FFT são as siglas de "Discrete Fourier Transform" e "Fast Fourier Transform", respectivamente. A primeira é "a transformada" e a última é, na realidade, um algoritmo eficiente para calcular a DFT. Ambas serão discutidas mais adiante, neste capítulo.

portados por normas internacionais.

Um ficheiro de imagem é habitualmente composto por um cabeçalho, seguido pelos valores dos pixels. Dependendo do formato do ficheiro, o cabeçalho contém informação sobre a resolução vertical e horizontal, a profundidade de pixel e eventualmente sobre a imagem original (se a que está no ficheiro tiver sido resultante do processamento da original).

Os formatos de uso mais comum são o BMP (este formato é usado pelos sistemas operativos Microsoft Windows e não comprime as imagens), o TIFF, o GIF, o PNG e o JPEG (o último é talvez o que apresenta a melhor relação entre capacidade de compressão e perda de informação da imagem original). Além destes formatos ainda há muitos que não iremos referir, mas [43] apresenta uma explicação detalhada sobre grande parte dos formatos existentes actualmente e os códigos fonte com que eles são implementados.

2.3 Fundamentos de Processamento Digital de Imagem

O PDI é feito recorrendo a uma grande variedade de técnicas e de algoritmos, e nesta secção vamos fazer uma revisão dos mais importantes para a sua (do PDI) compreensão, em sentido lato.

O primeiro passo no processo de processamento digital de imagem é a aquisição, que requer um sensor e um sistema com a capacidade de digitalizar o sinal que aquele produz (habitualmente o sistema é implementado com a electrónica apropriada). Após a obtenção da imagem, o passo seguinte consiste no seu pré-processamento, que serve para melhorar a imagem de modo a que os passos seguintes na cadeia de processamento obtenham sucesso. Tarefas comuns nesta fase são a remoção de ruído, a melhoria do contraste, etc.

A segmentação da imagem é a tarefa em que se tenta isolar (e reconhecer) na imagem alguns dos seus constituintes, e é o principal objectivo deste trabalho. Os algoritmos utilizados para a segmentação dependem da informação que se quer extrair da imagem e o processo deve ser interrompido a partir do momento em que o objecto de interesse é isolado. Seguidamente é necessário reconhecer o objecto segmentado (validando o procedimento) e retirar a informação pretendida.

2.3.1 Histograma e melhoramento de contraste

Um histograma representa matematicamente (por um vector), ou apresenta graficamente, o número de pixels que tomam cada valor (ou intervalo de valores) possível na representação da imagem. É uma ferramenta sintética de análise fundamental, pois descreve a distribuição dos valores dos pixels na imagem. O histograma de uma imagem com níveis de cinzento, onde os pixels tomam valores no intervalo $[0, \ell-1]$ é definido pela função discreta $H(k) = n_k$, onde k é um valor da escala de cinzentos, n_k é o número de pixels na imagem com valor igual a k e $\sum_{k=0}^{\ell-1} n_k = M \times N$ é o número de total de pixels na imagem (Fig. 2.4).

Um histograma normalizado consiste da função $p(r_k) = n_k/(M \times N)$, em que $p(r_k)$ dá a probabilidade de ocorrência do valor r_k.

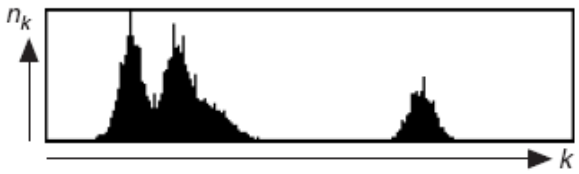


Figura 2.4: Exemplo de histograma de imagem.

Um histograma global de uma imagem a cores é subdividido em três histogramas, um por cada uma das três cores que definem a imagem (Fig. 2.5).

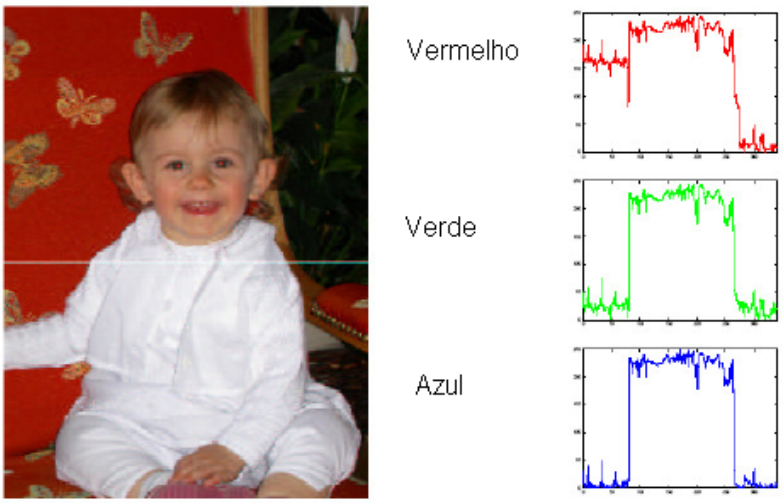


Figura 2.5: Histogramas dos pixels $u_{240,j}$ de uma imagem a cores. Imagem retirada de [41]

Ajuste de contraste

A selecção de um conjunto de valores específicos de níveis de cinzento é, muitas vezes, desejável para fazer sobressair na imagem original determinados detalhes. Isso pode ser alcançado através da escolha de um determinado intervalo de valores e "apagando" (ou "reduzindo" em valor) todos os pixels que não tenham esse valor, ou, pelo contrário, aumentando o valor dos pixels escolhidos (Fig. 2.6). Em todo o caso, este processamento envolve uma alteração da imagem feita a partir da análise do histograma original. O histograma da imagem modificada será, obviamente, diferente.

Uniformização de contraste

A uniformização do contraste de uma imagem é feita através da manipulação da função de probabilidade $p(r_k)$ do histograma. Se a variável r representar os níveis de cinzento de uma imagem a transformação

$$T(r) = \int_0^r p_r(w)dw \tag{2.3}$$

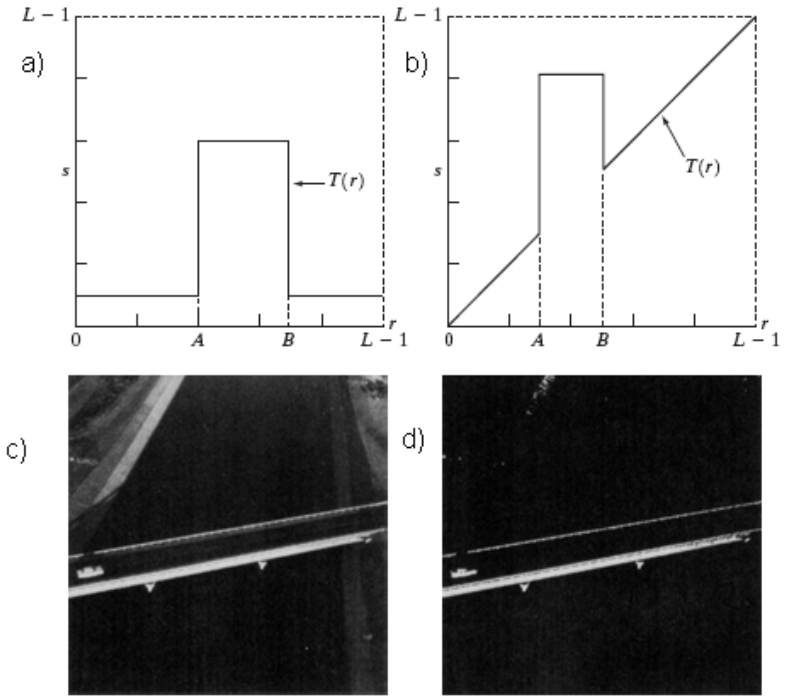


Figura 2.6: a) Esta transformação realça os valores situados no intervalo $[A, B]$ de cinzento e reduz todos os outros a um nível constante. b) Esta transformação realça os valores situados no intervalo $[A, B]$ mas preserva os restantes valores. c) Imagem exemplo. d) Resultado da transformação de (c) segundo (a). Exemplo retirado de [30].

permite uniformizar os níveis de cinzento da imagem (Fig. 2.7).

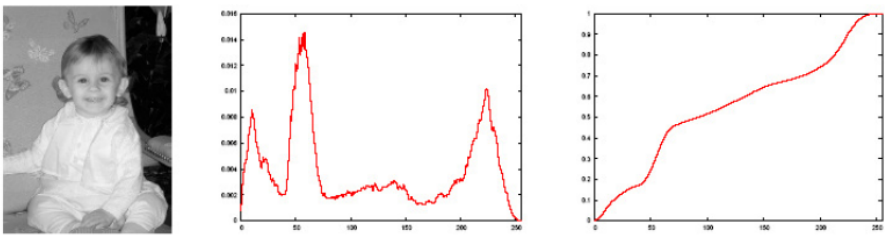


Figura 2.7: Exemplo de uniformização de contraste. Ao centro está o histograma da imagem original. À direita está o histograma resultante da uniformização [41].

Esta transformação também pode ser aplicada a sub-áreas quadradas contendo alguns pixels da imagem (7×7, 5×5, etc...) isoladamente, o que pode auxiliar a melhorar os detalhes em pequenas áreas.

2.3.2 Filtragem espacial

O conceito de filtragem espacial refere-se a outro tipo de processamento de imagem. O valor dos pixels é alterado não em função de características globais da imagem, mas em função dos valores dos pixels pertencentes a uma dada vizinhança do pixel em processamento.

A filtragem espacial consiste em mover um filtro (também chamado *kernel* ou máscara), pixel a pixel, por toda a imagem, criando-se uma nova imagem que é a versão "filtrada" da original. Calcula-se um valor para o pixel situado no "centro" do filtro a partir dos valores que este e que os pixels vizinhos possuem na imagem original. Assim, o *kernel* é uma função bidimensional que, quando aplicada a uma imagem em cinzentos (por exemplo), vai atribuir a um pixel um valor que é uma combinação (linear ou não) do seu valor e dos valores originais de alguns vizinhos. Os coeficientes do *kernel* definem a contribuição de cada vizinho, e a dimensão do *kernel* define o número de vizinhos que influenciam a filtragem. No caso de um *kernel* de 3×3, ilustrado na Fig. 2.8 a), o valor do pixel central (a negro) é obtido a partir do valor dos seus oito vizinhos. Na Fig. 2.8 b) mostra-se um *kernel* de 5×5 com 24 vizinhos. À semelhança do que acontece em processamento unidimensional, podem-se usar funções não lineares na realização dos filtros. Em geral, devido à natureza intrínseca das imagens, estes filtros são não causais (i.e., para processar o pixel (i, j) pode-se utilizar os valores dos pixels "avançados" na linha $i + 1$, na coluna $j + 1$, etc...

É preciso ter em atenção que, quanto maior for a máscara usada no filtro, mais tempo será gasto na filtragem. É esta a razão de as máscaras mais populares serem as mais pequenas, principalmente nas aplicações realizadas em tempo real.

A operação de filtragem envolve mover o *kernel* sobre todos os pixels (habitualmente procedendo-se desde o topo superior esquerdo até à base direita da imagem, isto é, incrementando os índices dos pixels desde $(0, 0)$ até (M, N)). A cada pixel "submetido" ao *kernel* é dado um novo valor, como se ilustra na Fig. 2.9.

Devido ao facto de, ao ser aplicado o *kernel* aos pixels nas fronteiras da imagem, parte da máscara se encontrar fora da imagem e poder diminuir a exactidão da filtragem, a aplicação deve ser feita só até aos pixels que se encontrem à distância $(n-1)/2$ do limite, onde n é a dimensão do *kernel*. Assim, a imagem filtrada será menor que a original.

O processo de filtragem é habitualmente dividido em dois tipos: linear (ou de convolução) e não-linear. A operação de convolução é definida (supondo uma máscara W):

$$v_{i,j} = \sum\sum_{k,\ell \in W} a_{k,\ell} u_{i-k,j-\ell} \qquad i,j \text{ na imagem} \tag{2.4}$$

onde $v_{i,j}$ é o novo valor atribuído ao pixel (i,j), $a_{k,\ell}$ é a matriz associada à máscara e $u_{i,j}$ é o valor do pixel (i,j) na imagem original.

Filtragem passa-baixo

A ideia por detrás da filtragem por média é reduzir as transições abruptas entre níveis de cinzento (ou de cores, se for aplicada aos três planos RGB de uma imagem a cores), substituindo o valor de cada pixel pela média dos valores dos vizinhos e do seu. Como o ruído é, tipicamente, a causa de muitas destas transições, a redução do ruído é a aplicação bandeira deste filtro. Contudo, as fronteiras reais entre objectos na imagem também vão ser atenuadas, diminuindo-se assim o seu contraste e podendo verificar-se o borramento da imagem (*blurring*). Alguns exemplos de máscaras de média encontram-se na Fig. 2.10.

Por exemplo, a máscara 3×3, de 5 pontos, à direita na Fig. 2.10, é correspondente à convolução:

$$v_{i,j} = \frac{1}{3} u_{i,j} + \frac{1}{6}(u_{i,j-1} + u_{i,j+1} + u_{i-1,j} + u_{i+1,j})$$

Filtragem passa-alto

Este tipo de filtro tem como principal aplicação o aumento do contraste da imagem, reforçando o respectivo conteúdo em altas frequências pela adição do gradiente da imagem

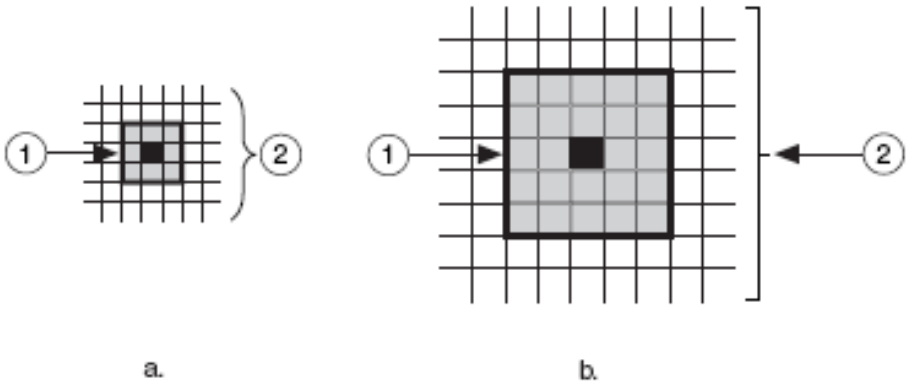


Figura 2.8: Exemplos de *kernels*.

Figura 2.9: Ilustração da filtragem com um *kernel*.

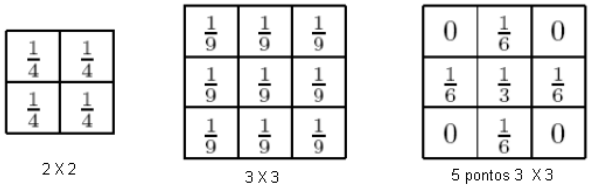


Figura 2.10: Máscaras clássicas de filtragem por média.

à imagem original. O ruído pode ser também reforçado por este filtro devido à sua natureza pontual e descontínua.

0	$-\frac{1}{4}$	0
$-\frac{1}{4}$	1	$-\frac{1}{4}$
0	$-\frac{1}{4}$	0

Figura 2.11: Máscara adequada para filtragem passa-alto.

A máscara mostrada na Fig. 2.11 é definida como:

$$v_{i,j} = u_{i,j} - \frac{1}{4}(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1})$$

Filtragem passa-banda

A aplicação simultânea dos filtros passa-alto e passa-baixo, com coeficientes apropriados, permite realizar filtros passa-banda, que são úteis para reforçar contornos na presença de ruído.

Filtragem baseada na mediana

O filtro por mediana[5] consiste em substituir o valor do pixel pela mediana dos valores dos pixels na vizinhança W correspondente à janela envolvente, isto é

$$v_{i,j} = \text{mediana}(u_{i-k,j-\ell}), \qquad k, \ell \in W. \tag{2.5}$$

[5]Este é um exemplo de filtro não-linear.

Para aplicar este filtro é necessário ordenar os pixels na janela por valor crescente, e escolher o valor central (para um número ímpar de pixels) ou a média dos dois valores centrais (para um número par). Este filtro preserva a resolução e apresenta bons resultados em ruído dos tipos impulso ou binário se o número de pixels com ruído (significativo) for menor do que metade do número de pixels na janela. Existem algoritmos eficientes que implementam o filtro da mediana apenas com $N_W log_2 N_W$ comparações, onde N_W é o número de pixels na janela. Existe também a possibilidade de realizar a filtragem por mediana em linhas ou em colunas separadamente. Este processo não é equivalente ao filtro da mediana a duas dimensões mas é mais rápido ("custa" menos comparações). O filtro de mediana tem características passa-baixo.

2.3.3 Detecção de arestas

Uma aresta é uma fronteira entre duas regiões com níveis de cinzento[6] diferentes. Basicamente, a ideia por detrás da detecção de arestas é considerar a imagem como uma superfície contínua $f(x, y)$ e calcular as suas derivadas direccionais. A Fig. 2.12 permite compreender melhor o conceito: nesta figura, a primeira derivada é positiva nos pontos de transição e zero nas áreas de nível de cinzento constante; a segunda derivada é positiva na transição associada à parte negra da figura e negativa na clara, sendo zero em toda a restante imagem. Podemos concluir, por isso, que a magnitude da primeira derivada pode ser usada para detectar uma aresta na imagem e a magnitude da segunda para determinar se um pixel de fronteira está do lado claro ou escuro da aresta. Vai-se discutir os operadores que actuam na detecção de arestas.

Operadores de gradiente

Estes operadores servem, em geral, para melhorar o contraste de uma imagem.
A definição de gradiente é dada por:

$$\nabla f = (G_x, G_y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \tag{2.6}$$

A implementação das primeiras derivadas em processamento de imagem é feita normalmente pela magnitude do gradiente

$$mag(\nabla f) = \left[{G_x}^2 + {G_y}^2 \right]^{1/2}. \tag{2.7}$$

Contudo, para promover a rapidez do algoritmo de cálculo, aproxima-se frequentemente a magnitude do gradiente pela soma dos valores absolutos das partes real e imaginária:

[6]Para efeitos ilustrativos considera-se imagens em cinzento, mas a detecção de arestas é aplicável a qualquer tipo de imagem.

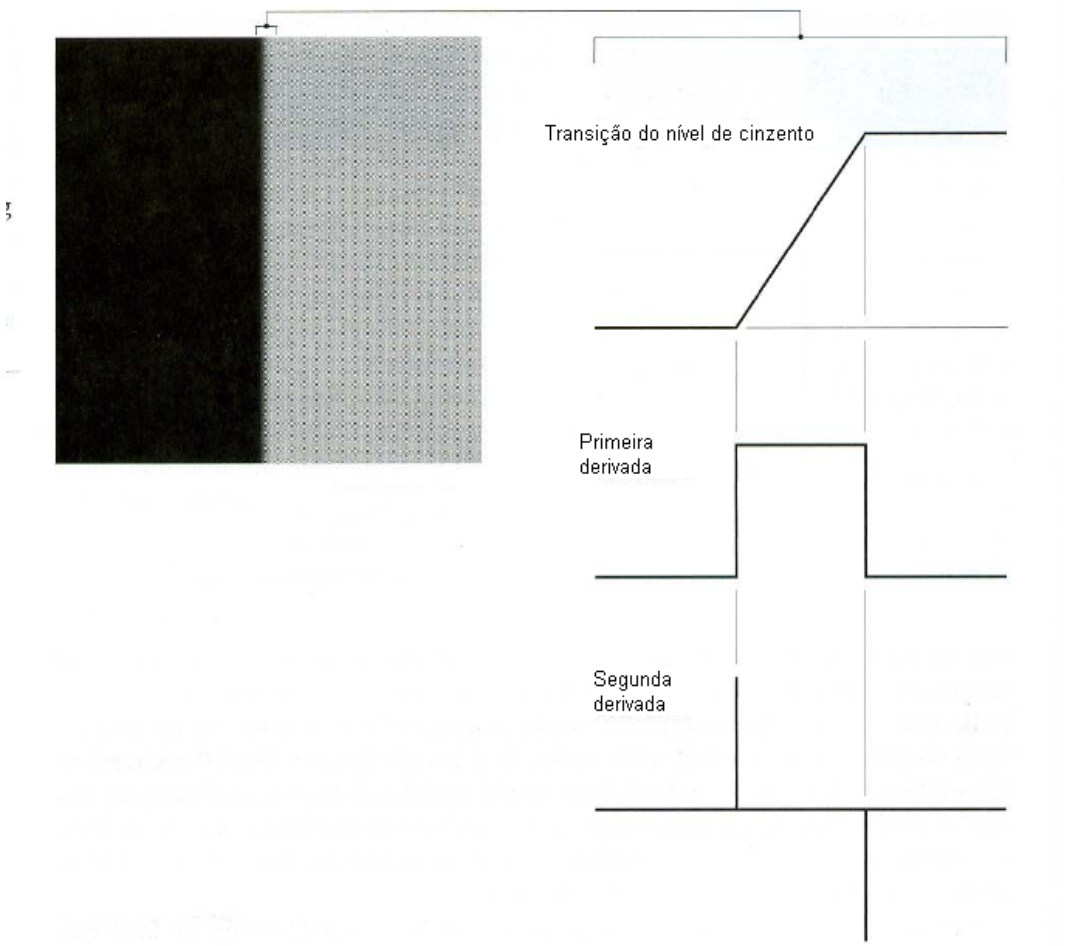


Figura 2.12: À esquerda está uma imagem com duas regiões de cinzento separadas por uma aresta vertical. À direita o gráfico das funções que definem a imagem e as suas derivadas segundo uma linha horizontal [30].

$$mag(\nabla f) \approx |G_x| + |G_y| \tag{2.8}$$

Para poder ser utilizada em processamento digital de imagem, esta função necessita de ser expressa numa forma discreta. Tendo em conta que a definição básica da primeira derivada de uma função unidimensional é habitualmente feita pela diferença progressiva[7]

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

e sabendo que segundo a direcção y a função será semelhante, podemos começar a definir as máscaras do gradiente.

A Fig. 2.13 apresenta uma máscara genérica 3×3. Com base na notação proposta nessa figura $f(x, y)$ será z_5, $f(x-1, y-1)$ será z_1 e por aí adiante. Com base nas funções anteriores podemos então concluir que a forma mais simples de definir os operadores de gradiente será

[7]Pode também ser feita pela diferença regressiva $\frac{\partial f}{\partial x} = f(x) - f(x-1)$.

$G_x = (z_i - z_j)$ e $G_y = (z_k - z_l)$. Como as máscaras mais usadas são do tipo 3×3 é necessário uma abordagem que inclua nove pontos simultaneamente. Uma das aproximações possíveis é a descrita pela seguinte fórmula:

$$\begin{aligned} mag(\bigtriangledown f)(z_5) \approx & |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| \\ & + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)| \end{aligned} \tag{2.9}$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Figura 2.13: Máscara genérica de dimensão 3×3.

−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	1	−1	0	1

Figura 2.14: Máscaras 3×3 utilizadas no cálculo de gradientes em imagens. São denominadas de *operadores de Sobel*.

As máscaras mostradas na Fig. 2.14 correspondem à aplicação da fórmula 2.9 e são chamadas de *Operadores de Sobel*. A máscara da esquerda aproxima a derivada na direcção y enquanto que a da direta a aproxima na direcção x. O factor de 2 serve para dar mais importância aos pontos centrais. É de notar que a soma de todos os coeficientes é zero, o que significa que a resposta numa área homogénea é nula, como seria de esperar num operador de cariz diferencial. Um exemplo da sua aplicação pode ser visto na Fig. 2.15.

Laplaciano

O método que permite usar o *Laplaciano* (um operador que, à semelhança do gradiente, habitualmente está associado a problemas em domínios contínuos) em processamento digital de imagem é muito semelhante àquele associado ao gradiente, ou seja, é necessário definir

uma formulação discreta da derivada de segunda ordem e construir uma máscara com base nessa formulação.

O Laplaciano de uma função contínua, $f(x, y)$, é definido como:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{2.10}$$

Tal como foi feito anteriormente para o gradiente, o Laplaciano pode ser descrito, de modo discreto, na direcção x, como:

$$\frac{\delta^2 f}{\delta x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y) \tag{2.11}$$

e na direcção y:

$$\frac{\delta^2 f}{\delta y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y) \tag{2.12}$$

A implementação digital do Laplaciano bidimensional faz uso da soma dos dois componentes:

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y) \tag{2.13}$$

A máscara da Fig. 2.16-a) é o resultado da implementação da equação anterior. O Laplaciano serve para realçar ainda mais as arestas (por comparação com o gradiente) mas possui a desvantagem de ser mais sensível ao ruído. Um exemplo da sua aplicação encontra-se na Fig. 2.17.

2.3.4 Operações morfológicas em imagens binárias

As operações morfológicas permitem transformar a estrutura de certas regiões das imagens, através da dilatação ou redução das suas fronteiras. Regra geral, são aplicadas a imagens binárias. Os operadores morfológicos alteram a forma das regiões baseando-se, também, nos valores dos vizinhos de um dado pixel, usando máscaras habitualmente denominadas *elementos estruturais*. Apesar da sua semelhança com os *kernels* utilizados na filtragem descrita na secção anterior, o mecanismo por detrás das operações morfológicas é diferente, pois neste caso os valores dos vizinhos não vão ser somados ou subtraídos ao valor do pixel em processamento, mas vão sim indicar se esse pixel deve ser apagado (definido como 0) ou não (Fig. 2.18). Os coeficientes dos elementos estruturais são binários e especificam quais os pixels vizinhos que irão ser considerados na aplicação da operação. Ou seja, se o valor de um coeficiente da máscara for 1 o pixel correspondente vai afectar o processamento do pixel central na aplicação da operação morfológica; se o valor na máscara for 0, o pixel é irrelevante nesta operação.

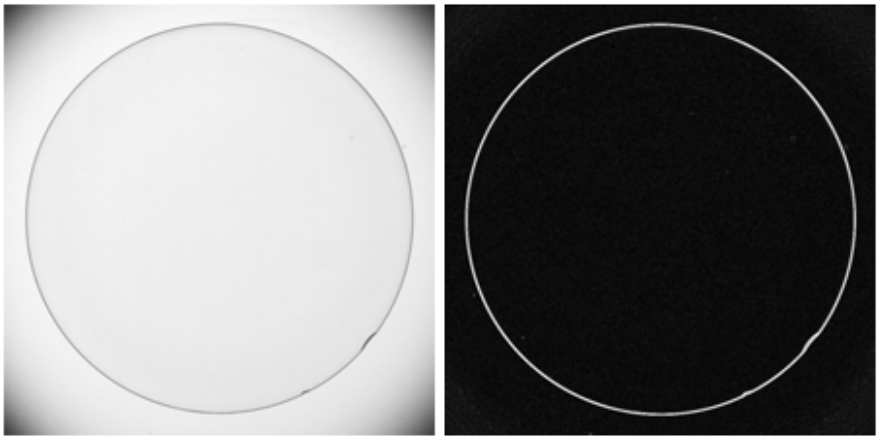


Figura 2.15: À esquerda uma imagem de uma lente de contacto. À direita a mesma imagem sob a acção de um gradiente de Sobel [30].

a

0	1	0
1	−4	1
0	1	0

b

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

c

−1	−1	−1
−1	8	−1
−1	−1	−1

d

Figura 2.16: a) Implementação digital do Laplaciano. b) Máscara Laplaceana que inclui também os pontos diagonais. c) e d): Duas implementações diferentes do Laplaciano.

Figura 2.17: À esquerda, uma imagem do Pólo Norte da Lua. À direita, a mesma imagem após ser filtrada com um Laplaciano [30].

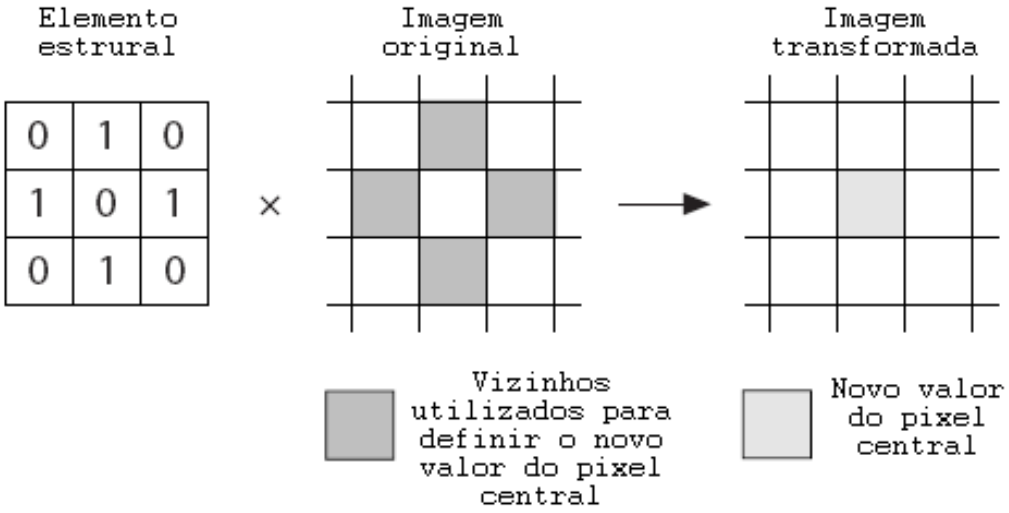


Figura 2.18: Exemplo de uma operação morfológica.

Nas operações morfológicas é definido, na máscara do elemento estrutural, um pixel central, p_0. Quando esta é colocada sobre uma dado pixel na imagem os pixels vizinhos que estejam sob a acção de um elemento da máscara com coeficiente 1 são definidos como p_i (no que se refere à descrição das operações que se segue).

Os dois tipos de operações morfológicas mais utilizados são a *erosão* e a *dilatação*.

A *erosão* permite eliminar pixels isolados no fundo da imagem e retira elementos ao contorno da imagem. É definida pelo seguinte elemento estruturante:

- Se $p_i = 0$, então $p_0 = 0$. De outro modo, $p_0 = 1$.
- Se $\text{AND}(p_i) = 1$, então $p_0 = 1$. De outro modo, $p_0 = 0$.

A *dilatação* elimina pequenos defeitos isolados em partículas e expande os contornos destas pela aplicação do seguinte elemento estruturante:

- Se $p_i = 1$, então $p_0 = 1$. De outro modo, $p_0 = 0$.
- Se $\text{OR}(p_i) = 1$, então $p_0 = 1$. De outro modo, $p_0 = 0$.

2.3.5 Filtragem em frequência

Ao contrário da filtragem espacial, os filtros na frequência não são aplicados directamente à imagem, mas sim às *frequências* presentes nas intensidades luminosas dos pixels. Esta representação é obtida através da *Transformada Discreta de Fourier* (DFT) da função, calculada por um algoritmo rápido habitualmente denominado de *Transformada Rápida de Fourier* (FFT)[8]. A DFT permite obter informação sobre a periodicidade e a dispersão dos padrões da imagem. A grande vantagem na realização do processamento em frequência deve-se à propriedade de que a convolução "no espaço" passa a ser um produto no domínio da frequência, à semelhança do que acontece a uma dimensão.

Esquematicamente, o processo de filtragem é composto por 3 passos distintos: i) a transformação da imagem espacial (coordenadas (x, y)) para a imagem na frequência (coordenadas (u, v)) recorrendo à DFT, calculada pelo algoritmo FFT; ii) a filtragem dessa imagem através de um produto das matrizes correspondentes à transformada da imagem e à transformada do filtro que se pretende aplicar; iii) e, para finalizar, a utilização a DFT inversa (também calculada por uma FFT) para reverter a imagem ao plano espacial (Fig. 2.19).

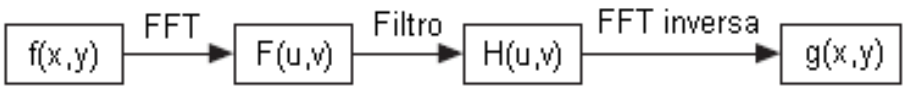


Figura 2.19: Esquematização da filtragem no espaço das frequências.

[8]DFT e FFT são as siglas de *Discrete Fourer Transform* e de *Fast Fourier Transform*, respectivamente.

A DFT de uma imagem é um vector bidimensional (ou matriz) de números complexos (também referido como *imagem complexa*), e é calculada executando primeiro uma transformação unidimensional sobre as linhas, seguida de uma transformação, também a uma dimensão, das colunas do resultado anterior. Numa imagem, os detalhes minuciosos e as arestas estão associados a altas ou moderadas frequências espaciais, porque introduzem mudanças de níveis de cinzentos (ou de cor) significativas num curto espaço (i.e., número de pixels). Os padrões com cores mais ou menos constantes estão associados a baixas frequências espaciais. Ao filtrar na frequência podemos remover, atenuar ou realçar as componentes espaciais correspondentes àquelas características. Note-se que isto era também feito com os filtros baseados em *kernels* atrás descritos.

No domínio contínuo a *Transformada de Fourier* bidimensional é definida por:

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(xu+yv)}dxdy \tag{2.14}$$

onde $f(x,y)$ é a intensidade luminosa no ponto (x,y) e u e v são os eixos horizontal e vertical das frequências espaciais, respectivamente. A Transformada de Fourier atribui um número complexo (em geral) a cada par (u,v).

No domínio discreto, a *Transformada Discreta de Fourier* é calculada pelo somatório

$$F(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} f(x,y)e^{-j2\pi(\frac{ux}{N}+\frac{vy}{M})} \tag{2.15}$$

onde $M \times N$ é a resolução da imagem.

O cálculo da DFT pode ser relativamente demorado, porque[9] $F(u,v)$ é composta por uma soma de senos e cosenos com muitas parcelas e, para cada par (u,v), todos os valores de $f(x,y)$ contribuem para o cálculo de $F(u,v)$. Por isso, são utilizados os algoritmos genericamente denominados de FFT que reduzem o número de computações de $\approx M^2N^2$ para $\approx M^2(\log_2 N)^2$ (presumindo que M e N têm valores aproximados) no cálculo da DFT da imagem. Existem muitas variantes da FFT que podem ser utilizadas. Muito rapidamente, pode-se dizer que todas elas se baseiam na periodicidade da exponencial complexa para reduzir o número de cálculos aparentemente necessários ao cálculo da DFT quando se examina a anterior equação.

A DFT inversa converte uma imagem transformada, $F(u,v)$, de novo numa imagem espacial, $f(x,y)$, através de cálculos muito semelhantes aos efectuados na DFT directa[10]:

$$f(x,y) = \frac{1}{NM}\sum_{u=0}^{N-1}\sum_{v=0}^{M-1} F(u,v)e^{j2\pi(\frac{ux}{N}+\frac{vy}{M})} \tag{2.16}$$

[9] A fórmula de Euler indica que $e^{-j2\pi ux} = \cos 2\pi ux$ - $j \sin 2\pi ux$.

[10] O termo $1/(NM)$ na DFT inversa pode ser distribuído entre esta e a DFT directa, colocando-se um factor $1/\sqrt{NM}$ em ambas.

A filtragem no espaço das frequências pode ser passa-baixo, passa-alto, passa-banda ou de qualquer dos tipos habitualmente encontrados em processamento analógico. A filtragem passa-baixo remove (ou atenua) as altas frequências, enquanto que a filtragem passa-alto executa o inverso. Nestas filtragens podemos optar por atenuar frequências ou, pura e simplesmente, removê-las acima ou abaixo de uma determinada frequência de corte (note-se que estamos a fazer processamento "numérico").

As operações que podem ser levadas a cabo com transformadas de Fourier são inúmeras (assim como as aplicações que a utilizam) e estão descritas no livro [44]. No processamento de imagem realizado neste trabalho não será utilizada, pelo que não examinamos aqui o processamento de Fourier em mais detalhe.

2.4 Fundamentos de processamento de imagem a cores

2.4.1 O sistema RGB

Como já foi referido existem diversos métodos de representar imagens a cores, sendo o sistema RGB[11] o mais comum. Este sistema constrói imagens de cor composta baseando-se nas cores vermelho, verde e azul. Estas três cores são conhecidas como cores primárias, visto todas as cores do espectro visível poderem ser construídas por um combinação daquelas três.

O espaço RGB pode ser visualizado como um cubo tridimensional com o verde, o vermelho e o azul nos cantos de cada eixo, como se pode ver na Fig. 2.20. A cor preta é a origem do cubo enquanto que o branco se encontra no vértice oposto. Cada lado do cubo tem um valor entre 0 e 1. Ao longo de cada eixo do cubo de RGB, as cores evoluem desde nenhuma contribuição (0) até ao ponto de saturação (1). Assim, cada ponto (cor) dentro do cubo é especificado por três números, um tripleto (R, G, B). A linha diagonal que liga o ponto que descreve o preto $(0, 0, 0)$ até ao branco $(1, 1, 1)$ representa a escala de valores de cinzento onde as componentes vermelho, verde e azul são iguais.

Também é possível tratar pixels coloridos como vectores. No sistema RGB cada ponto de cor pode ser interpretado como um vector que se estende da origem até ao ponto que define a sua cor no sistema de coordenadas referido.

Seja c um vector arbitrário:

$$c = \begin{vmatrix} c_R \\ c_G \\ c_B \end{vmatrix} = \begin{vmatrix} R \\ G \\ B \end{vmatrix} \tag{2.17}$$

Esta equação indica que as componentes de c são simplesmente as componentes RGB num dado pixel/ponto numa imagem a cores . Se tivermos em conta que as componentes de cor são função das coordenadas (x, y) aquela equação será, mais concretamente

[11] RGB é e sigla de "Red, Green, Blue".

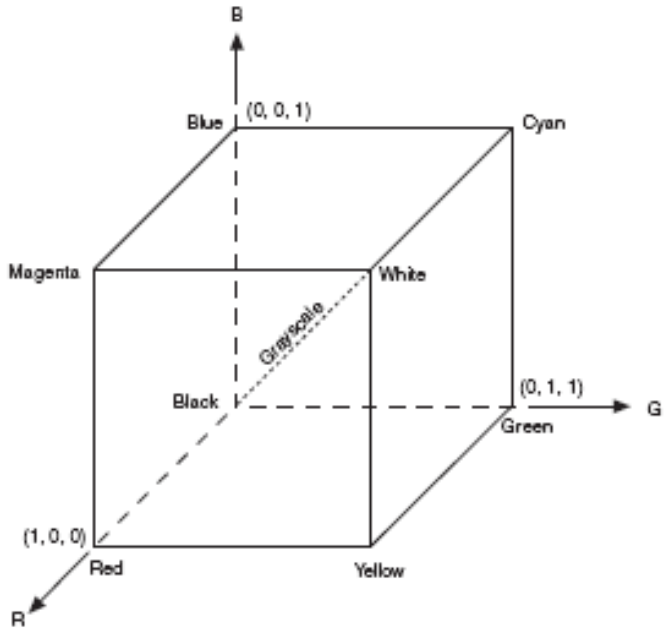


Figura 2.20: Cubo RGB.

$$c = \begin{vmatrix} c_R(x,y) \\ c_G(x,y) \\ c_B(x,y) \end{vmatrix} = \begin{vmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{vmatrix}. \tag{2.18}$$

Para um imagem de tamanho $M \times N$, existirão MN vectores $c(x,y)$, para $x = 0, 1, 2, ..., M-1$; $y = 0, 1, 2, ..., N-1$. Esta notação define, então, vectores cujas componentes são funções de x e y, o que nos permite utilizar técnicas de processamento de imagem aplicáveis espacialmente (Fig. 2.21). Sendo assim, podemos processar imagens a cor utilizando métodos de processamento de imagens em escala de cinzentos.

2.4.2 O sistema HSL

O espaço de cor HSL[12] foi desenvolvido de modo a descrever a cor em termos semelhantes à maneira como o olho humano a observa e interpreta. *Hue* corresponde ao comprimento de onda dominante na cor. A saturação refere-se à quantidade de branco que é adicionado ao *hue* e representa a pureza relativa da cor. Uma cor sem branco diz-se completamente saturada. Cores como o rosa, composto por vermelho e branco, são menos saturadas que o vermelho. A luminosidade descreve os níveis de cinzento da imagem.

O sistema coordenado do espaço de cor HSL é de simetria cilíndrica. As cores estão definidas num hexacone (Fig. 2.22). Os valores de hue vão de 0 a 360º, com o vermelho a 0º (é de notar que as cores complementares estão a 120º umas das outras). A saturação vai de 0 a 1, onde 1 representa a pureza da cor sem branco. A luminosidade vai de 0 a 1, onde 0 é branco e 1 é preto.

[12]"Hue", Saturação e Luminosidade.

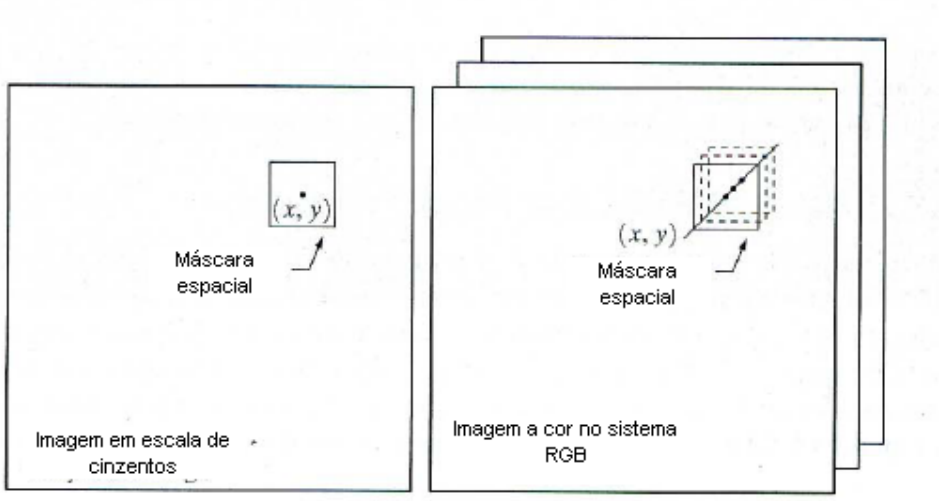


Figura 2.21: Aplicação de máscaras de processamento espaciais em imagens em escala de cinzentos (à esquerda) e em imagens RGB (à direita).

2.5 Algoritmos dedicados à monitorização automática

No tratamento de imagens com objectos em movimento existem três métodos de base para extrair informação sobre esse movimento:

1. *Ajuste de contraste*: numa imagem em que o objecto a estudar sobressaia do fundo em que está inserido, o contraste da imagem pode, por vezes, ser alterado de modo a que todos os pixels com valores abaixo/acima dos que constituem o objecto fiquem com o valor zero, e os restantes com o valor de 1, ou seja:

$$g(x,y) = \begin{cases} 1 & \text{se} \quad |f(x,y)| > T \\ 0 & \text{se} \quad |f(x,y)| < T \end{cases}$$

   onde T é o valor de contraste desejado, $f(x,y)$ é a imagem original e $g(x,y)$ a imagem resultante. Esta técnica produz bons resultados desde que o ambiente não contenha superfícies que reflectem a luz. A imagem resultante, $g(x,y)$, é binária.

2. *Subtracção de imagens*: na sua forma mais básica, esta operação envolve subtrair a imagem actual da imagem anterior. Nesta operação, os pixels que não mudarem de valor (objectos que não mudaram de posição) são tornados pretos. A diferença entre duas imagens adquiridas em instantes t_i e t_j é definida como:

$$d_{i,j}(x,y) = \begin{cases} 1 & \text{se} \quad |f(x,y,t_i) - f(x,y,t_j)| > T \\ 0 & \text{se} \quad |f(x,y,t_i) - f(x,y,t_j)| \leq T \end{cases}$$

   A desvantagem deste método é que só destaca a parte do objecto que está em movimento e no caso em que o objecto se mantém (quase) imóvel a sua visibilidade é

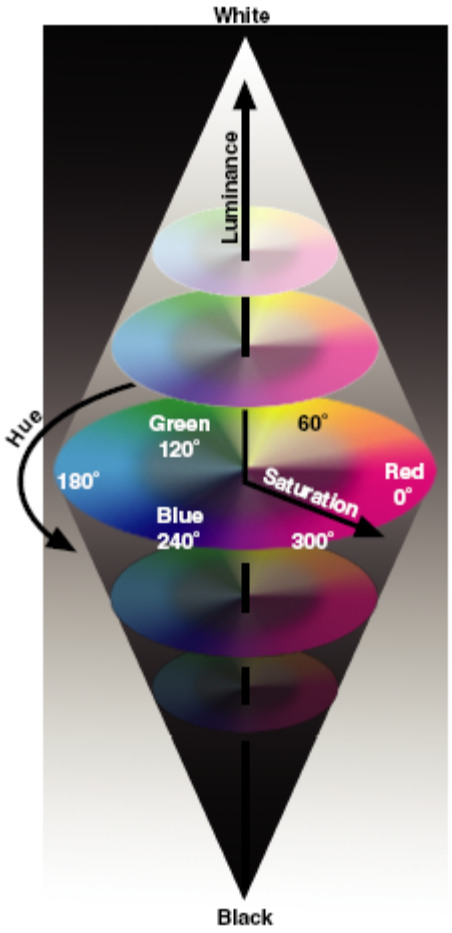


Figura 2.22: Espaço de cor HSL.

nula.

3. *Subtracção do fundo*: nesta técnica uma imagem estática do ambiente $g(x, y)$ é adquirida antes da introdução de objectos em movimento, no instante t_0. Esta imagem vai depois ser subtraída às imagens captadas $f(x, y, t_i)$:

$$d_i(x, y) = |f(x, y, t_i) - g(x, y, t_0)|$$

A principal desvantagem na sua aplicação é que o ambiente de fundo pode ir-se alterando gradualmente durante o processamento das imagens e tais mudanças tornam o fundo real diferente daquele que foi registado inicialmente, invalidando a eficácia da técnica.

2.5.1 Associação de observações com indivíduos

No caso da observação de vários objectos semelhantes, pode existir a necessidade de reconhecer e seguir cada um deles individualmente. O método usado para este fim é a monitorização da deslocação espacial dos objectos de imagem para imagem.

Passamos a descrever uma das técnicas apropriadas [21]. Sendo a posição inicial de todos os objectos conhecida, é criado um esquema de associação entre imagens sucessivas, assumindo que os objectos têm uma deslocação máxima limitada. Entre duas aquisições sucessivas, nos instantes t e $t+1$, os pontos correspondentes a cada objecto no tempo $t+1$ são agrupados à posição mais próxima que já foi detectada na aquisição efectuada em t. Este algoritmo *ganancioso*[13] é bastante eficaz, podendo, no entanto, gerar por vezes agrupamentos incorrectos, como se pode ver na Fig. 2.23. Contudo, a sua exactidão pode ser melhorada se forem levados em conta todos os agrupamentos possíveis entre duas aquisições sucessivas e for calculado o ajuste de cada conjunto globalmente segundo o critério dos mínimos quadrados:

$$\min \sum_{i=1}^{N_{assoc}} (dist(anter_i, actual_i))^2 \tag{2.19}$$

onde N_{assoc} é o número de possíveis associações entre objectos, $(anter_i, actual_i)$ é o i^{esimo} conjunto e a função $dist(p,q)$ representa a distância Euclideana. O agrupamento seleccionado como mais correcto (eventualmente o correcto...) será aquele que minimiza a função. Este algoritmo garante uma boa associação na prática, mas não consegue ultrapassar a limitação de que se dois objectos estiverem em contacto próximo num dado momento o sistema poderá contabilizá-los como um só e perderá, então, a individualização conseguida até então.

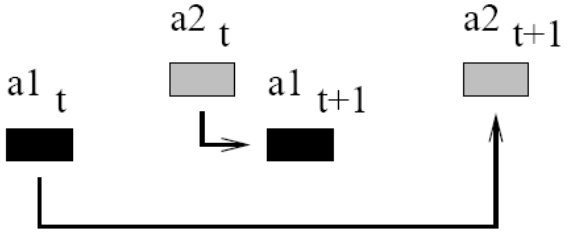


Figura 2.23: Caso em que o algoritmo de associação falha: dois objectos, $a1$ e $a2$, são representados em t e $t+1$. As setas demonstram a associação incorrecta ($a2$ é agrupado antes de $a1$) quando esta é baseada apenas no critério da distância miníma.

2.6 Conclusão

Neste capítulo foi dada uma panorâmica geral sobre imagem digital e respectivas técnicas de processamento.

Uma imagem digital é constituída por elementos discretos, os pixels. O processamento das imagens tem como base operações sobre o valor destes elementos e permite, no limite, aumentar a quantidade de informação que pode ser retirada de uma imagem.

[13] *Greedy*, em Inglês.

Também foram tratados algoritmos de monitorização automática, ou seja, métodos computacionais que permitem o seguimento de objectos numa imagem.

A área do processamento de imagem é extremamente vasta e existem muitos textos a ela dedicados. Por isso, aqui apenas foi esboçada a respectiva panorâmica geral destacando as técnicas mais importantes para a realização do nosso trabalho.

Capítulo 3

O Filtro de Kalman

3.1 Introdução

Na implementação prática do sistema de monitorização de animais pretende-se estimar a distância percorrida e as velocidades instantânea e média. Verificou-se, experimentalmente, que quando são usados apenas os algoritmos disponibilizados pelo sistema *LabVIEW/NI Vision* para obter as coordenadas dos centros de massa dos alvos de perseguição, a partir das quais são calculadas as grandezas mencionadas, estas estavam sujeitas ao efeito de erros de estimação bastante severos. Por essa razão, decidiu-se recorrer a técnicas que amenizassem este problema e, após alguma ponderação, foi decidido implementar um Filtro de Kalman (FK).

Na medida em que o Filtro de Kalman corresponde a um algoritmo que não é, propriamente, de processamento de imagem (embora seja sobejamente utilizado no seguimento de alvos em imagens), decidimos tratá-lo neste breve capítulo em que é feito o seu estudo.

3.2 A Génese do Filtro de Kalman

O *Filtro de Kalman* é um dos mais robustos algoritmos utilizados amiúde na estimação da posição de alvos em movimento. O FK é um *algoritmo recursivo de processamento de dados* que é *óptimo*, de acordo com critérios estatísticos: na verdade, de acordo com quase todos os critérios estatísticos que se revelam de utilidade prática.

O FK implementa um estimador linear útil para a previsão, para a correcção (filtragem) e para o alisamento ("smoothing") de sinais discretos ou contínuos. Aqui iremos considerar apenas o caso discreto, pois nele se enquandram as sequências de imagens às quais se aplica o FK. Na prática, uma vez que o FK é um programa executado por um processador, o filtro discreto é aquele que é importante pois as variáveis contínuas, em sistemas contínuos, são adquiridas habitualmente com uma frequência de amostragem constante.

O FK é simultaneamente simples, robusto e óptimo, no sentido em que minimiza a variância da estimativa do estado (mais precisamente, das variáveis de estado), porque incorpora todos os dados acessíveis desde o início da observação do alvo. A sua principal

vantagem é assentar na recursividade calculando, em cada iteração, estimativas *apenas* a partir da informação armazenada na iteração anterior: esta característica, que se traduz no facto de necessitar de escassos recursos de memória para ser implementado é, juntamente com a optimalidade, uma das razões principais para a sua popularidade.

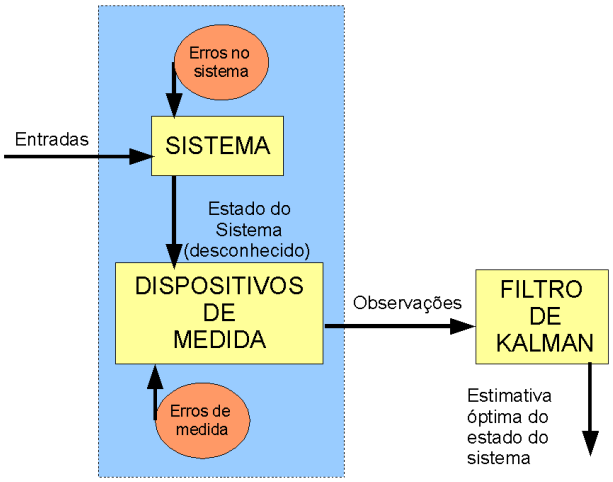


Figura 3.1: Aplicação típica de um Filtro de Kalman.

O exemplo típico de aplicação do FK é mostrado na figura 3.1. O processo sob observação engloba o sistema, propriamente dito, e o(s) dispositivo(s) de medida. O sistema está sujeito ao efeito de entradas que se presume serem conhecidas (e que podem não existir, se o sistema for autónomo). Existem duas fontes de erro: a primeira *incide no sistema* e modela os erros, as imprecisões e as aproximações inerentes à modelação, os efeitos imprevisíveis que alteram o comportamento idealizado do sistema (vento, ruído de origem física, vibrações,...); a segunda fonte de erro está associada ao processo de medida e entra em conta, por exemplo, com o ruído nos dispositivos de medida, o erro de quantização (normalmente é utilizado um conversor Analógico-Digital para criar os valores discretos).

O filtro de Kalman combina todos as observações já registadas com o conhecimento apriorístico da dinâmica do sistema para produzir uma estimativa das variáveis pretendidas, de tal maneira que a variância do erro de observação é minimizada, em termos estatísticos. Por outras palavras, se fossem aplicados, um grande número de vezes, um conjunto de filtros "candidatos" à mesma aplicação, então os resultados médios obtidos pelo FK seriam melhores que o de qualquer outro.

O FK é um algoritmo que propaga a estimativa óptima do estado do sistema, juntamente com a informação sobre a função de densidade de probabilidade desse estado durante todo o processo iterativo. Uma propriedade muito importante do filtro é necessitar apenas da informação guardada no instante anterior e da medida efectuada no actual instante para produzir a estimativa óptima, o que o torna particularmente atraente para a implementação em ambientes de cálculo em que a memória disponível é limitada.

3.2.1 O Filtro de Kalman Discreto

O modelo do processo

Um dos pressupostos subjacentes ao Filtro de Kalman é que o processo sob observação é necessariamente linear. A descrição, segundo o modelo das variáveis de estado, de um sistema linear consiste nas seguintes duas equações (denominadas, respectivamente, *equação de estado* e *equação de saída*):

$$\begin{aligned} x_{k+1} &= A\,x_k + B\,u_k \\ y_k &= C\,x_k \end{aligned} \tag{3.1}$$

sendo x o vector das variáveis de estado, u o vector das entradas no (ou o "controlo" do) sistema e y o vector de saída (ou vector das medidas) que corresponde às observações feitas ao sistema. A, B e C são matrizes de dimensões concordantes com as daqueles vectores. A matriz A é por vezes denominada de *matriz de transição (de estado)*[1]. No entanto, conforme já foi exposto, o erro do sistema, w, e o erro de observação, z, adicionam-se a cada uma daquelas equações:

$$\begin{aligned} x_{k+1} &= A\,x_k + B\,u_k + w_k \\ y_k &= Cx_k + z_k \end{aligned} \tag{3.2}$$

Nas equações acima, k é o índice discreto de tempo. u, um conjunto de entradas conhecidas, pode não existir em certos sistemas. Todas as variáveis são (em geral) vectores. O vector x contém toda a informação sobre o estado presente do sistema, mas não o podemos medir directamente: em vez disso medimos y, que se relaciona com uma estimativa de x corrompida por ruído.

O problema que o FK vem resolver é o seguinte: pretende-se usar as observações em y para estimar o estado do sistema x, apresentando esta estimativa, em termos estatísticos, a variância mínima. De seguida vai-se apresentar sumariamente a sua dedução.

As equações do filtro de Kalman

O Filtro de Kalman assenta em três pressupostos, que são:

1. O modelo da dinâmica do sistema é *linear*;
2. O ruído w associado ao processo é branco, Gaussiano e de média nula;
3. O ruído z associado à medição é também branco, Gaussiano e de média nula e, adicionalmente, é independente do erro do processo, w.

[1] No filtro de Kalman a matriz A pode depender do tempo, isto é, $A \equiv A_k$. Isto acontece, por exemplo, quando o sistema discreto se deve à discretização de um sistema contínuo com um período de amostragem variável. No desenvolvimento aqui apresentado vai-se presumir que é constante.

Matematicamente diz-se que w $N(0, S_w)$ e que z $N(0, S_z)$, onde $S_w = E[w\, w^T]$ e $S_z = E[z\, z^T]$ são as respectivas matrizes de covariância.

O ruído branco é caracterizado por uma densidade de potência espectral constante para qualquer frequência $\omega \in]-\infty, \infty[$. Na prática é suficiente que aquela grandeza seja aproximadamente constante na banda de frequências de funcionamento do sistema, que tipicamente é do tipo "passa-baixo". O qualificativo "Gaussiano" implica que a distribuição em amplitude do ruído é do tipo Gaussiana ou normal. O ruído branco limitado em frequência é frequentemente designado de "ruído rosa".

Aqueles pressupostos, relativamente ao processo, na prática raramente são restritivos. Quando a informação sobre o ruído consiste apenas numa média e numa variância, sem ser conhecida a sua distribuição de probabilidade exacta, a melhor "aposta" que se pode fazer relativamente a essa distribuição é que é uma Gaussiana. Por outro lado, a grande maioria dos processos geradores de ruído é do tipo branco e Gaussiano.

Para deduzir as equações correspondentes ao filtro de Kalman parte-se de uma estimativa do estado anterior do sistema, $\hat{x}_k$, a partir da qual se obtém uma estimativa *preliminar* (visto que irá ser posteriormente corrigida com a observação feita no corrente instante), denominada $\hat{x}_{k+1}^-$:

$$\hat{x}_{k+1}^- = A\,\hat{x}_k + B\,u_k \tag{3.3}$$

(em que o sinal (-) indica a natureza de uma estimativa provisória apriorística.) Sendo uma estimativa, $\hat{x}_k$ é caracterizada por uma matriz de covariância $P_k = E[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^T]$, onde $E[\cdots]$ denota o operador de expectância[2]. A covariância de $\hat{x}_{k+1}^-$ terá uma contribuição $A\,P_k\,A^T$ para a covariância de x_{k+1}^-. Para esta covariância existirá também a contribuição da covariância do erro w_k, denotada aqui por $S_w = E[w_k\, w_k^T]$. Então, uma estimativa *preliminar* (visto que irá ser também posteriormente corrigida) daquela covariância[3] é:

$$P_{k+1}^- = A\,P_k\,A^T + S_w \tag{3.4}$$

Relativamente à observação realizada no instante $k+1$ pode-se escrever

$$\hat{y}_{k+1} = C\,x_{k+1} + z_{k+1} \tag{3.5}$$

o que permite definir uma equação para a estimativa actualizada do estado do sistema, x:

$$\hat{x}_{k+1} = \hat{x}_{k+1}^- + K_{k+1}(\hat{y}_{k+1} - C\,\hat{x}_{k+1}^-) = (I - K_{k+1}\,C)\hat{x}_{k+1}^- + K_{k+1}\,\hat{y}_{k+1} \tag{3.6}$$

Então, o objectivo agora é obter a matriz K_{k+1}, denominada de *Ganho de Kalman*, que irá minimizar a covariância de $\hat{x}_{k+1}$. A partir da anterior equação esta covariância é:

$$P_{k+1} = (I - K_{k+1}\,C)\hat{P}_{k+1}^-(I - K_{k+1}\,C)^T + K_{k+1}S_zK_{k+1}^T \tag{3.7}$$

[2]As estimativas do estado que o FK produz são variáveis aleatórias e, por isso, vão ser caracterizadas por uma média e por uma (co)variância.

[3]Se uma variável aleatória n-dimensional, $\hat{X}$, é uma combinação linear de várias variáveis aleatórias independentes entre si, isto é $\hat{X} = \sum_i A_i\,\hat{X}_i$ (onde os símbolos A_i se referem a matrizes), a sua média e covariância serão dadas respectivamente por $\mu_X = \sum_i A_i\,\mu_{Xi}$ e $S_X = \sum_i A_i\,S_{Xi}\,A^T$.

Utilizando as propriedades da derivação de equações matriciais (consulte, por exemplo, [46]) para minimizar o traço[4] de P_{k+1} , conclui-se que

$$K_{k+1} = P_{k+1}^{-} C^T (C\, P_{k+1}^{-} C^T + S_z)^{-1} \tag{3.8}$$

o que conduz a que a covariância actualizada seja dada por

$$P_{k+1} = (I - K_{k+1}\, C) P_{k+1}^{-} \tag{3.9}$$

Sumariando, o conjunto de operações associadas a uma iteração do filtro de Kalman é o seguinte [47]:

$$\begin{aligned} \hat{x}_{k+1}^{-} &= A\,\hat{x}_k + B\,u_k \\ P_{k+1}^{-} &= A\,P_k\,A^T + S_w \\ K_{k+1} &= P_{k+1}^{-} C^T (C\, P_{k+1}^{-} C^T + S_z)^{-1} \\ \hat{x}_{k+1} &= \hat{x}_{k+1}^{-} + K_{k+1}(\hat{y}_{k+1} - C\,\hat{x}_{k+1}^{-}) \\ P_{k+1} &= (I - K_{k+1}\, C) P_{k+1}^{-} \end{aligned} \tag{3.10}$$

É esta a formulação que será utilizada mais adiante na implementação em computador do filtro.

3.3 O Filtro de Kalman Aumentado

Como foi já realçado, o FK resolve o problema de estimação de um estado $x \in \Re^n$ de um processo em tempo discreto que é governado por uma equação estocástica linear. Quando os processos são descritos por equações não lineares pode ser utilizado o *Filtro de Kalman Aumentado*[5] (EKF). Neste caso, as equações de estado e de saída subjacentes ao processo são:

$$\begin{aligned} x_k &= f(x_{k-1}, u_{k-1}, w_{k-1}) \\ y_k &= h(x_k, z_k), \end{aligned} \tag{3.11}$$

Linearizando as equações por expansão em série de Taylor, pode-se escrever o seguinte conjunto de equações, equivalentes ao conjunto 3.11:

$$\begin{aligned} \hat{x}_{k+1}^{-} &= f(\hat{x}_k,\, u_k, 0) \\ P_{k+1}^{-} &= A_k\,P_k\,A_k^T + W_k\,S_w\,W_k^T \\ K_{k+1} &= P_{k+1}^{-} C_{k+1}^T (C_{k+1}\, P_{k+1}^{-} C_{k+1}^T + S_{zk+1})^{-1} \\ \hat{x}_{k+1} &= \hat{x}_{k+1}^{-} + K_{k+1}[\hat{y}_{k+1} - h(x_{k+1}^{-}, 0)] \\ P_{k+1} &= (I - K_{k+1}\, C_{k+1}) P_{k+1}^{-} \end{aligned} \tag{3.12}$$

[4] O traço de uma matriz é a soma dos seus elementos diagonais.

[5] Extended Kalman Filter.

Note que agora as matrizes A_k, C_k dependem da iteração. Com efeito, são os Jacobianos das funções não lineares calculados nos instantes de tempo adequados, ou seja (onde, por exemplo, $A_{(k)(i,j)}$ denota o elemento i, j da matriz calculado no instante $k+1$)

$$A_{(k)(i,j)} = \left.\frac{\partial f_i}{\partial x_j}\right|_{(\hat{x}_{k-1},u_{k-1},0)} \qquad W_{(k)(i,j)} = \left.\frac{\partial f_i}{\partial w_j}\right|_{(\hat{x}_{k-1},u_{k-1},0)}$$

$$H_{(k)(i,j)} = \left.\frac{\partial h_i}{\partial x_j}\right|_{(\bar{x}_k,0)} \qquad Z_{(k)(i,j)} = \left.\frac{\partial h_i}{\partial w_j}\right|_{(\bar{x}_k,0)}$$

É conveniente recordar que f e h são funções vectoriais e que as variáveis são também vectores, em geral.

Uma questão que está ligada ao EKF é a alteração sofrida pelo ruído quando passa pelas funções não lineares. De facto, o ruído deixa em geral de ser branco e Gaussiano. Por outro lado, a linearização só será uma boa descrição do modelo quando $\hat{x}_{k+1}$ não está muito afastado do ponto de linearização anterior, $\hat{x}_k$, e quando na prática isto não se verifica, o EKF perde as propriedades de estimador óptimo.

Na prática, os Jacobianos são muitas vezes difíceis de calcular, a linearização leva a filtros com um desempenho extremamente instável se o período de amostragem não for suficientemente pequeno e, quando este período é encurtado na tentativa de evitar o anterior problema, verifica-se que a quantidade de operações aritméticas necessárias para gerar os Jacobianos e prever as estimativas do estado e da covariância são muito elevados.

Assim, outras propostas para construir filtros apropriados para sistemas não lineares têm vindo a ser publicadas e uma panorâmica recente pode ser vista em [45].

3.4 Conclusão

O Filtro de Kalman é considerado, desde o seu aparecimento, como o "cavalo de batalha" no que respeita à estimação do estado de sistemas lineares sujeitos ao efeito de ruído do tipo branco e Gaussiano. Foram desenvolvidas muitas extensões do FK para lidar eficazmente com desvios àquele modelo.

O FK é empregue em variadas situações práticas, algumas no mínimo curiosas: seguimento de mísseis, seguimento de cabeças/mãos/baquetes (de bateria), extracção de movimentos labiais a partir de vídeo, ajuste de curvas de Bézier a pontos de um gráfico, imensas aplicações em visão por computador, na economia, na navegação e na meteorologia.

Capítulo 4

O Ambiente de Desenvolvimento LabVIEW

O *LabVIEW*[1], criado pela companhia *National Instruments*, é uma linguagem de programação gráfica que utiliza preferencialmente ícones, por oposição a instruções textuais, na criação de aplicações.

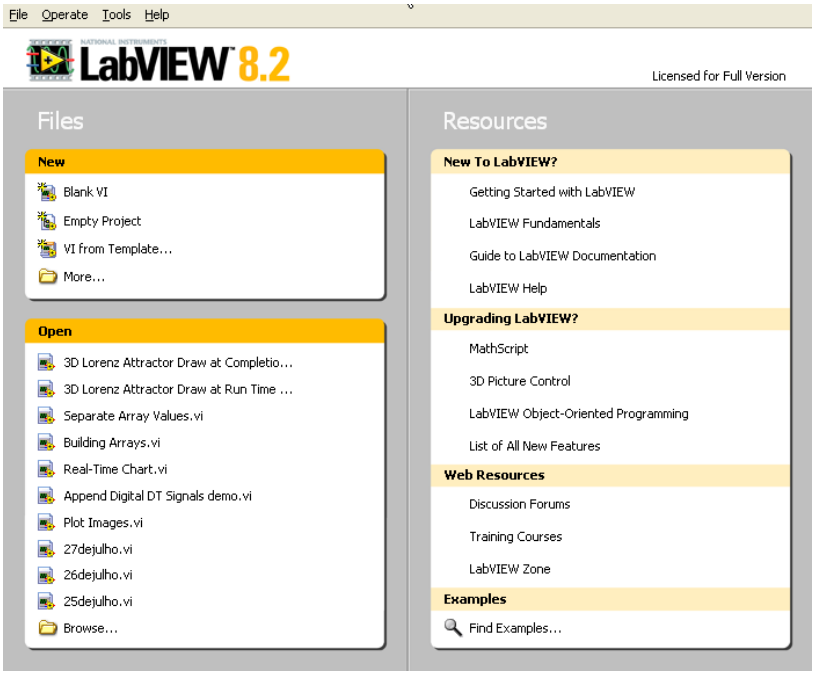


Figura 4.1: Painel inicial do LabVIEW 8.2.

O LabVIEW proporciona um ambiente interactivo (Fig.4.1) focado na rápida criação de programas vocacionados, sobretudo, para a aquisição e análise de dados sem, contudo, perder as capacidades das linguagens de programação tradicionais tais como o Microsoft Visual C, o Delphi, o Borland C, e outras. Um dos maiores benefícios em utilizar linguagens gráficas de programação consiste no aumento da rapidez de criação de aplicações, pois o modo de

[1] *Laboratory Virtual Instrument Engeneering Workbench.*

construcção do programa permite evitar, por exemplo, a maioria dos erros de sintaxe que infestam as linguagens baseadas em texto, a necessidade de alocar memória ou a necessidade de declarar variáveis.

A linguagem de programação usada no LabVIEW tem o nome de **G** e baseia-se no modelo de fluxo de dados, onde a execução das funções é efectuada pela ordem em que os diversos objectos (que representam funções) estão ligados. Esta particularidade é bastante útil na construção de programas, pois permite executar múltiplas operações em concorrência (isto é, um mesmo conjunto de dados pode ser fornecido a vários objectos ao mesmo tempo, enquanto que na maioria das outras linguagens de programação[2] de uso geral a execução do programa faz-se de acordo com linhas de código sequenciais o que impossibilita acções simultâneas).

Na prática, para todas as linguagens de programação existem editores e compiladores apropriados. Nas linguagens baseadas em texto o editor produz caracteres ASCII que definem o programa que é passado ao compilador para criar o "executável". Em programação gráfica o editor permite criar uma imagem que é passada ao compilador. Outra das vantagens do LabVIEW sobre as linguagens compiladas é que a criação da imagem e a sua compilação são executadas quase simultaneamente, o que o torna muito interactivo e rápido para trabalhar.

4.1 Programação em LabVIEW

Instrumentos Virtuais

Os programas do LV são habitualmente denominados Instrumentos Virtuais (IVs). Cada IV contém três componentes:

O **Painel Frontal** é a interface com o utilizador. Como interface, o Painel Frontal oferece um extenso conjunto de controlos (objectos de entrada ("input") que possibilitam o inserção de dados no programa) e de indicadores (objectos de saída ("output"), gráficos e tabelas, que apresentam os dados ao utilizador). Estes controlos e indicadores aparecem como terminais no Bloco de Diagramas. Os terminais são "portas" de entrada e de saída que trocam informação entre dois painéis (Fig. 4.2).

O **Diagrama de Blocos** é onde estão contidos os objectos de programação: o utilizador cria o programa ligando os objectos entre si com fios. A cor e o símbolo de cada ícone indicam o tipo de dados (inteiros ou decimais, "strings"[3] ou vectores, etc... que este pode

[2]Há excepções: as linguagens de descrição de hardware, tais como o VHDL e o Verilog, permitem modelar a execução concorrente de vários blocos para simular o que acontece com o "hardware" na realidade. Outros mecanismos para introduzir a concorrência de acções nos programas são o uso de "threads" ou a criação de múltiplos processos, que comunicam entre si por mensagens. Estas abordagens são cada vez nais populares em virtude de a grande maioria dos computadores actuais dispor de processadores *multicore*.

[3]A usual tradução deste termo para "cadeia de caracteres" é aqui preterida, mantendo-se a denominação

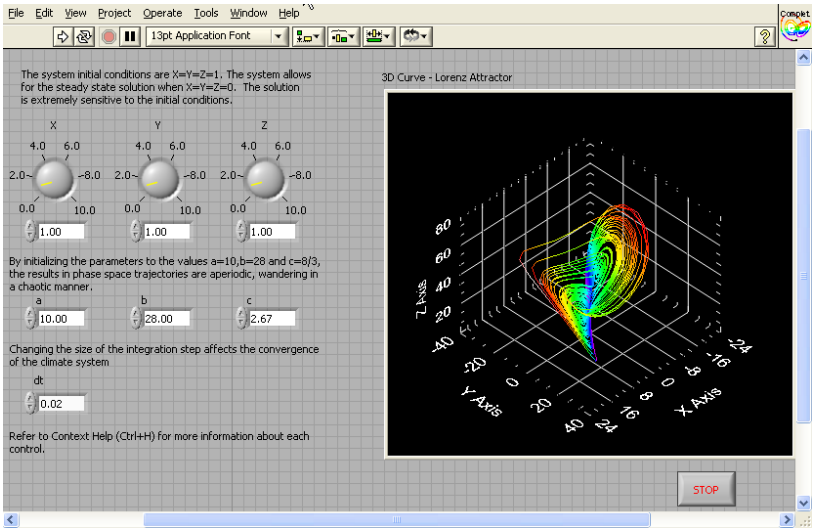


Figura 4.2: Painel frontal na execução de um dos exemplos fornecidos pela National Instruments: *3D Lorentz Attractor Draw at Completion using 3D Curve.vi*.

processar. Veja a Fig. 4.3.

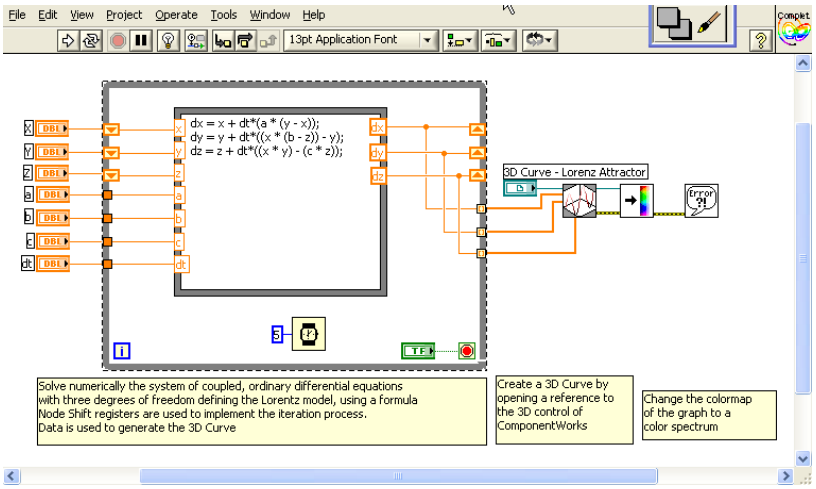


Figura 4.3: Diagrama de Blocos do exemplo *3D Lorentz Attractor Draw at Completion using 3D Curve.vi*.

O **Painel Conector** identifica o IV de modo a que ele possa ser usado dentro de um outro IV. Um IV é chamado de sub-IV quando é usado dentro de outro IV. Cada sub-IV pode ser usado como subprograma e chamado de dentro de outro programa. No entanto, devido ao modelo de funcionamento de fluxo de dados, as chamadas recursivas de IVs não são possíveis.

Como já foi referido, a programação em LabVIEW é fácil e intuitiva quando comparada com o que se passa com outras linguagens de programação. Passa-se agora a descrever as componentes básicas dos programas em LabVIEW.

em Inglês.

Fios

Os dados são transferidos entre os vários objectos do bloco de diagramas através de *fios*. Cada fio está associado a um só tipo de dados, mas pode-se ligar um fio a quantas funções/objectos se queira e os dados que o fio transporta estão acessíveis a todos os objectos a que ele está ligado. Os fios mudam de cor, estilo ou espessura consoante o tipo de dados que transportam. Quando um fio é ligado incorrectamente (i.e., quando se liga dois tipos de objectos que suportam dados diferentes, por exemplo) a sua aparência torna-se numa linha a tracejado negro espesso com um "X" vermelho no meio.

Na Fig. 4.4 os fios ligam dois algarismos a um nodo (objecto ou função) de soma que, por sua vez, é ligado a um terminal de indicação. Como já foi referido, o terminal de indicação liga o diagrama de blocos ao painel frontal onde é apresentado o resultado do programa. Como se pode verificar, a cor dos fios é azul, o que significa que os dados neles transportados são números inteiros.

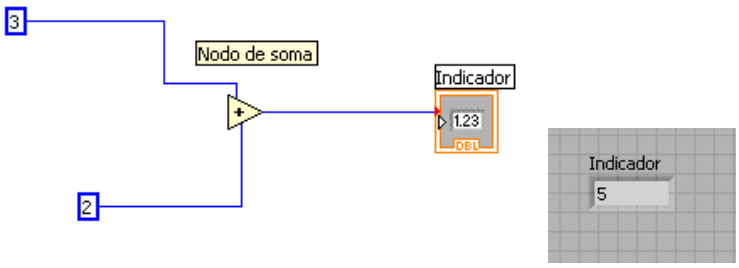


Figura 4.4: À esquerda: programa que executa a soma de dois algarismos; à direita: apresentação do resultado no painel frontal.

Nodos

Os *nodos* são objectos do bloco de diagramas, com números variáveis de entradas e de saídas ("inputs" e "outputs"), cuja função consiste em realizar certas operações enquanto o IV corre. São o análogo gráfico dos operadores lógicos e matemáticos encontrados nas linguagens de programação baseadas em texto (Fig. 4.5).

Figura 4.5: Alguns exemplos de nodos.

Estruturas

As *estruturas* são o equivalente gráfico dos ciclos (para efectuar repetições) nas linguagens de programação baseadas em texto. São usadas no bloco de diagramas para repetir a

execução de certos blocos de código ou para executar código sob determinadas condições ou numa ordem específica.

Na Fig. 4.6 podemos observar as estruturas equivalentes a um ciclo do tipo "for" e a um ciclo do tipo "do-while" que tipicamente existem noutras linguagens. O "quadrado" à esquerda é denominado "for-loop" e habitualmente é ligado ao seu terminal *N* (canto superior esquerdo do bloco) um algarismo. Esse algarismo indica o número de iterações que o ciclo vai realizar. Neste caso o ciclo vai ser executado 10 vezes, ou seja, um número aleatório entre 0 e 1 vai ser somado ao número 5 criando dez números aleatórios. O resultado de cada iteração é guardado em memória e, no final, os 10 algarismos são simultaneamente enviados para um vector ("vector de indicação" fora de ambos os ciclos), apresentados no painel frontal e injectados numa estrutura "while-loop". Dentro dessa estrutura os números vão ser somados e o resultado também é apresentado no ecrã. Este ciclo vai funcionar ininterruptamente até o botão de *STOP* ser pressionado.

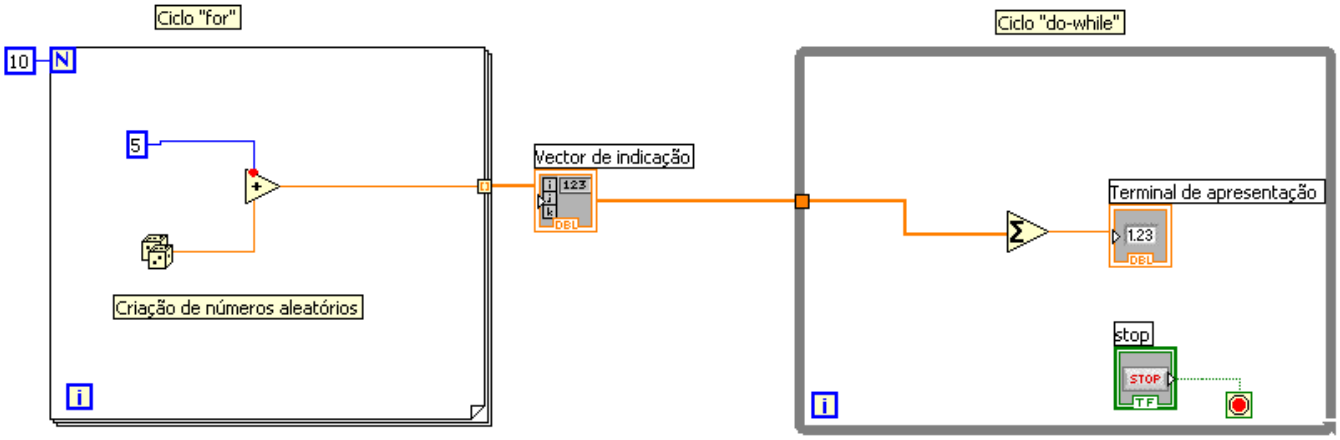


Figura 4.6: Diagrama de blocos que exemplifica a utilização de dois tipos de estruturas iterativas.

Menus

O menu de controlo é somente acessível no painel frontal. Este menu contém os já referidos controlos e os menus que permitem realizar a interface de dados com o utilizador. Estes estão organizados em submenus (Fig. 4.7).

O menu de funções é acessível pelo ambiente dos diagramas de blocos (ou janela de programação). Neste menu estão disponíveis os objectos de programação. Existem funções dedicadas a inúmeras aplicações: programação geral, realização de operações matemáticas e estatísticas, processamento de sinal, etc... (Fig. 4.8).

O menu de ferramentas inclui os modos de utilização do cursor e a opção que possibilita o uso dos fios que ligam os objectos e é acessível tanto na janela de programação como no painel frontal.

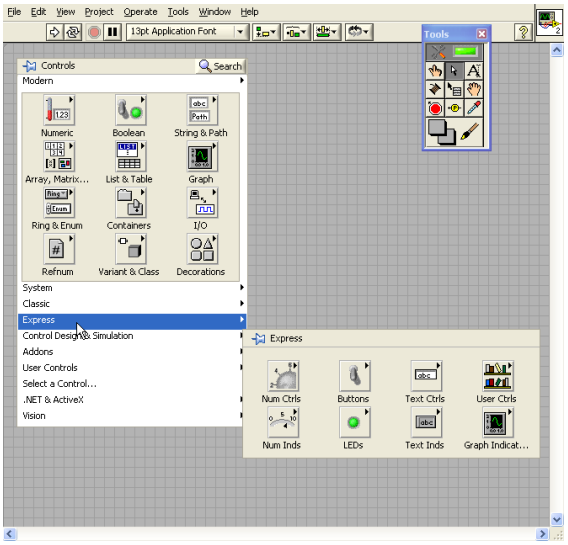


Figura 4.7: Menu de controlo com o menu de ferramentas no canto superior direito.

Gráficos

A apresentação gráfica dos dados no LabVIEW pode ser feita em "Graphs" ou em "Charts". Enquanto que no uso do "Graph" os dados são agrupados num vector e só depois são apresentados visualmente, quando o "Chart" é usado os dados são apresentados no instante em que são criados.

Vectores

Um vector é constituído por elementos e pode ter várias dimensões (i.e., pode servir para criar matrizes). Em LabVIEW os vectores podem ter até $2^{31} - 1$ elementos por dimensão e esses elementos podem ser dos tipos numérico, Booleano, "string" ou servir para armazenar funções de onda. Os vectores são geralmente usados para guardar dados resultantes da execução de ciclos, em que em cada iteração do ciclo se guarda um (ou mais) elemento do vector.

Não é possível criar "vectores de vectores", mas é possível criar "clusters" (agrupamentos) de vectores, onde cada "cluster" contém um ou mais vectores (ver a próxima secção).

O LabVIEW tem um sub-menu (Fig. 4.9) dedicado somente à criação e à manipulação de vectores. Por exemplo, pode-se extrair um determinado elemento de um vector, inserir ou apagar uma linha, coluna ou elemento.

Existe também um menu dedicado à criação e manipulação de "strings"; uma string é uma sequência de caracteres ASCII que permite criar mensagens de texto. Pode-se converter dados numéricos em strings de caracteres e vice-versa.

Clusters (agrupamentos)

Enquanto que num vector só podemos inserir dados de um mesmo tipo, num cluster os dados podem ser de diversos tipos. O cluster é o equivalente a uma estrutura das lingua-

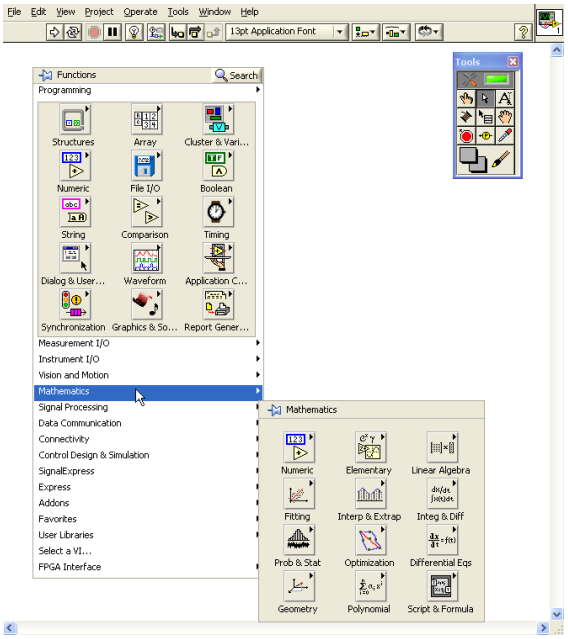


Figura 4.8: Menu de funções.

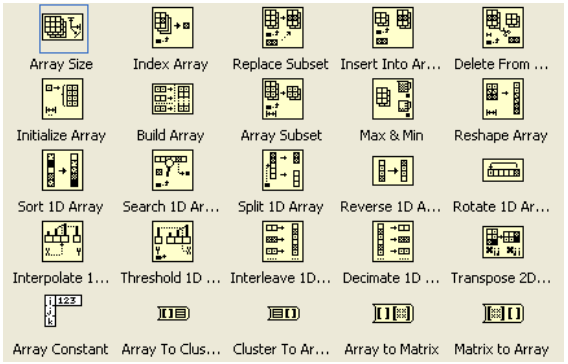


Figura 4.9: Sub-menu que agrupa as funções de manipulação de vectores.

gens de programação comuns. Têm uma utilização mais rebuscada comparativamente aos vectores, pois para trabalhar com um tipo de dados é, muitas vezes, necessário separá-lo do cluster e depois reagrupá-lo de novo.

Na Fig. 4.10 exemplifica-se a criação de um cluster com dois componentes: um vector constituído por dez números aleatórios e uma string contendo uma frase. O ícone à direita acede ao comando de criação de um cluster. Como se pode ver, os dois tipos de dados recebidos pelo cluster são diferentes (o fio proveniente da string é cor-de-rosa e aquele proveniente do vector é cor-de-laranja).

Ficheiros

O LabVIEW possui também uma extensa lista de funções para a manipulação de ficheiros. Elas possibilitam ler e escrever em ficheiros de dados ou em folhas de cálculo, criar ou renomear ficheiros ou alterar as suas características.

Na Fig. 4.11 ilustra-se a criação de um vector com um milhão de números aleatórios.

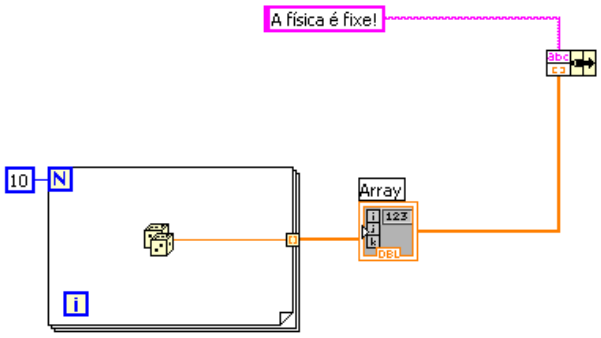


Figura 4.10: Exemplo da criação de um cluster.

Esses números são enviados para a sub-IV `writetospreadsheetfile.vi` onde são gravados no ficheiro indicado pela localização referida acima, na caixa de texto.

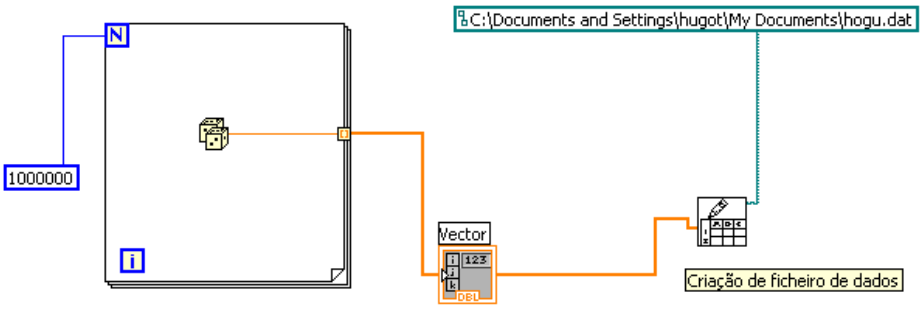


Figura 4.11: Exemplo da criação de um ficheiro de dados.

LabVIEW MathScript

Uma das utilidades do LabVIEW que o "aproxima" das outras linguagens de programação é o MathScript. O MathScript é uma linguagem de programação, optimizada para aplicações que façam uso intensivo de algoritmos, incluida no LabVIEW, e que permite escrever código em texto que pode ser usado dentro dos IVs. O utilizador pode, dentro de uma janela de MathScript, executar operações matemáticas, criar programas e até ver representações gráficas das variáveis. Na Fig. 4.12 está um exemplo de um programa complexo escrito em LabVIEW MathScript.

A sintaxe do MathScript é semelhante à de outros programas matemáticos, nomeadamente o MATLAB, e para ser inserido um bloco de MathScript num IV, basta ligá-lo a fios que transportem dados do tipo numérico adequados ao processamento que se pretende efectuar, tanto para as entradas como para as saídas de dados.

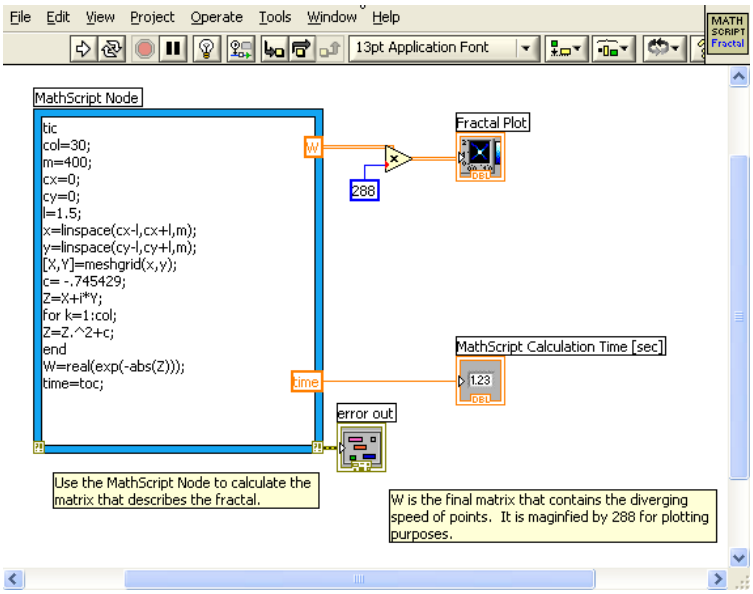


Figura 4.12: Um dos exemplos da National Instruments que cria um fractal: *MathScript Fractal.vi*.

4.2 NI Vision

O Módulo de Desenvolvimento de Visão denominado *NI Vision* é o conjunto de software desenvolvido pela National Intruments especialmente direccionado ao processamento digital de imagens. Este módulo inclui um ambiente interactivo de trabalho (Fig. 4.13) e bibliotecas de funções que permitem que o programa criado pelo utilizador possa ser desenvolvido concorrentemente em LabVIEW, em C e em Microsoft Visual Basic.

Este módulo permite executar, através de funções previamente desenvolvidas (nativas do NI Vision), a maior parte dos algoritmos de processamento de imagem referidos nas secções anteriores (através de menus bastante intuitivos, onde o utilizador dispõe de liberdade para variar diversos pârametros, como se pode ver na Fig. 4.14), sobre imagens fixas (fotos) ou sobre filmes, com a criação de *scripts*. Um script é um conjunto de funções de processamento de imagem em cadeia (Fig. 4.15). Para permitir o melhoramento dos programas criados pelos utilizadores, o NI Vision permite que o script seja transformado numa aplicação LabVIEW ou, então, gera automaticamente as correspondentes linhas de código em C ou VB.

Duas das mais interessantes ferramentas que o NI Vision disponibiliza são a *Calibração Espacial* e a *Análise de Partículas*.

A Calibração Espacial é extremamente importante nos casos em que é necessário fazer medições rigorosas numa imagem, e é um processo que permite transformar unidades de pixel em unidades SI levando em conta os erros inerentes a um estúdio de aquisição de imagens. Ou seja, dado que a imagem contém informação em pixels é possível converter "distâncias" especificadas nesta unidade para milímetros, por exemplo. A calibração faz-se seleccionando na imagem um determinado comprimento e fornecendo ao programa o seu equivalente "real". Também é possível definir um sistema de coordenadas que permite eliminar ângulos de perspectiva.

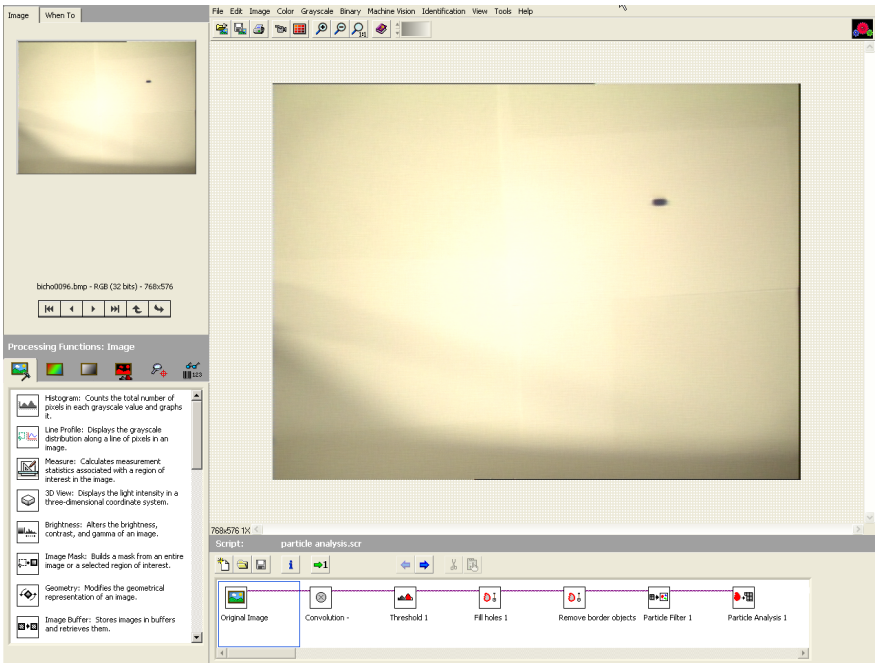


Figura 4.13: Ambiente de trabalho do NI Vision 8.0.

A definição de unidades "reais" permite-nos fazer *Análise de Partículas*, pois o NI Vision providencia um extenso conjunto de comandos que permitem fazer diversas medições em imagens binárias como, por exemplo, calcular o centro de massa, o perímetro ou área total dos pixels que constituem as partículas (ou objectos componentes) das imagens.

4.3 Conclusão

O objectivo deste capítulo foi o de dar uma panorâmica geral do funcionamento do LabVIEW, um ambiente interactivo focado na rápida criação de programas vocacionados, sobretudo, para a aquisição e análise de dados, e da forma como se podem construir programas utilizando métodos gráficos de programação. Também foi analisado o módulo de desenvolvimento de visão denominado *NI Vision*, o software desenvolvido pela National Intruments especialmente direccionado ao processamento digital de imagens.

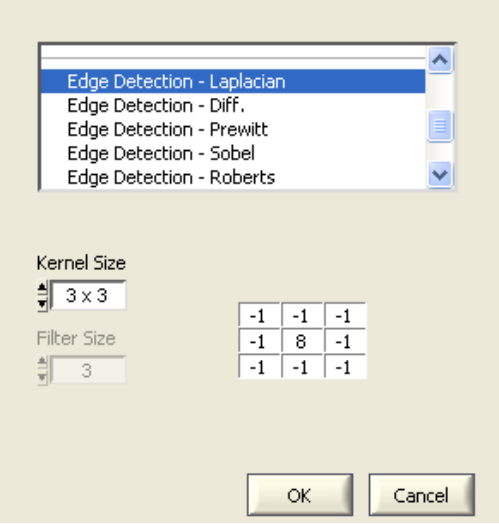


Figura 4.14: Um dos menus do NI Vision.

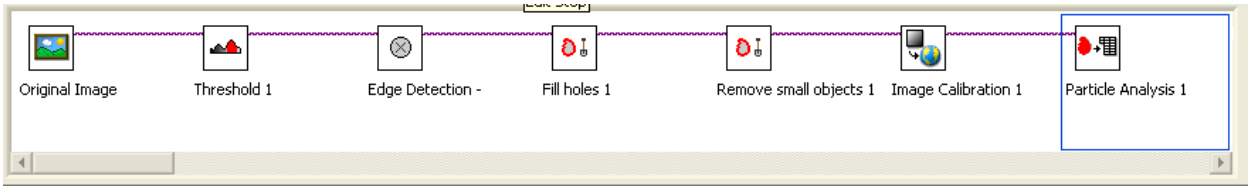


Figura 4.15: Exemplo de um script do NI Vision.

Capítulo 5

Implementação do sistema

5.1 Introdução

Após terem sido expostas várias técnicas utilizadas em processamento de imagem (capítulo 2), nomeadamente aquelas com mais interesse para o presente trabalho, e ter sido descrito o sistema onde o trabalho foi implementado, o LabVIEW (capítulo 4), neste capítulo descreve-se, com algum detalhe, a implementação do sistema de monitorização do movimento de animais. O sistema não está limitado a animais, podendo ser aplicado à monitorização das deslocações de outros objectos numa imagem com fundo fixo.

Ficou claro nas discussões anteriores que a necessidade de se efectuar o processamento de imagem em tempo real acarreta uma série de limitações aos algoritmos que podem ser implementados. Com efeito, para tentar manter um ritmo de aquisição de imagens na ordem de 10 Hz, não se pode aplicar algoritmos de reconhecimento de formas muito elaborados, porque o respectivo tempo de execução obrigaria a que fosse usada uma taxa de aquisições demasiado baixa para poder ser útil.

Por outro lado, os algoritmos simples e rápidos implementados quando se pretende fazer processamento em tempo real serão pouco robustos (pelo menos, serão muito menos robustos que outros mais complexos) na medida em que serão mais "facilmente" enganados por reflexos, variações de luminosidade, de pano de fundo, etc.

Assim, optou-se por implementar em paralelo dois sistemas que, ao invés de "competirem" entre si, poderão complementar-se. O primeiro sistema destina-se a ser executado em tempo real e emprega um conjunto de algoritmos rápidos e pouco sofisticados. O segundo, destina-se a processar conjuntos de imagens sequenciais armazenadas (isto é, "filmes"), e neste caso não foi restringido o tempo de execução dos algoritmos.

A disponibilidade simultânea destas duas ferramentas permite fazer um estudo das situações em que um se revela mais útil que o outro e vice-versa.

5.2 Equipamento do Sistema de Visão

O sistema de processamento foi realizado com as ferramentas LabVIEW 8.2 e NI Vision 8.0, da National Instruments, executadas em Windows XP instalado numa plataforma Pentium 5 Dual Core com 1 Gb de RAM. Este sistema é um computador "médio" à presente data (2007) mas mostrou-se eficiente para a realização do trabalho.

Para efectuar a aquisição de imagens registadas pela câmara de filmar digital foi instalada uma placa interna IMAQ PCI-1405, também da National Instruments, (Fig. 5.1). Na Fig. 5.2 encontra-se um esquema que ilustra a maioria das suas potencialidades de aquisição de imagem.

Figura 5.1: Imagem da IMAQ PCI-1405.

Esta placa permite adquirir imagens em tempo real, tanto a cores como monocromáticas, com uma frequência de 25 Hz (ou inferior), em modo PAL. Também foi instalado o módulo de software NI - IMAQ 3.7, que contém as rotinas para configurar a aquisição de imagens, a alocação de memória, o controlo do "trigger" (disparo) da câmara e as configurações da interface com o utilizador. A câmara utilizada neste trabalho foi uma *Sony DXC-101P*. É uma camara CCD que adquire imagens a cores e permite ajustar os níveis de brilho da imagem. A objectiva utilizada na câmara foi uma *Ernitec Auto Iris* de 16 mm com foco de 1,3.

5.3 Processamento de imagem em tempo real

Como foi já referido na secção sobre monitorização automática, existem três métodos diferentes para tratar imagens com objectos em movimento. Um deles, o método da subtracção de imagens, dificulta a visualização de animais imóveis (o que se pode revelar extremamente provável para certas espécies). O método de subtracção de fundo apresenta problemas quando os animais alteram o ambiente em que estão inseridos. Optou-se, por isso, por assentar o trabalho no método do ajuste de contraste. Este método permite uma fácil e rápida

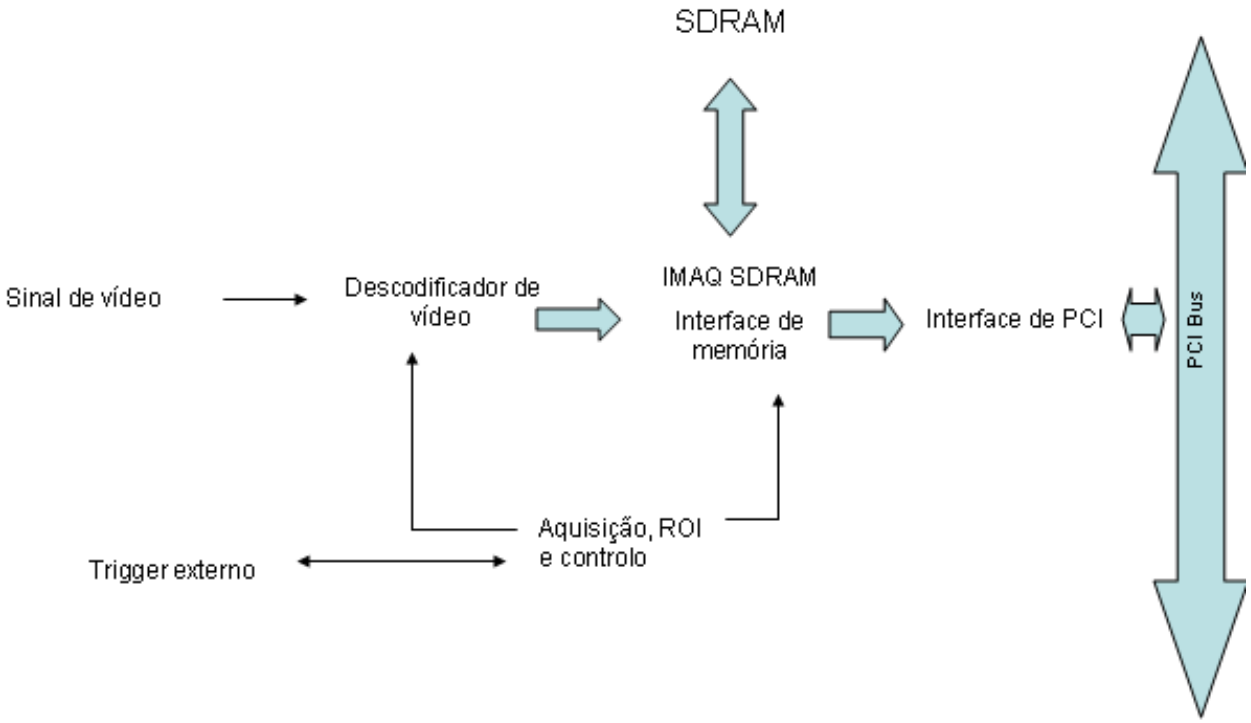


Figura 5.2: Diagrama que ilustra a funcionalidade da IMAQ 1405. Depois de recebido, o sinal de vídeo vai ser transferido para um descodificador. A placa tem 16 MB de memória dinâmica RAM (SDRAM), que permite gravar as imagens antes de serem transferidas para a memória principal (no PC). A imagem, antes de ser passada para a memória, pode ser sujeita à selecção de uma ROI ("Region Of Interest"). A interface da placa permite transferências para o PC a uma taxa máxima de 132 Mbytes/s.

alteração dos parâmetros sob estudo, em cenários variados.

Como já foi referido, a escolha dos algoritmos a serem implementados é dependente do facto de, eventualmente, o processamento da imagem ser feito em tempo real. O processamento é efectuado imagem a imagem, ou seja, é adquirida uma imagem, são executados os algoritmos de segmentação necessários e localizado o animal (ou animais). Depois, são gravados os dados pretendidos e é adquirida uma nova imagem, descartando-se a anterior. É necessário que exista um equilíbrio entre a complexidade do processamento e a velocidade de aquisição, ou seja, a complexidade tem de ser reduzida estritamente ao mínimo de modo a que a aquisição de imagens seja a mais rápida possível. Por isso, algoritmos que recorram a informação no espaço das frequências, apesar de eficazes, não podem ser utilizados devido ao facto de a transformada da imagem ser calculada com uma reduzida velocidade, mesmo usando a FFT.

5.3.1 Segmentação de imagem

A explicação do "programa" de detecção e seguimento ("tracking") de formas, que é usado para realizar o processamento em tempo real, é exemplificada com imagens da movi-

mentação de um bicho de conta sobre um fundo branco, filmadas com a câmara numa posição perpendicular ao plano do movimento e com a lente colocada a cerca de 0,75 m. A focagem foi feita manualmente, tendo sido regulada de modo a que a nitidez da imagem fosse optimizada (Fig. 5.3).

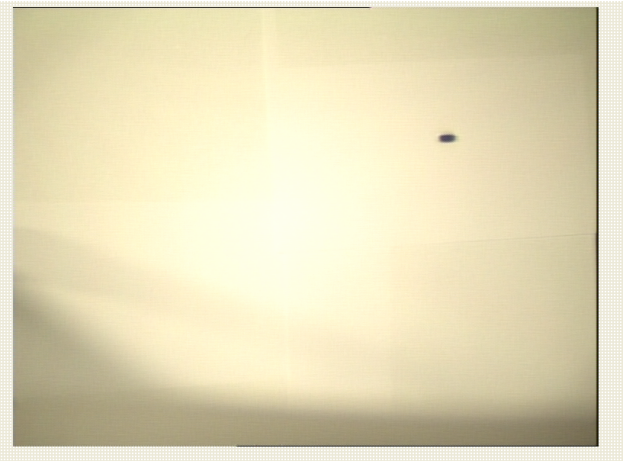


Figura 5.3: Imagem de um bicho de conta sobre um fundo branco. O processamento desta imagem será usado como exemplo.

O conjunto câmara-placa de aquisição aqui utilizado permite adquirir imagens RGB de 32 bits com uma resolução de 756×576 pixels. Contudo, os métodos necessários para o processamento em tempo real da imagem obrigam a que esta seja descrita numa escala de cinzentos (como já foi explicado, as imagens a cor são descritas por três planos "de cinzento" enquanto que as imagens cinzentas têm só um plano, o que possibilita a rápida execução dos algoritmos, pois apenas um plano tem de ser processado). A ferramenta NI Vision permite extrair um plano de cor da imagem colorida e descartar os outros dois planos. Assim, a imagem resultante será descrita numa escala de cinzentos. Esta operação será parametrizada no sistema, podendo-se escolher o plano (R, ou G, ou B) que mais se aproxime da imagem original.

Por análise da Fig. 5.4-a podemos observar que os planos têm histogramas muito semelhantes, o que seria de esperar, pois a maior parte de imagem é branca. Seleccionou-se aleatoriamente o plano vermelho e na Fig. 5.5-b apresenta-se a imagem-resultado após efectuar esta operação.

O passo de processamento seguinte consiste em detectar todas as arestas existentes na imagem, com o auxílio dos *Operadores de Sobel* vertical e horizontal (ver capítulo 2):

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{5.1}$$

Dado que as máscaras são mais eficazes segundo uma determinada direcção (horizontal ou vertical), ambas vão ser aplicadas ao pixel central, cujo valor vai ser o resultado da

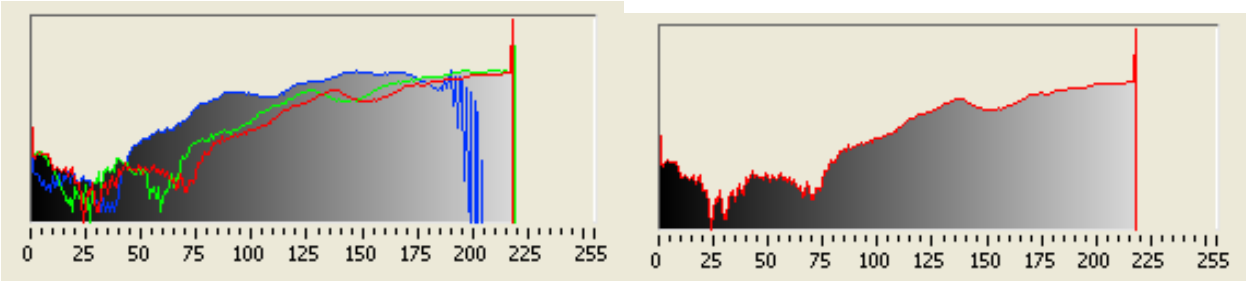


Figura 5.4: (a)- Histograma da imagem original. (b)- Histograma da imagem correspondente ao plano vermelho.

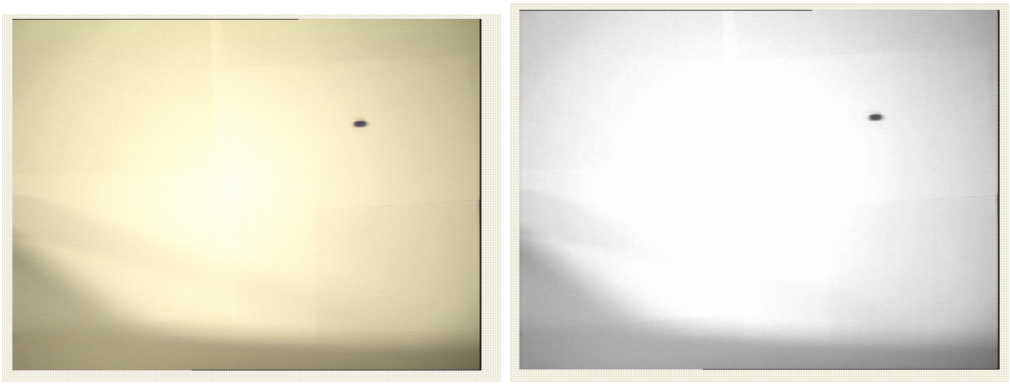


Figura 5.5: (a)- Imagem original. (b)- Imagem com o plano vermelho convertido para uma escala de cinzentos.

operação:

$$P_{(i,j)} = \max\Big[|P_{(i+1,j-1)} - P_{(i-1,j-1)} + 2P_{(i+1,j)} - 2P_{(i-1,j)} + P_{(i+1,j+1)} + P_{(i-1,j+1)}|,$$
$$|P_{(i-1,j+1)} - P_{(i-1,j-1)} + 2P_{(i,j+1)} - 2P_{(i,j-1)} + P_{(i+1,j+1)} + P_{(i+1,j-1)}|\Big]$$

A imagem resultante da sua aplicação é mostrada na Fig. 5.6.

Seguidamente ajustou-se o contraste da imagem, seleccionando um intervalo de valores de cinzento[1] e fazendo com que todos os pixels cujo valor estivesse incluído nesse intervalo ficassem com o valor de 1. Todos os outros pixels são colocados com o valor 0. Esta operação de "thresholding" transforma a imagem cinzenta numa imagem binária (a preto-e-branco). O referido intervalo de valores também é parametrizado na operação, para permitir ao utilizador escolher aquele que melhor se adapta à imagem a ser processada.

Neste caso, por análise do histograma da imagem resultante, concluiu-se que um valor limite adequado seria:

- pixels com valor ≤ 20 ficam com valor 0;
- pixels com valor > 20 ficam com valor 1.

A Fig. 5.7 mostra o resultado final desta operação, onde se pode observar o animal (insecto) já isolado.

[1] A escala de cinzentos varia de 0 a 255.

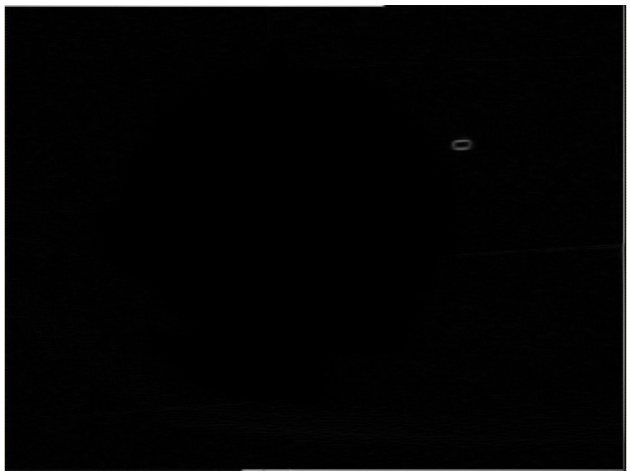


Figura 5.6: Imagem após a aplicação do Laplaciano.

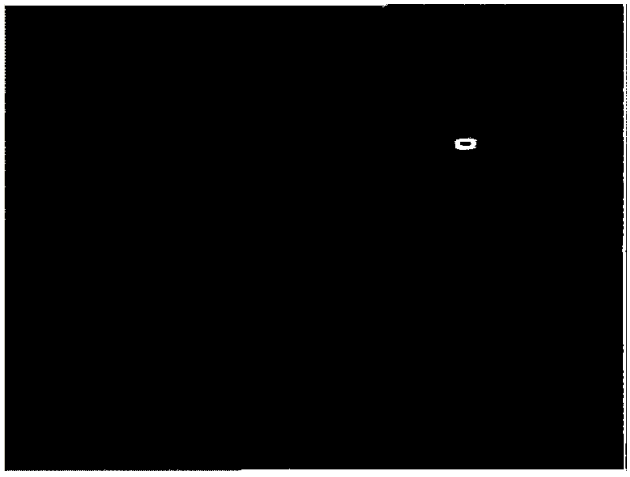


Figura 5.7: Imagem resultante do ajuste de contraste. A imagem tem 4347 pixels com o valor 1 e os restantes 438021 com o valor 0.

A espessura das arestas existentes na imagem pode ser facilmente aumentada, para que se tornem mais visíveis e para que eventuais pixels isolados possam ser removidos pela aplicação de um processo de erosão (Fig. 5.8) cujo elemento estruturante é, no caso em questão, 3×3:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{5.2}$$

Isto permite que todos os pontos na fronteira da imagem fiquem unidos para que se torne mais fácil a sua eliminação, ou seja, todas as regiões que toquem nas extremidades do plano que constitui a imagem são definidas com o valor 0. Este é o último passo de processamento, após o qual se pode verificar que a única partícula existente é o bicho-de-conta (Fig. 5.9) (é vísivel a eliminação dos pontos brancos na extremidade da Fig. 5.8).

Os passos finais do processo envolvem a calibração da imagem para que os resultados posicionais obtidos posteriormente sejam registados em unidades SI. O NI Vision facilita

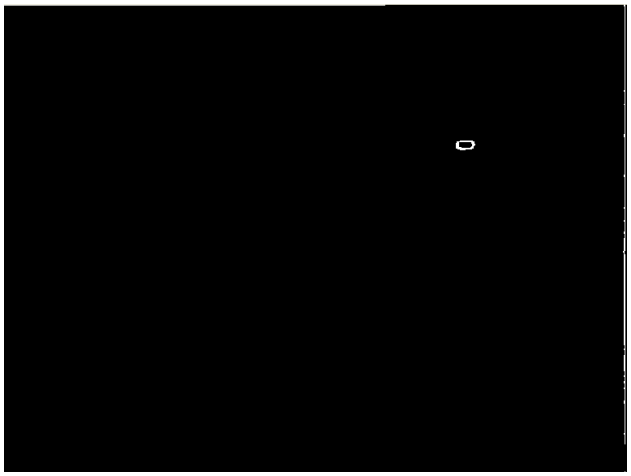


Figura 5.8: Imagem submetida ao processo da erosão.

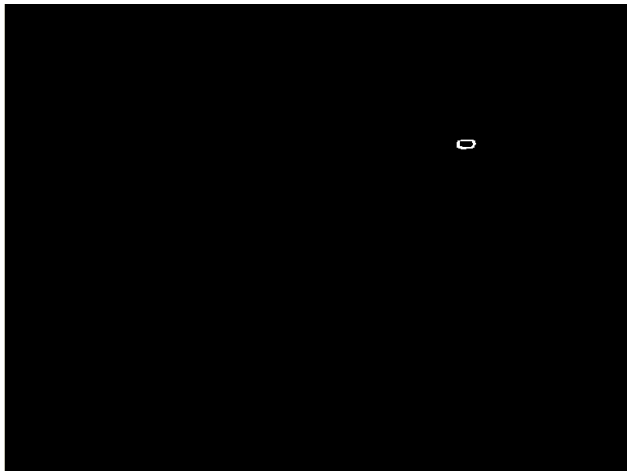


Figura 5.9: Imagem resultante da aplicação dos vários passos de processamento que levam à segmentação do animal.

essa tarefa com a função *Image Calibration* e, nesta imagem, sabe-se que 1 pixel = 0,51428 mm.

A posição do bicho-de-conta na imagem é dada pela localização do seu centro de massa (CM), definido como o ponto que representa a posição média da massa total da partícula.

Assumindo que todos os pontos da partícula têm densidade constante, as coordenadas do CM são calculadas pelas funções $\frac{\sum_x}{A}$ e $\frac{\sum_y}{A}$, onde A é a área da partícula. A área pode ser calculada aproximadamente assumindo que a partícula tem uma forma elipsoidal. A área de uma elipse é dada por:

$$Area_{elipse} = \pi ab, a = E_{2a}/2, b = E_{2b}/2 \tag{5.3}$$

onde E é definido na Fig. 5.10.

Sendo assim, neste exemplo a posição do bicho-de-conta num referencial (x, y) onde o ponto $x = 0$, $y = 0$ corresponde ao vértice superior esquerdo da imagem, tem as coordenadas $x = 291.7092$ e $y = 88.1890$.

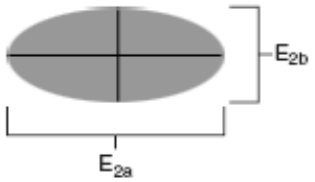


Figura 5.10: Elipse.

O ângulo do animal neste caso é de cerca de 5º. O NI Vision mede os graus de rotação ($0 \leq \theta \leq 180$) no sentido directo (contrário ao do andamento dos ponteiros do relógio) a partir do eixo x.

5.3.2 O sistema de processamento em tempo real

Passa-se agora a descrever a montagem de um sistema de processamento de imagem em tempo real. Para tal é criado um sub-IV (capítulo 4) que realiza as operações já descritas anteriormente (Fig. 5.11) mas agora incorporadas num processo de aquisição de imagem.

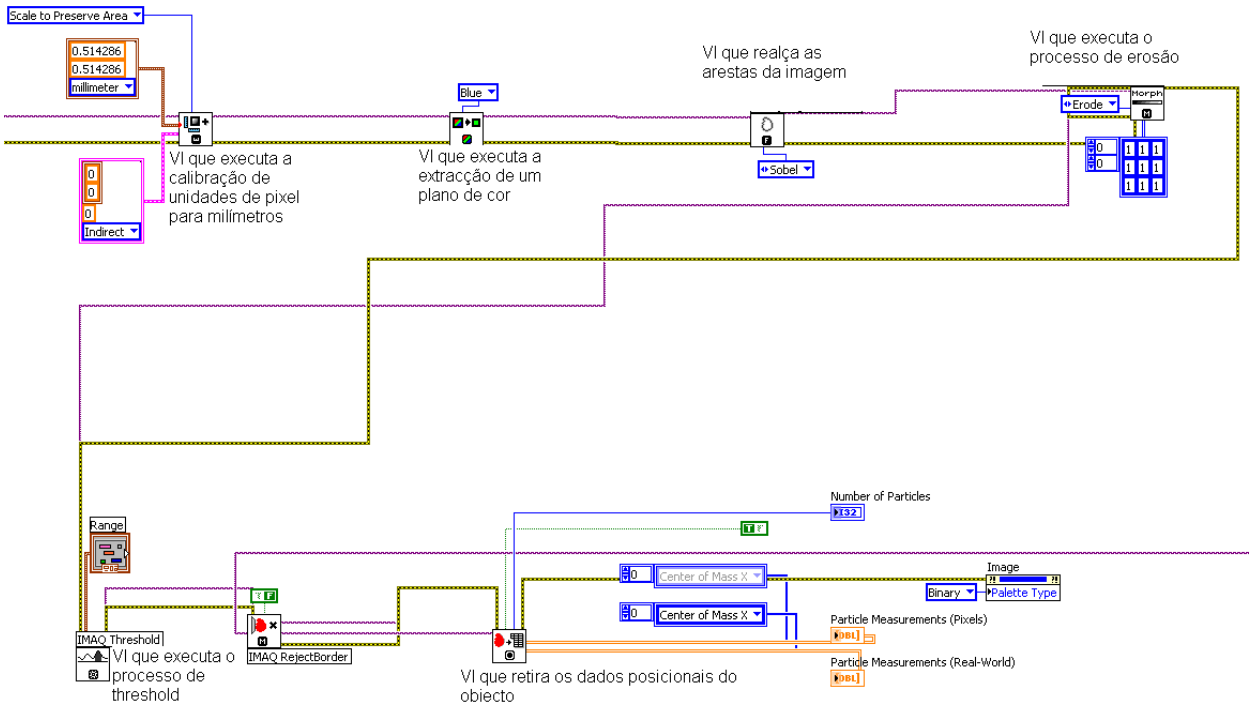


Figura 5.11: Sub-IV contendo os algoritmos de segmentação.

Para inicializar a aquisição de imagens é necessário executar as seguintes funções:

- *IMAQ Init* - Configura o sistema de aquisição;
- *IMAQ Create* - Cria a imagem;
- *IMAQ Grab Setup* - Inicializa o sistema de aquisição contínua de imagens.

Os "fios" (ou ligações) "transportam" a imagem até uma estrutura cíclica `do-while` que contém os sub-VIs que implementam os algoritmos de processamento. Este ciclo permite que a aquisição possa ser interrompida quando o utilizador quiser. A função *IMAQ Grab* "importa" a imagem proveniente do hardware de visão (de cada vez que o ciclo é executado é adquirida uma nova imagem).

Os fios que transportam a imagem são ligados ao sub-IV onde é feita a segmentação da imagem. O resultado da segmentação é um vector que contém a posição e o ângulo do objecto (ou partícula, em terminologia de processamento de imagem). Esse vector pode ser guardado num ficheiro de dados e posteriormente analisado.

Uma parte do IV é mostrada na Fig. 5.12. Os elementos do vector que contêm os dados da partícula são extraídos e ligados a uma função que permite desenhar o gráfico da sua posição em tempo real e simultaneamente a um bloco de *MathScript* onde são executados os cálculos que permitem saber, em tempo real também, a velocidade instantânea da partícula e a distância percorrida até então. Este valor foi utilizado para o cálculo da velocidade do animal. Este sistema consegue processar cerca de 10 imagens por segundo.

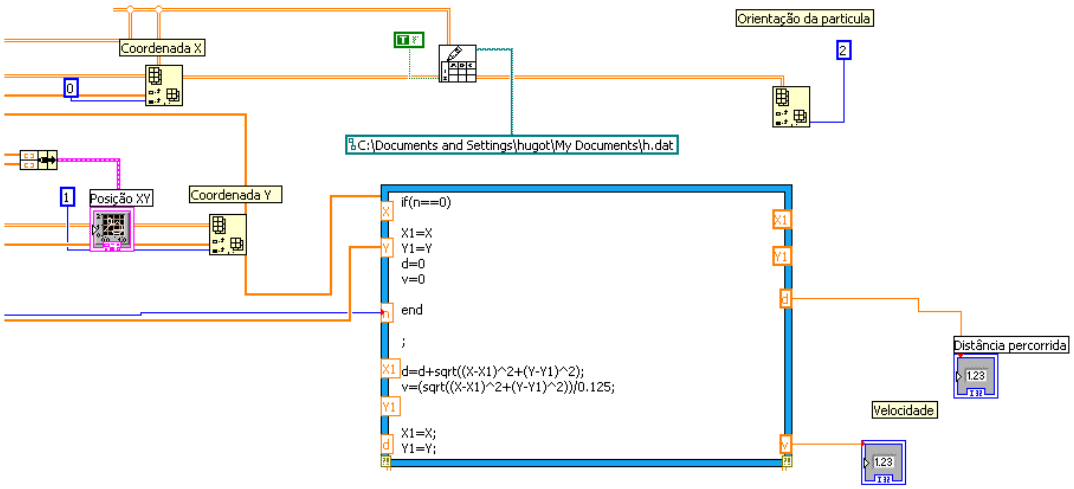


Figura 5.12: Sub-IV que contém os algoritmos de localização de uma partícula numa imagem.

Caso existam dois animais a interagir, é utilizado o algoritmo que permite que a posição de um animal seja associada às suas posições seguintes (capítulo 2), para que os dados referentes a cada um deles não sejam trocados. Este algoritmo foi implementado num bloco de MathScript que retorna os dados referentes a cada animal, ou seja, que torna possível que a posição do animal 1 não seja confundida com a do animal 2, mesmo que se cruzem no intervalo de tempo entre a aquisição de duas imagens sucessivas. Os dados dos animais podem ser observados de imediato ou gravados em ficheiros de dados. É também construído um gráfico com as posições das partículas.

5.4 Processamento de imagens arquivadas

A preocupação com a frequência de aquisição de imagens deixa de fazer sentido caso se pretenda processar um conjunto de imagens já adquiridas "off-line". Neste caso não há uma restrição tão apertada ao uso de algoritmos de processamento mais lentos. Obviamente, o uso destes algoritmos só será justificado caso a informação obtida justifique a penalização em tempo.

5.4.1 O sistema de processamento "off-line"

A gravação de filmes pode ser de grande utilidade na observação animal, até em ambientes não laboratoriais. Um sistema de monitorização automática que actue sobre estas gravações pode permitir a observação de diversos comportamentos ou mesmo a descoberta de novos padrões de comportamento não observáveis facilmente, em tempo real, por um observador humano.

O sistema de observação criado para este caso foi idealizado para trabalhar com imagens a cores sem qualquer filtro aplicado (espacial ou na frequência), utilizando as funções do LabVIEW ligadas ao reconhecimento de padrões e de cores. Os filmes deverão ser armazenados no formato AVI[2](o único formato vídeo que o LabVIEW processa) e não existe nenhum limite para a sua duração (desde que exista espaço no disco do computador).

A primeira acção implementada no sistema consiste em pedir ao utilizador que seleccione o filme. A função *File Dialog* abre uma caixa de diálogo onde o utilizador pode especificar a localização do ficheiro de vídeo. Depois de definida esta localização, a função *IMAQ AVI Open* disponibiliza o filme para ser manipulado pelo IV. O *IMAQ AVI Get Info* retira as informações sobre o filme que vão ser necessárias mais tarde, tais como o número de quadros (*frames*), o filtro de compressão utilizado para criar o filme AVI, a altura e a largura das imagens e outros dados associados ao ficheiro. Temos, então, toda a informação necessária para começar a processar o filme, quadro a quadro.

Aquela informação é transmitida para uma estrutura sequencial (*Stacked Sequence Structure*), uma ferramenta que permite que os sub-VIs criados sejam executados numa ordem pré-determinada. O IV é constituído por três sequências (o sistema vai ser ilustrado com imagens retiradas de um filme de um *crash test* com o simples propósito de demonstrar a versatilidade do programa que foi desenvolvido):

1. A forma escolhida para começar o processamento da imagem é deixar ao critério do utilizador a escolha do objecto que pretende seguir. Neste sub-IV o utilizador selecciona um ROI[3] na imagem, que vai ser o padrão (*template*) que o sistema vai tentar seguir (Fig. 5.13).

   A imagem é transferida para a função *IMAQ AVI Read Frame*, o que permite seleccionar o quadro que se quer analisar. O quadro 0 (quadro inicial) é a escolha óbvia

[2]Audio Video Interleave.
[3]Region Of Interest.

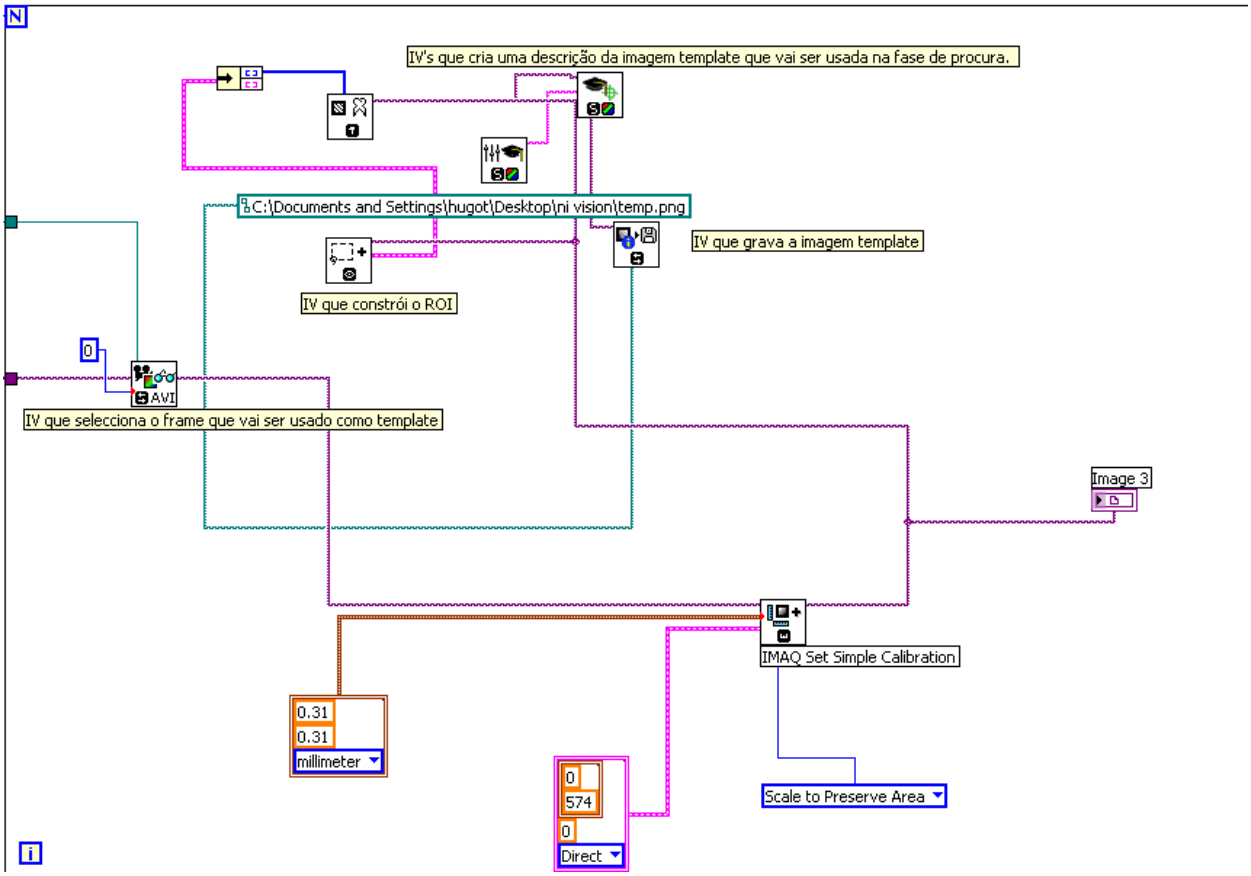


Figura 5.13: Código que cria a imagem padrão (*template*).

neste caso. Este quadro é depois transferido para a função *IMAQ CostructROI*, que apresenta um ecrã com a imagem e que dispõe de uma série de ferramentas que o utilizador pode usar para seleccionar a forma geométrica da ROI (rectangular, circular ou definida por um conjunto de linhas que permitem recortar um objecto do fundo). Na Fig. 5.14 o utilizador escolheu como padrão a seguir no filme o logotipo visível na camisola do "boneco" (*dummy*) do *crash test*. O quadro vermelho visível na imagem é um cronómetro.

Os dados extraídos do ROI são utilizados para criar um padrão de imagem para seguimento. O *IMAQ Write Image And Vision Info* grava a imagem como um ficheiro PNG na localização definida pelo utilizador, o que permite que a imagem possa ser usada posteriormente sempre que seja necessário.

2. A imagem padrão é reinserida no sistema pela função *IMAQ Read Image And Vision Info* e, deste modo, pode começar a ser feita a localização do objecto nos diversos quadros do filme. O módulo de visão do *LabVIEW* é (supostamente) suficientemente robusto para permitir localizações correctas mesmo com variações de intensidade luminosa, desfocagens, ruído (desde que as alterações introduzidas não sejam muito severas) ou transformações geométricas, tais como rotações ou translações, combinando a localização de cores com a técnica de correlação-cruzada normalizada aplicada a

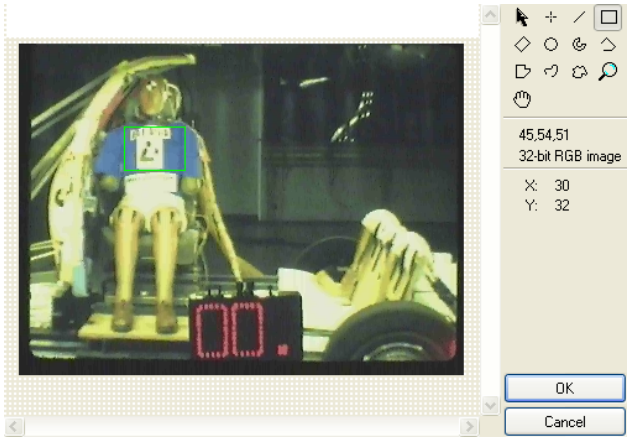


Figura 5.14: Selecção de um objecto (a verde) a ser seguido num filme.

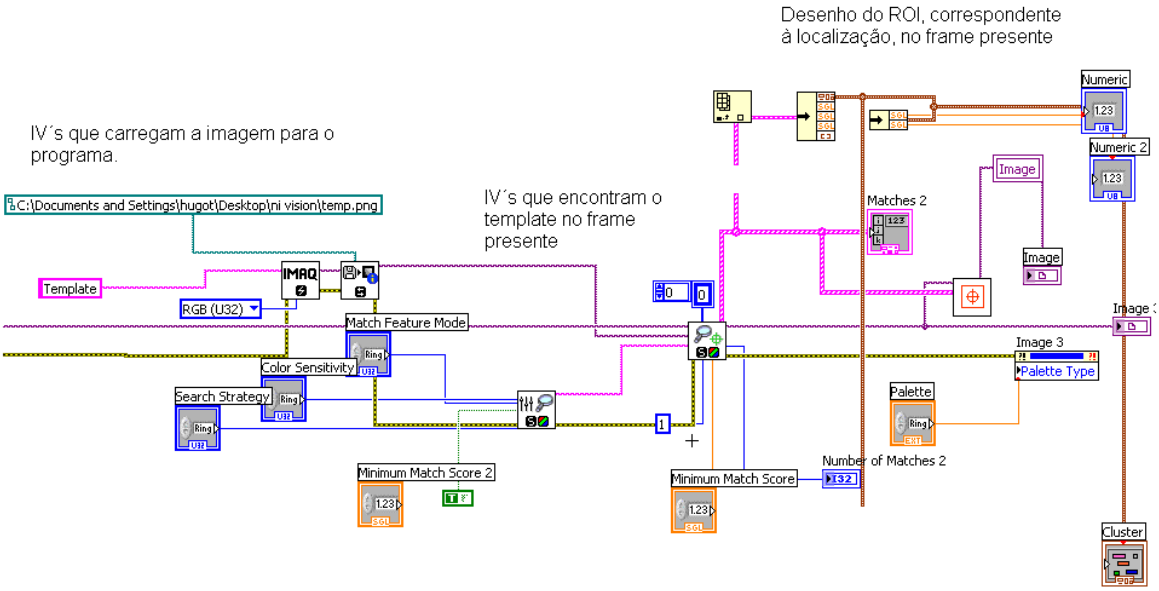


Figura 5.15: Código em que se faz o reconhecimento dos objectos.

imagens descritas numa escala de cinzentos. Ao utilizar as funções de localização de cores o *LabVIEW* sobrepõe a imagem padrão à imagem correntemente a ser analisada e procura semelhanças nos padrões de cor (Fig. 5.15). O reconhecimento de cores é feito no espaço de cores HSL, razão pela qual as cores são transformadas de RGB para HSL segundo as fórmulas:

$$
\begin{aligned}
V2 &= \sqrt{3}(G - B) \\
V1 &= 2R - G - B \\
L &= 0.299R + 0.587G + 0.114B \\
H &= 256tan^{-1}(V2/V1)/(2\pi) \\
S &= 255(1 - 3min(R, G, B)/(R + G + B))
\end{aligned} \tag{5.4}
$$

Deste modo, são comparados os histogramas de cores da imagem padrão com regiões da imagem a reconhecer.

Seguidamente, às regiões com maior número de sucessos é aplicada a técnica da correlação-cruzada normalizada para imagens na escala de cinzentos para se tentar descobrir a posição exacta do objecto.

A técnica de correlação-cruzada normalizada consiste em sobrepor uma sub-imagem $w(x, y)$, de dimensões $K \times L$, no interior de uma imagem $f(x, y)$, de dimensões $M \times N$, onde $K \leq M$ e $L \leq N$. A correlação entre $w(x, y)$ e $f(x, y)$, centrada no ponto (i, j), é dada por:

$$C(i,j) = \sum_{x=0}^{L-1} \sum_{y=0}^{K-1} w(x,y) f(x+i, y+i) \tag{5.5}$$

onde

- $i = 0, 1, ...; M - 1$,
- $j = 0, 1, ...; N - 1$, com o somatório realizado apenas na região da imagem onde w e f se sobrepõem.

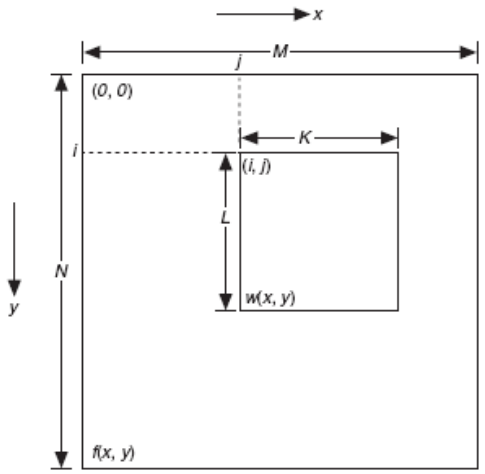


Figura 5.16: Ilustração do processo de correlação cruzada entre imagens.

A Fig. 5.16 ilustra a técnica de correlação. Assumindo a origem da imagem f no canto superior esquerdo do ecrã, a correlação é calculada movimentando-se a sub-imagem w em toda a área de f, calculando-se o valor de $C(i, j)$ em cada iteração. Isto envolve efectuar uma multiplicação para cada pixel em que há sobreposição e somar os resultados para todos os pixels na imagem padrão. O valor máximo de $C(i, j)$ indicará idealmente a melhor "aproximação" entre w e f.

A correlação pode ser optimizada para, por exemplo, diminuir a sua sensibilidade a alterações de intensidade luminosa nas imagens através da normalização do coeficiente de correlação:

$$R(i,j) = \frac{\sum_{x=0}^{L-1}\sum_{y=0}^{K-1}(w(x,y)-\overline{w})(f(x+i,y+j)-f(i,j))}{\left[\sum_{x=0}^{L-1}\sum_{y=0}^{K-1}(w(x,y)-\overline{w})^2\right]^{\frac{1}{2}}\left[\sum_{x=0}^{L-1}\sum_{y=0}^{K-1}(f(x+i,y+j)-f(i,j))^2\right]^{\frac{1}{2}}} \tag{5.6}$$

onde $\overline{w}$ (calculado apenas uma única vez) é o valor médio dos valores dos pixels no padrão w. O valor de R encontra-se entre -1 e 1 e é independente de variações de escala nos valores de intensidade de f e de w.

A transformação de RGB para escala de cinzentos é linear. O *LabVIEW* faz a conversão, pixel a pixel, utilizando a fórmula (sendo P_C o valor da intensidade luminosa do pixel em cinzento):

$$P_C = 0.299\,R + 0.587\,G + 0.114\,B. \tag{5.7}$$

A função *IMAQ Match Color Pattern* recebe como argumentos as duas imagens (o quadro e a imagem padrão) e devolve a informação posicional associada ao objecto reconhecido. Na Fig. 5.17 ilustra-se o resultado do processo. O padrão escolhido na Fig. 5.14 foi seguido durante 22 segundos. Os dados posicionais do centro do ROI são gravados num ficheiro de dados.

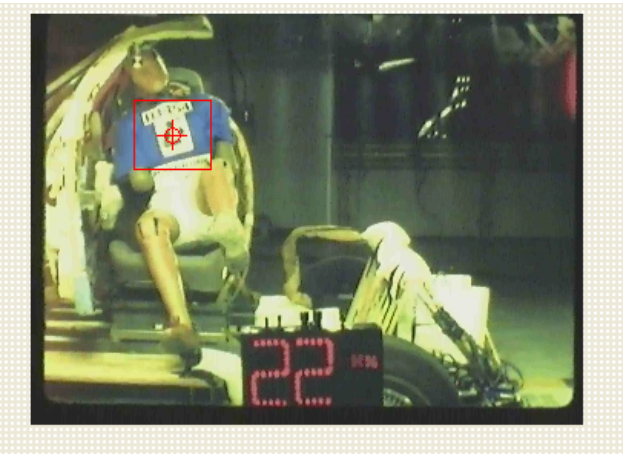


Figura 5.17: Posição do objecto seguido 22 segundos após o começo do programa.

3. O ficheiro de dados é lido e as diversas posições do objecto são desenhadas na imagem com a ajuda da função *IMAQ Draw*, o que permite visualizar a deslocação do objecto. Na Fig. 5.18 é possível observar a deslocação da imagem padrão durante o filme.

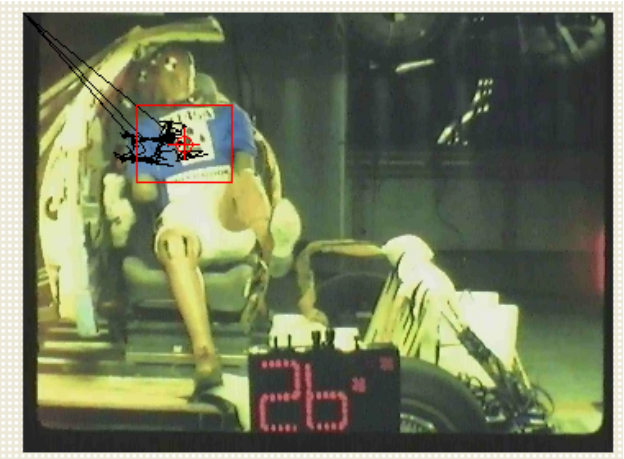


Figura 5.18: Deslocações do objecto no filme do *crash test* utilizado como exemplo (as rectas que partem do canto superior esquerdo do ecrã referem-se a localizações mal sucedidas).

5.5 Calibração

O funcionamento de ambos os sistemas foi explicado em pormenor nas secções anteriores, faltando somente referir, e realçar, algumas das suas características.

Uma das vantagens do sistema de aquisição em tempo real deve-se à existência de um comando que permite regular os parâmetros de detecção (por exemplo, podemos regular o sistema para que só capte objectos com dimensões entre 1 e 3 mm^2), o que significa que é possível haver objectos junto do animal que não são captados, desde que tenham dimensões suficientemente diferentes daquele. A calibração dos sistemas também é parametrizável, ou seja, a conversão de pixels para centímetros ou milímetros pode ser alterada a qualquer momento caso a posição da câmara relativamente ao cenário seja alterada.

5.6 Conclusão

Neste capítulo foi explicado a construção e funcionamento dos sistemas de monitorização em pormenor. Os principais conteúdos referem-se aos métodos utilizados para segmentação de imagem e aos instrumentos que realizam essas operações mas incorporados num processo de aquisição de imagem em tempo-real; e ao sistema de observação criado para filmes pré-gravados, sendo este caso idealizado para trabalhar com imagens a cores sem qualquer filtro aplicado (espacial ou na frequência), utilizando as funções do LabVIEW ligadas ao reconhecimento de padrões e de cores.

Capítulo 6

Resultados e discussão

6.1 O sistema em tempo real

Até aqui foi feito o estudo dos fundamentos teóricos subjacentes às técnicas de processamento de imagem utilizadas neste trabalho, e a descrição da implementação do sistema de processamento de imagem, numa perspectiva dupla focando quer o material utilizado ("hardware"), quer os programas desenvolvidos ("software"). O trabalho foi desenvolvido sobre uma plataforma LabVIEW da National Instruments.

É importante referir que o Windows não é o melhor sistema operativo (SO) para ser utilizado em processamento de imagens em tempo real dado que não é um *Sistema Operativo de Tempo Real*, ou seja, não foi desenvolvido, nem é particularmente adequado, para a execução de tarefas onde o tempo máximo de resposta a um evento está pré-definido. Como o SO não é dedicado à tarefa específica de aquisição de imagens (ou não lhe é atribuída a a prioridade de execução máxima), ele vai executando várias operações (ou processos) em concorrência, distribuindo o tempo de processamento entre eles (e eventualmente atendendo solicitações assíncronas internas ou externas), o que vai levar a um retardamento global de todas as operações em execução. Isto vai fazer com que o número de aquisições de imagens por segundo seja menor que o ideal e, essencialmente, que este seja um parâmetro não controlável do sistema.

O bicho de conta (*Crustaceos Isopoda*) foi o ser vivo escolhido para ser estudado pelo programa de monitorização em tempo real. Este animal desloca-se a uma velocidade relativamente lenta , o que permite simultaneamente fazer a observação do seu comportamento e decidir sobre possíveis ajustes a fazer no sistema. O sistema em tempo real revelou-se eficaz na segmentação e no seguimento destes animais. Contudo, revelou-se também muito sensível a flutuações luminosas do ambiente experimental (cenário), pois estas frequentemente produzem sombras que o sistema reconhece erroneamente como objectos.

O bicho de conta foi colocado numa caixa com área 18×10 cm e observado durante 10 minutos. Nesse espaço de tempo o animal deslocou-se 1578 mm e a sua velocidade máxima foi de 2 mm/s. A distância percorrida pelo animal medida pelo programa foi comparada

com a distância percorrida na realidade e apresentou um erro inferior a 1%.[1]. A deslocação do animal foi observada num gráfico posicional e foram calculadas a velocidade instantânea e a orientação, tudo em tempo real. Na Fig. 6.1 pode ser observada a deslocação do animal.

Aquando da interacção entre dois bichos de conta, o sistema conseguiu distinguir entre os dois a $2,0 \pm 0,5$ mm de distância, em ambas as direcções. A partir dessa distância o sistema assume que há apenas um objecto em observação e o algoritmo de monitorização deixa de ser eficiente.

Figura 6.1: Gráfico da deslocação do bicho de conta durante 10 minutos.

Também foram gravados os dados relativos às diversas orientações do animal (Fig. 6.2)[2]. Está fora do âmbito deste trabalho o estudo das razões da preferência do bicho de conta por determinadas direcções.

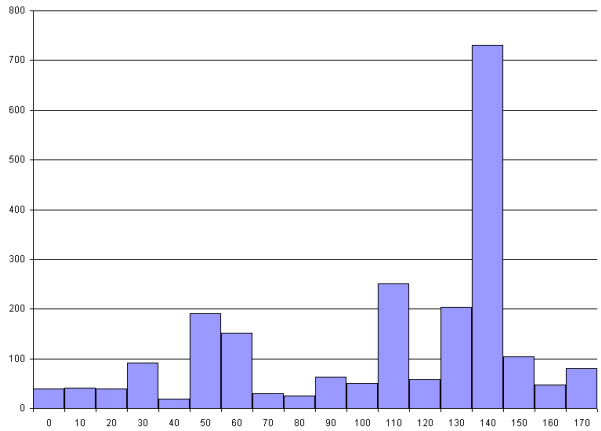


Figura 6.2: Histograma relativo às diversas orientações do bicho-de-conta. Ângulo que um eixo que atravesse o animal faz com o eixo X em graus vs o número de contagens nessa direcção.

[1]Calculado a partir do maior desvio em relação à média.

[2]Dado que é impossível ao programa distinguir a cabeça da cauda do animal, o histograma só vai de 0 a 170 graus.

6.2 O sistema de processamento de imagens arquivadas

Como já foi referido, para complementar o programa de monitorização em tempo real, foi desenvolvido um programa de localização e seguimento de alvos em filmes previamente gravados.

O sistema de localização do animal dispõe de diversos pârametros que podem ser alterados pelo utilizador. Por exemplo, o sistema permite regular a contribuição que a cor do objecto deve ter no processo de decisão do seu reconhecimento, sendo que o outro parâmetro decisório é a forma.

Aquele parâmetro varia entre 0 e 1000: se o peso escolhido for de 1000, o algoritmo só utiliza a contribuição da cor para a localização do objecto; se for de 0, a localização baseia-se somente na forma; para um peso de 500 a contribuição dos dois factores é igual. A estratégia de procura pode ser alterada, de modo a optimizar a velocidade, entre conservativa (maior número de iterações, mais lento) e agressiva (menos iterações, mais rápido). Também se pode alterar a sensibilidade do sistema a variações de cor (alta, média ou baixa), ou seja, quanto menor for a diferença entre as cores do objecto e o fundo, mais este pârametro deve ser aumentado para melhorar a eficácia. Um parâmetro importante programável pelo utilizador é a regulação da semelhança mínima entre objectos: ou seja, no reconhecimento de um objecto móvel o sistema determina um coeficiente de semelhança que varia entre 0 e 1000. Ao estabelecer um valor mínimo de semelhança impede-se que o sistema apresente reconhecimentos "falsos" (i. e., falsos positivos).

Para o processamento de imagens arquivadas foram utilizados dois tipos de filmes: gravações de peixes em aquários e um curto filme da deslocação de um percevejo (*Hemiptero da familia Pentatomidae*) na já referida caixa rectangular.

O caso mais simples de analisar será o do percevejo, pois a diferença cromática entre o animal e o ambiente é muito bem definida (Fig. 6.3). Como seria de esperar o processamento do filme é bastante lento (um filme de 22 segundos demorou 1 minuto e 45 segundos a ser processado), pois os algoritmos de reconhecimento são bastante complexos neste caso.

O primeiro passo para a obtenção de bons resultados é a selecção da imagem padrão. Na medida em que este padrão (template) representa sinteticamente o objecto alvo, é necessário que sejam capturadas todas as características que o tornam bem definido e o singularizam na imagem. A escolha do padrão é da responsabilidade do utilizador.

Este processo está detalhado nos gráficos da Fig. 6.4. Foi feita uma localização do percevejo com um valor mínimo (limiar) de semelhança de 800 e uma estratégia de busca com ponderações iguais (500,500) de cor e forma . No gráfico da Fig. 6.4-a a ROI consistia de um quadrado que englobava tanto o animal, como uma parte do fundo. O sucesso de localização foi de 90%, mas os centros de massa das localizações são muito dispersos. Na Fig. 6.4-b) o ROI foi construído recortando completamente o animal do fundo onde estava inserido e, neste caso, o sucesso de reconhecimento foi de 99%, com um gráfico de deslocação

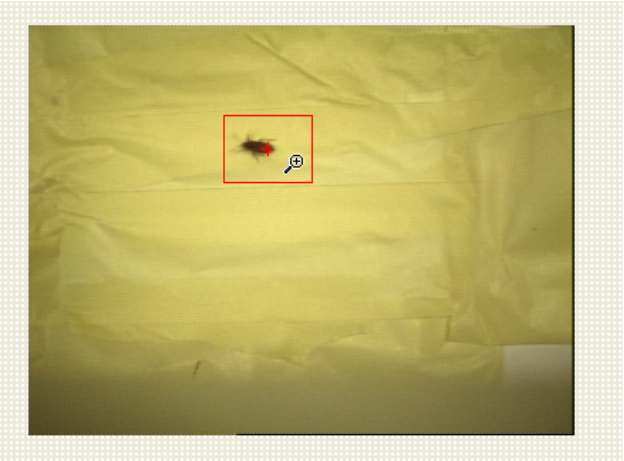


Figura 6.3: Localização do percevejo na imagem.

correspondendo muito fielmente à deslocação real do animal.

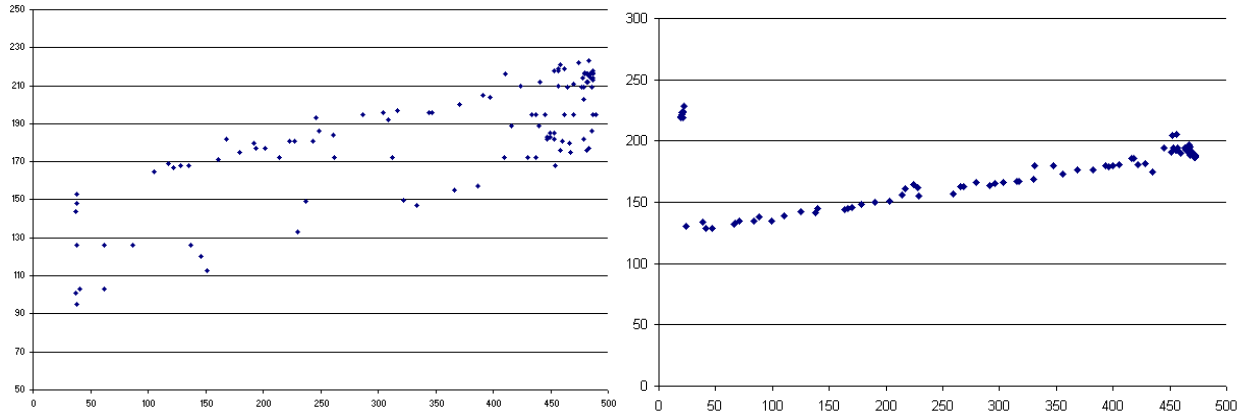


Figura 6.4: Gráficos da deslocação do percevejo em mm, plano XY. a) Localização com um ROI com fundo incluído; b) Localização com uma ROI sem o fundo incluido.

Sendo assim, está provado que o sistema funciona robustamente com animais com uma forma bem definida e que se distingam claramente do fundo em que estão inseridos, ou seja, funciona bem em condições muito perto da ideal.

Contudo, em fundos pouco homogéneos e com animais que mudam de forma durante o deslocamento não é tão eficaz. No caso do reconhecimento de um peixe, a percentagem de localizações foi bastante mais baixa (57%) (Fig. 6.5-a), mesmo com uma ROI bem definida e com diversas tentativas de valores de ajuste dos parâmetros de busca. São efectuados bastantes reconhecimentos mas, em certas movimentações do peixe, não é possível uma localização positiva quer pela cor quer pela forma do animal (Fig. 6.5-b). Este caso é paradigmático das dificuldades de um sistema de monitorização pela cor, pois a cor das pedras no fundo do aquário é semelhante à cor do peixe e a localização pela forma é difícil pois a forma do peixe varia constantemente, consoante a sua orientação tridimensional.

Figura 6.5: a) Imagem em que o reconhecimento do peixe foi possível. b) Imagem em que o sistema não conseguiu reconhecer o peixe.

6.2.1 Janela adaptativa

Para ultrapassar os problemas referidos anteriormente, foi implementado no sistema de reconhecimento o conceito de *janela adaptativa*. Note-se que esta janela não faz parte, de todo, dos algoritmos implementados de raiz no sistema de processamento de imagem da NI e que, portanto, traduz-se numa melhoria (talvez original, embora a ideia seja simples) "proprietária" associada ao presente trabalho. Passa-se a explicá-la.

O utilizador, na janela de controlo, pode seleccionar o perímetro de uma janela que limita a área em que o sistema vai fazer a localização do alvo. As coordenadas centrais da janela, no início do processamento de um dado quadro, vão ser as coordenadas do centro de massa obtido na localização do animal no quadro anterior. Sendo assim, a procura de um animal vai ser delimitada e baseada na sua localização anterior. Isto por um lado acelera a aplicação do algoritmo (a janela tem menor área que a imagem total) e, por outro, mantém o foco da localização na sub-região da imagem que mais interessa.

Obviamente que o sucesso da técnica depende do facto de o animal não "fugir" da janela entre dois quadros sucessivos: a dimensão da janela deverá pois depender da velocidade máxima estimada para o animal e da separação temporal entre quadros. Ao possuir uma localização adaptativa (as dimensões mantêm-se fixas), a janela vai acompanhando o deslocamento do animal ao longo dos quadros, mantendo-se aproximadamente centrada nele. Quando uma das arestas da janela ultrapassa os limites da imagem, deixa de ser utilizada pelo programa.

No caso do filme do peixe, foi inserida uma janela de 100 mm de lado. A escolha das dimensões da janela têm de ser cuidadosas de modo a não serem inferiores à área do ROI. Como foi referido a localização da janela vai acompanhado o deslocamento do animal e, para este caso, a taxa de sucessos de localização subiu para 80%, o que foi uma melhoria significativa em função da simplicidade da técnica utilizada.

A experiência seguinte descreve a localização e a caracterização do movimento de um peixe (*Cichlidae*) num aquário (Fig.6.6). Note-se que este filme é diferente do anterior e, em

particular, que o peixe vermelho sobressai bastante mais do cenário que o peixe estudado no filme anterior.

O peixe vermelho foi monitorizado na sua deslocação e todos os centros de massa localizados (bem ou mal) estão representados na Fig.6.7. Este gráfico foi realizado sem a janela adaptativa aplicada, ou seja, usou somente os algoritmos nativos do sistema de processamento de imagem. Foi feita uma localização com um valor mínimo de semelhança de 800 e com uma estratégia de busca baseada apenas na cor.

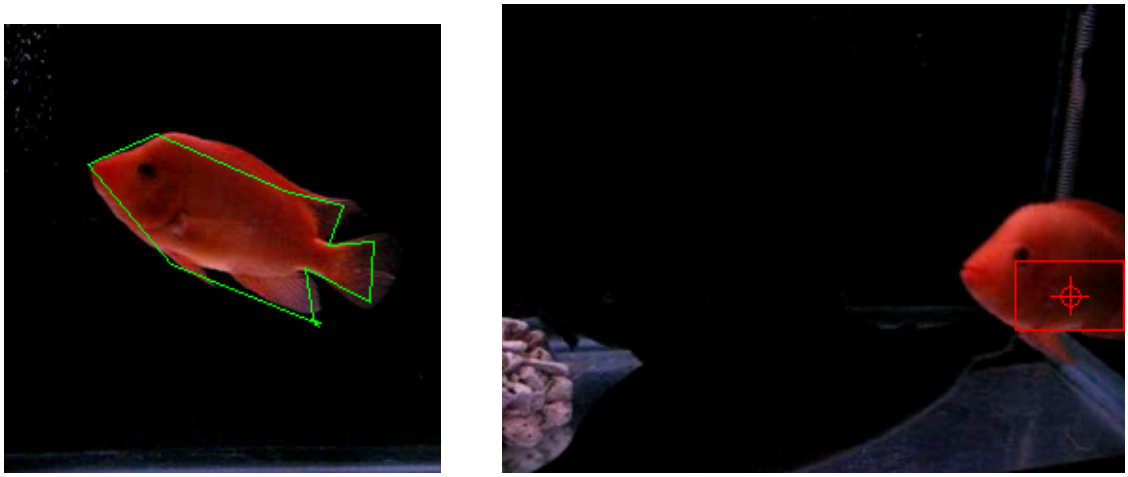


Figura 6.6: a) ROI utilizada na localização de um peixe num filme; b) Localização do peixe na imagem.

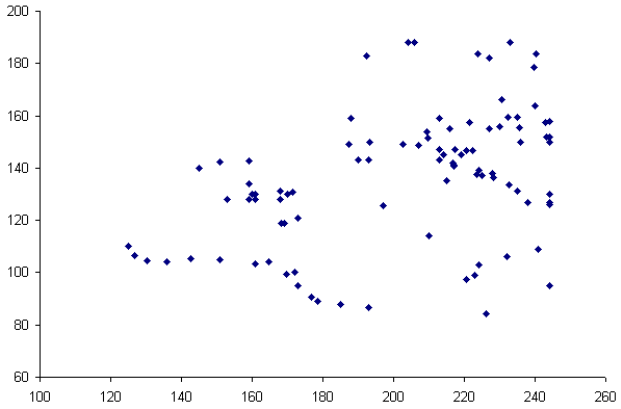


Figura 6.7: Gráfico representando a deslocação do peixe em mm (plano XY) sem a janela adaptativa aplicada.

A utilização da janela adaptativa melhorou significativamente os resultados, conforme se pode observar na Fig.6.8.

Contudo, pela análise dos gráficos, é óbvio que é muito difícil retirar dados quantitativos sobre o movimento do animal. Os dados referentes à posição do animal são um conjunto de pontos bastante dispersos, e algo desorganizados (a informação temporal não está patente na Fig.6.8) pois as coordenadas do centro de massa localizado pelo algoritmos apresentam

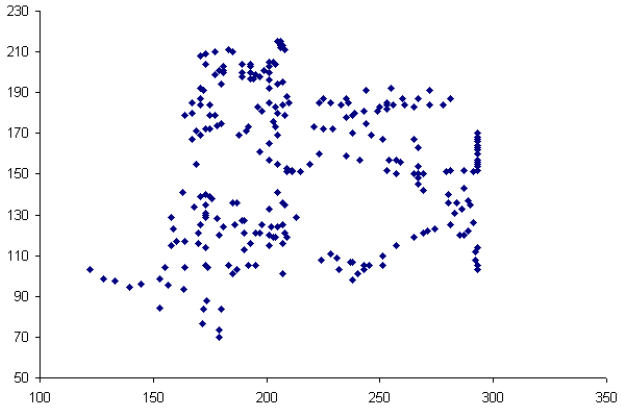


Figura 6.8: Gráfico representando a deslocação do peixe em mm (plano XY) com a janela adaptativa aplicada.

"oscilações" significativas relativamente à verdadeira posição (observável a olho nu pelo utilizador).

Este problema só foi convenientemente atenuado pela utilização de um filtro de Kalman, cuja aplicação se passa a descrever de seguida.

6.2.2 Aplicação do Filtro de Kalman

Para se poder aplicar o filtro de Kalman, cuja exposição foi feita no capítulo 3, é necessário desenvolver o modelo apropriado para o problema concreto, enquadrado pelas equações 3.2. Neste caso trata-se de um modelo de deslocamento típico em que o alvo é caracterizado por vectores de posição e de velocidade a duas dimensões, descrevendo estas variáveis o estado do sistema.

Suponhamos então que um animal se desloca sobre uma trajectória e é caracterizado pelas posição p e velocidade v. Vamos assumir, também, que o animal tem aceleração nula[3]. A variável y é a medida da posição (i.e., a observação) em cada T segundos (T é o intervalo de tempo que medeia entre duas imagens sucessivas). Neste modelo, as equações do movimento serão:

$$\begin{aligned} p_{k+1} &= p_k + T\, v_k + \tilde{p_k} \\ v_{k+1} &= v_k + \tilde{v_k} \end{aligned} \tag{6.1}$$

onde $\tilde{p_k}$ e $\tilde{v_k}$ representam, respectivamente, os ruídos da posição e da velocidade, assumidos no filtro de Kalman como sendo do tipo branco e Gaussiano. Neste modelo simplesmente se "prevê" que a posição em $k+1$ é a posição anterior adicionada do termo de deslocamento dado pela velocidade anterior vezes o tempo, e que a velocidade se mantém.

[3]É claro que se a aceleração fosse sempre nula o alvo continuaria a seguir uma trajectória em linha recta, com velocidade constante, o que não acontece na prática. Este modelo é, porém, justificado no presente problema assumindo a aceleração como fazendo parte do "ruído" do modelo, w. Na verdade, as mudanças de direcção do animal são bastante imprevisíveis e inesperadas (isto é, aleatórias) e, sendo assim, não podem ser modeladas deterministicamente.

O vector de estado, denominado h, é assim definido por:

$$h_k = \begin{bmatrix} p_k \\ v_k \end{bmatrix} \tag{6.2}$$

e as equações lineares do sistema (ver o capítulo 3) são então:

$$h_{k+1} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} h_k + \tilde{w_k} \tag{6.3}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} h_k + \tilde{z_k} \tag{6.4}$$

onde $\tilde{w_k}$ e $\tilde{z_k}$ representam respectivamente os vectores de ruído do processo e da observação.

Monitorização do peixe

Para descrever a movimentação do peixe, o modelo de deslocamento anterior é escrito a duas dimensões. Sendo os eixos da imagem denominados de xx (horizontal) e yy (vertical), pode-se escrever então que:

$$h_k^x = \begin{bmatrix} p_k^x \\ v_k^x \end{bmatrix} \qquad h_k^y = \begin{bmatrix} p_k^y \\ v_k^y \end{bmatrix} \tag{6.5}$$

Optou-se por modelar o movimento em cada eixo separadamente[4]. Em cada quadro da imagem fazemos uma nova medição e, assumindo o quadro como a unidade de tempo no modelo de deslocamento, define-se $T = 1$. Assim, a posição será "medida" em *mm* e a velocidade em *mm/quadro*. Como foi referido no capítulo 3, A é denominada de *matriz de transição de estado*, relacionando o vector de estado h em instantes discretos sucessivos, e C é a *matriz de observação* que relaciona y com h, ou seja (lembre-se que $T = 1$):

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

As especificações do ruído foram estimadas a partir da análise dos dados da posição e da velocidade do peixe em alguns segmentos do filme. É óbvio que os erros de medição e do processo serão diferentes de animal para animal e, por isso, os respectivos valores podem ser alterados pelo utilizador. Foi assumido um erro de medição com desvio padrão igual a 10 mm, em cada direcção, e os valores do desvio padrão associados ao modelo de movimento são $\epsilon_{px} = 2$, $\epsilon_{py} = 1$, $\epsilon_{vx} = 1$ e $\epsilon_{vy} = 0,2$. Sendo assim, as matrizes de ruído são:

$$w^x = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \qquad z^x = \begin{bmatrix} 10 \end{bmatrix}, \qquad w^y = \begin{bmatrix} 1 \\ 0,2 \end{bmatrix}, \qquad z^y = \begin{bmatrix} 10 \end{bmatrix}$$

assumindo-se que são constantes durante todo o tempo (i.e., não dependem de k).

As matrizes de co-variância serão então dadas por $S_w^x = w^x \, (w^x)^T$ e $S_z^x = z^x \, (z^x)^T$ e $S_w^y = w^y \, (w^y)^T$ e $S_z^y = z^y \, (z^y)^T$.

[4]Poder-se-ia usar um modelo a 4 dimensões, mas nesse caso seria necessário fazer operações com matrizes 4×4 o que oneraria desnecessariamente o tempo de cálculo.

Temos, então, todos os parâmetros da modelação necessários para utilizar as equações do Filtro de Kalman. Os dados resultantes da aplicação da janela adaptativa foram pré-tratados de modo a que os não-reconhecimentos, ou seja, as coordenadas de centro de massa que o Labview define como $(0, 0)$, passassem a apresentar um valor igual ao último reconhecimento válido efectuado. Esta hipótese é empiricamente justificada.

Inicializamos h com os valores das coordenadas da localização do objecto no 1º quadro do filme (seleccionadas pelo utilizador do Labview sobre o filme) e com velocidade nula, pois assume-se que inicialmente o peixe está parado, e a matriz de covariância com $P = S_w$, para designar a incerteza do estado inicial. A implementação dos algoritmos segundo as duas coordenadas é igual, variando apenas os parâmetros associados aos erros. As equações do FK são executadas quadro a quadro com os resultados referentes a 300 quadros (cerca de 12 segundos de filme) mostrados na Fig. 6.9, onde se pode constatar uma muito significativa melhoria da definição do movimento, com a deslocação do peixe completamente reconhecível.

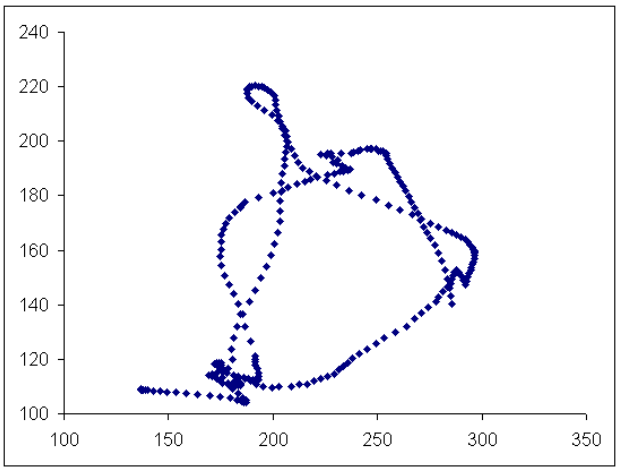


Figura 6.9: Trajectória do peixe com o Filtro de Kalman aplicado.

A sobreposição dos valores originais e dos resultantes da aplicação do FK pode ser observada na Fig. 6.10.

Os melhoramentos trazidos pela aplicação do FK também podem ser observados em cada direcção separadamente na Fig. 6.11.

Uma das grandes vantagens do uso do FK está patente na Fig. 6.12. Dado que a velocidade do animal está incluída no estado do sistema h, em resultado da sua aplicação obtém-se também a velocidade instantânea estimada para o animal segundo as duas coordenadas. Podemos observar naquelas figuras as velocidades antes e depois de aplicar o FK, sendo evidentes os efeitos de "filtragem" alcançados. É de salientar que as velocidades não são observadas.

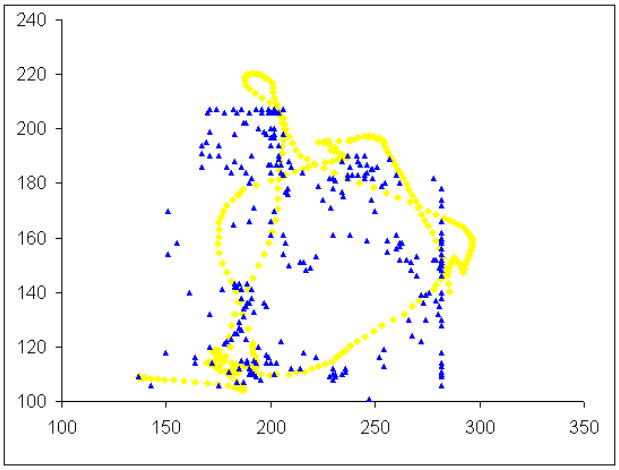


Figura 6.10: A amarelo, os valores obtidos pela aplicação do FK; a azul, os valores devolvidos pelo sistema de monitorização.

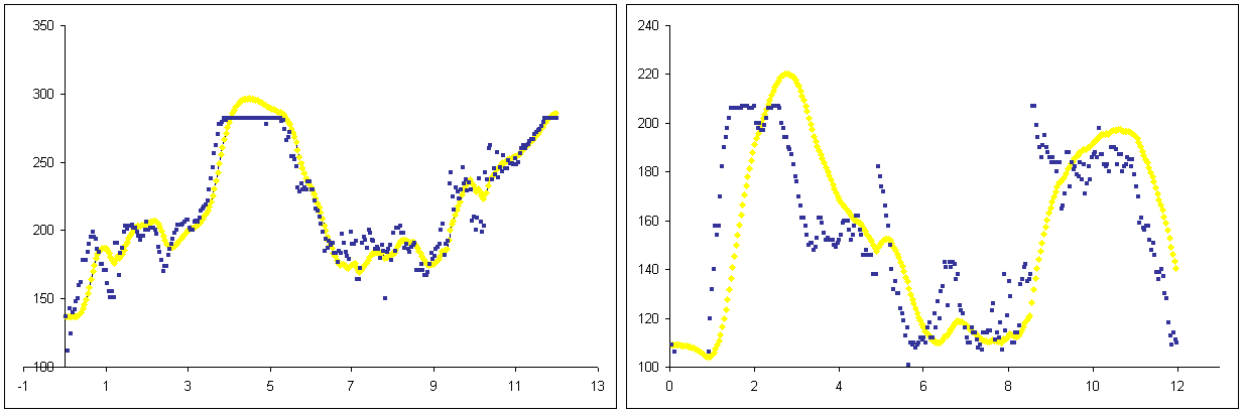


Figura 6.11: À esquerda, posição segundo o eixo xx em função do tempo; à direita, posição segundo o eixo yy em função do tempo. A amarelo, os valores devolvidos pelo FK; a azul, os valores resultantes do sistema de monitorização.

6.3 Conclusão

Neste capítulo procedeu-se à descrição dos resultados obtidos com o sistema de reconhecimento e seguimento de animais. Foram implementadas versões ajustadas a processamento em tempo real e a processamento em modo "off-line", através de compromissos entre a velocidade de processamento dos algoritmos e a sua complexidade (e robustez).

Depois de construído o sistema de processamento de imagem, o mesmo revelou-se algo inferior às expectativas, pois a taxa de insucessos na detecção foi bastante significativa. Para melhorar este aspecto adoptaram-se estratégias suplementares que conduziram a resultados francamente positivos, isto é, francos melhoramentos no seguimento, permitindo fazer estimativas da posição e velocidade instantânea dos alvos. O centro da estratégia consistiu na aplicação de um filtro de Kalman para prever a movimentação de uma janela adaptativa que cerca o animal.

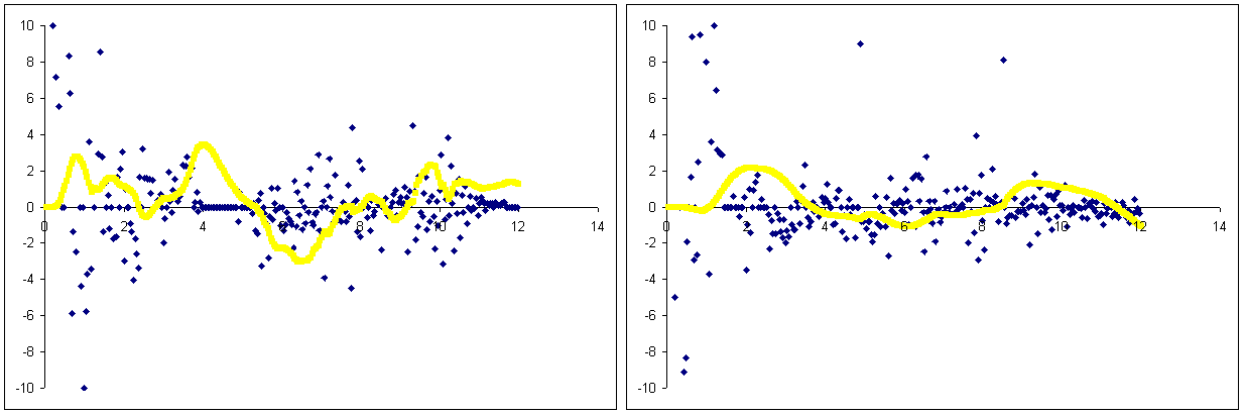


Figura 6.12: a - Velocidade segundo o eixo xx em função do tempo. b - Velocidade segundo o eixo yy em função do tempo. A amarelo os valores resultantes da aplicação do FK, a azul os valores resultantes do sistema de monitorização.

Capítulo 7

Conclusão e trabalho futuro

7.1 Conclusão

No projecto descrito nesta dissertação foram construídos dois sistemas de monitorização automática de animais. O primeiro funciona em tempo real e, quando o cenário apresenta um fundo homogéneo, apresenta uma elevada percentagem de sucessos de localização e retira dados que podem ser importantes no estudo dos respectivos comportamentos e interacções, tais como a velocidade (instantânea e média), a distância percorrida e a orientação angular. O sistema de processamento de imagem disponibilizado pela plataforma LabVIEW é bastante versátil e dispõe de um elevado número de funções nativas que satisfazem a quase totalidade das necessidades de processamento exigidas neste trabalho. No entanto, o comportamento "em campo" revelou-se algo inferior às expectativas, pois a taxa de insucessos na detecção foi bastante significativa.

Resolver este problema, terá sido, talvez, a parte mais original do trabalho, pois a aplicação de um filtro de Kalman a um sistema de "tracking", em filmes pré-gravados, para prever a movimentação de uma janela adaptativa que cerca o animal e se centra no seu centro de massa e filtra as "observações" desse mesmo centro de massa, permitiu fazer estimativas da posição e velocidade instantâneas dos alvos. O modelo do filtro foi adaptado ao presente problema e escolhida uma caracterização dos ruídos adequada. Os resultados obtidos foram extremamente positivos.

Ambos os programas podem servir como ferramentas valiosas no âmbito das ciências biológicas, quando aplicados a estudos que envolvam a interacção entre animais, sendo possível afirmar que os pontos propostos na formulação do problema foram alcançados com sucesso e eficiência.

7.2 Trabalho futuro - Utilização de duas câmaras em simultâneo

A utilização de duas câmaras em simultâneo no sistema poderá ser feita de várias maneira e recorrendo a diversas tecnologias. Porém, parece-nos que a melhor opção (ponderando a complexidade de implementação e o custo do sistema) será a que recorre a câmaras "firewire" (interface IEEE-1394), devido à sua boa velocidade de transmissão de dados (400 Mb/s para a especificação IEEE-1394a e 800 Mb/s para a especificação IEEE-1394b) e possibilidade de ligação simultânea de vários dispositivos [48],[49].

Contudo, o NI-IMAQ baseado na especificação IEEE-1394 não garante que o sistema de aquisição funcione se a informação transferida for superior à capacidade do barramento ("bus"). Para que a aquisição de imagens seja feita à maior velocidade possível, a melhor solução será decerto a aquisição de uma placa "firewire" com várias entradas, o que irá permitir a ligação de duas câmaras simultaneamente. Uma boa opção para aquisição de imagens simultâneas seria o sistema **Compact Vision System NI CVS-145x** (Fig. 7.1), uma placa externa da NI com três entradas "firewire" criada especificamente para captar imagens de câmaras deste tipo. Para seleccionar as câmaras é preferível optar por dispositivos que apresentem uma maior rapidez de aquisição por oposição àqueles que tenham uma melhor resolução. A **Basler A601F**, com resolução de 640×480, permite captar imagens a uma velocidade de 60 frames por segundo. O NI-IMAQdx é o "driver" de hardware fornecido pela National Instruments que permite fazer a interface com o bus "firewire".

Figura 7.1: Placa NI CVS-145x.

Configurar em LabVIEW um sistema deste género é conceptualmente simples: basta atribuir a cada câmara um nome (e.g., *camO* e *cam1*) e criar dois sistemas de aquisição no mesmo IV. Se criarmos um IV para adquirir imagens para uma câmara "firewire" basta, neste caso, construir mais um IV quase exactamente igual, pois apenas estará ligado a uma câmara diferente. Estas funções de aquisição com os dois dispositivos vão ser executadas

simultaneamante, no que diz respeito ao utilizador.

A relevância deste sistema multi-câmara seria o possibilitar do reconhecimento de trajectórias de animais num espaço a três dimensões o que, no caso de animais que se movimentam em meios aquáticos ou aéreos, se pode revelar de extrema utilidade.

Capítulo 8

Apêndices

8.1 Aquisição de imagens

Na Fig. 8.1 estão exemplificados alguns conceitos importantes para o processo de aquisição de imagens de modo a que a sua qualidade permita extrair facilmente toda a informação pretendida. Ao adquirir imagens é necessário ter em conta:

- O tamanho do sensor de gravação- as dimensões da área activa do sensor, normalmente definidas pela sua dimensão horizontal;
- A distância de gravação- a distância desde a parte frontal da lente até ao objecto;
- O campo de visão- a área máxima que a câmara pode cobrir;
- A distância focal da lente- uma medida da qualidade da lente que pode ser obtida pela equação: *Distância focal = (Tamanho do sensor de gravação* $\times$ *Distância de gravação) / Campo de visão).*

8.2 Manual do utilizador

Nenhum dos programas construídos necessitam de correr num computador com LabVIEW instalado. É necessário, contudo a instalação do software necessário, ou seja, o *Run Time Engine* fornecido pela *National Instruments* em [38] e que permite executar programas, mas não alterar os seus códigos. Os programas são de fácil utilização sendo a única dificuldade a regulação dos parâmetros de localização de modo a que estes se adaptem ao cenário. Recomenda-se por isso alguns ensaios prévios para que o utilizador possa reconhecer os melhores parâmetros para a experiência em questão. Todos os programas podem ser inicializados ou parados quando o utilizador quiser.

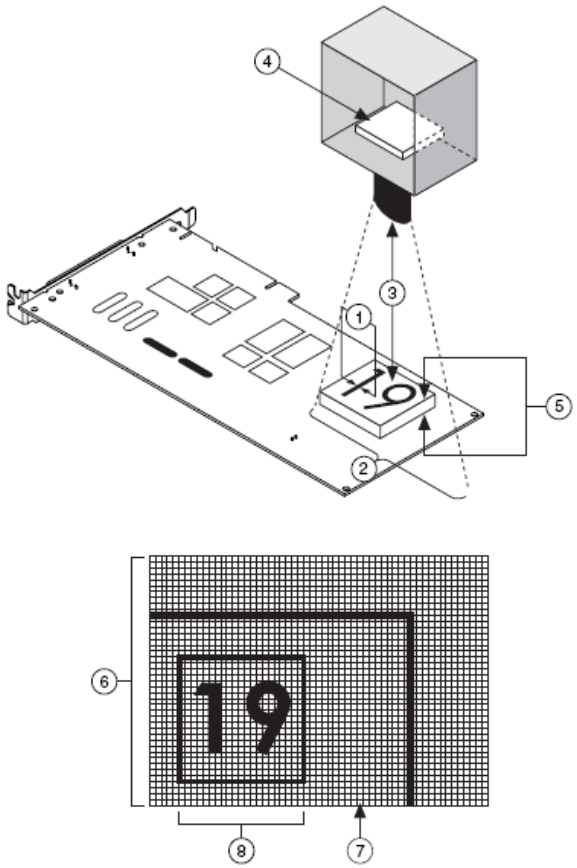


Figura 8.1: Parâmetros fundamentais de um sistema de aquisição de imagem. 1- Resolução; 2- Campo de visão; 3- Distância de gravação; 4- Tamanho do sensor de gravação; 5- Profundidade do campo de visão; 6- Imagem; 7- Pixel; 8- Resolução de pixel.

8.2.1 Programa de localização em tempo real

O programa de localização de animais em tempo real exige que o computador tenha uma placa de aquisição de imagens instalada. Depois de instada a mesma e de colocada a camâra na direcção dos objectos a localizar, basta carregar no botão de ínicio e o programa começa imediatamente funcionar. Os dados são gravados numa pasta escolhida pelo utilizador (recomenda-se que sejam gravados como um ficheiro ".dat") e para isso basta carregar no ícone correpondente. Depois do programa estar a correr o utilizador pode variar diversos parâmetros de visualização tais como a área máxima e mínima das partículas da imagem (Fig. 8.2).

Existem dois programas diferentes, um para localizar um animal e outro para localizar dois animais. No programa para localizar dois animais é necessário inserir duas pastas diferentes - uma para cada animal. O programa grava automaticamente os dados posicionais dos objectos até o programa ser interrompido.

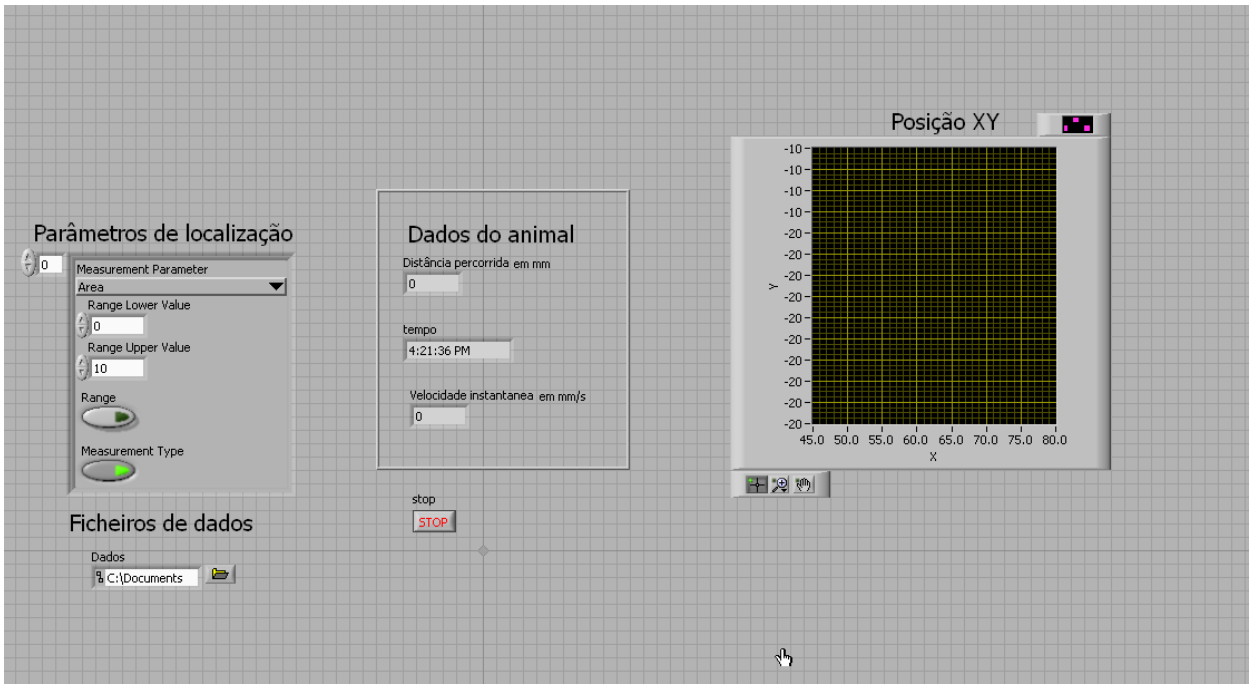


Figura 8.2: Painel frontal do programa de aquisição em tempo real.

8.2.2 Programa de localização em imagens arquivadas

O programa da localização de animais em imagens arquivadas é mais complexo que o anterior, devido ao maior número de parâmetros que podem ser alteráveis (Fig. 8.3). Uma regra básica será definir se é necessária uma estratégia de busca mais baseada na cor ou na forma de um objecto. A localização de animais que mudem de forma necessita de parâmetros de busca diferentes do que a de animais com forma fixa. O mesmo se aplica a um animal com uma cor semelhante ao cenário e um com uma cor completamente diferente. O utilizador pode optar por variar os parâmetros que a seguir se explicam. Percentagem de cor na busca - este parâmetro varia entre 0 e 1000: se o peso escolhido for de 1000, o algoritmo só utiliza a contribuição da cor para a localização do objecto; se for de 0, a localização baseia-se somente na forma; para um peso de 500 a contribuição dos dois factores é igual. Estratégia de busca - pode ser alterada, de modo a optimizar a velocidade, entre conservativa (maior número de iterações, mais lento) e agressiva (menos iterações, mais rápido). Sensibilidade de cor - Também se pode alterar a sensibilidade do sistema a variações de cor (alta, média ou baixa), ou seja, quanto menor for a diferença entre as cores do objecto e o fundo, mais este pârametro deve ser aumentado para melhorar a eficácia. Percentagem de reconhecimento - Um parâmetro importante programável pelo utilizador é a regulação da semelhança mínima entre objectos: ou seja, no reconhecimento de um objecto móvel o sistema determina um coeficiente de semelhança que varia entre 0 e 1000. Ao estabelecer um valor mínimo de semelhança impede-se que o sistema apresente reconhecimentos “falsos” (i. e., falsos positivos). Modo de busca - Permite escolher se a busca é só baseada na cor, na forma ou em ambas. Existe ainda outro parâmetro que permite escolher o tamanho da janela (em mm) em que queremos concentrar as buscas do animal. A gravação de dados é exactamente igual

ao programa anterior.

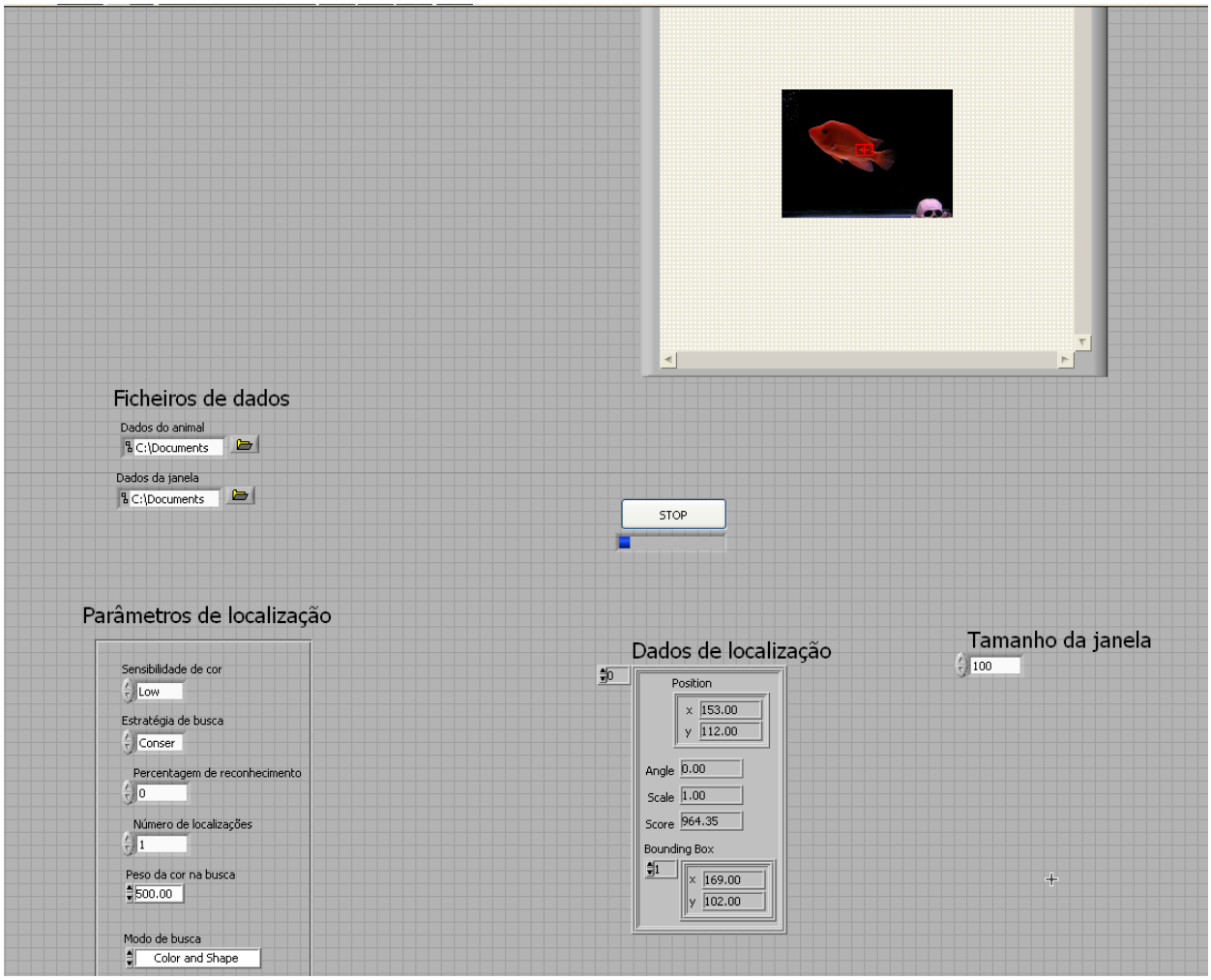


Figura 8.3: Painel frontal do programa de aquisição em imagens pré-gravadas.

Bibliografia

[1] L.P.J.J. Noldus, A.J. Spink e R.A.J. Tegelenbosch. EthoVision: A versatile video tracking system for automation of behavioral experiments, *Behavior Research Methods, Instruments, & Computers* **33**, p. 398 (2001)

[2] R.L. Clark, R.F. Smith e D.R. Justesen. An infrared device for detecting locomotor activity, *Behaviour Research Methods, Instruments and Computers* **17**, p. 519 (1985)

[3] O. Gapenne, P. Simon e J. Lannou. A simple method for recording the path of a rat in an open field, *Behaviour Research Methods, Instruments and Computers* **22**, p. 443 (1990)

[4] R.W. Silverman, A.S Chang e R.W. Russel. A microcomputer-controled system for measuring reactivity in small animals, *Behaviour Research Methods, Instruments and Computers* **20**, p. 495 (1988)

[5] R.L. Clark, R.F. Smith e D.R. Justesen. A programmable proximity-contact sensor to detect location or locomotion of animals, *Behaviour Research Methods, Instruments and Computers* **24**, p. 515 (1992)

[6] W.H. Akaka e B.A. Houck. The use of an ultrasonic monitor for recording locomotor activity, *Behaviour Research Methods, Instruments and Computers* **12**, p. 514 (1980)

[7] P.H. Martin e D.M.Unwin. A microwave Doppler radar activity monitor, *Behaviour Research Methods, Instruments and Computers* **12**, p. 517 (1980)

[8] P. Pereira e R.F. Oliveira. A simple method using a single video camera to determine the three-dimensional position of a fish, *Behaviour Research Methods, Instruments and Computers* **26**, p. 443 (1994)

[9] K. Klapdor, K. Dulfer e Van Der Staay. A computer-aided method to analyse foot print patterns of rats, mice and humans, *Proceedings of Measuring Behavior ´96, International Workshop on Methods and Techniques in Behavioral Research*, p. 60 (1996)

[10] D.G. Lowe. Three-Dimensional Object Recognition from Single Two-Dimensional Images, *Artificial Intelligence* **31**, p. 355 (1987)

[11] M.J. Swain e D.H. Ballard. Color Indexing, *International Journal of Computer Vision* **7**, p. 11 (1991)

[12] H.J. Wolfson e I. Rigoutsos. Geometric Hashing: An Overview, *IEEE Computational Science and Engineering* **4**, p. 10 (1997)

[13] E. Hadjidemetriou e S.K. Nayar. Appearance-Matching with Parcial Data, *DARPA Image Understanding Workshop*, p.1071 (1998)

[14] P.N. Belhumeur, J.P. Hespanha e D.J. Kriegman. Eidenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**, p.1395 (1997)

[15] P. Viola e M. Jones. Robust Real-time Object Detection, *Second International Workshop on Statistical and Computational Theories of Vision-Modeling, Learning, Computing and Sampling* (2001)

[16] H. Tao, H. Sawhneye e R. Kumar. A Sampling Algoritm for Tracking Multiple Objects, *Workshop on Vision Algorithms* (1999)

[17] Z. Khan, T. Balch e F. Dellaert. MCMC Data Association and Sparce Factorization Updating for Real Time Multitarget Tracking with Merged and Multiple Measurements, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**, p.1960 (2006)

[18] I.D. Jonsen, R.A. Myers e J.M. Flemming. Meta-Analysisof Animal Movement Using State-Space Models, *Ecology* **84**, p. 3055 (2003)

[19] M. Isard e J. MacCormick. BraMBLe: A Bayesian Multiple-Blob Tracker, *Proc. Int. Conf. Computer Vision* **2** p.34 (2001)

[20] K. Branson e S. Belongie. Tracking Multiple Mouse Contours (without Too Many Samples), *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **VOL 1**, 1039 (2005)

[21] T. Balch, Z. Khan e M. Veloso. Automatically Tracking and Analyzing the Behavior of Live Insect Colonies, *Proc. 5th Int. Conf. Autonomous Agents*, p.521 (2001)

[22] T. Balch *et al.*. How Multirobot Systems Research Will Accelerate Our Understanding of Social Animal Behavior, *Proceedings of the IEEE* **97**, p.1445 (2006)

[23] A.P. French. Visual Tracking: From an Individual to Groups of Animals. *PHD Thesis to the University of Nottingham*

[24] R.A. Dielenberg, P. Halasz e T.A. Day. A method for tracking rats in a complex and completely dark enviroment using computerized video analysis, *Journal of Neuroscience Methods* **158**, p.279 (2006)

[25] D. Walther, D.R. Edgington e C. Koch. Detection and Tracking of Objects in Underwater Video, *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR' 04)*, p.1063 (2004)

[26] S. Colantonio *et al.*. Object tracking in a stereo and infrared vision system *Infrared Physics & Technology* **49**, p.266 (2007)

[27] A. Fernández-Caballero, *et al.*. On motion detection through a multi-layer neural network architecture *Neural Networks* **16**, p.205 (2003)

[28] K. Tabb *et al.*. The recognition and analysis of animate objects using neural networks and active contour models *Neurocomputing* **43**, p.145 (2002)

[29] B.C. Carter, G.T. Shubeita e S.P. Gross. Tracking single particles: a user-friendly quantitative evaluation *Physical Biology* **2**, p.60 (2005)

[30] R.C. Gonzalez e R.E. Woods, *Digital Image Processing*, **Prentice Hall** (1992)

[31] J.R. Parker, *Algorithms for Image Processing and Computer Vision*, **Wiley** (1996)

[32] M. Seul, L. O´Gorman e M.J. Sammon, *Practical Algorithms for Image Analysis: Descriptions, Examples, and Code*, **Cambridge University Press**(2000)

[33] T. Klinger, *Image Processing wiht LabView and IMAQ Vision*, **Prentice Hall** (2003)

[34] S. Erturk, *Digital Image Processing* (2003)

[35] P.S. Sriram, *Developing Neural Networks Applications Using LabVIEW*, **University of Missouri-Columbia** (2005)

[36] J. Travis e J. Kring, *LabVIEW for Everyone: Graphical Programming Made Easy and Fun, Third Edition*, **Prentice Hall** (2006)

[37] R. Bitter, T. Mohiuddin e M. Nawrocki *LabVIEW Advanced Programming Techniques*, **CRC Press** (2001)

[38] www.ni.com

[39] L.G. Shapiro e G.C. Stockman *Machine Vision*, **Prentice Hall** (2001)

[40] National Instruments, *NI Vision Concepts Manual* (2005)

[41] J. Augusto e L. Deniau, *Processamento de Imagens e Aplicações em Imagiologia Médica* (2005)

[42] www.wikipedia.org

[43] J. Miano, *Compressed Image File Formats: JPEG, PNG, GIF, XBN, BMP*, **ACM Press** (1999)

[44] A.E. Zonst, *Understanding FFT Applications, Second Edition*, **Citrus Press** (2003)

[45] D. Simon, *Kalman Fitering*, **Embedded Systems Programming** (2001)

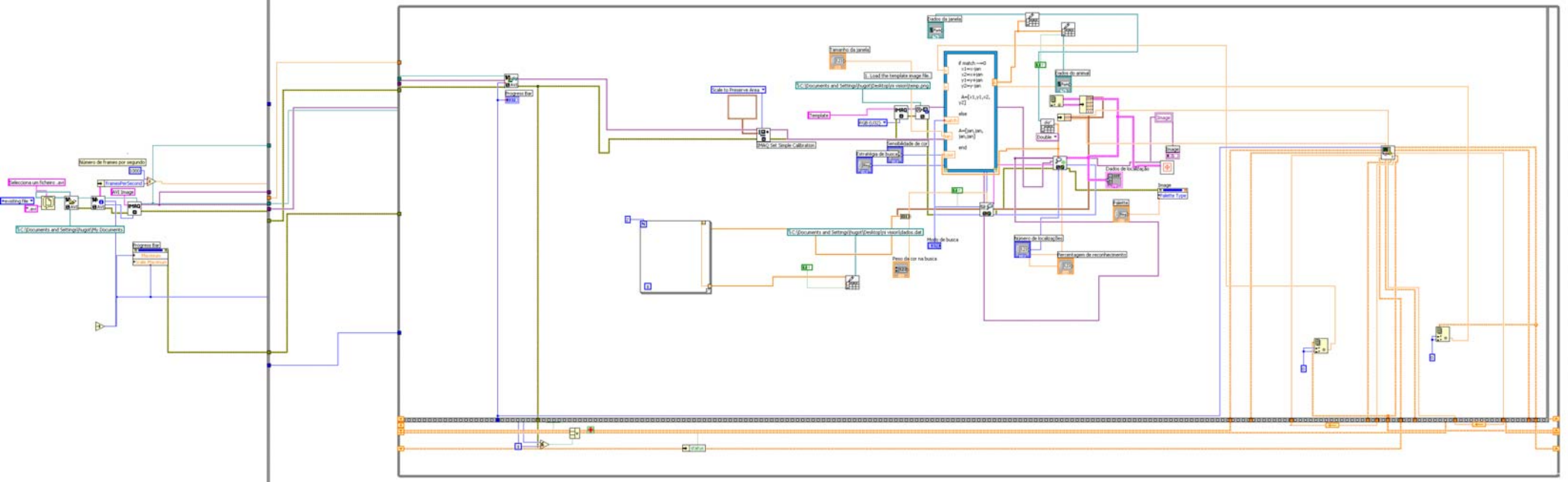
[46] P.J. Hargrave, *A Tutorial Introduction to Kalman Filtering*, **STC Technology Ltd.**

[47] G. Welch e G. Bishop, *An Introduction to the Kalman Filter*, **University of North Carolina at Chapel Hill** (2006)

[48] National Instruments, *Acquiring from Firewire Cameras with National Instruments NI-IMAQdx and Legacy NI-IMAQ for IEEE 1394* (2006)

[49] 1394 Trade Association, *IIDC 1394 - based Digital Camera Specification - Version 1.30* (2005)

Programa de monitorização com utilização do filtro de Kalman



Código do Filtro de Kalman