UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FERRAMENTAS DE PRÉ-PROCESSAMENTO DE IMAGENS

FABRÍCIO MENDES DE OLIVEIRA

ORIENTADORA:

JULIANA FERNANDES CAMAPUM WANDERLEY

PROJETO FINAL DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

BRASÍLIA-DF, 17 DE FEVEREIRO DE 2003

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FERRAMENTAS DE PRÉ-PROCESSAMENTO DE IMAGENS

FABRÍCIO MENDES DE OLIVEIRA

PROJETO FINAL DE GRADUAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE ENGENHARIA ELÉTRICA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO

APROVADA POR:

_____

JULIANA FERNANDES CAMAPUM WANDERLEY, PhD, UnB, (ORIENTADORA)

_____

FRANCISCO ASSIS DE O. NASCIMENTO. DOUTOR, UnB (EXAMINADOR INTERNO)

_____

LÚCIO MARTINS DA SILVA, DOUTOR, UnB (EXAMINADOR INTERNO)

BRASÍLIA-DF, 17 DE FEVEREIRO DE 2003.

FICHA CATALOGRÁFICA

OLIVEIRA, FABRÍCIO M.
Ferramentas de Pré-processamento de Imagens [Distrito Federal] 2003
(xv), (100)p., 297 mm (ENE/FT/UnB, Engenheiro, Engenharia Elétrica, 2003).

Projeto Final de Graduação – Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica.

1. Imagem
2. Processamento de Imagens Digitais
3. Segmentação de Imagens
4. Compressão de Imagens

I. ENE/FT/UnB.
II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

OLIVEIRA, FABRÍCIO M. (2003). Ferramentas de Pré-processamento de Imagens. (Projeto Final de Graduação), Publicação /2003, Departamento de Engenharia Elétrica, Universidade de Brasília , Brasília , DF, (100)p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Fabrício Mendes de Oliveira
TÍTULO DA DISSERTAÇÃO: Ferramentas de Pré-processamento de Imagens
GRAU/ANO: Engenheiro/2002.

_____

FABRÍCIO MENDES DE OLIVEIRA

QNC 13 CASA 31

CEP 72.115-630 – TAGUATINGA/DF – BRASIL

“Apenas fazer uma tarefa é labuta. Expressar uma qualidade através dela é riqueza. Aprender é a recompensa por respeitar a vida”

Anthea Church

" Navegar é preciso, viver não é preciso"

Fernando Pessoa

" Saúde e paz, o resto a gente corre atrás"

Pedro Bial

Aos meus pais Simonides e Lourdes.

AGRADECIMENTOS

- a Deus, por tudo;
- ao meu pai Simonides, pela referência de garra, educação e por ter propiciado-me condições de poder cursar uma faculdade;
- a minha mãe Lourdes, grande responsável por minha formação social, pelo carinho e por todo o apoio ao longo da minha vida;
- a minha orientadora, Juliana, pela confiança e dedicação, que tanto contribui com minha formação acadêmica;
- ao meu irmão Fernando, que agüentou a luz acesa durante as várias noites de estudos passadas ao longo do curso;
- a minha irmã Wanessa e minha prima Simone, pelo incentivo;
- a minha namorada Paôla, pelo incentivo e auxílio nesse final de curso;
- ao meu chefe Jefferson, referência de profissional, ao meu chefe Marcelo, pelas horas que me liberou para poder me dedicar ao projeto e aos demais do departamento de engenharia da ITSA Telecomunicações: Leandro Assunção, Cláudio Navarro, Francisco de Assis, Elaine, Renildo e Aires;
- aos meus amigos: Diórgenes (e sua mãe Graça), Eduardo Wú, Daniel Andrade, Daniel Araújo, Endrizzo, Michelle, Isabel, Gustavo e Domingos, pelo apoio durante o curso;
- aos meus familiares e demais colegas da faculdade.

RESUMO

O rápido desenvolvimento dos computadores e da área de telecomunicações vem exigindo e propiciando o surgimento de uma geração de aplicações relacionadas com imagens digitais. As aproximações convencionais estão alcançando uma saturação em termos de eficiência. A necessidade de novas técnicas que objetivem a manipulação de informações visuais, demanda o conhecimento das variadas ferramentas de processamento de imagens.

Este trabalho pretende fornecer uma visão geral sobre essa área, despertando o interesse na utilização das técnicas de pré-processamento. A maior parte dos exemplos e algoritmos desenvolvidos, visa a segmentação de imagens, que é a extração de objetos de interesse.

ABSTRACT

The fast development of computers and telecommunications techniques has urged and motivated the arisen of a new generation of applications based on digital image processing. The conventional approaches are reaching saturation in terms of codification efficiency. The need for new techniques aiming the manipulation of visual information, demands the knowledge of the extensive tools of image processing.

This work is devoted to the investigation of digital image processing techniques. It intends to supply a general view of this field, showing the interesting applications of daily image processing techniques. The majority of the examples and developed algorithms aim at the object extraction, or either, the segmentation of the images.

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1. Introdução

Os tópicos abordados neste trabalho foram escolhidos de modo a propiciar uma base ao leitor em processamento de imagens digitais, enfatizando as técnicas de segmentação. Alguns tipos de compressão também foram abordados, objetivando o entendimento de alguns padrões de vídeo.

1.1 Organização do Trabalho

O Capítulo 2 aborda os conceitos básicos de processamento de imagens digitais iniciando com um breve histórico. Em seguida, vários exemplos de aplicações são citados mostrando a sua importância em diversas áreas da ciência. Os fundamentos de representação e geração de uma imagem digital são detalhados num terceiro tópico.

O Capítulo 3 apresenta as técnicas de realce mais comuns. Elas são bastante úteis em qualquer algoritmo desenvolvido para o processamento de imagens, sendo de essencial importância para o pré-processamento de várias aplicações.

O Capítulo 4 apresenta ferramentas matemáticas morfológicas, que são mais adequadas para alguns tipos de processamento sobre as imagens. Vários exemplos são citados. A evolução dessas técnicas tem propiciado a obtenção de bons resultados com um menor tempo de processamento.

O Capítulo 5 cita alguns métodos de segmentação de imagens. As ferramentas básicas apresentadas nos capítulos 3 e 4 são usadas para propiciar a separação de um objeto de interesse da imagem. Algumas técnicas baseadas no movimento também são abordadas.

No Capítulo 6 é intitulado como compressão de imagens, apresentando técnicas de codificação e sistemas de compressão com e sem perdas.

No Capítulo 7 aborda de forma superficial alguns padrões de vídeo, e detalha o H.261 e possibilidades de melhoria do seu sistema.

A conclusão e sugestões para trabalhos futuros são feitas no capítulo 8.

No final se encontram os anexos e o glossário. Eles contêm os códigos de alguns programas utilizados ao longo do projeto, um resumo do padrão H.261 e uma publicação que foi implementada no capítulo 5.

As referências bibliográficas utilizadas se encontram no final de cada capítulo.

1.2 Metodologia de Estudo

O trabalho foi baseado em livros, publicações de especialistas e nos padrões definidos por órgãos de regulamentação.

Algumas imagens exemplo foram tiradas da Internet. Nas fontes onde não havia uma referência impressa, foi utilizada a descrição dos endereços URL. Devido à natureza de constante mudança da internet, não há garantias que as referências continuem disponíveis.

2. Conceitos Básicos

A visão humana está limitada a uma banda do espectro eletromagnético, chamada de banda visível. As câmeras capturam esta faixa do espectro gerando imagens que são percebidas pelo sistema visual humano. Existem máquinas que cobrem outras faixas do espectro, indo dos raios gama às ondas de rádio, criando imagens que não tem associação direta com a percepção humana. Isso gera uma enorme quantidade de aplicações, como o ultra-som, a microscopia eletrônica, entre várias outras que serão citadas adiante.

O interesse em métodos de processamento de imagem deriva de duas principais áreas de aplicação: a melhoria da informação visual para a interpretação humana e o processamento de dados, para armazenamento, transmissão e representação para percepção automática através das máquinas [1].

Apesar de não haver um consenso na delimitação das áreas no processamento computacional de imagens, geralmente, são considerados três níveis: o processamento de baixo-nível, onde há o envolvimento de operações de pré-processamento, como redução de ruído, ajuste de contraste e melhoria na nitidez da imagem, onde tanto a entrada quanto a saída são imagens como um todo; o processamento de nível-médio, envolve tarefas como a segmentação, o ajuste de objetos para um processamento computacional, a classificação e o reconhecimento deles numa imagem, sendo essa tarefa caracterizada pela entrada que geralmente é uma imagem, e a saída que é um objeto de interesse; o último tipo é o de alto-nível, envolve o reconhecimento de objetos com significado semântico, que geralmente requer uma informação contextual e inteligência artificial [1]. Neste trabalho serão abordados processamentos de baixo e médio nível.

2.1 Histórico

Uma das primeiras aplicações de técnicas de processamento de imagens foi no melhoramento de imagens digitalizadas para jornais, que eram enviadas por meio de cabo submarino de Londres para Nova York. Esse sistema desenvolvido no século 20 reduziu o tempo requerido de mais de uma semana, para menos de três horas no transporte de imagens no Oceano Atlântico. Um equipamento especializado de impressão codificava as imagens para transmissão a cabo, as quais eram reconstruídas no receptor, onde havia a necessidade de uma melhor distribuição dos níveis de brilho. Este exemplo não se enquadra totalmente na definição de processamento de imagem digital, pelo fato dos computadores terem surgido 35 anos depois dessa primeira aplicação, mas sem dúvida, esse foi

um marco no surgimento e desenvolvimento de várias técnicas que foram colocadas em prática com os computadores digitais de grande porte.

A utilização de técnicas de computação para o melhoramento de imagens produzidas por uma sonda espacial iniciou-se no *Jet Propulsion Laboratory*, na Califórnia, USA, em 1964, quando imagens da Lua, transmitidas pelo Ranger 7, foram processadas por um computador para corrigir vários tipos de distorção de imagens inerentes à câmera de televisão de bordo. Essas técnicas serviram de base para métodos melhorados no realce e restauração de imagens das missões *Surveyor* para a Lua, a série Mariner de missões para Marte e os vôos tripulados da Apolo para a Lua [1].

Desde então, a área de processamento de imagens vem crescendo com toda a força; em medicina, por exemplo, procedimentos computacionais melhoram o contraste ou codificam os níveis de intensidade em cores, de modo a facilitar a interpretação de imagens de raios X e outras imagens biomédicas. Geógrafos usam técnicas similares para estudar padrões de poluição em imagens aéreas ou de satélites. O realce e a restauração de imagens podem ser aplicados na arqueologia, na astronomia, biologia, medicina nuclear, apoio à lei, aplicações industriais, dentre várias outras áreas.

O aumento da freqüência com a qual as informações vêm sendo transmitidas, armazenadas, processadas, e visualizadas no formato digital, a acentuada queda de preços que os dispositivos eletrônicos vem sofrendo, principalmente os da área de processamento digital de sinais e a melhoria da performance e da expansão de redes de comunicação mais velozes, estão criando oportunidades sem precedentes na área de processamento de imagens digitais, bem como a necessidade do desenvolvimento de técnicas eficientes para todas essas tarefas, de forma a melhorar a integridade visual da informação de interesse.

No próximo tópico serão citados alguns exemplos de processamento de imagens digitais em diversas áreas da ciência.

2.2 Exemplos de Processamento de Imagens Digitais

O processamento de imagens digitais vem trazendo grandes facilidades em diversas áreas, pois a visualização das características de interesse tem sido possíveis devido ao constante aprimoramento que as técnicas de aquisição e tratamento de imagens têm alcançado. Devido a essa grande variedade, uma maneira simples de agrupar as imagens é classificá-las de acordo com a sua fonte de energia. A principal é o espectro de energia (Figura 2.1). Outras fontes são de origem acústica, ultra-sônica, eletrônica (microscópio eletrônico) e imagens sintéticas que são aquelas geradas pelo computador.

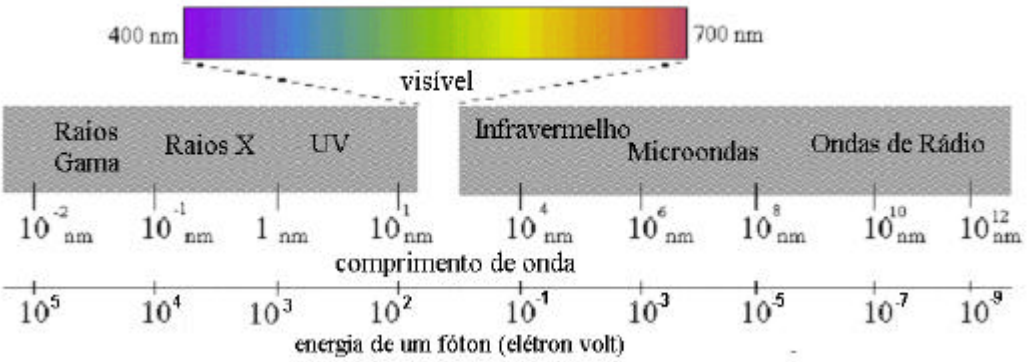


Figura 2.1: Espectro eletromagnético organizado de acordo com a energia por fóton e comprimento de onda.

As imagens mais familiares são as pertencentes às bandas visíveis e as de raios X.

Na área militar, nos sistemas de defesa e inteligência, há aplicações na interpretação automática de imagens de satélite e no reconhecimento e rastreamento de alvo em tempo real. No processamento de documentos, existe a varredura, arquivamento e transmissão de documentos, a detecção e reconhecimento automático de caracteres impressos. Na automação de fábricas, se usa a inspeção visual de produtos, para que haja a verificação da qualidade dos mesmos. No apoio à lei, têm-se: a identificação de pessoas através do processamento automático das impressões digitais; o reconhecimento facial; o casamento de DNA; a proteção de copyright com a utilização de marca d´água; e a segurança de dados com o uso da codificação.

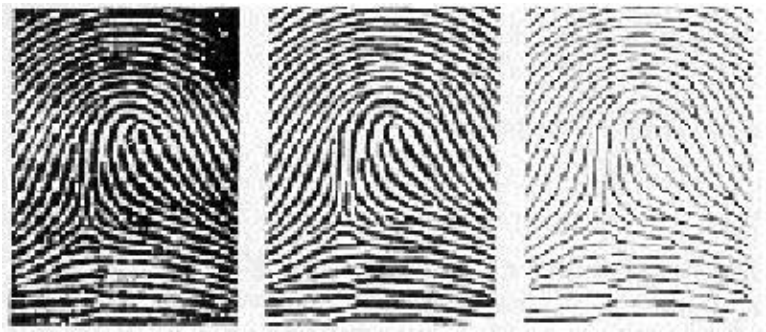
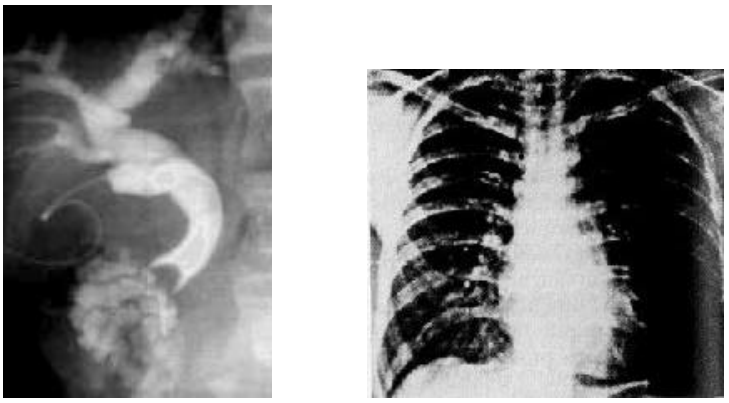


Figura 2.2. Exemplo de tratamento de uma impressão digital [1].

Na Biologia tem-se uma vasta quantidade de aplicações como a análise automática de amostras biológicas e a análise de ossos, tecidos e células. É na área médica que comumente há o relacionamento das imagens com sua origem, baseada no espectro eletromagnético. A geração de imagens através dos raios gamas inclui a medicina nuclear e observações astronômicas. Outras imagens bastante conhecidas são as baseadas em raios X, que são usadas em diagnósticos médicos e também na astronomia. A Figura 2.3 exemplifica duas aplicações desse tipo de fonte. O

desenvolvimento da câmara de cintilação por Hal Anger em 1952, conhecida como a câmara Anger, é a base da medicina nuclear e do desenvolvimento da tomografia axial computadorizada (CAT ou CT), que também é proveniente do uso dos raios X [2].



(a) **(b)**

Figura 2.3. (a) Angiografia com realce de contraste [2] e (b) Radiografia de tórax [1].

Aplicações da faixa de ultravioleta são variadas, e inclui litografia, inspeção industrial, microscópios, lasers, observações astronômicas e análises na área de biologia.

As imagens oriundas da banda visível, que são as mais comuns, englobam inúmeras aplicações. As imagens da faixa de infravermelho são comumente usadas com as da faixa visível para a obtenção de resultados específicos. A Tabela 2.1 mostra um exemplo de aplicação.

Tabela 2.1: Bandas temáticas definidas no satélite LANDSAT da NASA [1].

Nome	Usos
Azul Visível	Maior penetração na água
Verde visível	Mede o vigor de plantas
Vermelho visível	Discriminação de vegetação
Perto do Infravermelho	Mapeamento de biomassa
Infravermelho médio	Umidade de solos e plantas
Infravermelho térmico	Mapeamento de temperatura
Infravermelho alto	Mapeamento de minerais

Na faixa de microondas a aplicação mais usada é o radar, que tem a característica de coletar dados de uma região independente do clima ou condições de iluminação. Algumas ondas de radares podem penetrar roupas, vegetações, gelo e areia. Na faixa de rádio encontram-se, novamente, aplicações na astronomia e na medicina. Em 1966, os trabalhos do Suíço Richard R. Ernest com a aplicação da modulação com sinais de rádio e o uso da transformada de Fourier no sinal da NMR (ressonância nuclear magnética) lhe garantiu o prêmio Nobel em química em 1991. Este desenvolvimento pôde finalmente ser utilizado em 1973 por Damadian e Lauterbur na geração das primeiras imagens por ressonância magnética (MRI), Figura 2.4 [2].

Figura 2.4: MRI evidenciando veias e órgãos [2].

Mudando a fonte de geração de imagens, tem-se a ultra-sonografia, que é comumente usada na área de obstetrícia, Figura 2.5.

Outro exemplo é a microscopia eletrônica que gera imagens onde se evidencia o formato, e dá a sensação de profundidade de células e outros objetos de interesse.

Apenas alguns dentre vários exemplos foram citados, mostrando a vasta quantidade de aplicações possíveis e a importância que o processamento de imagens digitais tem na vida de todos.

No próximo tópico haverá uma breve explicação dos passos fundamentais dessa área de processamento.

Figura 2.5: Ultra-sonografia de um bebê na barriga da mãe [3].

2.3 Fundamentos de Imagens Digitais

O desenvolvimento de um modelo de representação no processamento de uma imagem digital tem suas origens relacionadas com a capacidade visual do olho humano de distinguir detalhes, que é uma função da disposição dos elementos fotossensíveis dentro da retina. Esta capacidade é denominada de acuidade visual, sendo ela o fator determinante da quantização espacial e da resolução horizontal e vertical que a imagem deve ter para que apareçam os detalhes de interesse em uma aplicação específica.

A aquisição da imagem é o primeiro passo nessa área. Ela é feita através de sensores que captam radiações refletidas ou alteradas (aquelas que atravessam um objeto) transformando-as em um sinal elétrico. Depois dessa etapa, é feita a digitalização desse sinal, ou seja, faz-se uma conversão analógico-digital que é composta da amostragem e quantização, obtendo-se no final a imagem digital.

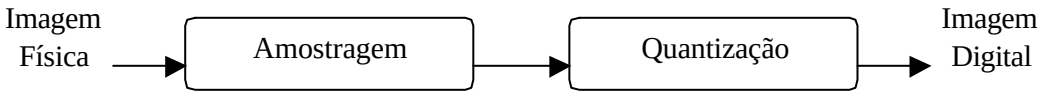


Figura 2.6 – Etapas de Digitalização de uma Imagem

Uma imagem pode ser definida como uma função bidimensional *f(x,y)*, onde x e y são coordenadas espaciais num plano bidimensional, e *f* representa a intensidade (brilho) ou o nível de cinza da imagem naquele ponto. Quando x, y e *f* são valores finitos e discretos, a imagem passa a ser digital. O campo de processamento de imagens digitais se refere ao processamento dessas pelos computadores digitais. Quando *f(x,y)* está numa posição particular e com um valor específico, ele é referenciado como elemento da imagem, ou pel, ou, o termo mais usado, pixel. Essa função pode ser

caracterizada por duas componentes, que são chamadas de iluminação e reflectância, representados por $i(x,y)$ e $r(x,y)$, logo:

$$\begin{aligned} f(x,y) &= i(x,y).r(x,y) \\ 0 &< i(x,y) < \infty \\ 0 &< r(x,y) < 1 \end{aligned} \qquad \textbf{(2.1)}$$

Ao longo deste trabalho, os algoritmos são feitos para imagens monocromáticas. Logo cada intensidade de f será denominado de nível de cinza (l). Com isso, l ficará restrito em um intervalo:

$$L_{\min} \leq l \leq L_{\max} \qquad \textbf{(2.2)}$$

O intervalo $[L_{\min}, L_{\max}]$ é denominado escala de cinza. Na prática se desloca esse intervalo para $[0, L]$, onde $l = 0$ é considerado negro e $l = L$ é o branco [1].

É claro que as imagens visualizadas por todos diariamente não são limitadas a níveis de cinza, pois a cor é um aspecto extremamente importante, ela transmite uma variedade de informações que melhor descrevem, de forma qualitativa, os objetos constantes numa imagem. No olho humano, as cores são percebidas pela combinação de comprimentos de ondas curtos, médios e longos, que correspondem as três cores primitivas: vermelho, verde e azul [4]. Essas cores são usadas em vários padrões de câmeras de vídeo, que é conhecido como RGB. A formação das outras cores no olho humano se dá através da adição dessas componentes.

O fato da maioria dos algoritmos serem desenvolvidos para imagens em níveis de cinza, não limita a aplicação dos mesmos. A imagem colorida pode ser dividida em três camadas, cada uma correspondendo a uma das cores primitivas RGB. Aplica-se, então, o algoritmo a cada uma delas e no final combinam-se as três camadas processadas, obtendo a imagem colorida.

A amostragem é o processo de conversão do sinal no espaço contínuo em um sinal no espaço discreto. Suponha que uma imagem contínua $f(x,y)$ é aproximada por amostras igualmente espaçadas, arranjadas na forma de uma matriz NxM como mostrada na Equação (2.3).

Erro! Não é possível criar objetos a partir de códigos de campo de edição. **(2.3)**

Cada elemento dessa matriz representa um elemento de imagem, ou seja, um pixel.

A outra parte da digitalização é a quantização que faz a conversão de um valor contínuo em um valor discreto, em termos da amplitude. Uma imagem depois de amostrada fica discretizada no tempo, mas continua com uma infinidade de amplitudes, então se define uma quantidade de níveis de

cinza que representa bem a imagem de interesse, a partir disso têm-se os níveis de quantização que correspondem aos níveis de cinza. A escolha do tamanho da máscara de amostragem, ou seja, do tamanho NxM, e da quantidade de níveis de quantização definem o tamanho da imagem sem nenhum tipo de compressão. Logo o número de bits necessários para armazenar uma imagem é dado por:

$$b = NxMxm$$
$$G = 2^m \qquad \textbf{(2.4)}$$

onde G é o número de níveis de cinza.

A resolução de uma imagem depende fortemente do processo de digitalização, a Figura 2.7 mostra o efeito de se amostrar uma imagem com diferentes tamanhos de máscara. As imagens foram redimensionadas para um só tamanho, para facilitar a visualização dos efeitos. Com a diminuição da resolução espacial nota-se uma perda da definição das bordas, ou seja, perdem-se detalhes de cada objeto, devido ao efeito xadrez que a imagem sofre.

Já a Figura 2.8 mostra os efeitos da redução dos níveis de cinza na representação das imagens. Com essa diminuição dos níveis de quantização, nota-se o surgimento de sulcos. Esse efeito é denominado de falso contorno.

(a) **(b)** **(c)**

Figura 2.7: (a) Imagem Original de 256x256 com 256 níveis de cinza, (b) Imagem amostrada novamente a 64x64 e (c) Imagem amostrada novamente a 32x32.

(a) **(b)** **(c)**

Figura 2.8: (a) Imagem com 64 níveis de cinza. (b) Imagem com 16 níveis de cinza. (c) Imagem com 04 níveis de cinza.

Outros fundamentos importantes no processamento de imagens digitais são os relacionamentos entre os pixels. Um pixel pode ter vizinhos horizontais, verticais, e diagonais, com isso, pode-se definir que se um ou mais desses vizinhos tiverem características semelhantes, como a cor, por exemplo, eles estarão conectados. Essa propriedade de conectividade é um conceito importante no estabelecimento das bordas de um objeto e componentes das regiões de uma imagem.

A rotulação, onde há a criação de uma máscara que marca os pixels de interesse, bem como medidas de distância e operações lógico-aritméticas entre os pixels, são operadores bastante utilizados nos algoritmos de processamento de imagens. Podemos citar como transformações básicas, a translação, a mudança de escala e a rotação.

No próximo capítulo será feita uma revisão sobre as técnicas de realce, que são úteis em quase todas as aplicações de processamento de imagens digitais.

2.4 Referências

[1] R. C. Gonzalez e R. E. Woods, “Digital Image Processing”, Prentice Hall, 2ª edição, ISBN: 0-201-18075-8, 2002.

[2] Imagens Médicas na página da Universidade Federal do Rio Grande do Sul, disponível em http://www.if.ufrgs.br/ast/med/imagens/node2.htm.

[3] http://www.parenthood.com

[4] Al BOVIK, “Introduction to Digital Image and Video Processing”, 2000

3. Realce de Imagens Digitais

As técnicas de realce têm como principal objetivo o processamento de uma imagem de modo que o resultado seja mais adequado a uma aplicação específica, ou seja, as técnicas são empregadas de acordo com necessidades peculiares. O julgamento do efeito da técnica empregada é feito visualmente, onde o propósito do realce é avaliado de forma subjetiva.

O realce de imagens pode ser dividido em duas categorias: no domínio espacial, onde se trabalha no próprio plano da imagem, manipulando diretamente os pixels da imagem; e no domínio da freqüência, onde as técnicas são baseadas em modificações das transformadas de Fourier [1]. Neste capítulo serão abordadas estas duas categorias.

Existem diversas técnicas de realce, mas neste trabalho serão citadas aquelas mais utilizadas.

3.1 Realce de imagens no domínio espacial

Um processo no domínio espacial pode ser denotado pela seguinte expressão:

$$g(x, y) = T[f(x, y)] \qquad \textbf{(3.1)}$$

onde $f(x, y)$ é a imagem de entrada, $g(x, y)$ é a imagem processada e T é um operador sobre f definido sobre alguma vizinhança (x, y), ou sobre um conjunto de imagens.

Uma técnica bastante comum e fácil de ser aplicada é o negativo de uma imagem, onde há a reversão do preto com o branco, de modo que a intensidade de saída diminua à medida que a de entrada aumente. Outra transformação é a compressão da escala dinâmica que pode ser dada por curvas logarítmicas, por potenciação, ou mesmo por exponenciação, sendo essa última, conhecida como correção gama. A Figura 3.1 mostra um exemplo de transformada negativa e de correção gama.

(a) **(b)** **(c)**

Figura 3.1: (a) Imagem original. (b) Negativo da Imagem original. (c) Imagem original com correção gama.

Pode-se fazer também o fatiamento das intensidades, objetivando um maior contraste para uma região em específico. Um exemplo disso é a aplicação de um valor limiar (*threshold*), através do qual todos os pixels com nível abaixo desse valor vão para preto, e todos acima, para o branco.

Para uma representação matemática das técnicas de realce, pode-se definir "r" como o nível de cinza de entrada e "s" como o nível de cinza de saída, ambos na faixa de $[0, L-1]$, sendo L o número total de níveis de cinza, [1]. A Figura 3.2 mostra algumas transformadas.

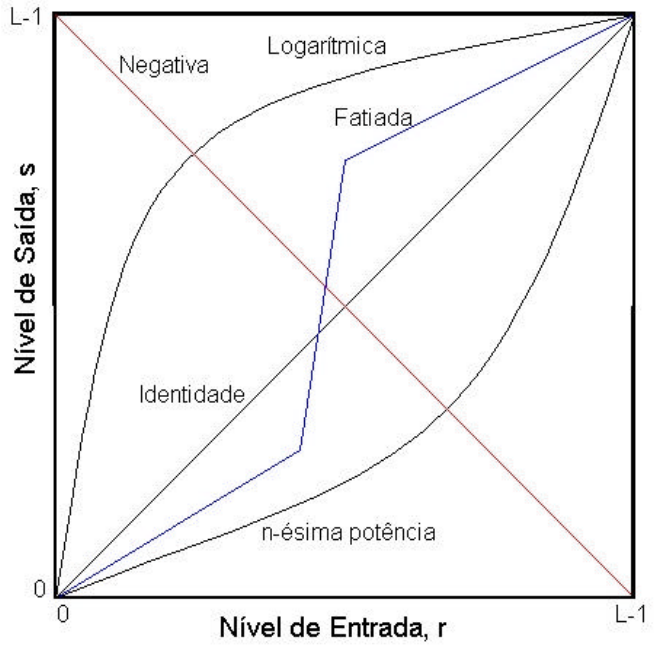


Figura 3.2: Algumas transformações básicas de níveis de cinza.

Uma importante função no realce de imagens é o processamento de histograma. Ele é um gráfico que indica quantos pixels, pertencentes a uma imagem, estão num certo nível de cinza. O histograma pode ser mostrado de forma normalizada, onde a quantidade de pixels com um determinado nível de cinza é dividida pelo número total de pixels. A Figura 3.3 mostra o histograma de uma imagem.

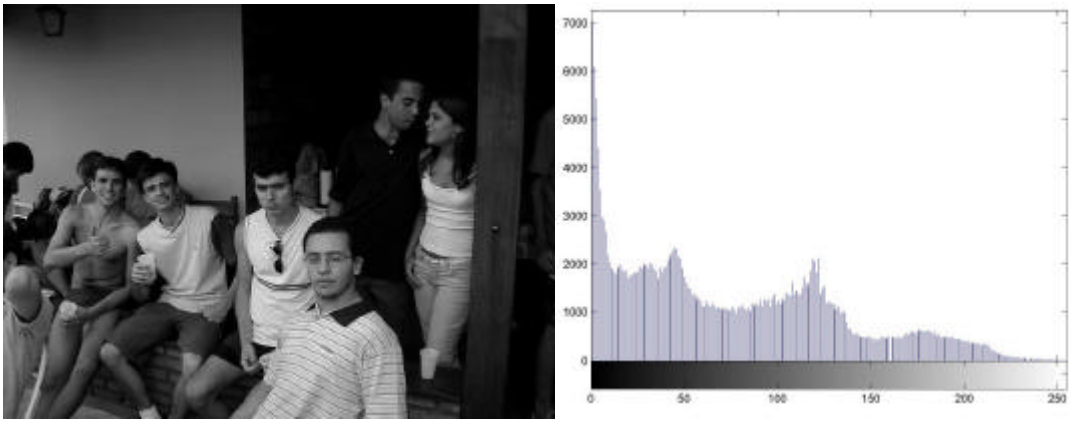


Figura 3.3: Uma imagem e seu respectivo histograma.

Dentre as operações não-lineares tem-se a equalização de histograma [2], que através da função de densidade de probabilidade equaliza uma imagem de modo a melhorar sua visualização. A Figura 3.4 mostra a imagem da Figura 3.3 equalizada.

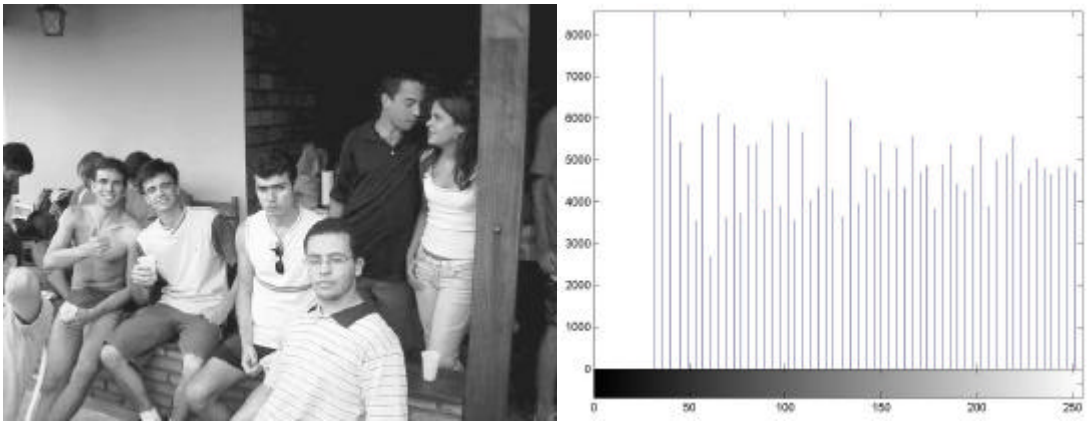


Figura 3.4: A imagem equalizada e seu respectivo histograma.

Pode-se observar que através do histograma dá para dizer se a imagem é escura, clara, de baixo ou de alto contraste.

Para a manipulação do histograma de uma imagem ser mais útil em determinadas aplicações é desejável poder especificá-lo de modo a realçar certas escalas de cinza, através da determinação da

função de densidade desejada. Isso se torna possível, fazendo com que essa manipulação torne-se uma poderosa ferramenta.

Os métodos discutidos até agora são ditos globais, pois os pixels são modificados através de uma função de transformação baseada na distribuição dos níveis de cinza sobre uma imagem completa. Uma outra maneira de utilizar essa ferramenta é o realce de pequenas áreas, onde a alteração de cada pixel vai depender dos seus vizinhos. A média de intensidade e a variância são propriedades usadas para modificar um pixel de acordo com a sua vizinhança, onde a média é uma medida de brilho e a variância de contraste.

Operações lógicas e aritméticas também são bastante usadas. A subtração de imagens por exemplo, pode ser eficientemente usada na detecção de diferenças de uma imagem para outra, e se tornar uma ferramenta na detecção de movimento [2]. A média aritmética de imagens, bem como a mediana, pode ser usada para a redução do ruído. A Figura 3.5 mostra uma imagem com ruído aditivo gaussiano, depois mostra essa imagem com realce local através da média de pixels vizinhos e por último a média de doze imagens com ruído distribuído de forma aleatória. Quanto maior a quantidade de imagens usadas na média aritmética, melhor é o resultado obtido.

(a) **(b)** **(c)**

Figura 3.5: (a) Imagem com ruído. (b) Imagem com realce local – Média. (c) Imagem obtida da média de doze imagens com ruído.

Outra técnica utilizada é o uso de máscaras espaciais, conhecidas por filtros espaciais. Através da definição de valores dentro dessa máscara, obtêm-se diversos efeitos de realce. A Figura 3.6 mostra o processo de filtragem espacial, onde:

$$z_5' = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 \qquad \textbf{(3.2)}$$

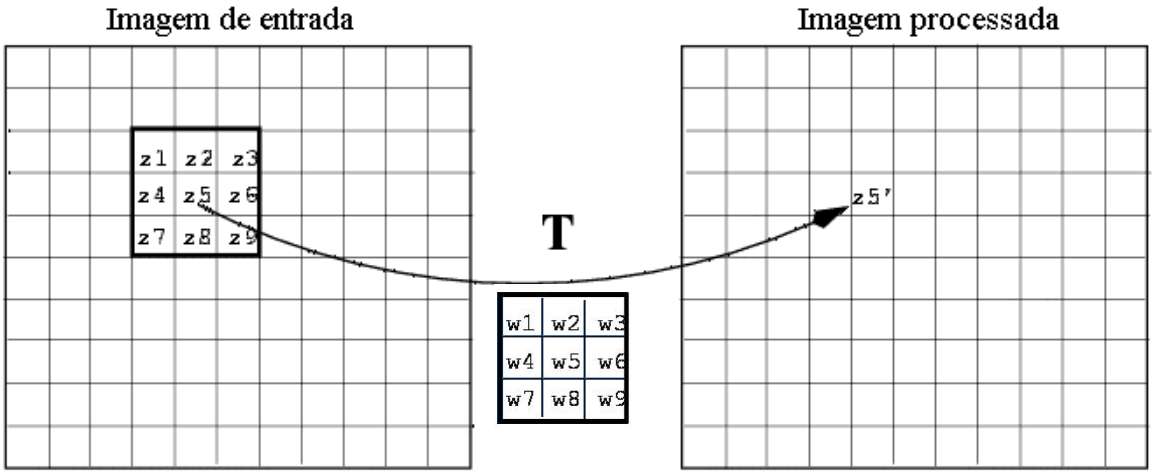


Figura 3.6: Filtragem de um pixel por uma máscara de tamanho 3x3 (w_1 a w_9).

Filtros como os de suavização são usados para o borramento, que tira pequenos detalhes de uma imagem antes da extração de objetos maiores. Filtros por derivadas têm efeito contrário, ou seja, há o aguçamento de detalhes na imagem, como por exemplo, o uso do gradiente, que será utilizado mais adiante, em alguns processos de segmentação. O gradiente de f nas coordenadas (x, y) é definido como:

$$\nabla f = \left[\frac{\partial f}{\partial x}; \frac{\partial f}{\partial y}\right] \qquad \textbf{(3.3)}$$

Figura 3.7: Aplicação do gradiente (Sobel) na detecção de bordas.

A segunda derivada também é utilizada na construção de filtros isotrópicos, que tem uma resposta independente da direção das descontinuidades da imagem, ou seja, invariante a rotações da imagem.

3.2 Realce de Imagens no Domínio da Freqüência

O conhecimento da transformada de Fourier e conseqüentemente representações no domínio da freqüência é de essencial importância para um bom entendimento de diversas aplicações em processamento de imagens.

Antes de descrever algumas técnicas de realce no domínio da freqüência, será abordado a Transformada Discreta de Fourier (DFT) e sua relação com características de uma imagem.

Para que a transformada de Fourier seja útil no processamento de imagens, faz-se uma extensão dela para duas variáveis. Logo.

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2p(ux+vy)}dxdy \qquad \textbf{(3.4)}$$

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2p(ux+vy)}dudv \qquad \textbf{(3.5)}$$

onde $F(u,v)$ é a transformada de Fourier de $f(x,y)$ [1].

Como a aplicação é sobre funções discretas, usa-se a Transformada Discreta de Fourier, dada por:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2p\left(\frac{ux}{M}+\frac{vy}{N}\right)} \qquad \textbf{(3.6)}$$

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2p\left(\frac{ux}{M}+\frac{vy}{N}\right)} \qquad \textbf{(3.7)}$$

O espectro de Fourier, o ângulo de fase e o espectro de potência são dados, respectivamente, pelas equações (3.8), (3.9) e (3.10).

$$|F(u,v)| = \left|R^2(x,y) + I^2(x,y)\right|^{1/2} \quad \textbf{(3.8)}$$

$$f(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right] \quad \textbf{(3.9)}$$

$$P(u,v) = R^2(u,v) + I^2(u,v) \quad \textbf{(3.10)}$$

onde $R(u,v)$ e $I(u,v)$ é a parte real e imaginária de $F(u,v)$, respectivamente.

O valor da transformada no ponto $(u,v) = (0,0)$ é chamado de valor médio. Ele na imagem corresponde a média dos níveis de cinza.

Pode-se observar que a transformada de Fourier contem todos os valores de $f(x,y)$ modificados por uma exponencial. Isso dificulta a relação direta de componentes específicos de uma imagem com a sua transformada. Mas em contrapartida, sabe-se que a freqüência é diretamente relacionada com a taxa de mudança, ou seja, isso permite estabelecer analogia das freqüências contidas na transformada, com as variações de intensidade numa imagem. Nesse âmbito as baixas freqüências correspondem às pequenas variações contidas numa imagem, ou seja, aqueles locais onde as mudanças de níveis de cinza são pequenas, suaves, por outro lado, as altas freqüências correspondem às variações mais bruscas, sendo geralmente as bordas dos objetos dentro de uma imagem.

Para uma melhor visualização da transformada de Fourier, multiplica-se a imagem por $(-1)^{x+y}$ com o intuito de se centralizar o espectro.

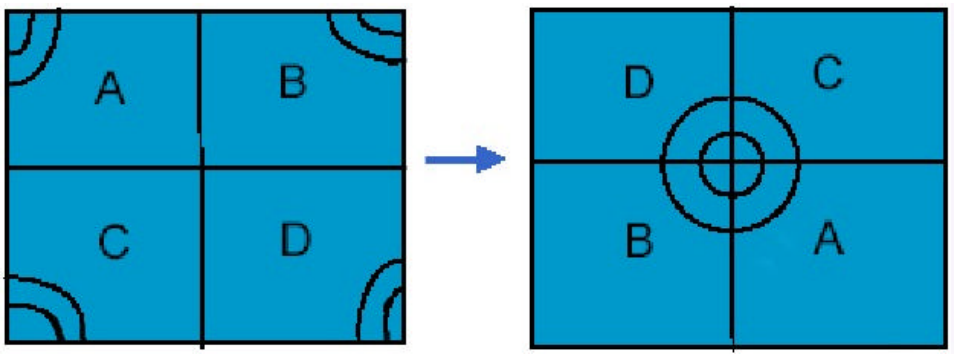
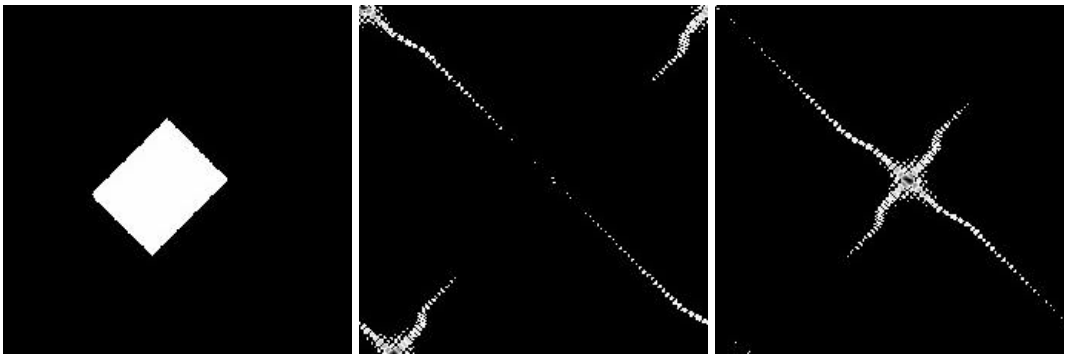


Figura 3.8: Centralização do espectro pela multiplicação da imagem por $(-1)^{x+y}$.



(a) **(b)** **(c)**

Figura 3.9: (a) Exemplo de uma imagem $f(x,y)$. (b) Sua transformada $|F(u,v)|$. (c) Centralização do espectro com $|F[f(x,y).(-1)^{x+y}]|$.

As técnicas de realce no domínio da freqüência são feitas através do uso de filtros que correspondem a funções de transferência $H(u,v)$. Eles suprimem certas freqüências de acordo com o propósito de cada um.

Os passos básicos para uma filtragem nesse domínio são os seguintes [1]:

1. Multiplica-se a imagem de entrada por $(-1)^{x+y}$;
2. Calcula-se a DFT da imagem obtida no passo 1;
3. Multiplica $F(u,v)$ pela função do filtro $H(u,v)$;
4. Calcula-se a DFT inversa da imagem obtida no passo 3;
5. Separa-se a parte real do resultado obtido no passo 4;
6. Multiplica o resultado obtido no passo 5 por $(-1)^{x+y}$.

A utilização de filtros passa-baixas causa uma suavização na imagem, pois corta as freqüências que caracterizam as mudanças bruscas de nível de cinza. Dentre os filtros de suavização temos: o passa-baixas ideal, o Butterworth e o Gaussiano. O filtro passa-altas tem efeito contrário, ele aguça a imagem. Alguns filtros de aguçamento são duais dos passa-baixas, logo, têm-se também o passa-altas ideal, o Butterworth e o Gaussiano. Outro tipo bastante utilizado é o Laplaciano, que realça a imagem através da equação (3.11).

$$g(x,y) = f(x,y) - \nabla^2 f(x,y) \quad \textbf{(3.11)}$$

onde $f(x,y)$ e $g(x,y)$ é a imagem original e aguçada respectivamente.

(a) **(b)** **(c)**

Figura 3.10: (a) Imagem original. (b) Imagem suavizada. (c) Imagem aguçada.

3.3 Relação entre os Domínios e Propriedades da FFT

Como foi visto, é possível se trabalhar tanto no âmbito espacial quanto no da freqüência. A relação fundamental entre esses dois domínios é o teorema da convolução. Uma convolução no domínio espacial equivale a uma multiplicação no domínio da freqüência e vice-versa. Logo, considerando $f(x,y)$ a imagem original, e $h(x,y)$ o filtro que se vai utilizar, têm-se:

$$f(x,y) * h(x,y) \Leftrightarrow F(u,v).H(u,v) \quad \textbf{(3.12)}$$

$$f(x,y).h(x,y) \Leftrightarrow F(u,v) * H(u,v) \quad \textbf{(3.13)}$$

A convolução discreta de duas funções de tamanho MxN, é dada pela seguinte expressão:

$$f(x,y) * h(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n).h(x-m, y-n) \quad \textbf{(3.14)}$$

O desenvolvimento da transformada rápida de Fourier revolucionou e possibilitou o desenvolvimento de diversas técnicas em toda a área de processamento de sinais, pois reduziu significativamente o número de operações para realizar a transformada de Fourier, além disso possui propriedades que facilitam o trabalho com imagens, como a translação, o escalonamento, a rotação, a

periodicidade e a separabilidade, sendo que essa última característica, permite o cálculo da transformada bidimensional através do cômputo de duas transformadas unidimensionais, facilitando a implementação de algoritmos aplicáveis em imagens.

Existem outras transformadas além da FFT, como a de Walsh-Hadamard, cosseno discreta (DCT), Haar, Slant, LOT, MLT, ELT, Wavelets entre outras. A DCT será explicada no capítulo de compressão.

No próximo capítulo será abordado o processamento morfológico de imagens, que aborda uma matemática mais apropriada para imagens.

3.4 Referências

[1] R. C. GONZALES, R. E. WOODS, "Digital Image Processing", Prentice Hall, 2ª edição, ISBN: 0-201-18075-8, 2001.

[2] Al BOVIK, "Basic Gray-Level Image Processing", Handbook of Image and Video Processing, Academic Press, 2000.

4. Processamento Morfológico de Imagens

As transformadas abordadas no capítulo anterior foram inicialmente definidas para sinais unidimensionais e mais tarde adaptadas para imagens. Logo, elas não privilegiam as formas dos objetos, e nem sempre são as mais adequadas.

A morfologia matemática é uma ferramenta para a extração de componentes de imagens que sejam úteis na representação e descrição da forma de uma região [1]. Ela foi inicialmente (década 60) desenvolvida para trabalhar com imagens binárias, tendo como primeira aplicação, o estudo de porosidade de minerais. Mais tarde, foi estendida para imagens de níveis de cinza e atualmente há estudos no contexto de imagens coloridas [2].

A linguagem da morfologia matemática se baseia na teoria dos conjuntos, por isso, oferece uma abordagem unificada e poderosa para numerosos problemas. Em imagens binárias, os conjuntos em questão são membros do espaço bidimensional de números inteiros Z^2 em que cada elemento do conjunto é um vetor bidimensional cujas coordenadas (x, y) correspondem aos pixels pretos. Imagens de níveis de cinza podem ser representadas por conjuntos cujos componentes estejam em Z^3, tendo dois elementos indicando a posição, e um terceiro correspondendo ao valor discreto da intensidade. Abordagens com conjuntos de maiores dimensões podem ser usados para conter outros atributos, como a cor e componentes que variem com o tempo [1].

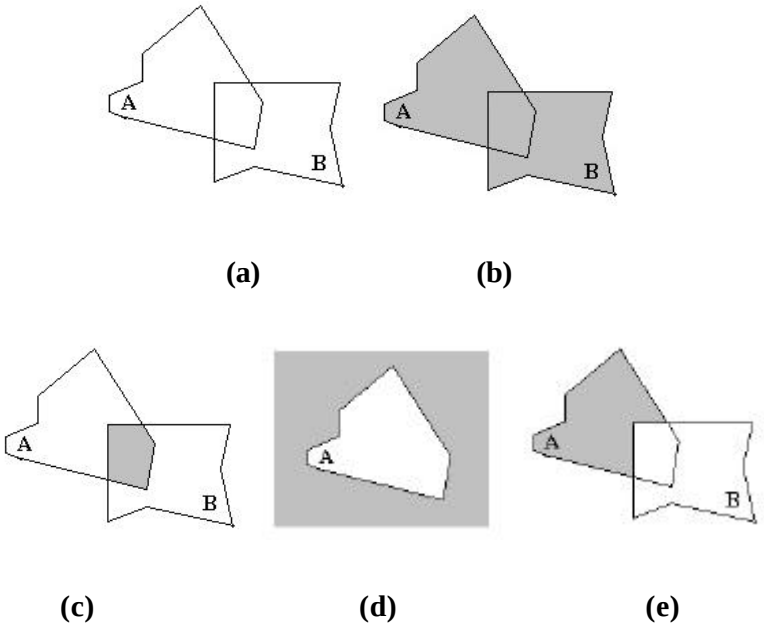
Os operadores de morfologia matemática quando usados corretamente preservam as características essenciais das formas de um objeto e removem detalhes irrelevantes. Análise de imagens microscópicas, inspeção visual e reconhecimento de caracteres são domínios de aplicações.

Neste capítulo serão abordados os conceitos básicos de morfologia para imagens binárias, com uma posterior extensão para imagens em níveis de cinza. Alguns algoritmos usados para exemplificar o modo de atuação dos operadores estão descritos no Anexo.

4.1 Conceitos Preliminares

Considere os conjuntos A e B de Z^2. Se $a = (a_1, a_2)$ é um elemento de A e não é de B, então pode-se afirmar que $a \in A$ e $a \notin B$. Se todos os elementos de A pertencerem a B, então diz-se

que A está completamente contido em B, $A \subseteq B$. A união de dois conjuntos A e B é denotado por $A \cup B$ e a intersecção por $A \cap B$. Se os conjuntos forem mutuamente exclusivos, tem-se $A \cap B = \varnothing$ (vazio). O complemento de A é formado por todos os elementos que não estão contidos nele, logo $A^c = \{w | w \notin A\}$. A diferença entre A e B é denotada por $A - B = \{w | w \in A, w \notin B\} = A \cap B^C$.

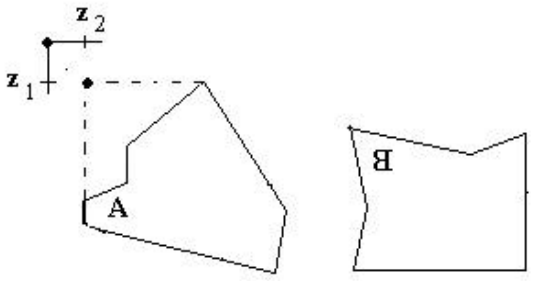


(a) **(b)**

(c) **(d)** **(e)**

Figura 4.1: a) Conjuntos A e B. (b) $A \cup B$. (c) $A \cap B$. (d) A^c. (e) $A - B$.

Outras duas definições adicionais que são amplamente usadas em morfologia são a reflexão e a translação. A reflexão de B, é dada por: $\widehat{B} = \{w | w = -b\}$ para $b \in B$. A translação do conjunto A para o ponto $z = (z_1, z_2)$, é definido por $(A)_c = \{c | c = a + z\}$ para $a \in A$.



(a) **(b)**

Figura 4.2: (a) $(A)_c$. (b) $\widehat{B}$.

Os operadores lógicos são ferramentas poderosas na matemática morfológica sobre imagens binárias. Os mais utilizados são o NOT, AND e OR. Outras funções podem ser geradas a partir da combinação desses operadores básicos.

Tabela 4.1: Os três operadores lógicos de operações binárias.

x_1	x_2	x_1 AND x_2	x_1 OR x_2	NOT x_1
		$(x_1.x_2)$	$(x_1 + x_2)$	$\overline{x}_1$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

A Lei DeMorgan é bastante utilizada, afirmando que [3]:

$$NOT[AND(x_1,...,x_n)] = OR[NOT(x_1),...,NOT(x_n)] \quad \textbf{(4.1)}$$

$$NOT[OR(x_1,...,x_n)] = AND[NOT(x_1),...,NOT(x_n)] \quad \textbf{(4.2)}$$

Com isso, a matemática morfológica atua sobre os objetos usando operações lógicas, que são aplicadas por máscaras denominadas de elementos estruturantes. O seu formato depende do resultado que se quer obter nos objetos contidos na imagem. Inicialmente define-se o seu tamanho e seu ponto de origem.

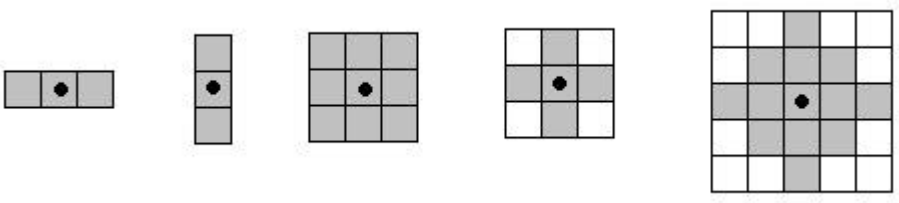


Figura 4.3: Exemplos de elementos estruturantes com origem no centro.

Nos tópicos seguintes, poderão ser observados os resultados da aplicação de diferentes elementos estruturantes.

4.2 Dilatação e Erosão

As duas principais operações em morfologia matemática são a dilatação e a erosão, elas estão presentes na maioria dos algoritmos.

A dilatação de uma imagem binária A pelo elemento estruturante B é definida por:

$$A \oplus B = \delta_B(A) = \left\{ z \middle| (\hat{B})_z \cap A \neq \varnothing \right\} \tag{4.3}$$

que nada mais é que uma operação OR entre A e B.

Como o próprio nome diz, a dilatação aumenta os objetos. Ela tem o efeito de eliminar buracos menores que o elemento estruturante, podendo também, conseqüentemente, diminuir o número de componentes presentes na imagem.

A erosão de uma imagem binária A pelo elemento estruturante B é definida por:

$$A \ominus B = \varepsilon_B(A) = \left\{ z \middle| (B)_z \subseteq A \right\} \tag{4.4}$$

A erosão é simplesmente uma operação AND entre A e B. Ela tem a característica de diminuir o tamanho dos objetos, podendo até mesmo eliminá-los caso sejam menores que o elemento estruturante. O número de componentes pode aumentar na imagem.

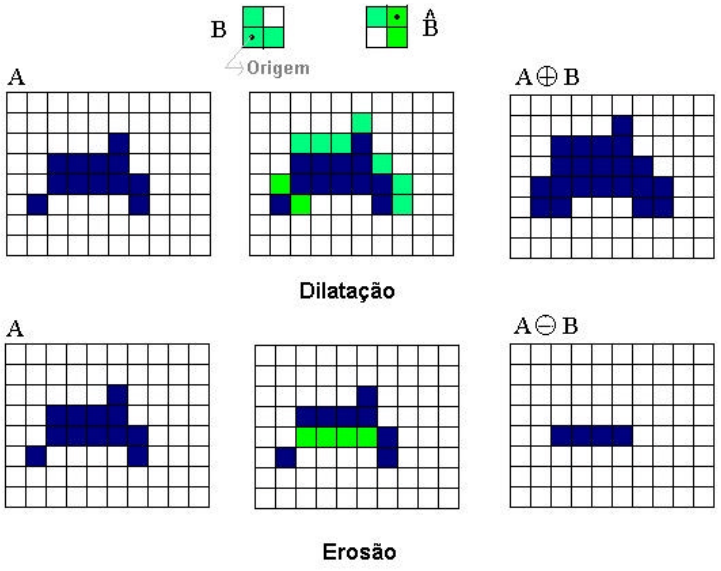
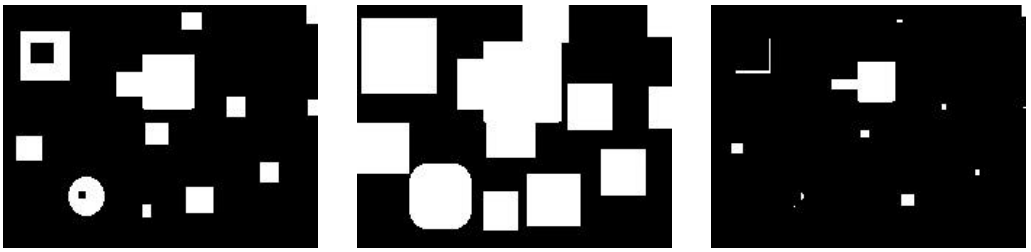


Figura 4.4: Processos de dilatação e erosão de uma imagem A pelo elemento estruturante B. Os pixels pintados representam nível 1.



(a) **(b)** **(c)**

Figura 4.5: (a) Imagem original. (b) Imagem com dilatação. (c) Imagem com erosão.

Pode-se observar na Figura 4.5 que na dilatação os objetos aumentaram e alguns chegaram a se conectar. Na erosão, eles diminuíram e os componentes menores foram quase eliminados. Nota-se também, o uso de um elemento estruturante de forma quadrada, pois o círculo da imagem original teve seu formato modificado, enfatizando que a escolha desse subconjunto deve tentar acompanhar os contornos presentes na imagem.

A dilatação e a erosão são duais, isso pode ser verificado pela expressão (4.5).

$$(A \ominus B)^c = \left(A^c \oplus \hat{B}\right) \tag{4.5}$$

4.3 Abertura e Fechamento

Outras duas importantes operações são a de fechamento e abertura, que são originadas da erosão e dilatação. A abertura é definida como:

$$A \circ B = (A \ominus B) \oplus B \tag{4.6}$$

ou seja, é a dilatação de uma imagem A com erosão, ambas pelo mesmo elemento estruturante B.

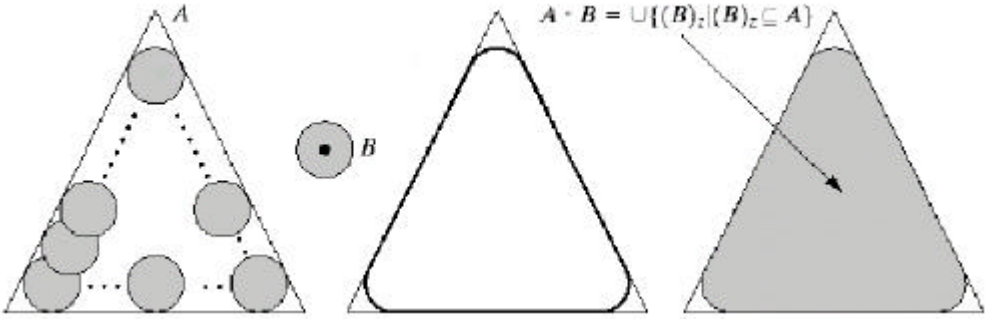
O fechamento, de forma similar, é definido por:

$$A \bullet B = (A \oplus B) \ominus B \tag{4.7}$$

logo, é a erosão de uma imagem A dilatada, ambas pelo mesmo elemento estruturante B.

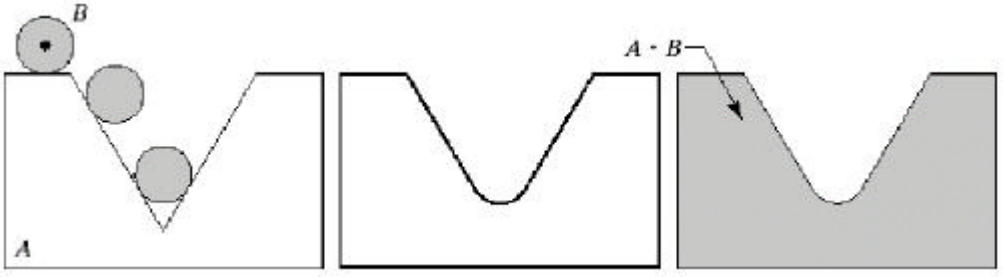
A abertura geralmente suaviza os contornos, quebra istmos estreitos e elimina protusões. O fechamento também tende a suavizar os contornos, mas tem o efeito de fundir as quebras em pequenos golfos, eliminando pequenos buracos e preenchendo fendas de um contorno. Para um melhor entendimento desses operadores, suponha a utilização de um elemento estruturante com a forma de um

círculo sobre uma imagem contendo um triângulo. Na abertura ele irá preencher esse triângulo internamente sem ultrapassar o seu contorno. No fechamento, ele também fará esse preenchimento, mas de forma externa ao contorno. A Figura 4.6 e a Figura 4.7 ilustram esse exemplo.



(a) **(b)** **(c)**

Figura 4.6: (a) Elemento estruturante passando internamente ao contorno do objeto. (b) Novo contorno gerado pela abertura. (c) Imagem produzida com a abertura [1]



(a) **(b)** **(c)**

Figura 4.7: (a) Elemento estruturante passando externamente ao contorno do objeto. (b) Novo contorno gerado pelo fechamento. (c) Imagem produzida com o fechamento [1]

A abertura e o fechamento também são duais.

$$(A \bullet B)^c = \left(A^c \circ \hat{B}\right) \qquad \textbf{(4.8)}$$

A vantagem desses dois operadores está no seu pequeno efeito no tamanho original dos objetos, ao contrário da erosão e dilatação.

4.4 Transformação Hit-or-Miss

A transformada morfológica *hit-or-miss* é uma ferramenta básica para a detecção de formas. Ela é caracterizada por um par de elementos estruturantes (B_1, B_2). A idéia é que um certo ponto fará parte da imagem resultante se e somente se o elemento B_1 "atingir" totalmente uma imagem entrada nesse pixel, e se o elemento B_2 "perder" totalmente a imagem entrada nesse pixel, ou seja, deve estar contido no complemento da imagem [3]. Pode-se representar essa operação pela Equação (4.9).

$$A \circledast B = (A \ominus B_1) \cap (A^c \oplus B_2) = (A \ominus B_1) - (A \oplus \hat{B}_2) \quad \textbf{(4.9)}$$

Para exemplificar essa operação, aplicou-se sobre a imagem original mostrada na Figura 4.8 a operação hit-or-miss, com B_1 do tamanho do menor quadrado e B_2 de tamanho um pouco maior. Com isso os pixels centrais dos objetos desejados foram identificados, em seguida dilatou-os para melhorar a visualização dos mesmos.

(a) **(b)** **(c)**

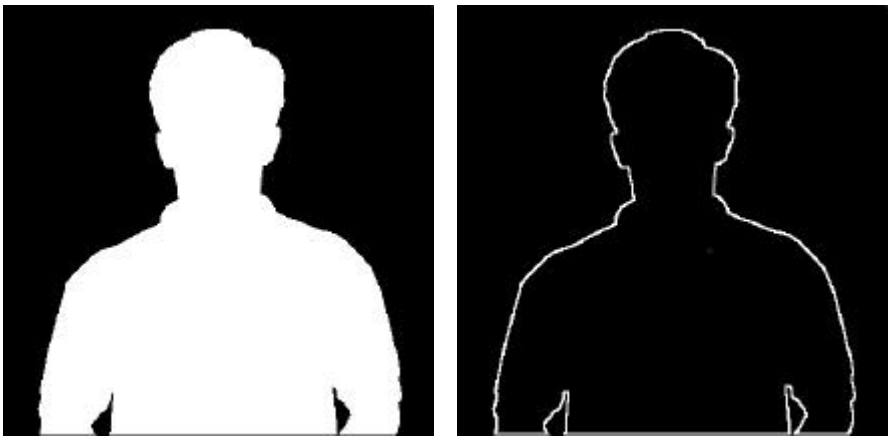
Figura 4.8: (a) Imagem original; (b) Resultado do hit-or-miss; (c) Dilatação de (b).

4.5 Algoritmos Morfológicos em Imagens Binárias

A principal aplicação de morfologia, no caso de imagens binárias, é a extração de componentes da imagem que sejam úteis na representação e na descrição de formas. Particularmente, algoritmos que fazem o preenchimento, o afinamento, o espessamento, a extração de fronteiras, de componentes conectados, fecho-convexo e o esqueleto de uma região.

A extração de contorno pode ser obtida através da subtração da imagem A por ela mesma com erosão. Essa operação pode ser definida como mostra a Equação (4.10).

$$b(A) = A - (A \ominus B) \quad \textbf{(4.10)}$$



(a) **(b)**

Figura 4.9: (a) Imagem binária original. (b) Imagem com a extração morfológica de contorno.

O preenchimento de regiões pode ser feito através da dilatação de conjuntos, da complementação e de interseções. O primeiro passo é definir B como um elemento estruturante com conexidade desejada, depois faz-se a seleção de um pixel p dentro do buraco a ser preenchido da imagem A. Cria-se uma matriz X_0 preenchida com zeros, com exceção do pixel p, então se faz iterações conforme a Equação (4.11).

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1,2,3,... \tag{4.11}$$

as iterações são feitas até $X_k = X_{k+1}$. A interseção limita a dilatação a região de interesse.

Uma importante função na análise automática de imagens é a extração de componentes conectados. Sendo Y uma região conectada contida em um conjunto A, e assumindo que um ponto p de Y seja conhecido, então a expressão iterativa seguinte leva à todos os pixels de Y:

$$X_k = (X_{k-1} \oplus B) \cap A \quad k = 1,2,3,... \tag{4.12}$$

sendo: $X_0 = p$; B um elemento estruturante adequado. As iterações são feitas até $X_k = X_{k+1}$, fazendo-se então $Y = X_k$.

O afinamento é uma operação onde se remove os pixels de um grupo de componentes conectados, até restar apenas um estreito conjunto. É usado para revelar estruturas em aplicações de reconhecimento de padrões. O afinamento, Equação (4.13), pode ser definido através da transformada *hit-or-miss*.

$$A \otimes B = A - (A \circledast B) = A \cap (A \circledast B)^c \tag{4.13}$$

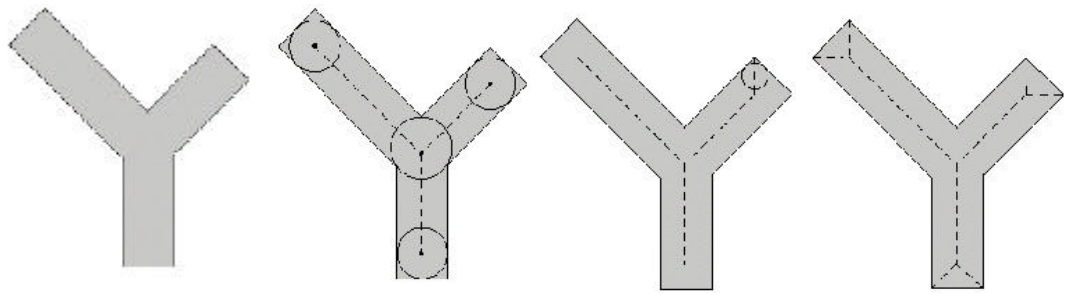
onde B é uma seqüência de elementos estruturantes.

O espessamento é o dual morfológico do afinamento e é definido por:

$$A \odot B = A \cup (A \circledast B) \tag{4.14}$$

O algoritmo pode ser programado diretamente da definição. Entretanto, descobriu-se que é mais efetivo fazer o afinamento do complemento de A.

Uma outra importante abordagem para a representação estrutural da forma de um objeto consiste na obtenção do seu "esqueleto" através do algoritmo de afinamento. Em vez de bordas pode-se extrair seu eixo medial, que são linhas finas que condensam a informação original enquanto se tenta preservar a homotopia dos objetos. Esse processo é bastante utilizado em problemas de inspeção automática de circuitos impressos, e também bastante útil na segmentação morfológica.



(a) **(b)** **(c)** **(d)**

Figura 4.10: (a) Imagem A. (b) Várias posições dos discos máximos. (c) Outro disco máximo num diferente segmento de A. (d) Esqueleto completo de A [1].

Definindo o esqueleto da imagem mostrada na Figura 4.10 como $S(A)$, ele pode ser expresso através de erosões e aberturas.

$$S_k(A) = (A \ominus kB) - (A \ominus kB) \circ B \tag{4.15}$$

onde k indica as sucessivas erosões de A.

4.6 Extensão para Imagens em Níveis de Cinza

Com o tempo, as idéias usadas na morfologia binária foram aplicadas em imagens de níveis de cinza. Os operadores básicos passaram por uma reformulação para trabalhar com esse tipo de imagem, possibilitando o desenvolvimento de algoritmos para a extração de contorno e o particionamento de conteúdos baseados na textura.

Para que os valores de intensidade do nível de cinza estejam presentes nessa nova formulação, há a necessidade do elemento estruturante ser definido como $b(x,y)$, onde b vai assumir um valor de cinza no ponto (x,y). A imagem de entrada volta a ser representada por $f(x,y)$.

A definição de dilatação passa a ser da forma mostrada na Equação (4.16).

$$(f \oplus b)(s,t) = \max\{ f(s-x,t-y)+b(x,y) \,|\, (s-x),(t-y) \in D_f ; (x,y) \in D_b \} \qquad \textbf{(4.16)}$$

onde D_f e D_b são domínios de f e b respectivamente [1]. A condição de que $(s-x)$ e $(t-y)$ deve estar no domínio de f, e (x,y) estar no domínio de b, é similar à condição da dilatação morfológica binária, onde os dois conjuntos devem estar superpostos por no mínimo um elemento.

Com essa definição de dilatação, se todos os valores do elemento estruturante são positivos, a imagem de saída aumenta a luminosidade. Os detalhes escuros ou são reduzidos ou são eliminados, dependendo de como os seus valores e formas estão relacionados com o elemento estruturante usado.

A erosão é definida de forma dual como mostra a Equação (4.17).

$$(f \ominus b)(s,t) = \min\{ f(s+x,t+y)-b(x,y) \,|\, (s+x),(t+y) \in D_f ; (x,y) \in D_b \} \qquad \textbf{(4.17)}$$

Novamente a condição de $(s+x)$ e $(t+y)$ estarem no domínio de f, e (x,y) estar no domínio de b é similar a condição da erosão morfológica binária, onde o elemento estruturante deve estar completamente contido pelo conjunto erodido.

Com essa erosão, se todos os elementos do elemento estruturante são positivos, a imagem de saída fica mais escura. O efeito de detalhes claros na imagem de entrada menores que o elemento estruturante, é reduzido. O grau de redução é determinado pelos níveis de cinza dos vizinhos e pela forma e amplitude de $b(x,y)$.

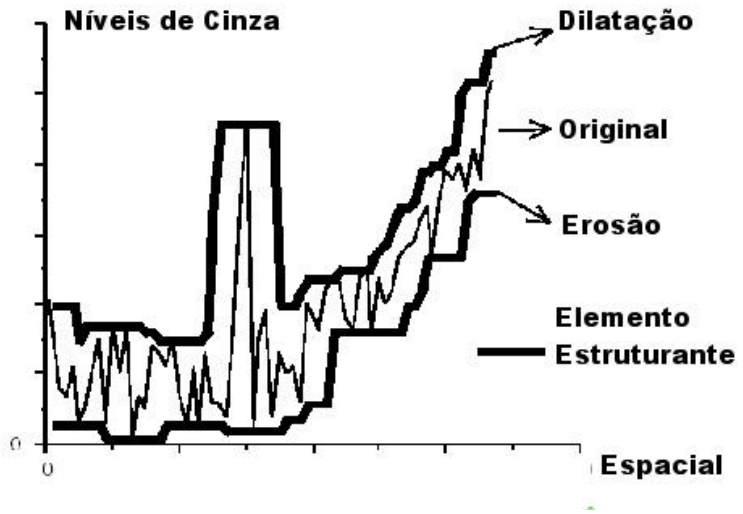


Figura 4.11: Exemplo no plano bidimensional dos efeitos da dilatação e da erosão.

A dualidade entre a dilatação e a erosão para essas imagens pode ser expressa conforme a Equação (4.18).

$$(f \ominus b)^c(s,t) = (f^c \oplus \hat{b})(s,t) \tag{4.18}$$

(a) **(b)** **(c)**

Figura 4.12: (a) Imagem original. (b) Imagem com dilatação. (c) Imagem com erosão.

As expressões para abertura e fechamento são as mesmas do caso binário, e são dadas respectivamente pelas equações abaixo.

$$f \circ b = (f \ominus b) \oplus b \tag{4.19}$$

$$f \bullet b = (f \oplus b) \ominus b \tag{4.20}$$

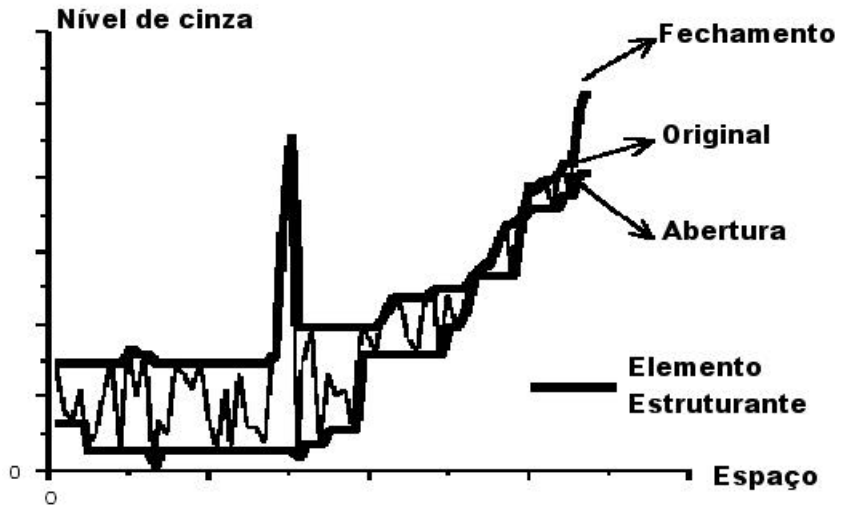


Figura 4.13: Exemplo no plano bidimensional dos efeitos da abertura e do fechamento.

A dualidade é expressa por:

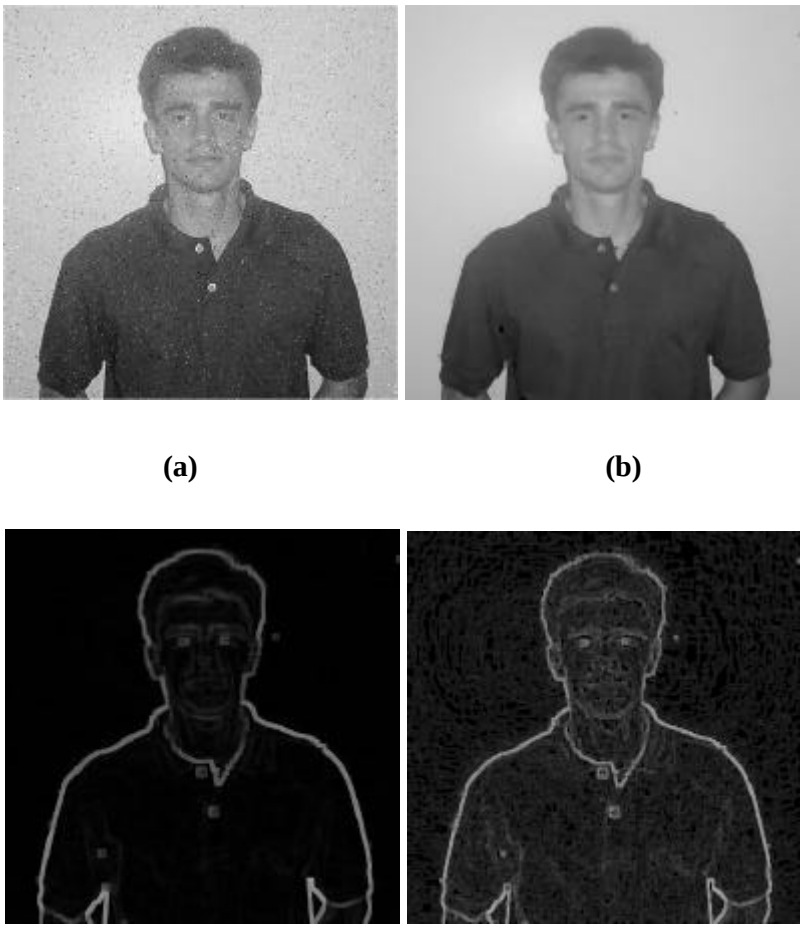
$$(f \bullet b)^c = f^c \circ \hat{b} \qquad \textbf{(4.21)}$$

onde $f^c = -f(x,y)$.

Na abertura, os picos estreitos em relação ao diâmetro do elemento estruturante são reduzidos em amplitude e conseqüentemente suavizados. Assim ela é usada para remover detalhes pequenos claros, deixando características claras e grandes intocadas. A erosão inicial remove os detalhes e torna a imagem mais escura, a dilatação subseqüente aumenta a intensidade da imagem sem introduzir os detalhes removidos na erosão.

No fechamento, os picos são deixados na forma original, assim, ele é usado para remover detalhes pequenos escuros, deixando características claras intocadas. A dilatação inicial remove os detalhes escuros e torna a imagem mais clara, a erosão subseqüente escurece a imagem sem introduzir os detalhes removidos na dilatação.

Existem várias aplicações para morfologia matemática em imagens de níveis de cinza. Um primeiro exemplo é a suavização com uma abertura seguida de um fechamento, retirando ruídos da imagem. O gradiente morfológico também pode ser usado, através da diferença entre uma imagem dilatada por ela mesma com erosão. Outra operação é o laplaciano, que é dado pelo resíduo entre o gradiente por dilatação e por erosão.



(a) **(b)**

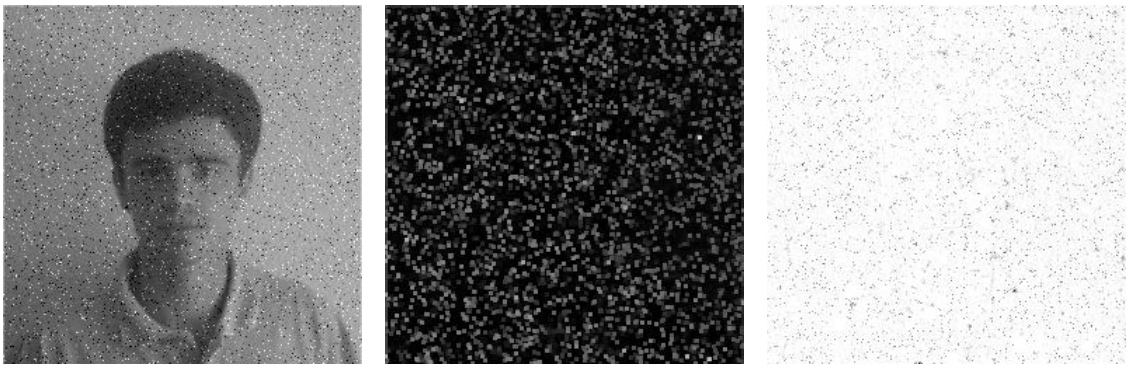
(c) **(d)**

Figura 4.14: (a) Imagem com ruído f. (b) Suavização $g = (f \circ b) \bullet b$. (c) Gradiente morfológico $(g \oplus b) - (g \ominus b)$. (d) Laplaciano morfológico $[(g \oplus b - g) - (g - g \ominus b)]$.

Uma outra transformação é a *top-hat*, também conhecida como chapéu mexicano, Equação (4.22). Ela detecta picos numa imagem, ou seja, separa as regiões que tem valores bem maiores que os seus vizinhos, desde que, o seu tamanho seja menor que o elemento estruturante. Um aplicação desse operador é no realce de detalhes na presença de sombra. O dual dessa operação é a detecção de vales, Equação (4.23), que são as regiões com valores bem menores que os seus vizinhos [4]. Exemplificando esses operadores, se for escolhido um elemento estruturante bem pequeno, pode-se detectar o ruído existente numa imagem, Figura 4.15.

$$h = f - (f \circ b) \tag{4.22}$$

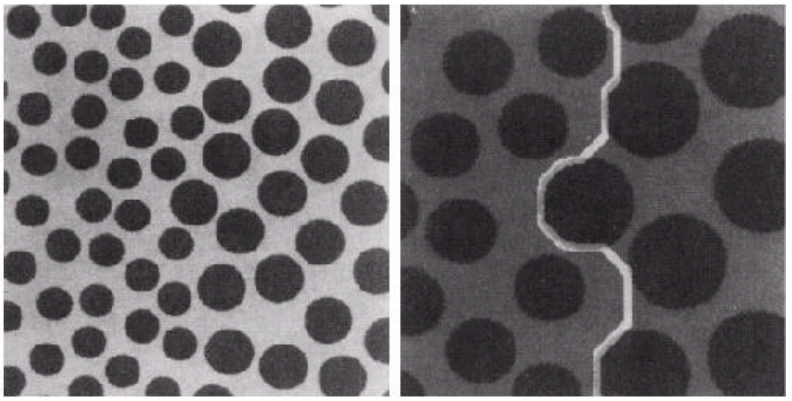
$$v = (f \bullet b) - f \tag{4.23}$$



(a) **(b)** **(c)**

Figura 4.15: (a) Imagem com ruído. (b) Aplicação do *top-hat* seguido de uma dilatação para uma melhor visualização do ruído, nesse caso os pixels brancos. (c) Detecção do ruído representado pelos pixels pretos pela detecção de vales.

A segmentação através da textura também pode ser feita com a matemática morfológica. Para a Figura 4.16, o primeiro passo foi fazer o fechamento da imagem usando elementos estruturantes sucessivamente maiores do que as pequenas bolhas, com isso houve a remoção dos detalhes escuros da imagem deixando apenas um fundo claro à esquerda e bolhas maiores à direita. Em seguida, aplicou-se uma abertura com elemento estruturante grande em relação à separação entre as bolhas maiores, removendo as regiões claras entre elas e deixando as escuras à direita. Tendo, finalmente, uma região clara à esquerda e uma região escura à direita, pode-se usar um limiar para gerar a fronteira entre as duas texturas.



(a) **(b)**

Figura 4.16: (a) Imagem original. (b) Imagem evidenciando o contorno entre regiões de diferentes texturas [1].

4.7 Operações Geodésicas e Reconstrução

Com o intuito de adequar mais ainda a morfologia matemática no processamento de imagens, surgiu uma classe chamada de transformações geodésicas. Elas envolvem a imagem a ser processada e uma outra que pode ser chamada de máscara. A idéia é que ao aplicar os filtros morfológicos sobre a imagem original, os seus efeitos fiquem limitados à máscara.

Representando a imagem máscara por g, pode-se definir a dilatação geodésica por:

$$\delta_g^{(1)}(f) = \delta^{(1)}(f) \wedge f \tag{4.24}$$

onde $\wedge$ é a operação de ínfimo, que no caso das imagens binárias corresponde à intersecção de conjuntos e no caso de níveis de cinza, à operação de mínimo.

A erosão geodésica é definida na equação abaixo:

$$\varepsilon_g^{(1)}(f) = \varepsilon^{(1)}(f) \vee f \tag{4.25}$$

onde $\vee$ é a operação de supremo, que no caso das imagens binárias corresponde a união de conjuntos e no caso de níveis de cinza, à operação de máximo.

Para a dilatação de tamanho n, tem-se que $\delta_g^n(f) = \delta_g^{(1)}(\delta_g^{(n-1)}(f))$, com $\delta_g^n(f) \leq \delta^n(f) \wedge g$. A erosão é definida de forma análoga.

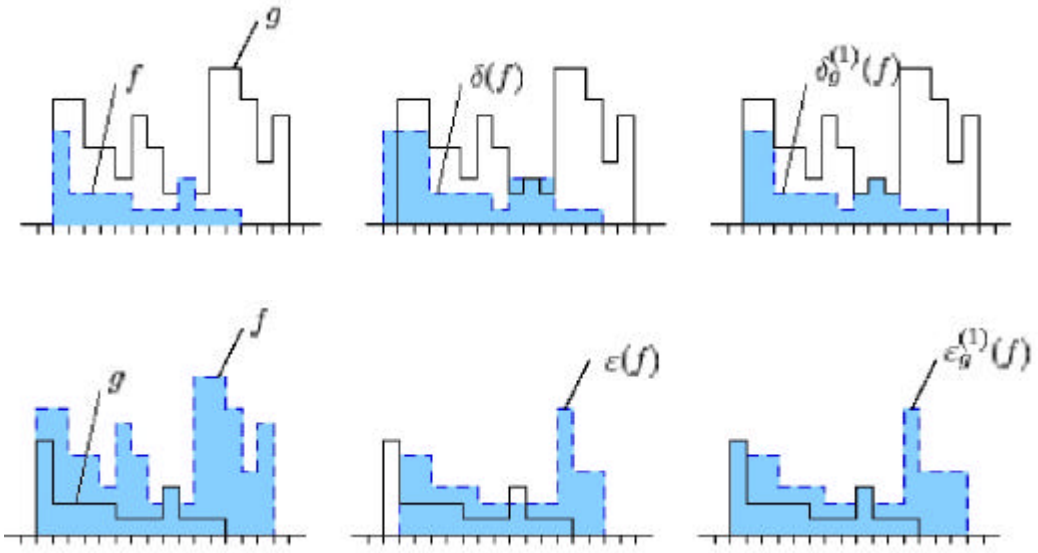


Figura 4.17: Exemplo mostrando a diferença entre a dilatação e erosão normal para a geodésica.

Através desses operadores geodésicos introduz-se a reconstrução. Ela é bastante utilizada em diversos tratamentos de imagens, inclusive em alguns tipos de segmentação presentes no próximo capítulo.

Supondo $f \leq g$, a reconstrução por dilatação de uma imagem máscara g, à partir de uma imagem marcadora f, é definida como a dilatação geodésica até a estabilidade de f com respeito a g.

$$R_g(f) = \delta_g^{(i)}(f) \tag{4.26}$$

onde i é o menor inteiro positivo tal que $\delta_g^{(i)}(f) = \delta_g^{(i+1)}(f)$.

A reconstrução por erosão, supondo que $f \geq g$, é definida conforme mostra a Equação (4.27).

$$R_g^*(f) = \varepsilon_g^{(i)}(f) \tag{4.27}$$

observando que foi usado um asterisco para diferencia-la da reconstrução por dilatação.

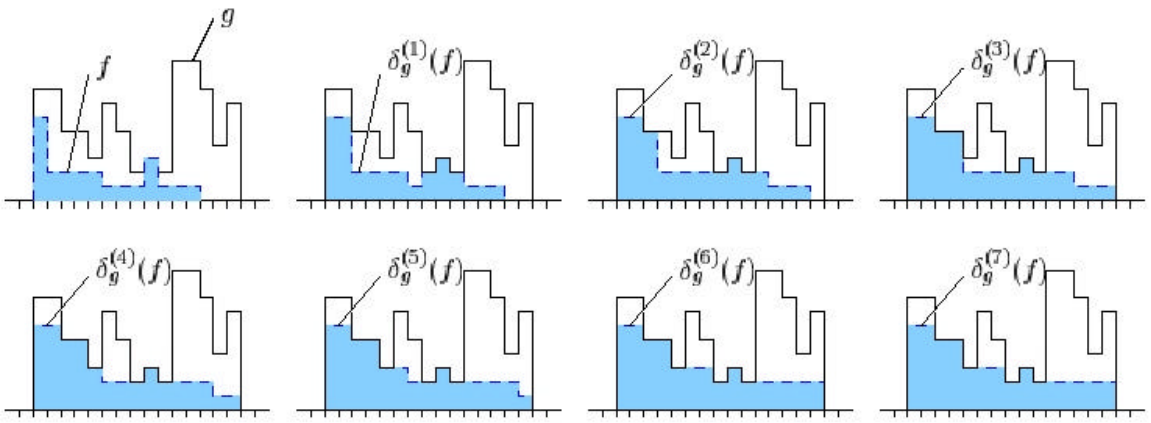


Figura 4.18: Exemplo de reconstrução por dilatação.

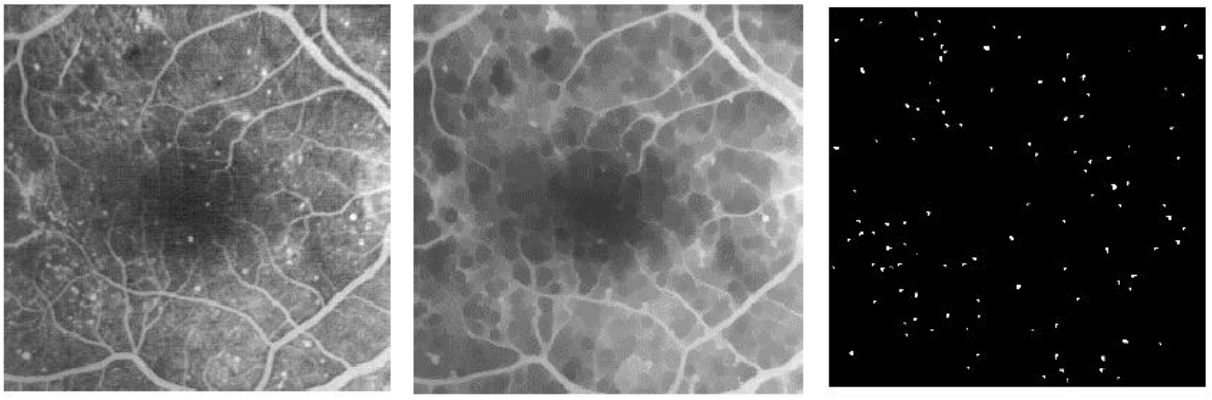
A abertura por reconstrução é definida conforme a Equação (4.28).

$$g_R^{(n)}(f) = R_f[\varepsilon^{(n)}(f)] \tag{4.28}$$

ou seja, apenas os objetos presentes após a erosão de tamanho n serão reconstruídos pela dilatação.

O fechamento por reconstrução de tamanho n é definido pela Equação (4.29).

$$j_R^{(n)}(f) = R_f^*[d^{(n)}(f)] \quad \textbf{(4.29)}$$



(a) **(b)** **(c)**

Figura 4.19: Extração de aneurismas. (a) Imagem original. (b) Fechamento seguido de uma abertura, ambos por reconstrução. (c) Subtração da imagem (a) pela (b) seguida de uma limiarização.

Até este capítulo foram apresentadas algumas ferramentas básicas de processamento de imagens. No próximo, será abordada a segmentação de imagens, que é o uso das técnicas vistas para a extração de objetos de interesse da imagem.

4.8 Referências

[1] R. C. GONZALES, R. E. WOODS, "Digital Image Processing", Prentice Hall,2ª edição, ISBN: 0-201-18075-8, 2001.

[2] N. S. T. Hirata, "Noções de Morfologia Matemática", Departamento de Ciência da Computação, IME-USP, 2002.

[3] A. C. Bovik e M. D. Desai, "Basic Binary Image Processing" Handbook of Image and Video Processing, Academic Press, ISBN: 0-12-119790-5, 2000.

[4] P. Maragos e L. F. C. Pessoa, "Morphological Filtering for Image Enhancement and Detection" Handbook of Image and Video Processing, Academic Press, ISBN: 0-12-119790-5, 2000.

5. Segmentação de Imagens

Nos capítulos anteriores foram apresentadas várias ferramentas e técnicas usadas no tratamento de imagens. Esse embasamento serve para uma outra área que é a análise de imagens. A segmentação de imagens faz parte dessa área e tem uma vasta quantidade de aplicações.

A segmentação subdivide uma imagem em suas partes ou objetos constituintes, dependendo do problema a ser resolvido. Um exemplo é a inspeção automática de circuitos integrados, onde os objetos de interesse são as anomalias presentes, como caminhos rompidos, ou falta de componentes, o que exige um certo grau de detalhamento. Cuidados especiais são tomados para se melhorar as chances de segmentação robusta, incluindo outros fatores como iluminação e sensores mais adequados de captura.

Os algoritmos de segmentação para imagens monocromáticas são geralmente baseados em uma das seguintes propriedades básicas de níveis de cinza: descontinuidade e similaridade. Na primeira categoria, a abordagem é o particionamento da imagem baseado em mudanças bruscas dos níveis de cinza. As principais áreas de interesse nessa categoria são a detecção de pontos isolados e detecção de linhas e bordas na imagem. Na segunda categoria tem-se a limiarização, crescimento, divisão e fusão de regiões. O conceito de segmentação de uma imagem baseado em descontinuidade ou em similaridade dos valores de nível de cinza de seus pixels pode ser aplicado tanto em imagens estáticas como em imagens dinâmicas. Sendo que nesse último caso, o movimento pode ser usado como uma ferramenta para o melhoramento dos algoritmos de segmentação [1].

Neste capítulo serão exploradas as técnicas baseadas em imagens estáticas e em seqüências que caracterizam o movimento.

5.1 Segmentação de Imagens Estáticas

Nesta seção serão apresentadas algumas técnicas de segmentação aplicadas em imagens estáticas. A maioria das operações usadas foram expostas nos capítulos anteriores. A maioria das imagens que serão usadas como exemplos, podem ser geradas usando-se os algoritmos que estão em anexo.

5.1.1 Detecção de Descontinuidades

Existem três tipos básicos de descontinuidades em imagens digitais: pontos, linhas e bordas. Uma maneira de se encontrar essas características é através da filtragem por convolução com máscaras específicas no domínio espacial. Relembrando a resposta de uma máscara de tamanho $3x3$, tem-se a equação abaixo.

$$R = w_1 z_1 + w_2 z_2 + ... + w_9 z_9 = \sum_{i=1}^{9} w_i z_i \qquad \textbf{(5.1)}$$

onde w_i é o i-ésimo coeficiente da máscara e z_i o pixel associado da imagem a ser filtrada.

A detecção de pontos mede as diferenças ponderadas entre o ponto central e seus vizinhos, tendo a idéia de que o nível de cinza de um ponto isolado será completamente diferente do nível de cinza dos seus vizinhos. A máscara que realiza essa operação está mostrada na Figura 5.1.

Indo para um próximo nível, tem-se a detecção de linhas, que é feita aplicando-se o conjunto de máscaras mostradas abaixo.

-1	-1	-1	-1	-1	-1	-1	+2	-1	-1	-1	+2	+2	-1	-1						
-1	8	-1	+2	+2	+2	-1	+2	-1	-1	+2	-1	-1	+2	-1						
-1	-1	-1	-1	-1	-1	-1	+2	-1	+2	-1	-1	-1	+2	+2						
(a)			**(b)**			**(c)**			**(d)**			**(e)**								

Figura 5.1: (a) Máscara para detecção de pontos. (b) Para detecção de linhas horizontais. (c) De linhas verticais. (d) De linhas a +45º. (e) De linhas a –45º.

A abordagem mais comum de detecção de descontinuidades é através das bordas contidas na imagem. Uma borda é o limite entre duas regiões com propriedades relativamente distintas de nível de cinza. A maioria das técnicas aplicadas nessa categoria se baseia na aplicação de um operador local diferencial. Como foi definido no Capítulo 3, o gradiente corresponde a primeira derivada, enquanto o laplaciano corresponde a segunda derivada. As máscaras que realizam estas operações são mostradas na Figura 5.3. Observe que se usa apenas uma máscara na aplicação do laplaciano, evidenciando a sua indiferença à direção das descontinuidades.

(a) **(b)** **(c)**

Figura 5.2: (a) Imagem original. (b) Imagem com detecção de pontos. (c) imagem com detecção de linhas.

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

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

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

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

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

(a) **(b)**

Figura 5.3: (a) Conjunto de máscaras que implementam o gradiente – Prewitt. (b) Máscara correspondente ao laplaciano.

(a) **(b)** **(c)**

Figura 5.4: (a) Detecção usando gradiente. (b) Detecção usando o laplaciano. (c) Laplaciano de uma imagem com ruído.

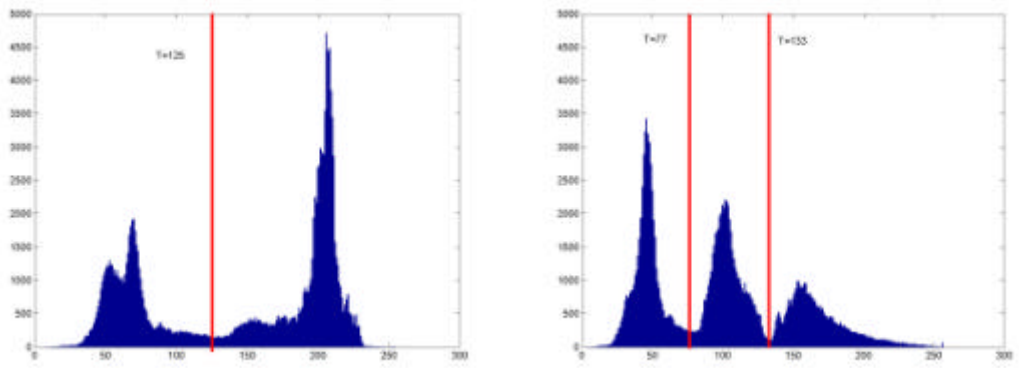
Embora o laplaciano responda as transições de intensidade, ele é raramente usado na prática de forma direta, devido a sua alta sensibilidade a ruídos presentes na imagem. Ele é mais utilizado, quando se deseja encontrar os pontos de cruzamento dos zeros, que indicam as descontinuidades.

A detecção de continuidades por si só, raramente caracteriza completamente uma fronteira, isso devido ao ruído, quebras, iluminação não uniforme e outros efeitos de descontinuidades de intensidade espúrias. Por isso, algoritmos de detecção de bordas são tipicamente seguidos de procedimentos de ligação e de outros de detecção de fronteiras, desenvolvidos para juntar e organizar os pixels das bordas.

5.1.2 Limiarização

A limiarização aparece freqüentemente nos algoritmos de segmentação. Ela é uma ferramenta bastante simples, que ao ser aplicada numa imagem, passa todos os níveis abaixo de um limiar (T) para preto e os acima para branco.

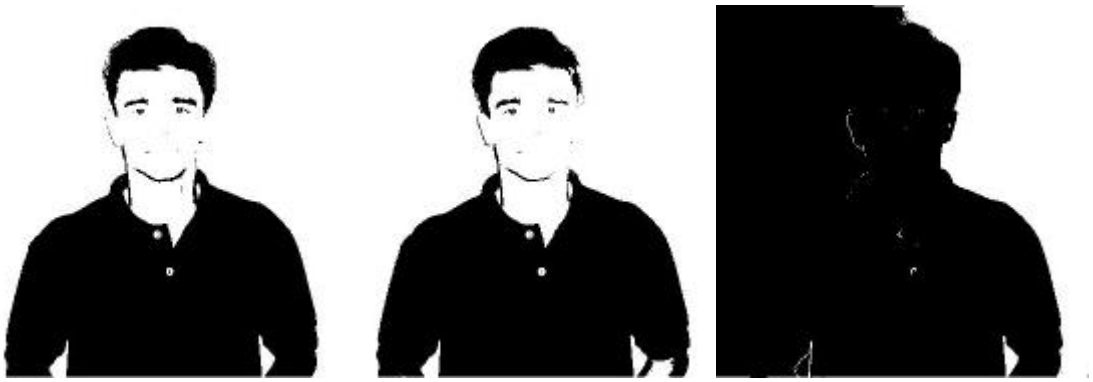
Uma forma de se escolher o limiar, é através da observação das regiões do histograma da imagem. A Figura 5.5 mostra dois histogramas, o primeiro evidencia que há duas regiões com contraste bastante distinto. Já o segundo, que é originado da mesma imagem, mas com iluminação não uniforme, apresenta três modos.



(a) **(b)**

Figura 5.5: (a) Histograma da Figura 5.2 (a), com uma reta indicando o limiar escolhido ($T = 125$). (b) Histograma da mesma imagem com iluminação não uniforme ($T = 77$ e $T = 133$).

A Figura 5.6 mostra o resultado da aplicação dos limiares obtidos pelo histograma.



(a) **(b)** **(c)**

Figura 5.6: (a) Aplicação do limiar $T = 125$. (b) Limiar $T = 77$ na imagem com maior iluminação no canto superior direito. (c) Limiar $T = 133$ da mesma imagem (b).

Pode-se observar que a iluminação influencia fortemente no histograma e conseqüentemente no resultado de uma limiarização [3].

Outras maneiras de se achar um limiar podem ser utilizadas, como as que levam em conta probabilidades, médias, variâncias, gradientes e laplacianos.

5.1.3 Segmentação Orientada à Regiões

Nesta seção, as técnicas que serão discutidas se diferenciam das apresentadas por se basearem diretamente na descoberta das regiões.

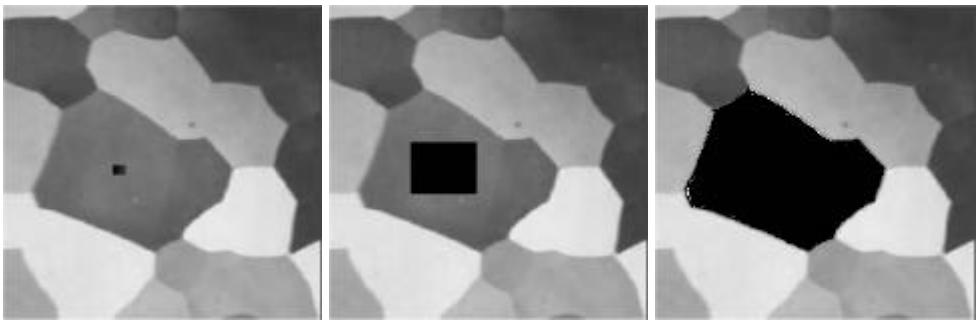
Iniciando com uma formulação básica, seja R uma região completa de uma imagem, e que possa ser particionada em n regiões R_1, R_2, ..., R_n, tal que:

- $\bigcup_{i=1}^{n} R_j = R$;
- R_i é uma região conexa, $i = 1,2,...,n$;
- $R_i \cap R_j = \varnothing$ para todo $i \neq j$;
- $P(R_i) = VERDADEIRO$ para $i = 1,2,...,n$;
- $P(R_i \cup R_j) = FALSO$ para $i \neq j$.

onde $P(R_i)$ é um predicado lógico que pode ser uma característica qualquer, mas que deve ser igual em toda a região, como por exemplo, a mesma intensidade de nível de cinza.

Com as condições citadas, sabe-se que cada pixel deve pertencer a uma região, e que eles sejam conexos. Que uma região não faz intersecção com a outra, e que pelo menos uma característica dos pixels é igual para toda região R_i.

O crescimento de regiões é um procedimento que agrupa pixels ou sub-regiões em regiões maiores. A mais simples forma de se fazer isso é através da agregação de pixels que se baseia em pontos denominados sementes. Os seus pixels vizinhos são analisados e se possuírem o mesmo predicado lógico passam a pertencer à região, em seguida se avaliam os vizinhos desses novos pixels, seguindo o mesmo procedimento. Um exemplo de predicado pode ser: se a diferença de intensidade entre a semente e o pixel analisado for menor que um determinado valor, este pixel passará a pertencer à nova região.



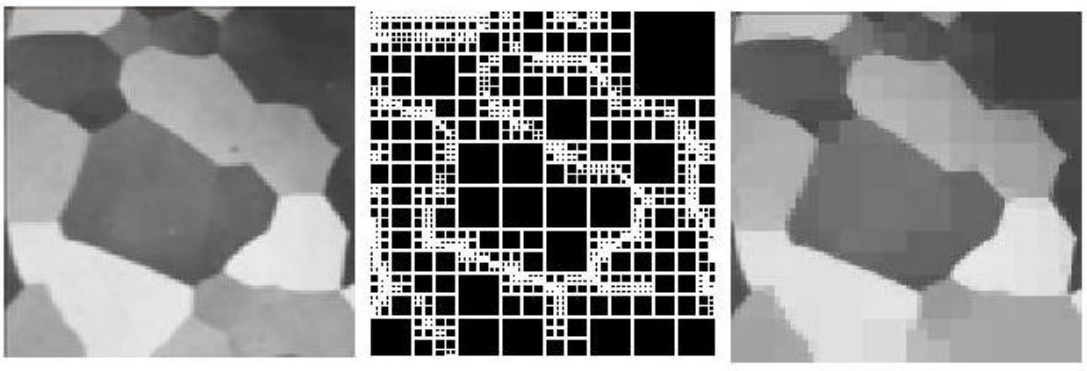
(a) **(b)** **(c)**

Figura 5.7: (a) Imagem mostrando um ponto semente. (b) Estágio intermediário de crescimento de uma região. (c) Região Final – toda preta.

A escolha da semente vai depender da natureza do problema. Sem um conhecimento a priori, pode-se calcular o histograma e escolher entre os valores de níveis de cinza dos picos mais fortes. O critério de similaridade (predicado), também vai depender da aplicação. Informações como textura, forma e tamanho das regiões, podem ser usadas na análise.

Uma alternativa ao crescimento de regiões a partir de um conjunto de sementes, é a subdivisão da imagem em várias regiões arbitrárias e disjuntas. Realizando então divisões e/ou fusões delas na tentativa de se estabelecer às condições propostas na formulação básica.

Uma maneira de se aplicar essa técnica consiste em subdividir uma imagem sucessivamente em vários quadrantes (*quadtree*), até que o predicado $P(R_i)$ seja verdadeiro. Dependendo da escolha do predicado, se obterá uma imagem bastante segmentada, por isso pode-se fazer a fusão das regiões que atenderem a uma outra característica pré-definida. Esse procedimento é conhecido como *"split and merge with quadtree"*.



(a) **(b)** **(c)**

Figura 5.8: (a) Imagem original. (b) Linhas de contorno das regiões após sucessivas divisões. (c) Imagem final após a fusão de regiões que atendam a um predicado pré-definido.

Esses operadores serão bastante úteis na implementação de alguns algoritmos que serão utilizados mais à frente.

5.1.4 Segmentação por Morfologia

No capítulo referente a processamento morfológico foram vistas algumas técnicas de aplicabilidade direta na segmentação de uma imagem, como a extração de contorno, o gradiente e o laplaciano morfológico. Eles funcionam muito bem quando se trata de imagens com iluminação uniforme e os objetos a serem segmentados têm uma borda bem definida com relação ao fundo. As operações de detecção de descontinuidades e limiarização também não são muito robustas. Uma técnica que produz uma segmentação menos sensível às condições dos objetos na imagem é o *watershed* morfológico, ou linhas de partição de água (LPA).

Como já foi visto, uma imagem em níveis de cinza pode ser considerada um relevo onde se encontram partes que se assemelham a vales e outras à montanhas. Quando chove, à medida que a água vai enchendo os vales e formando poças, elas vão crescendo e juntando-se a outras poças, formadas pela acumulação de água em vales vizinhos. Analogamente, suponha que são feitos buracos em cada mínimo local da imagem e faz-se uma inundação da superfície deles. Progressivamente, o nível da água vai aumentando. Para evitar a união de duas bacias, uma represa é construída progressivamente para cada ponto de contato. A união de todas as represas constitui as linhas de partição de águas.

Outra maneira de se criar as LPA é através de marcadores, onde a inundação começa á partir deles, só parando quando a água de um marcador começar a se encontrar com a do outro.

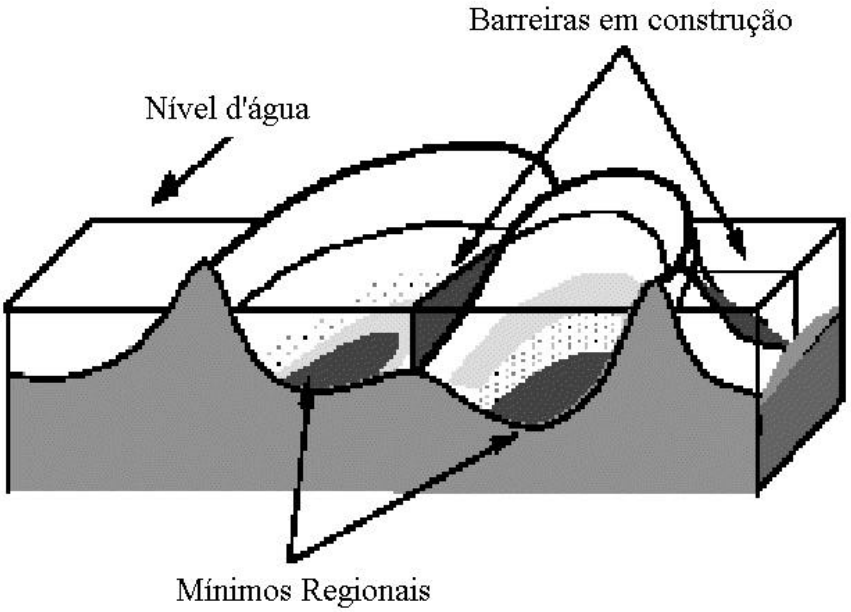


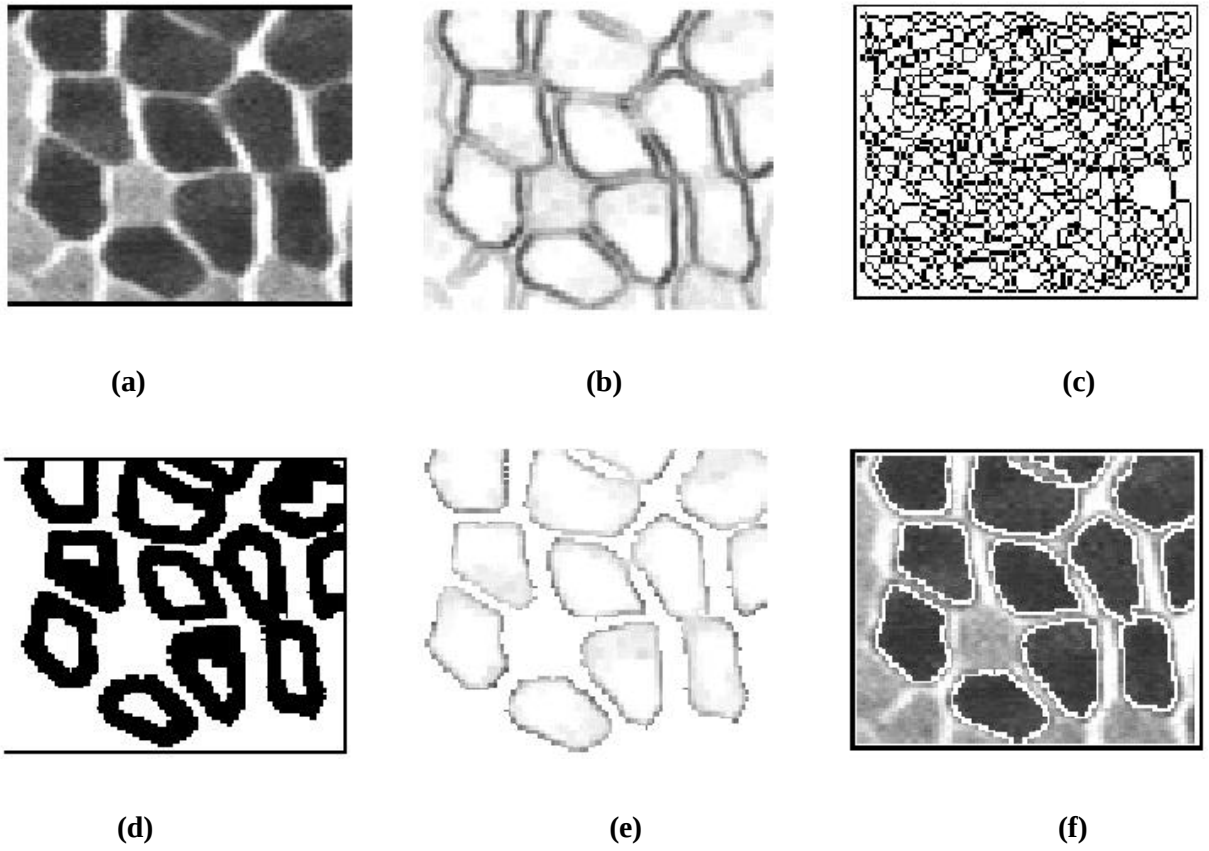
Figura 5.9: Exemplo de inundação e criação das LPA.

Quando se usa o watershed diretamente sobre uma imagem, tem-se, geralmente, como resultado uma supersegmentação. Para resolver esse problema, alguns pré-processamentos podem ser feitos na imagem. Esse procedimento é chamado de Paradigma de Beucher, e consiste basicamente em realçar e identificar os objetos que devem ser segmentados e eliminar os vales nas outras regiões.

A técnica mais eficiente para solução do problema de supersegmentação é a mudança de homotopia do gradiente, que elimina as descontinuidades não desejadas. Para isso, há a necessidade de um conhecimento prévio dos objetos a serem segmentados, que podem ser escolhidos de forma manual ou automática, definindo assim os marcadores. A suavização da imagem nem sempre é a melhor solução pelo fato de poder causar perda de descontinuidades.

A mudança de homotopia do gradiente começa com a negação dos marcadores. Em seguida, se calcula o ínfimo dessa imagem negada com a original. Esse resultado é usado para se fazer a mudança. No final, obtém-se uma imagem com contornos dos objetos de interesse, realçados e sem as descontinuidades causadas pelos ruídos. Como se o relevo fosse um pouco aplanado, tirando as pequenas colinas e enchendo os vales insignificantes [3].

A Figura 5.10 mostra o tecido de um músculo e o resultado da sua segmentação, com e sem o uso da filtragem homotópica. Pode-se observar a importância da análise da imagem, antes de se adotar os procedimentos de segmentação, evidenciando que características são realmente importantes.



(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figura 5.10: (a) Tecido muscular. (b) Gradiente invertido. (c) Supersegmentação. (d) Criação de marcadores (automático). (e) Filtragem homotópica. (f) Segmentação final.

5.1.5 Combinando Técnicas na Segmentação

Para exemplificar a segmentação de uma imagem, foi implementado a publicação "*Image segmentation towards new image representations methods*" [4]. O objetivo do programa desenvolvido é o de possibilitar a introdução de novas técnicas de segmentação na codificação de vídeo em redes de baixas taxas. Tanto a descrição do método, quanto o código do programa se encontram em anexo.

O programa diminui a quantidade de regiões necessárias para representar uma imagem. O primeiro passo é basicamente um pré-processamento, onde se usa a morfologia para eliminar detalhes menos relevantes ao sistema visual humano, e conseqüentemente retirar os ruídos que podem atrapalhar a segmentação. Para suavizar a imagem sem perder a forma e a posição correta das fronteiras, é utilizada uma abertura por reconstrução de erosão, seguida de um fechamento por reconstrução de dilatação. Os detalhes claros e escuros são suavizados sem corromper as bordas presentes. A Figura 5.11 mostra essa operação utilizando diferentes elementos estruturantes.

(a) **(b)** **(c)**

Figura 5.11: (a) Imagem Original. (b) Imagem com abertura seguida por fechamento, ambos por reconstrução, com um elemento estruturante quadrado 3x3. (c) Com elemento estruturante cruz 3x3.

No passo seguinte, foi feita a divisão da imagem em vários quadrantes, até a satisfação do predicado estabelecido. Essa decisão de divisão pode ser baseada em dois critérios: o de homogeneidade, onde se têm o critério de faixa dinâmica (DR) e a variância, por exemplo; e o de similaridade, onde é usado, a média das regiões. O critério utilizado no programa foi o DR, ou seja, se dentro de uma região, a diferença entre a máxima intensidade e a mínima for maior que um limiar, essa região é dividida.

$$DR \rightarrow \max\{R\} - \min\{R\} \geq t_{DR}^{s} \qquad \textbf{(5.2)}$$

A escolha de t_{DR}^{s} vai depender muito da imagem de entrada, devido às condições de iluminação e contraste. A Figura 5.12 mostra as regiões de divisão e o limiar escolhido.

(a) **(b)** **(c)**

Figura 5.12: (a) *Quadtree* com DR - $t_{DR}^{s} = 64$ gerando 1243 regiões. (b) Com DR - $t_{DR}^{s} = 102$ - 640 regiões. (c) Com DR - $t_{DR}^{s} = 18$ - 4906 regiões.

Como pode se observar, a aplicação da divisão gera uma imagem bastante segmentada. Por isso, se faz numa terceira etapa, a fusão das regiões, baseada também, num critério de homogeneidade ou similaridade. Como se deseja fazer uma união, pode-se comparar os blocos de acordo com a média calculada em cada um. Logo, o critério usado corresponde a seguinte equação:

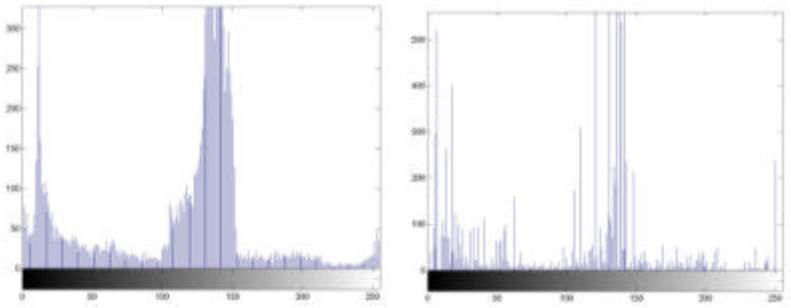
$$AVG \rightarrow \left| m_{R_i} - m_{R_j} \right| \geq t_{AVG}^{m} \qquad \textbf{(5.3)}$$

O limiar t_{AVG}^{m} vai ser escolhido de acordo com o grau de definição que se deseja na imagem resultante. A Figura 5.13 mostra as imagens mostradas na Figura 5.12, processadas com a fusão das regiões conforme o critério AVG.

(a) **(b)** **(c)**

Figura 5.13: (a) Fusão com AVG - $t_{AVG}^{m} = 5$, reduzindo para 802 regiões. (b) Com AVG - $t_{AVG}^{m} = 15$ - 266 regiões. (c) Com AVG - $t_{AVG}^{m} = 7$ - 1916 regiões.

Nota-se uma mudança significativa na quantidade de regiões, causando uma diminuição da quantidade de níveis de cinza usados para representar as imagens. Os histogramas mostrados abaixo comprovam esse efeito.



(a) **(b)**

Figura 5.14: (a) Histograma da Figura 5.11(a). (b) Histograma da Figura 5.13(a).

Vários parâmetros podem ser mudados para a melhoria do resultado final. A Tabela 2.1 mostra alguns ajustes e quantidade de regiões obtidas.

Tabela 5.1: Parâmetros de ajuste e respectivos resultados.

Nome	Elemento Estruturante	Limiar Divisão – DR	Limiar Fusão - AVG	Nº regiões após a divisão	Nº regiões após a fusão
Claire	Cruz 3x3	77	10	1024	510
Claire	Quadrado 3x3	77	10	979	475
Claire	Cruz 3x3	64	5	1243	802
Claire	Cruz 3x3	102	15	640	266
Claire	Cruz 3x3	102	70	640	60
Fabrício	Cruz 3x3	26	10	3664	1249
Fabrício	Cruz 3x3	18	6	4906	1916

Alguns problemas relacionados com a não iluminação uniforme da imagem, podem ser percebidos no resultado final da segmentação. Na Figura 5.13(c), o lado esquerdo do fundo é mais subdividido que o direito, evidenciando esse efeito.

É preciso evidenciar que, em sistemas onde não se admite perda de qualidade da imagem, como em aplicações médicas, esse método não é recomendado. Contudo, em esquemas de codificação de vídeo, ele causa um grande impacto, pois leva à uma otimização e reduz significativamente a quantidade de dados utilizados na representação da imagem.

5.2 Segmentação com Utilização de Movimento

A segmentação é uma ferramenta crucial nos esquemas de codificação de imagem e vídeo. Essa última aplicação permite a utilização do movimento para a extração dos objetos, melhorando significativamente a segmentação e possibilitando o desenvolvimento de novos padrões.

Um vídeo pode ser trabalhado como uma seqüência de imagens estáticas. Quando um objeto se move, diferenças na posição do pixel relacionado a ele ocorrerão. Diversas são as maneiras de se detectar movimento. Podem ser do tipo mais simples, onde se compara, pixel a pixel, uma imagem

com a outra, ou com técnicas mais avançadas, que usam várias imagens e permitem uma melhor segmentação. Neste tópico será citado um método baseado em gradiente.

5.2.1 Fluxo Ótico

Quando se têm imagens resultantes de um movimento do objeto diante da câmera, ou dela em relação ao objeto a ser filmado, ou ainda de ambos, se têm mudanças, chamadas de movimentos reais. Se essas variações são observadas em uma região, que muda de intensidade ao longo do tempo, tem-se o movimento aparente [5].

As mudanças podem ser modeladas por equações diferenciais parciais, normalmente chamadas de equações de restrição (*constraint equations*). O campo de vetores de velocidade obtido pela solução dessas equações é normalmente chamado de fluxo ótico (*optical flow*).

Para o entendimento do fluxo ótico, considere $I(x_1, x_2, t)$ a distribuição de intensidade contínua no domínio do espaço e do tempo. Se a intensidade se mantém constante ao longo da trajetória do movimento, tem-se:

$$\frac{dI(x_1, x_2, t)}{dt} = 0 \qquad \textbf{(5.4)}$$

onde a posição indicada por x_1 e x_2 variam de acordo com a trajetória do movimento.

A Equação (5.4) é uma expressão de derivada total e denota a taxa de variação de intensidade ao longo da trajetória. Usando a regra de derivação da cadeia e representando as coordenadas por um vetor posição $\vec{x}$ pode-se reescrever esta equação da seguinte maneira:

$$\frac{\partial I(\vec{x}, t)}{\partial x_1} v_1(\vec{x}, t) + \frac{\partial I(\vec{x}, t)}{\partial x_2} v_2(\vec{x}, t) + \frac{\partial I(\vec{x}, t)}{\partial t} = 0 \qquad \textbf{(5.5)}$$

onde:

$$v_1(\vec{x}, t) = \frac{\partial x_1}{\partial t} = \frac{dx_1}{dt} \qquad \textbf{(5.6)}$$

$$v_2(\vec{x}, t) = \frac{\partial x_2}{\partial t} = \frac{dx_2}{dt} \qquad \textbf{(5.7)}$$

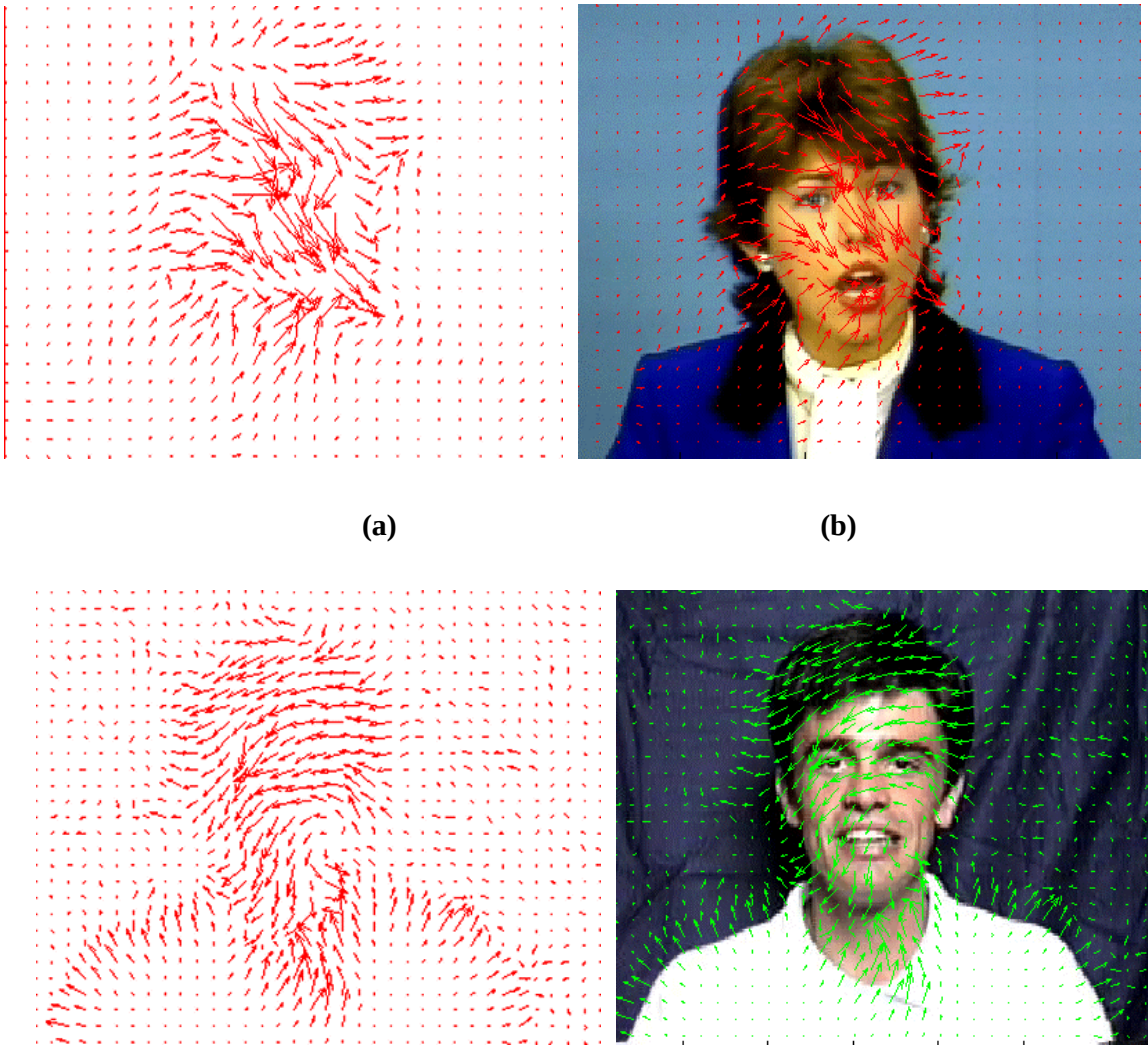
O termo em (5.5) é conhecido como equação de restrição do fluxo ótico (OFC). As velocidades (5.6) e (5.7) contidas na OFC é o fluxo ótico.

(a) **(b)**

(c) **(b)**

Figura 5.15: (a) Claire – quadro 141. (b) Claire – quadro 150. (c) Fabrício – quadro 46 (d) Fabrício – quadro 50.

Para calcular o fluxo ótico, foi escrito um programa que se encontra em anexo. Tomando dois vídeos como referência, foram tirados dois quadros de cada um (Figura 5.15). O fluxo ótico é mostrado na Figura 5.16, onde o tamanho das setas indica a intensidade do movimento e o ângulo em relação a horizontal, a direção na qual o objeto se moveu.



(a) **(b)**

(c) **(d)**

Figura 5.16: (a) Setas indicando o fluxo ótico de Claire. (b) Setas sobrepostas na Claire. (c) Setas indicando o fluxo ótico de Fabrício. (d) Setas sobrepostas no Fabrício.

Como o cálculo do fluxo ótico é baseado na mudança de intensidade do nível de brilho, pode-se observar que nas regiões mais uniformes, o movimento não é eficientemente detectado.

O uso de apenas duas imagens também interfere no resultado final. No próximo tópico, será abordado um algoritmo que de acordo com os parâmetros de entrada, ele usa várias imagens de uma seqüência para o cálculo dos vetores de velocidade.

5.2.2 Método de Horn & Schunck

O intuito desse método é a suposição de que o campo de movimento satisfaz a OFC, com a variação de intensidade pixel, a pixel ao longo de toda a imagem constante. A partir da Equação (5.5), se obtém a expressão que denota o erro na equação do fluxo ótico.

$$e_{OF}(\bar{v}(\bar{x};t)) = \left\langle \nabla \bar{I}(\bar{x};t), \bar{v}(\bar{x};t) \right\rangle + \frac{\partial \bar{I}(x;t)}{\partial t} \tag{5.8}$$

Se a equação do OFC foi satisfeita, o erro é igual a zero. Na ausência de situações de ruídos e outros problemas relacionados com o movimento, pode-se minimizar o quadrado do erro e assim formar um sistema de duas equações, visto que a OFC não é suficiente para a determinação do vetor de velocidade.

A variação pixel a pixel dos vetores de velocidade pode ser quantificada pela soma dos quadrados das magnitudes dos gradientes espaciais das componentes da velocidade [6]. Logo:

$$e_{OF}^2(\bar{v}(\bar{x};t)) = \left\| \nabla \bar{v}_1(\bar{x},t) \right\|^2 + \left\| \nabla \bar{v}_2(\bar{x},t) \right\|^2 \tag{5.9}$$

Que é a soma dos Laplacianos das componentes do vetor velocidade $\bar{v}$. Podendo ser reescrita por:

$$e_{OF}^2(\bar{v}(\bar{x};t)) = \left(\frac{\partial v_1(\bar{x},t)}{\partial x_1} \right)^2 + \left(\frac{\partial v_1(\bar{x},t)}{\partial x_2} \right)^2 + \left(\frac{\partial v_2(\bar{x},t)}{\partial x_1} \right)^2 + \left(\frac{\partial v_2(\bar{x},t)}{\partial x_2} \right)^2 \tag{5.10}$$

cujas coordenadas espaciais e temporais são consideradas variáveis contínuas. A Equação (5.10) quando minimizada, suaviza o vetor velocidade e portanto, é chamada de equação de suavização. Horn e Schunck [7] propõem a seguinte solução interativa para determinar os valores das componentes de $\bar{v}$, partindo de um ponto inicial. As equações usadas são descritas abaixo.

$$\bar{v}_1^{(n+1)}(\bar{x};t) = \bar{v}_1^{(n)}(\bar{x};t) - \frac{\partial I(\bar{x};t)}{\partial x_1} \cdot \frac{\left(\frac{\partial I(\bar{x};t)}{\partial x_1} \cdot \bar{v}_1^{(n)}(\bar{x};t) + \frac{\partial I(\bar{x};t)}{\partial x_2} \cdot \bar{v}_{21}^{(n)}(\bar{x};t) + \frac{\partial I(\bar{x};t)}{\partial t} \right)}{\alpha^2 + \left(\frac{\partial I(\bar{x};t)}{\partial x_1} \right)^2 + \left(\frac{\partial I(\bar{x};t)}{\partial x_2} \right)^2} \tag{5.11}$$

$$\bar{v}_2^{(n+1)}(\bar{x};t) = \bar{v}_{21}^{(n)}(\bar{x};t) - \frac{\partial I(\bar{x};t)}{\partial x_2} \cdot \frac{\left(\frac{\partial I(\bar{x};t)}{\partial x_1} \cdot \bar{v}_1^{(n)}(\bar{x};t) + \frac{\partial I(\bar{x};t)}{\partial x_2} \cdot \bar{v}_{21}^{(n)}(\bar{x};t) + \frac{\partial I(\bar{x};t)}{\partial t} \right)}{\alpha^2 + \left(\frac{\partial I(\bar{x};t)}{\partial x_1} \right)^2 + \left(\frac{\partial I(\bar{x};t)}{\partial x_2} \right)^2} \tag{5.12}$$

O parâmetro α é selecionado heuristicamente e controla a restrição de suavização da imagem. Ou seja, é a partir da escolha de α que a determinação das componentes de $\bar{v}$ convergem mais ou menos. Os parâmetros $n+1$ e n são respectivamente a interação a ser calculada e a anterior. Os valores iniciais da velocidade média são freqüentemente usados como zero. Os gradientes são todos estimados usando diferenças finitas.

O programa usado foi escrito por Travis Burkitt e modificado por John Barron em 1992. A linha de comando segue o padrão mostrado abaixo:

```
horn2 nome alfa d_p I_C n /input /output –Type /outdata –MH –T limiar
```

onde: “nome” se refere à imagem a ser analisada; “alfa” o multiplicador de Lagrange que controla a restrição de suavização; “d_p” o desvio gaussiano da imagem central, que determina a quantidade de imagens vão ser usadas; “I_C” é o número da imagem central; “n” o número máximo de iterações; “input” o diretório que contem a seqüência de imagens; “output” é o diretório onde vai ser armazenado o resultado final; “-Type” indica o tipo da imagem de entrada; “outdat” é o diretório com as imagens pré-processadas (suavizadas), usadas na interação; “-MH” habilita a suavização das imagens de entrada; “-T limiar” indica o limiar que se deseja aplicar na imagem resultante do optical flow.

Para exemplificar o método, foi escolhida uma seqüência de imagem da *Miss America*. A Figura 5.17 mostra a primeira e a última imagem das usadas no processamento.

(a) **(b)**

Figura 5.17: (a) Quadro 63 da *Miss America*. (b) Quadro 80.

Os parâmetros usados são mostrados na linha de comando abaixo.

```
horn2 miss 0,5 0,25 71 100 /input /output –P /outdata –MH
```

O resultado obtido usando a limiarização do próprio programa, não foi muito eficiente, logo ela foi feita com a utilização de um outro programa. O resultado obtido é mostrado na Figura 5.18.

(a) **(b)**

Figura 5.18: (a) Resultado do fluxo ótico pós-processada com uma limiarização igual a 44. (b) Segmentação da Miss, através da operação de máximo entre o quadro 63, 80 e máscara obtida em (a).

A estimação de movimento é de grande importância nos sistemas que é limitado por baixas taxas de transmissão. A qualidade desse cálculo depende fortemente das condições em que o vídeo é capturado. A câmeras tradicionais, geralmente produzem vídeos com erros ao longo do tempo, principalmente quando se trata de objetos em movimento. As técnicas de tratamento de imagens são convenientes por justamente tratar essas limitações e produzir um efeito melhor.

5.3 Referências

[1] R. C. GONZALES, R. E. WOODS, “Digital Image Processing”, Prentice Hall,2ª edição, ISBN: 0-201-18075-8, 2001.

[2] A. C. Bovik e M. D. Desai, “Basic Binary Image Processing” Handbook of Image and Video Processing, Academic Press, ISBN: 0-12-119790-5, 2000.

[3] N. S. Hirata, “Segmentação: Watershed” Depto. De Ciência da Computação – IME – USP, 2002.

[4] D. CORTEZ, P. NUNES, M. M. SEQUEIRA, F. PEREIRA, “Image segmentation towards new image representation”, Instituto Superior Técnico, Secção de Telecomunicações, Portugal, 1994.

[5] K. N. Ngan, T. Meier, D. Chai, “Advanced Video Coding Principles and Techniques” , Elsevier, Science, ISBN: 0 444 82667 X, 1999.

[6] I. N. S. OLIVEIRA, V. O. RODA, “Metodologia para Detecção Rápida do Movimento em Seqüências de Imagens”, Departamento de Engenharia Elétrica da Escola de Engenharia de São Carlos, Universidade de São Paulo.

[7] B. HORN, B. SCHUNCK, "Determining Optical Flow", Artificial Intelligence 17, 185-203, 1981.

[8] A. M. Tekalp, “Digital Video Processing”, University of Rochester, Prentice Hall, ISBN: 0-13-190075-7, 1995.

6. Compressão de Imagens

Diariamente, uma grande quantidade de informação é armazenada, processada e transmitida na forma digital. Dados financeiros, relatórios anuais, estudos de longos anos, bibliotecas eletrônicas, entre vários outros documentos, correspondem, atualmente, a seqüências de milhões de bits. A TV digital, por exemplo, demanda espaço de armazenamento, e linhas com capacidade de transmissão desses dados. Se atualmente tem-se um avanço nas aplicações voltadas à imagem e vídeo digital, é graças aos métodos de compressão [1].

A compressão trata o problema de reduzir a quantidade de dados necessários para representar uma imagem digital, sendo à base dessa redução, a representação, a remoção de dados redundantes e o conhecimento do sistema visual humano.

A compressão é composta de duas etapas, uma de modelagem, onde se decide o código de codificação baseado na origem da informação e a própria codificação, que usa um conjunto de símbolos para representar os dados.

Neste capítulo, serão citados fundamentos, modelos de compressão e a transformada DCT.

6.1 Fundamentos

Ao se falar de dados, deve-se ter em mente que eles representam um meio pelo qual as informações são transmitidas. Com isso, diferentes quantidades de dados podem representá-las. Textos escritos, falas, músicas, imagens estáticas e dinâmicas, são apenas alguns exemplos de informações.

A redundância de dados é baseada num sólido conceito matemático. Para iniciar essa formulação, tem-se que a redundância de dados relativos (R_D) é definida conforme a equação abaixo.

$$R_D = 1 - \frac{1}{C_R} \qquad \textbf{(6.1)}$$

onde $C_R = (n_1 / n_2)$. O valor n_1 e n_2 correspondem ao número de unidades de transporte de informação em dois conjuntos de dados que representam a mesma informação [2].

Se $R_D = 0$, a primeira representação da informação não contém dados redundantes em relação a segunda. Se $n_2 << n_1$ a compressão é significativa. E se $n_2 >> n_1$ o segundo conjunto de dados tem várias redundâncias em relação ao original. Usualmente, designa-se a taxa de compressão por uma fração, como por exemplo, 10:1, significando que a quantidade de dados foi reduzida em dez vezes.

Ao se tratar de imagens digitais, existem basicamente três tipos de redundância: a de codificação, a interpixels e a psicovisual.

Na redundância de codificação, há a atribuição de um código com tamanho variável, em termos de números de bits [1]. Os bits de um determinado tom de cinza que têm maior ocorrência são representados pelos códigos de comprimento menor, ao contrário, se forem pouco freqüentes são representados por um código maior. Um exemplo seria pegar uma imagem de 256 níveis de cinza, que usa 8 bits para representar cada um deles, e calcular o seu histograma. Isso forneceria a probabilidade com que cada nível aparece, permitindo se atribuir códigos pequenos para os de maior ocorrência. Isso reduziria a quantidade de bits usados para representar a imagem.

A redundância interpixel permite estimar o valor de um pixel pelo valor dos seus vizinhos; esta correlação espacial está ligada ao relacionamento geométrico entre os objetos na imagem. Nomes como a redundância espacial, a geométrica e a entre quadros, são usados dentro dessa categoria.

Sabendo-se que o olho não responde com a mesma sensibilidade a todas informações visuais, pode-se definir um grau de importância para elas, podendo eliminar algumas sem que haja perda significativa na percepção da imagem. A redundância psicovisual é associada com a informação visual quantificável ou real. Desde que a eliminação dos dados psicovisualmente redundantes resulta numa perda de informação quantitativa, ela é comumente denominada quantização [1]. Essa operação traz perdas de informação de forma irreversível, o que resulta em compressão de dados com perda.

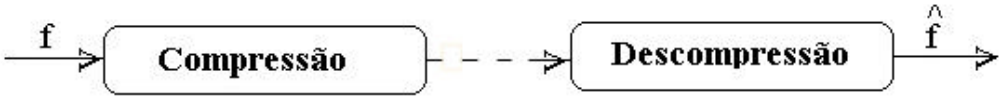


Figura 6.1: Caminho simplificado que a imagem percorre num sistema de compressão.

Com o objetivo de se avaliar os efeitos da redução de redundância na imagem de saída após a descompressão, foram criados duas classes de critérios, os de fidelidade objetivos, e os de fidelidade subjetivos. O primeiro expressa a categoria onde o nível de perda é expresso em função da imagem original. O erro raiz média quadrática (RMS) e a razão sinal-ruído média quadrática (SNR_{ms}) são parâmetros dessa classe. O segundo critério é baseado na observação das imagens resultantes por um

ser humano, permitindo uma avaliação subjetiva da qualidade da imagem, dizendo se está boa, regular ou ruim.

6.2 Modelos de Compressão de Imagens

As técnicas de redução de redundâncias são geralmente combinadas para formar um sistema de compressão. Um modelo genérico desse sistema é composto por um codificador e um decodificador, como mostra a figura abaixo.

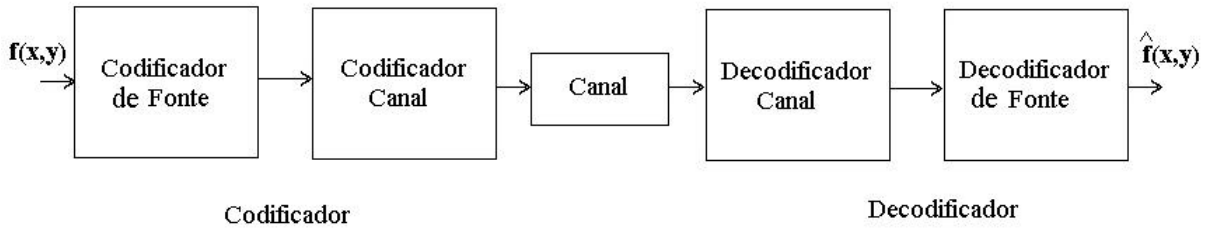


Figura 6.2: Modelo de compressão genérico.

O codificador fonte é responsável pela redução de redundâncias da imagem de entrada. Ele é modelado de acordo com os requisitos de critérios de fidelidade, mas no geral, é constituído por uma seqüência de três operações. O primeiro é o mapeador, que transforma os dados de entrada num formato projetado para reduzir as redundâncias interpixels. Os mapeamentos por comprimento de corrida e por coeficientes de transformação são dois exemplos desse bloco. O segundo operador é o quantizador, que reduz a acurácia da saída do mapeador, de acordo também, com algum critério de fidelidade. Esse bloco é utilizado para compressões com erros. A última operação é o codificador de símbolos, que cria um código de comprimento fixo ou variável e mapeia a saída de acordo com ele.

O decodificador fonte contém apenas o decodificador de símbolos e um mapeador inverso, que são os duais dos contidos no codificador fonte.

O codificador de canal é utilizado quando há ruídos no caminho de transmissão que podem acarretar erros na recuperação dos dados. Ele é projetado de forma a inserir redundâncias controladas nesses dados, tornando-os mais robustos e possibilitando a sua recuperação. Técnicas de justaposição de bits, como a desenvolvida por Hamming, é bastante utilizada nesse codificador. O decodificador de canal faz o papel inverso, corrigindo os dados recebidos.

6.3 Compressão Livre de Erro

Algumas aplicações exigem que os processos de compressão e descompressão sejam livres de perdas de informação, como imagens médicas, documentos de negócios, banco de dados, entre outros. Os algoritmos de compressão de dados sem perdas são baseados no comportamento estatístico das amostras, e/ou dicionários com códigos de comprimentos variáveis.

6.3.1 Codificação por Tamanho Variável

Como já foi citado, um dos meios de se diminuir redundâncias de codificação, é através da atribuição de códigos de tamanho variável para cada nível de intensidade, dependendo da sua freqüência. O código de Huffman é um dos mais antigos métodos de compactação. Baseando-se em métodos estatísticos, ele leva ao menor número possível de símbolos de código. A primeira etapa é a ordenação das probabilidades dos símbolos, e da combinação dos de menor probabilidade em um único símbolo. A segunda etapa corresponde à codificação de cada fonte reduzida, começando com a menor fonte e continuando para trás até a fonte original.

Tabela 6.1: Tabela mostrando o procedimento do código de Huffman.

Fonte Original			Redução de Fonte							
Símb.	Prob.	Cód	1	Cód	2	Cód	3	Cód	4	Cód
a_2	**0,4**	**1**	**0,4**	**1**	**0,4**	**1**	**0,4**	**1**	**0,6**	**0**
a_6	**0,3**	**00**	**0,3**	**00**	**0,3**	**00**	**0,3**	**00**	**0,4**	**1**
a_1	**0,1**	**011**	**0,1**	**011**	**0,2**	**010**	**0,3**	**01**		
a_4	**0,1**	**0100**	**0,1**	**0100**	**0,1**	**011**				
a_3	**0,06**	**01010**	**0,1**	**0101**						
a_5	**0,04**	**01011**								

Outra codificação bastante utilizada é a de Lempel-Ziv-Welch (LZW). Ela é baseada na construção de um dicionário de um ou mais caracteres, chamados frases, que são feitas à partir dos dados de entrada. Ao se analisar uma imagem, as seqüências constantes no dicionário são substituídas por um símbolo (token) que o faz referência. Esta técnica está presente em vários formatos de figura, como o TIFF, GIF e PDF.

6.3.2 Codificação por Planos de Bit

Outra técnica que trata da redução de redundâncias interpixel é a codificação por planos de bits. Ela se baseia no conceito de decomposição de uma imagem multiníveis em uma série de imagens binárias, seguida da compressão de cada uma delas.

Uma imagem em níveis de cinza de m bits pode ser representada na forma de um polinômio de base 2, ou seja, consegue-se separar a imagem num plano correspondendo ao primeiro bit, outro ao segundo, e assim por diante. Essa abordagem tem a desvantagem das pequenas mudanças terem impacto significativo na complexidade dos planos de bits, por isso, dependendo da necessidade, pode-se representar a imagem por um código de Gray, que após a sua decomposição, vai revelar planos que diferem de forma intensa nos bits mais significativos [2].

Com a imagem decomposta, pode-se usar a codificação de área constante, que representa blocos de imagens, provavelmente correspondendo a grandes áreas de 0's ou 1's em cada plano, por palavras-código especiais. A representação por essas palavras diminuirá significativamente a quantidade de dados necessários.

Outro tipo de codificação é o de corrida (Run-Lenght), que pode ser unidimensional, ou bidimensional. Com ele há a representação dos planos por seqüências de seqüências que descrevem trechos de pixel. Esse tipo de codificação é o padrão usado em fac-símiles (FAX) [1].

6.3.3 Codificação Preditiva sem Perdas

Este tipo de codificação se baseia na eliminação de redundâncias entre os pixels poucos espaçados, através da extração e codificação da informação de um pixel em relação ao outro, ou seja, apenas a diferença entre o valor do pixel predito e o valor do pixel original é transmitida. Com isso, atribuem-se códigos menores para representar o ruído. Este método é adequado para sinais nos quais os valores sucessivos não diferem muito entre si.

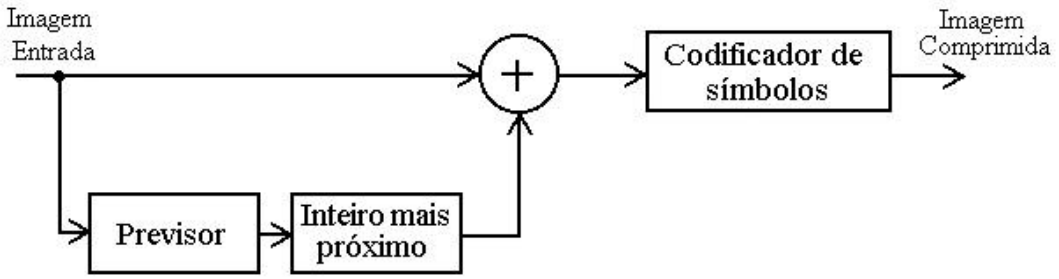


Figura 6.3: Modelo do codificador previsor sem perdas.

No próximo tópico será visto o uso da predição na compressão com perdas.

6.4 Compressão com Perdas

As técnicas discutidas até agora fornecem compressão de aproximadamente 3:1, o que ainda é muito pouco para se viabilizar algumas aplicações. Com a necessidade de aumentar esse fator, muitas sistemas de compressão utilizam codificadores, onde há perda de informação. Imagens monocromáticas podem ser reduzidas em torno de vinte vezes, sem muita perda de qualidade, às vezes de forma imperceptível para o olho humano.

Este tópico contem a codificação previsora e a por transformada, sendo que nesta última, será dada ênfase à DCT, por estar presente em vários padrões de vídeo.

6.4.1 Subamostragem do Espaço de Cor

Antes de entrar nas técnicas de codificação propriamente ditas, uma maneira utilizada em alguns padrões de vídeo como uma primeira fase de compressão é a subamostragem do espaço de cor. Este processo trabalha com imagens coloridas e através de suas propriedades subamostra a crominância em relação ao sinal de luminância [3].

Os sistemas típicos utilizam uma relação de duas amostras de luminância para cada amostra dos sinais de crominância, em cada uma das coordenadas verticais e horizontais da imagem (compressão 4:1:1). Pode-se também desprezar totalmente uma das componentes de crominância, e recuperá-la a partir das outras (compressão 4:1:0). A Figura 6.4 mostra como normalmente se toma as amostras de crominância em relação a luminância.

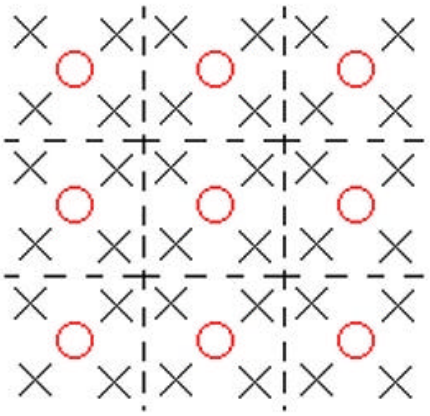


Figura 6.4: Amostragem 4:1, onde o círculo indica a amostra de crominância, o X de luminância e as linhas tracejadas, o limite dos blocos.

6.4.2 Codificação Previsora com Perdas

A principal diferença deste modelo de codificação ao relatado anteriormente, é o fato dele possuir um quantizador.

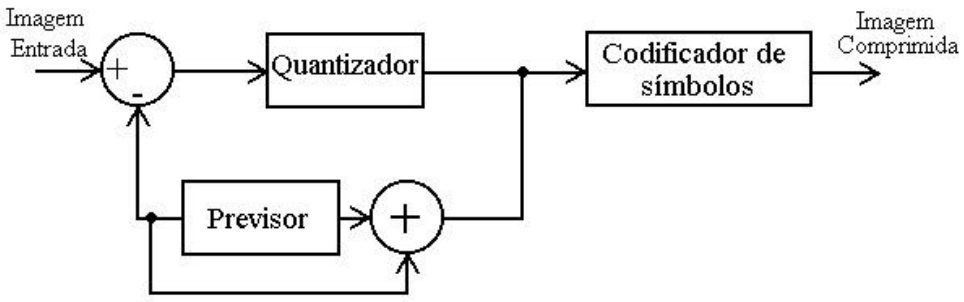


Figura 6.5: Um modelo de codificador previsor com perdas.

Várias distorções podem ocorrer dependendo dos métodos de previsão e quantização empregados. Previsores ótimos são projetados para minimizar o erro médio quadrático e quantizadores podem usar medidas estatísticas ou psicovisuais para definir os níveis de quantização, melhorando significativamente a qualidade da imagem recuperada dos sistemas de compressão.

A modulação de código de pulso diferencial (DPCM) é um exemplo bastante conhecido que pode empregar previsores ótimos e quantizadores adaptativos.

Quando se trabalha com imagens e vídeos digitais, várias são as possibilidades do uso da predição. Três modos interessantes podem ser aplicados:

- Codificação de diferenças espaciais (intra-quadro): Cada amostra de uma imagem é predita usando amostras vizinhas no espaço, explorando as redundâncias espaciais;

- Codificação inter-quadro: ao se trabalhar com vídeo, pode-se fazer a predição dos valores dos pixels de uma imagem se baseando na anterior, transmitindo-se apenas o erro de predição para as zonas das cenas que mudaram, explorando assim, redundância temporal;

- Codificação inter-quadro com compensação de movimento: considerando que durante o vídeo os objetos se movem ao longo dos quadros, pode-se diminuir o erro de predição detectando-se e compensando o movimento, ou seja, usando não só a amostra do quadro anterior, como também as correspondentes deslocadas em caso de movimento.

6.4.3 Codificação por Transformada

O princípio da codificação por transformada se baseia na capacidade de se poder passar uma informação no domínio espacial ou temporal para um outro domínio através de operadores matemáticos, sendo que esse processo deve ser reversível. A idéia principal é pegar a imagem transformada e priorizar a codificação dos coeficientes que contêm maior energia.

Várias transformadas podem ser usadas com esse propósito, dentre as mais conhecidas têm-se a Transformada Discreta de Wavelets (DWT), a de Karhunen-Loève (KLT), a Walsh-Hadamard (WHT), e as relacionadas diretamente com a de Fourier, a DFT e a de Cosseno Discreto (DCT).

A DFT é talvez o exemplo mais comum de transformadas de comprimento finito. Mas outra bastante relacionada com ela, é a DCT. Ela tem como base seqüências reais devido ao uso apenas dos cossenos. A DCT é usada em muitas aplicações de compressão de dados em preferência à DFT, devido a uma de suas propriedades. Comumente chamada de compactação de energia [4].

Da mesma forma que a DFT foi ampliada para trabalhar com duas variáveis, a DCT-2 ou bidimensional e sua inversa podem ser expressas pelas Equações (6.2) e (6.3).

$$F(u,v) = \frac{C(u)}{2}\frac{C(v)}{2}\sum_{y=0}^{N-1}\sum_{x=0}^{M-1} f(x,y)\cos\left[\frac{(2x+1)u\pi}{2M}\right]\cos\left[\frac{(2y+1)v\pi}{2nN}\right] \qquad \textbf{(6.2)}$$

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{M-1}\frac{C(u)}{2}\frac{C(v)}{2}F(u,v)\cos\left[\frac{(2x+1)u\pi}{2M}\right]\cos\left[\frac{(2y+1)v\pi}{2nN}\right] \qquad \textbf{(6.3)}$$

Onde:

$$\begin{gathered} C(u) = 1/\sqrt{M} \ \textbf{ para } u = 0 \textbf{ e } C(v) = 1/\sqrt{N} \ \textbf{ para } v = 0 \\ C(u) = \sqrt{2/M} \ \textbf{ para } 1 \le u \le M-1 \textbf{ e } C(v) = \sqrt{2/N} \ \textbf{ para } 1 \le v \le N-1 \end{gathered} \qquad \textbf{(6.4)}$$

A DCT de uma imagem de tamanho MxN vai retornar uma matriz de mesmo tamanho, mas geralmente, devido à grande quantidade de cálculos, usam-se um M e N pequenos e faz-se uma convolução sobre toda a imagem.

A propriedade de compactação da energia, vem do fato de que os coeficientes de maior energia ficam concentrados nos índices menores dessa matriz, correspondendo ao seu canto superior direito. Observe a Figura 6.6 que mostra a energia dos coeficientes da DCT com $M = N = 8$.

(a) **(b)**

Figura 6.6: (a) Imagem original. (b) Imagem mostrando a energia dos coeficientes da DCT, de acordo com a escala normalizada de –10 a 10.

Visto que a energia está concentrada nos coeficientes do canto direito superior, surge uma variedade de possíveis codificações para diminuir a quantidade de dados usados na representação da imagem. Poderia, por exemplo, definir níveis de quantização menores para esses coeficientes do que os menos representativos, e em seguida, usar uma codificação de comprimento variável. Reduzindo ainda mais, uma parte dos coeficientes poderia ser descartada. A Figura 6.7 mostra a imagem correspondente a DCT inversa de apenas 15% dos gerados originalmente, e uma outra correspondendo ao erro, com um limiar de 20 para facilitar a visualização. Observe que pelo fato das bordas corresponderem as componentes de alta freqüência, elas ficam um pouco suavizadas devido ao truncamento feito.

(a) **(b)**

Figura 6.7: (a) Imagem obtida da DCT inversa usando aproximadamente 15% dos coeficientes. (b) Imagem, que passou por uma limiarização ($T = 20$), mostrando os erros do truncamento.

6.5 Observações Importantes

Ao falar sobre compressão e codificação pode-se expor o assunto baseado não só na compressão com perdas ou sem, mas também dividir as técnicas em dois grupos, chamados de codificação de entropia e codificação de fonte. A primeira se refere às técnicas que não levam em conta a natureza da informação a ser comprimida, ou seja, trata os dados como seqüência de bits, sem a tentativa de se otimizar através do conhecimento de parâmetros estatísticos ou outro qualquer que indique a semântica da informação de entrada. Já no segundo, ocorre o contrário, dependendo das características da informação original pode-se ter fatores de compressão bem mais elevados. As técnicas de codificação de comprimento variável se enquadram na codificação de entropia, enquanto o DPCM e as por transformadas, se enquadram na codificação por fonte.

As técnicas explicadas possuem vantagens e desvantagens. A codificação por transformada, por exemplo, tem um bom fator de compressão, é menos suscetível a ruídos do canal, mas é mais complexa de ser implementada. Já a codificação por predição é mais vulnerável a ruídos, mas geralmente é mais simples [5]. Nos últimos anos, as dificuldades relacionadas à velocidade de processamento têm sido bastante amenizadas, possibilitando sistemas de compressão com várias técnicas de codificação em conjunto. A codificação híbrida, onde o DPCM opera em conjunto com as transformadas, é bastante utilizada para aumentar o fator de compressão sem aumentar muito as perdas de qualidade das imagens.

Este capítulo forneceu uma noção geral para os padrões de vídeo a serem introduzidos no próximo capítulo.

6.6 Referências

[1] P. D. SYMES, “Video Compressing”, McGraw-Hill, ISBN: 0-07-063344-4, 1998.

[2] R. C. GONZALES, R. E. WOODS, “Digital Image Processing”, Prentice Hall,2ª edição, ISBN: 0-201-18075-8, 2001.

[3] M. C. CHEHAB, “Implementação e Avaliação de Desempenho na Codificação de Imagens Coloridas”, Dissertação de Mestrado em Engenharia Elétrica, Universidade de Brasília, Brasil.

[4] A. V. OPPENHEIM, R. W. SCHAFER, J. BUCK, “Discrete-Time Signal Processing”, Prentice Hall, ISBN: 0-13-754920-2, 1998.

[5] K. R. RAO, J. J. HWANG, “Techniques & Standarts for Image Video & Audio Coding”, Prentice Hall PTR, ISBN 0-13-309907-5, 1996.

7. Padrões de Compressão

O progresso de padrões para áudio, imagem e vídeo, tem auxiliado no desenvolvimento de produtos e sistemas tanto físicos (hardware) quanto de métodos computacionais (software) em diversas disciplinas. O principal objetivo desses padrões é a compatibilidade e a interoperabilidade entre sistemas feitos por diferentes empresas.

Este capítulo abordará de forma superficial, alguns fóruns responsáveis pela padronização, e alguns padrões, como JPEG, MPEG e H.261. Este último será mais bem detalhado, com o objetivo de verificar a introdução de novas técnicas no melhoramento de seu desempenho.

7.1 Entidades de Padronização

Os formatos e algoritmos de compressão de imagens necessitam de um forte grau de padronização, de forma a possibilitar a troca de informações entre diferentes organizações. Os principais responsáveis por isso se dividem em três grupos [1]:

- Organismos oficiais de padronização a nível internacional – ITU, ISO/IEC;
- Organismos de padronização de fato – IETF;
- Fóruns de usuários, fornecedores e operadores de serviço – MERCI, DAVIC, IMTC.

A União Internacional de Telecomunicações (ITU) é um dos mais importantes fóruns de padronização. Ele é dividido em basicamente dois setores: o de Padronização de Telecomunicações (ITU-T), responsável pelos padrões na área de telegrafia, telefonia e telecomunicações de uma forma geral; e o de Padronização de Rádio (ITU-R), responsável pela padronização de sistemas de rádio, incluindo aspectos de propagação, difusão e canalização.

A Organização de Padrões Internacional (ISO), em conjunto com a Comissão Eletrotécnica Internacional (IEC), é responsável por diversos padrões de transmissão e armazenamento de imagens e vídeos. A ISO juntamente com a ITU-T definem padrões como o JPEG, MPEG e H.261.

O IETF é o organismo responsável por definir os padrões na internet, como por exemplo, protocolos da família TCP/IP.

Embora a padronização seja implícita para operações de codificação básica, há uma liberdade e flexibilidade para o desenvolvimento de dispositivos, pois somente a sintaxe e a semântica da seqüência de bits são especificadas pelos padrões.

7.2 JPEG

Este padrão foi especialmente desenvolvido para a compressão de imagens fotográficas, pois na natureza não se têm mudanças muito bruscas de intensidade, o que permite o uso de transformadas e filtros passa-baixas em seu código. Ele é largamente usado em várias aplicações por poder ter uma implementação onde não se exige muita velocidade de processamento e conseguir uma boa relação de compressão e qualidade das imagens [2].

O JPEG específica um ou mais algoritmos para cada modo listado a seguir [3]:

- Codificação seqüencial: cada componente de cor é codificada em uma única passada, da esquerda para direita, do alto para baixo (*zig-zag*), em blocos de tamanho 8x8. Usa-se a transformada DCT para fazer a compressão com perdas. Os coeficientes resultantes são quantizados através de sua divisão por um fator de escala que aumenta proporcionalmente a freqüência associada a eles. O bloco é reorganizado e codificado através da metodologia de Huffman;

- Codificação progressiva: cada componente é codificado em várias passadas. Em cada uma delas, codifica-se sub-amostras da imagem original, de maneira a possibilitar o decodificador obter rapidamente uma aproximação grosseira da imagem, e ir melhorando a cada passo;

- Codificação sem perdas: utiliza o método de compressão DPCM sem perdas. Tem um menor fator de compressão;

- Codificação hierárquica: cada componente pode ser codificada em múltiplas resoluções, possibilitando a decodificação na resolução mais adequada ao receptor de forma a otimizar o esforço computacional necessário.

A atual taxa de compressão desse padrão varia de 2:1 até 100:1, sendo essa escolha baseada na complexidade do sistema e na qualidade mínima para uma determinada aplicação. O JPEG é usado em fac-símile colorido, jornais que contenha fotos, imagens médicas, câmeras digitais, scanners, entre outros.

7.3 MPEG

Com o desenvolvimento das tecnologias em vídeo digital, a utilização da compressão se tornou necessária para viabilizar vários tipos de aplicações. O grupo MPEG que está sob comando do ISO/IEC desenvolve padrões para codificação de vídeo, áudio e a combinação dos dois [4]. Dois padrões já foram estabelecidos, sendo eles designados de MPEG-1 e MPEG-2, e outros dois estão em desenvolvimento, o MPEG-4 e o MPEG-7.

O MPEG-1 trabalha com vídeos com resolução de 352x240 (CIF), com uma amostragem de 4:1:0. Usa a transformada DCT, quantização escalar, predição de movimento entre quadros subseqüentes e o código de Huffman. Opera numa taxa de aproximadamente 1,5 Mbps. É usado em mídias de armazenamento como CD-ROM, discos rígidos, ópticos e em aplicações de multimídia. Tem qualidade similar à do videocassete VHS.

O MPEG-2 trabalha com taxas maiores, de 1,5 a 20 Mbps [5], operando com ótima qualidade na faixa de 4 a 6 Mbps. Trabalha com três tipos de resolução, 352x240, 720x486 e 1920x1080. Usa amostragens de 4:1:0, 4:1:1 e 4:4:4. Tem uma predição de movimentos aperfeiçoada e escalabilidade temporal e espacial. Usa também a DCT, quantização escalar e código de Huffman [1].

O MPEG-4 tem o objetivo de prover algoritmos para trabalhar com baixas taxas de codificação de dados audiovisuais. Mas devido às necessidades da nova geração de aplicações, algumas características têm sido mudadas. Ele busca uma maior eficiência na codificação; uma escalabilidade baseada no conteúdo; eficiência na representação tanto de objetos naturais como sintéticos e uma maior robustez a erros.

Muitas das aplicações multimídia em desenvolvimento necessitam que as informações audiovisuais sejam efetivamente e eficientemente acessadas e manipuladas [6]. O MPEG-7 busca uma codificação baseada no conteúdo, o que vai permitir uma incrível evolução e surgimento de aplicações.

7.4 H.261

A recomendação H.261 foi escrita para ser compatível com os padrões de vídeos mais comuns, descrevendo métodos de codificação e decodificação de imagens em movimento, de serviços que operem a taxas de px64 kbits/s, com p variando de 1 até 30. Foi principalmente desenvolvida para serviços de videotelefonia e videoconferência.

A descrição geral da recomendação ITU-T H.261 está em anexo. Neste tópico será feito um resumo de forma analítica do mesmo.

O formato de vídeo do padrão H.261 específica o sistema de cor YCrCb como formato do "video data". O Y representa a luminância, enquanto Cr e Cb as componentes de crominância. Cr e Cb são subamostrados por um fator de 4 se comparado a Y, devido a maior sensibilidade do olho humano à luminância. O tamanho dos vídeos suportados pelo padrão H.261 são o CIF e o QCIF. CIF – "Common Intermediate Format" tem uma resolução de 352x288 pixels de Y e 176x144 pixels para Cr e Cb. O "Quarter-CIF" contêm um quarto do CIF, logo, tem uma resolução de 176x144 pixels e 88x72 pixels para luminância e crominância respectivamente.

O codificador fonte do H.261 emprega compensação de movimento em conjunto com a DCT (MC-DCT). Ele pode operar em dois modos. No modo intra-picture, aplica-se a DCT em um bloco 8x8, faz a quantização, e manda para o codificador de multiplexação de vídeo. No modo inter-picture, o compensador de movimento é usado para comparar o macrobloco do quadro atual com os blocos de dados do quadro que foi enviado anteriormente. Se a diferença, que também é conhecida como erro de predição, estiver abaixo de um limiar pré-determinado, nenhum dado é mandado por esse bloco, caso contrário, o bloco-diferença sofre a DCT, a quantização, e é enviado para o codificador de multiplexação de vídeo. O estimador de movimentos é usado apenas quando a diferença entre o vetor de movimento do macrobloco atual e o anterior tiver sido mandado. Um filtro "loop" é usado para melhorar a qualidade do vídeo através da remoção de ruídos de alta freqüência, enquanto o controle de codificação é usado para selecionar o modo intra-picture ou inter-picture e também para controlar o tamanho dos níveis usados na quantização. No codificador de multiplexação de vídeo, o "bitstream" é melhor comprimido quando os coeficientes DCT quantizado são "scaneados" em uma ordem "zigzag" e então usa-se um código de comprimento variável e mapeamento por Huffman. A saída do codificador de multiplexação de vídeo é colocada em um buffer de transmissão. Então uma estratégia de controle da taxa que controla o quantizador vai ser usada para regular a saída de "bitstream" (seqüência de bits).

A seqüência de dados comprimidos é arranjada hierarquicamente em 4 camadas, chamadas: Imagem, Grupo de blocos (GOB) , Macroblocos (MB) e Bloco;

A imagem é a camada topo, ela pode ser QCIF ou CIF. Cada imagem é dividida em grupos de blocos (GOBs). Uma imagem CIF tem 12 GOBs enquanto uma QCIF tem 3. Cada GOB é composto de 33 macroblocos (MBs) em um arranjo 3x11, e cada MB é feito de 4 blocos de luminância e 2 de crominância (Cr e Cb). O bloco é um arranjo 8x8 de pixels.

A transmissão de dados de vídeo no H.261 começa pela camada de imagem. O cabeçalho da imagem contém o código de início da imagem, a referência temporal, o tipo de imagem e outras informações. A camada GOB contém um cabeçalho seguido pela camada MB. O cabeçalho GOB inclui o código de início do GOB, o número do grupo, os valores de quantização do GOB e outras informações. Um MB tem um cabeçalho seguido pelos dados da camada bloco. Um MB típico consiste de um endereço de MB, tipo, valor de quantização, vetor de movimento e o molde do bloco codificado. Os dados de uma camada de bloco contêm os coeficientes DCT quantizados e um comprimento fixo EOB "codeword" para sinalizar o fim do bloco.

O padrão H.261 não inclui algumas decisões de codificação [7]. Entre elas têm-se:

- o critério de escolha de um ou outro para transmitir ou manter um MB;
- o controle do mecanismo de codificação intra-picture ou inter-picture;
- o uso e a derivação do vetor de movimento;
- a opção de aplicar um filtro linear para o quadro previamente decodificado antes de usa-lo para predição;
- a estratégia do controle da taxa e, portanto, o ajuste do tamanho dos níveis de quantização.

7.5 Referências

[1] M. C. CHEHAB, "Implementação e Avaliação de Desempenho na Codificação de Imagens Coloridas", Dissertação de Mestrado em Engenharia Elétrica, Universidade de Brasília, Brasil.

[2] P. D. SYMES, "Video Compressing", McGraw-Hill, ISBN: 0-07-063344-4, 1998.

[3] K. R. RAO, J. J. HWANG, "Techniques & Standarts for Image Video & Audio Coding", Prentice Hall PTR, ISBN 0-13-309907-5, 1996.

[4] S. ARAMVITH, M. T. SUN, "MPEG-1 and MPEG-2 Video Standarts", Academic Press, Handbook of Image and Video Processing, ISBN: 0-12-119790-5, 2000.

[5] A. M. Tekalp, "Digital Video Processing", University of Rochester, Prentice Hall, ISBN: 0-13-190075-7, 1995.

[6] B. EROL, A. DUMITRAS, F. KOSSENTINI, "Emerging MPEG Standarts: MPEG-4 and MPEG-7", Academic Press, Handbook of Image and Video Processing, ISBN: 0-12-119790-5, 2000.

[7] K. N. NGAN, T. MEIER, D. CHAI, "Advanced Video Coding Principles and Techniques", Elsevier, ISBN: 0-444-82667-X, 1999.

8. Conclusão

Com a evolução que o mundo digital tem passado a cada dia, se torna cada vez mais importante o desenvolvimento de novas técnicas. Este trabalho tenta contribuir com o conhecimento básico das ferramentas de processamento de imagens digitais, dos termos e blocos presentes nos padrões de vídeos existentes, bem como, procura despertar a curiosidade do leitor para as inúmeras possibilidades de aplicação.

Os diversos exemplos evidenciaram que a escolha das técnicas deve ser escolhida de acordo com a origem e características desejadas em uma imagem. As necessidades encontradas para visualização ou representação das informações visuais geram ferramentas mais específicas e eficientes. Os diferentes tipos de realce obtidos de diversas maneiras são um passo fundamental no pré e pós-processamento, viabilizando e melhorando os resultados obtidos.

As segmentações abordadas foram baseadas em descontinuidades, similaridades e no movimento presente numa seqüência de imagens. A detecção de descontinuidade se mostrou muito sensível a ruídos, quebras de borda e a presença de regiões com textura, mas foi rápida e eficiente para imagens com bom contraste e uniforme. Na aplicação da limiarização deve-se levar em conta as condições de iluminação e modos caracterizados pelo histograma da imagem, para obtenção do resultado desejado. A segmentação orientada à regiões se mostrou eficiente, mas com um elevado tempo de processamento. O uso da morfologia melhora a eficiência de alguns operadores por levar em consideração as formas do objeto. A potencialidade da técnica *watershed* depende da escolha dos parâmetros envolvidos no processo, como disposição, forma e tamanho dos objetos de interesse. Bem escolhidos, obtêm-se resultados impressionantes.

A combinação de técnicas para obtenção de uma melhor segmentação foi mostrada com o uso de morfologia, divisões e fusões de regiões. Os resultados mostraram a eficiência na redução do número de níveis de cinza usados na representação da imagem. Isso viabilizaria um melhoramento do uso dos codificadores que usam representações de tamanho variável. Mas mostrou-se vulnerável a iluminação não-uniforme e a presença de texturas, bem como, a necessidade de um tempo razoável de processamento. O desenvolvimento de critérios de fusão que levem em conta informações como textura, cor e movimento podem melhorar significativamente o método, podendo ser adaptado a criação de máscaras indicadoras das regiões de interesse.

A segmentação utilizando o movimento, é prática e rápida na criação de máscaras. Os problemas de detecção, apresentados com o uso do fluxo ótico simples, são parcialmente resolvidos

com a metologia apresentada por Horn e Schunck. As falhas como as várias descontinuidades na região de movimento e a indicação errônea em alguns locais do fundo, causados por ruídos gerados pelos dispositivos de captura das imagens, podem ser processadas com ferramentas de morfologia para melhoria do resultado.

A DCT mereceu destaque por ter se mostrado uma ferramenta poderosa na redução de dados necessários na representação da imagem.

A necessidade de novas técnicas que objetivem a manipulação de informações visuais, demanda o conhecimento das variadas ferramentas de processamento de imagens, bem como, o conhecimento dos padrões existentes, e das possibilidades de melhoria com o aproveitamento do que não é especificado pelas normas.

Finalmente, esse trabalho serve para uma iniciação no mundo de processamento de imagens digitais e no despertar de idéias para o desenvolvimento de novas aplicações.

I. ANEXO - GLOSSÁRIO

CIF	*Common Intermediate Format* – padrão de imagem – 352x288
DAVIC	Digital Audio Video Interoperability Council - URL: http://www.cnm.bell-atl.com/davic/davic.html
DCT	Discrete Cossine Transform –Transformada baseada na de Fourier sendo ela mais eficaz no quesito compressão de imagens.
DFT	Discrete Fourier Transform
DPCM	Differential Pulse Code Modulation – Tipo de modulação que usa um sistema de codificação com previsor.
DWT	Discrete Wavelets Transform -
GIF	Formato de imagem – Graphic Interchange Format
H.261	Recomendação de codificador de vídeo – descreve métodos de codificação e decodificação de vídeo para componentes de imagens em movimento para serviços audiovisuais a taxas de px64 kbit/s.
IEC	International Eletro-technical Commission - URL: http://www.iec.ch
IETF	Internet Engineering Task Force - URL: http://www.ietf.cnri.reston.va.us
IMTC Inc.	International Multimedia Teleconferencing Consortium - URL: http://www.csn.net/imtc
ISO	International Standards Organization - URL: http://www.iso.ch
ITU	International Telecomunications Union – URL: http://www.itu.ch
JPEG	*Joint Photographic Experts Group*
KLT	Karhunen-Loève Transform -
MERCI	Multimidia European Research Conferencing Integration - URL: http://www.cs.ulc.ac.uk/mice/merci
MPEG	*Motion Picture Experts Group*
QCIF	*Quarter Common Intermediate Format* - padrão de imagem – 176x144
PDF	Portable Document Format – Formato de Imagem
RGB	Red Green Blue – Vermelho Verde Azul – Cores que correspondem aos comprimentos de onda percebidos pelo olho humano.
RMS	*Root Mean Square Error* – Erro raiz de média quadrática – Critério de medida para expressar erro de variáveis processadas em relação a suas originais (ou teóricas).
TIFF	Tagged Image File Format – Formato de Imagem
WHT	Walsh-Hadamard transform – Transformada matemática

II. ANEXO – CÓDIGOS PRÓPRIOS

Este anexo contêm alguns dos algoritmos implementados para exemplificar os capítulos.

```
%--------------------------------------------------------------------
% Programa que muda os valores de resoluçao espacial e os exibe em %
% 256x256  - MATLAB
%--------------------------------------------------------------------
%
clear;
%
A=imread('Fabricio01_256_256.jpg'); % Le a imagem e armazena em uma matriz
B=A;
%
imshow(A)  %Mostra a imagem de referencia com 256x256x256
pause
%
%-------------------------------------------------------------------------
-
% Bloco que reduz a imagem para o tamanho desejado
%
k=[256 128 64];  %Valores de origem reduçao(/2) da resoluçao espacial
C=zeros(256,256);   % Cria uma matriz de 256x256 preenchida com zeros.
for i=k
    B(:,1:2:i)=[];   %Deleta as colunas de 2 em duas ate o valor k
    B(1:2:i,:)=[];   %Deleta as linhas de 2 em duas ate o valor k
%
    s=2*(256/i);     %Fator de escala para utilizaçao dos proximos loops
%
    %-----------------------------------------------------------------------
    %Bloco que retorna a imagem reduzida para o tamnho de 256x256
%
    for j=1:s
        for m=1:s
            C(j:s:256,m:s:256)=B(1:i/2,1:i/2);
        end
    end
    %-----------------------------------------------------------------------
%
    imshow(C,[0 255])  %Resultado da reduçao da resoluçao espacial
    pause              %com niveis de cinza de 0 a 255
end
```

```
%--------------------------------------------------------------------
% Programa que reduz o numero de bits na representaçao de cada pixel
% MATLAB
%--------------------------------------------------------------------
%
clear;
%
A=imread('Fabricio01_256_256.jpg');   % Le a imagem e armazena em uma
matriz
```

```
%
%-------------------------------------------------------------------
%Definindo os niveis de cinza
%
map(1,1)=0;   %Valor do 1º nivel de quantizaçao
for k=2:2:6
    map(2^k,1)=1;   %Valor do ultimo nivel de quantizaçao
    for i=2:1:((2^k)-1)
        map(i,1)=map(i-1,1)+(1/((2^k)-1)); %divide linearmente a faixa de 0
a 1
    end                                     %na quantidade de niveis

%-------------------------------------------------------------------
%Montando uma matriz de tamanho 256x3 para o funcionamento correto
%do imshow e imwrite
%
    p=1;
    for v=1:(2^k)      %Passa a matriz map de (2^k)x1 para 256x1
        for m=p:1:256
            for g=1:3
                nmap(m:1:256,g)=map(v,1);
            end
        end
        p=v*(256/(2^k));
    end

imshow(A,nmap)
pause
end
close
```

```
%-------------------------------------------------------------------
%   Algoritmos do Capitulo 4 relacionados com Morfologia - MATLAB
%-------------------------------------------------------------------

%-----------------------------------------------------
% Exemplo de Dilataçao e Erosao Imagens binarias

clear;
El_es=[0 1 0
       1 1 1
       0 1 0];  % Definindo um elemento estruturante

I1=imread('Original_teste2.jpg');  %Lendo a imagem
I1_d=imdilate(I1,El_es);           %Dilata a imagem I1
imshow (I1_d);                       %Mostra a imagem dilatada
imwrite(I1_d,'Original_teste2_dilatada.jpg','jpg'); %Salva resultado

I1_e=imerode(I1,El_es);           %Faz a erosao da imagem I1
imshow (I1_e);
imwrite(I1_e,'Original_teste2_erodita.jpg','jpg');


%-----------------------------------------------------
% Exemplo Hit-or-Miss

clear;
I2=imread('teste.bmp');
B1=ones(7,7);      %Elemento estruturante B1
B2=ones(11,11);
B2(3:9,3:9)=0;   % Elemento estruturante B2=complemento(B2)
I2_e=imerode(I2,B1);
I2_d=imdilate(I2,B2);
I2_f=imsubtract(I2_e,I2_d);  %Faz a subtracao de imagens
imshow (I2_f);
imwrite(I2_f,'teste_hitormiss.jpg','jpg');


%-----------------------------------------------------------
%Exemplo de Extracao de Contorno

clear;
I3=imread('Fabricio02_binaria.jpg');
B3=strel('diamond',1)    %Elemento estruturante em cruz
I3_e=imerode(I2,B1);
I3_f=imsubtract(I3,I3_e);  %resulta na extraçao de contorno
imshow (I3_f);
imwrite(I3_f,'Fabricio02_ext_contorno.jpg','jpg');

%-----------------------------------------------------------
% Exemplo de Dilataçao e Erosao Imagens Niveis Cinza

clear;
I4=imread('Fabricio01.jpg');
B3=strel('diamond',1);
I4_d=imdilate(I4,B3);
imshow (I4_d);
imwrite(I4_d,'Fabricio01_dilatado.jpg','jpg');

I4_e=imerode(I4,B3);           %Faz a erosao da imagem I1
imshow (I4_e);
imwrite(I4_e,'Fabricio01_erodita.jpg','jpg');
```

```
%-------------------------------------------------------------
% Exemplo de Suavizaçao - Abertura seguida Fechamento

clear;
I5=imread('Fabricio02.jpg');
B3=strel('diamond',1);
I5_e=imerode(I5,B3);
I5_d=imdilate(I5,B3);
I5_a=imdilate(I5_e,B3);      %Faz a abertura da imagem I5
I5_fe=imerode(I5_d,B3);      %Faz o fechamento da imagem I5
I5_suav=imerode(I5_a,B3);    %Imagem suavizada
imshow (I5_suav);
imwrite(I5_suav,'Fabricio02_suavizado.jpg','jpg');


%-------------------------------------------------------------
% Exemplo de Gradiente Morfologico

clear;
I5=imread('Fabricio02.jpg');
B3=strel('diamond',1);
I5_e=imerode(I5,B3);
I5_d=imdilate(I5,B3);
I5_grad=imsubtract(I5_d,I5_e);
imshow (I5_grad);
imwrite(I5_grad,'Fabricio02_gradiente_morf.jpg','jpg');


%-------------------------------------------------------------
% Exemplo de Gradiente Morfologico

clear;
I5=imread('Fabricio02.jpg');
B9=strel('diamond',3);
I5_e=imerode(I5,B9);
I5_d=imdilate(I5,B9);
I5_laplac=imsubtract(imsubtract(I5_d,I5),imsubtract(I5,I5_e));
imshow (I5_laplac);
imwrite(I5_laplac,'Fabricio02_laplaciano_morf.jpg','jpg');
```

```
%-----------------------------------------------------------------
%   Algoritmos do Capitulo 5 relacionados com Segmentacao
%-----------------------------------------------------------------

%-----------------------------------------------------------------
% DETECCAO DE DESCONTINUIDADES - MATLAB
%-----------------------------------------------------------------

clear;

F=imread('Fabricio02.jpg');

% Detecçao de pontos
Mdp=[-1 -1 -1
     -1 +8 -1
     -1 -1 -1];    % Mascara para detecçao de pontos.

DP=imfilter(F,Mdp);   % Filtrando a imagem usando a mascara Mdp
imwrite(DP,'Fabricio02_Detec_pontos.jpg','jpg'); % Salvando a imagem
imshow(DP)  % Mostrando o resultado final
pause;

%-----------------------------------------------------------------
% Detecçao de linhas
Mlh=[-1 -1 -1
     +2 +2 +2
     -1 -1 -1];    % Mascara para detecçao de linhas horizontais.
Mlv=[-1 +2 -1
     -1 +2 -1
     -1 +2 -1];    % Mascara para detecçao de linhas verticais.
Mldp=[-1 -1 +2
      -1 +2 -1
      +2 -1 -1];    % Mascara para detecçao de linhas diagonais +45º
Mldn=[+2 -1 -1
      -1 +2 -1
      -1 -1 +2];    % Mascara para detecçao de linhas diagonais -45º

DL1=imfilter(F,Mlh);
DL2=imfilter(F,Mlv);
DL3=imfilter(F,Mldp);
DL4=imfilter(F,Mldn);
DLT=imadd(imadd(imadd(DL1,DL2),DL3),DL4);    %Soma das imagens filtradas
imwrite(DLT,'Fabricio02_Detec_linhas.jpg','jpg');
imshow(DLT)
pause;

%-----------------------------------------------------------------------
% Detecçao de bordas por gradiente - Metodo de Prewitt.
Mdh=[-1 -1 -1
      0  0  0
     +1 +1 +1];    % Mascara para detecçao de bordas horizontais.
Mdv=[-1  0  +1
     -1  0  +1
     -1  0  +1];    % Mascara para detecçao de bordas verticais.
Mddp=[-1 -1  0
      -1  0 +1
       0 +1 +1];    % Mascara para detecçao de bordas diagonais +45º
Mddn=[ 0 +1 +1
      -1  0 +1
      -1 -1  0];    % Mascara para detecçao de bordas diagonais -45º
```

```
DD1=imfilter(F,Mdh);
DD2=imfilter(F,Mdv);
DD3=imfilter(F,Mddp);
DD4=imfilter(F,Mddn);
DDT=imadd(imadd(imadd(DL1,DL2),DL3),DL4);    %Soma das imagens filtradas
imwrite(DDT,'Fabricio02_Detec_Bordas_grad.jpg','jpg');
imshow(DDT)
pause;

%-----------------------------------------------------------------------
% Detecçao de bordas por laplaciano - Metodo de Prewitt.
MBg=[ 0  0 -1  0  0
      0 -1 -2 -1  0
     -1 -2 16 -2 -1
      0 -1 -2 -1  0
      0  0 -1  0  0];

DBg=imfilter(F,MBg);
imwrite(DBg,'Fabricio02_Detec_Bordas_lapl.jpg','jpg');
imshow(DBg)
pause;

%-----------------------------------------------------------------------
%Exemplo de limiarizaçao
imhis(F)
FL = im2bw(F,125/256);  % aplicaçao de limiarizaçao
imshow(FL)

close all;
clear;
```

```
%-----------------------------------------------------------
%   Algoritmo do Capitulo 5 relacionado com a implementação do
% paper “Image segmentation towards new image representation
% methods”
%-----------------------------------------------------------

clear;
% Imagem='Fabricio02_256';
Imagem='claire128';
N_nome=strcat(Imagem,'.bmp');
I=imread(N_nome);
tam=size(I);



%-----------------------------------------------------------------------
% Simplificaçao da imagem
%-----------------------------------------------------------------------

% Abertura por reconstruçao de erosao
El_es=mmsecross(1);
% El_es=mmsebox(1);
I_ar=mmopenrec(I,El_es);

% Seguida por fechamento por reconstruçao de dilatacao
I_Simple=mmcloserec(I_ar,El_es);
```

```
N_nome=strcat(Imagem,'_simplificada.bmp');
imwrite(I_Simple,N_nome,'bmp');
imshow(I_Simple)
pause
close

%-----------------------------------------------------------------------
% Extraçao das caractristicas
%-----------------------------------------------------------------------

%-----------------------------------------------------------------------
% Split

Lim_split=0.4;
I_S=qtdecomp(I_Simple,Lim_split);

% Cria as bordas de cada bloco.
Marca=im2uint8(zeros(max(size(I_S))));
for i=1:1:max(size(I_S))
    for j=1:1:max(size(I_S))
        if I_S(i,j)~=0
            Marca(i,j:j+(I_S(i,j))-1)=255;
            Marca(i:i+(I_S(i,j))-1,j)=255;
        end
    end
end

I_Ma=imadd(I,Marca);        %mostra a mascara sobre a imagem

N_nome=strcat(Imagem,'_Marcada.bmp');
imwrite(I_Ma,N_nome,'bmp');
imshow(I_Ma)
pause
close


%-----------------------------------------------------------------------
% Merge

% Uniao dos blocos pelo criterio da media

%calculo da media
M_Media=im2uint8(zeros(max(size(I_S))));
for i=1:1:max(size(I_S))
    for j=1:1:max(size(I_S))
        if I_S(i,j)~=0
            M_Media(i,j)=round(mean(mean(I(i:i+I_S(i,j)-1,j:j+I_S(i,j)-
1))));
        end
    end
end

Lim=15;   % limiar entre as medias de cada bloco

%Faz a comparaçao da media do bloco e de seus vizinhos.
%Se a diferenca ficar abaixo do limiar, os blocos sao unidos.
for i=1:1:max(size(I_S))
    for j=1:1:max(size(I_S))
        if I_S(i,j)~=0
            for t=i:i+I_S(i,j)
                for v=j:j+I_S(i,j)
```

```
                    if
((t<=tam(1))&(v<=tam(2))&((i+I_S(i,j))<=tam(1))&((j+I_S(i,j))<=tam(2)))
                        if M_Media(i+I_S(i,j),v)~=0
                            VI_1=im2double(M_Media(i,j));
                            VI_2=im2double(M_Media(i+I_S(i,j),v));
                            if abs(VI_1-VI_2)<=Lim/255
                                M_Media(i+I_S(i,j),v)=M_Media(i,j);
                            end
                        end
                        if M_Media(t,j+I_S(i,j))~=0
                            VI_1=im2double(M_Media(i,j));
                            VI_2=im2double(M_Media(t,j+I_S(i,j)));
                            if abs(VI_1-VI_2)<=Lim/255
                                M_Media(t,j+I_S(i,j))=M_Media(i,j);
                            end
                        end
                    end
                end
            end
        end
    end
end


% Preenchendo toda a matriz pela media do bloco
M_Result1=im2uint8(zeros(max(size(I_S))));
for i=1:1:max(size(I_S))
    for j=1:1:max(size(I_S))
        if I_S(i,j)~=0
            for r=i:i+I_S(i,j)-1
                for s=j:j+I_S(i,j)-1
                    M_Result1(r,s)=M_Media(i,j);
                end
            end
        end
    end
end

N_nome=strcat(Imagem,'_Final.bmp');
imwrite(M_Result1,N_nome,'bmp');
imshow(M_Result1)
pause
close


%--------------------------------------------------------------------------
%Conta quantos niveis de cinza sao usados na Imagem e mostra o Histograma

imhist(I);
pause
H1=imhist(I);
imhist(M_Result1);
pause
close
H2=imhist(M_Result1);
Increm=0;
Increm1=0;
for b=1:256
    if H1(b)~=0
        Increm=Increm+1;  %Qtdade de regioes na imagem original
    end
```

```
    if H2(b)~=0
        Increm1=Increm1+1;  %Qtdade de regioes na Imagem modificada
    end
end


%-------------------------------------------------------------------------
%Conta quantas regioes existem nas imagem

Incr=0;
for i=1:1:max(size(I_S))
    for j=1:1:max(size(I_S))
        if (I_S(i,j)~=0)
            Incr=Incr+1;
        end
    end
end

%Rotula de acordo com conexidade de 4
T1=mmlabelflat(M_Result1);
Incr1=max(max(T1));

%Mostra os indices calculados
P=[Increm,Increm1,Incr,Incr1]
```

```
%-------------------------------------------------------------------
% Cálculo do Fluxo Ótico – Capítulo 5 - MATLAB
%-------------------------------------------------------------------

clear;

% inicializa os parametros
a=1/12;
b=1/6;
F=[a b a;b 0 b;a b a];
NoI=300;
lambda=0.1;

% Le as imagens
im1=imread('claire141.jpg','jpg');
im2=imread('claire150.jpg','jpg');
I1=double(im1(:,:,1));
I2=double(im2(:,:,1));

% normaliza as imagens
I1=I1/max(I1(:));
I2=I2/max(I2(:));

% Passa pelo filtro gaussiano
G = fspecial('gaussian',5,5);   % cria filtro gaussiano passa-baixa com
                                %desvio de 5
I1 = filter2(G,I1,'same');
I2 = filter2(G,I2,'same');

% calcula fx, fy, ft
[fx fy]=gradient(I1);
ft=I2-I1;
```

```
% Passo 1: Inicializa U e V

% Tamanho da Imagem
[r c] = size(I1);
U=zeros(r,c);             %Aloca matriz U e V
V=zeros(r,c);

% Passo 2: Iteraçao para escolher U e V
for i=1:NoI
    % media U e V
    UA=conv2(U,F,'same');   % convolui U com F para pegar a
                            %media dos vizinhos
    VA=conv2(V,F,'same');   % convolui V com F para pegar a media dos
                            % vizinhos
   % atualiza P e D
    D=lambda^2+fx.^2+fy.^2;
    P=fx.*UA+fy.*VA+ft;
    % atualiza U e V
    U=UA-fx.*(P./D);
    V=VA-fy.*(P./D);
end

% plota o resultado
[X,Y]=meshgrid((1:size(I1,2)),(1:size(I1,1)));
colormap('gray')
imagesc(im1);                            % Decide a imagem que vai servir
de fundo%
hold on
e=10;
X = X(1:e:end,1:e:end);                 % Apenas plota a cada "e" setas
Y = Y(1:e:end,1:e:end);
U = U(1:e:end,1:e:end);
V = V(1:e:end,1:e:end);
quiver(X,Y,U,V,2,'r');                  % Plota o vetor de velocidade, com
escala de 3.
pause
close all;
```

```
%-------------------------------------------------------------
%   Algoritmos do Capitulo 6 relacionados com Compressao
%-------------------------------------------------------------

%-----------------------------------------------------------------
% Compressão por DCT


I = imread('Fabricio02.jpg');

J = dct2(I);
%imshow(log(abs(J)),[]), colormap(jet(64)), colorbar % Para mostrar
colorido
imshow(log(abs(J)),[]), colormap(gray(256)), colorbar %Mostra a energia dos
coeficientes da DCT
pause

I = im2double(I);
T = dctmtx(8);
B = blkproc(I,[8 8],'P1*x*P2',T,T');          %Faz um processamento com uma
DCT 8x8
mask = [1   1   1   1   0   0   0   0
        1   1   1   0   0   0   0   0
        1   1   0   0   0   0   0   0
        1   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0];
B2 = blkproc(B,[8 8],'P1.*x',mask);
I2 = blkproc(B2,[8 8],'P1*x*P2',T',T);
imshow(I), figure, imshow(I2)
```

III. ANEXO – H.261

O diagrama de blocos que representa o H.261 está ilustrado na Figura 1.

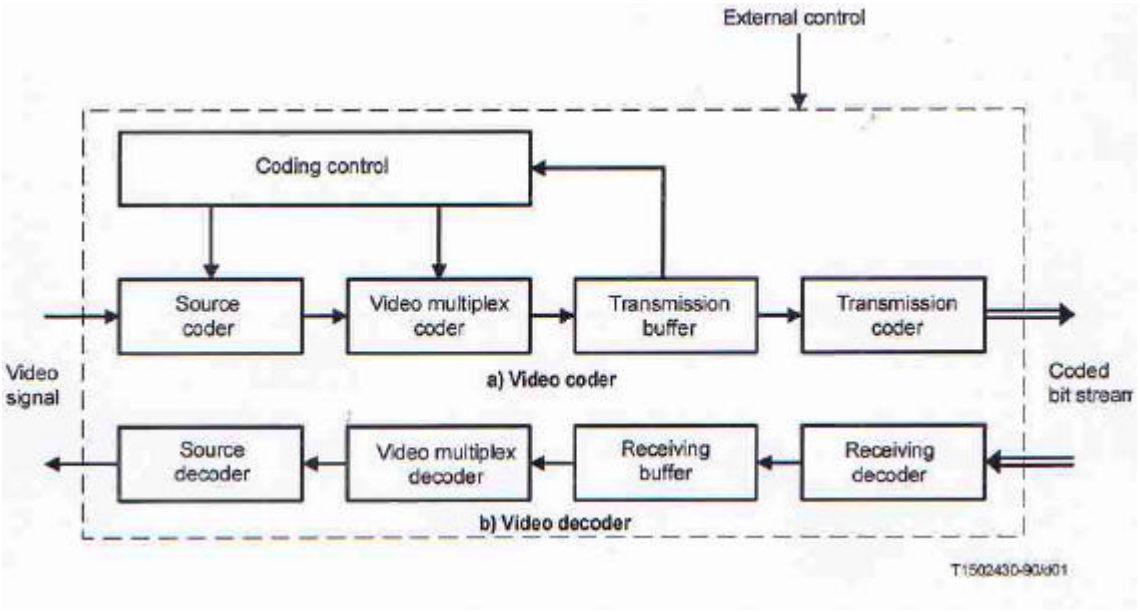


Figura 8 – Esquema retirado da recomendação H.261

ESPECIFICAÇÕES GERAIS

Os padrões de vídeo de cada país se diferenciam no número de linhas, na freqüência de campo, na freqüência de quadro, na banda do canal de vídeo, e no tipo de modulação de som.

Um dos primeiros passos para resolver esse problema de formato adotado por esta recomendação foi o projeto de um codificador fonte que trabalha-se com imagens baseadas num formato intermediário comum (CIF). Este formato será explicado mais a frente ao descrevermos o codificador fonte.

Outros padrões de entrada e saída de sinais de televisão não são sujeitos dessa regulamentação.

O codificador de vídeo despacha uma contida seqüência de bits que pode ser combinada com outros sinais.

As imagens devem ser amostradas numa taxa que deverá ser um número múltiplo inteiro do número de linhas do vídeo.

Também é adotada uma técnica de redução da redundância espacial e temporal baseados em transformadas e predições respectivamente. O decodificador tem a capacidade de estimação de movimento, que pode ou não ser usada.

A recomendação, também, pede para se usar taxas de bits de vídeo que estejam entre 40 kbit/s e 2 Mbit/s.

O codificador pode ser usado para uma comunicação uni ou bi-direcional.

As seqüências de bits transmitidas contêm um código de correção de erro BCH. O uso dele no decodificador é opcional.

CODIFICADOR FONTE.

O codificador fonte pode ser esquematizado conforme a Figura 2.

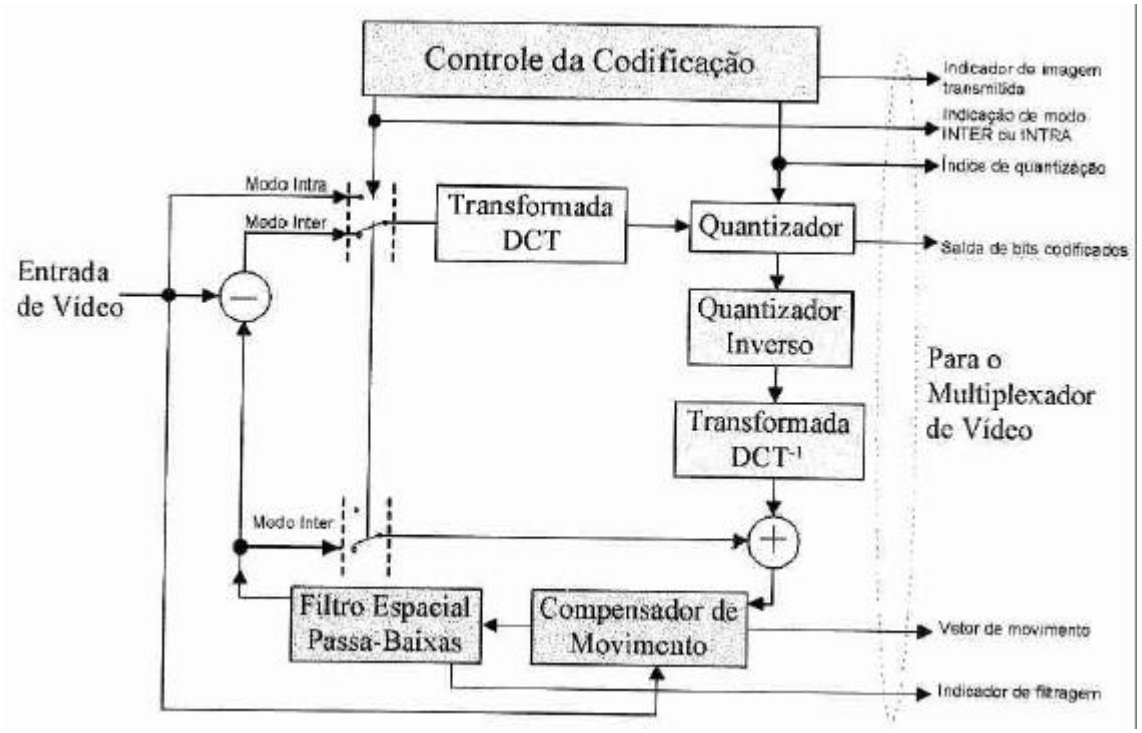


Figura 9 – Codificador Fonte

FORMATO DA FONTE

O codificador fonte opera com imagens não interlaçadas que ocorrem a uma taxa de 30000/10001 (~29,97) vezes por segundo, com uma tolerância de ± 50 ppm.

As imagens são codificadas em luminância e duas componentes de cores distintas, provendo o formato Y, C_B, e C_R. As definições dessas componentes são estabelecidas na recomendação CCIR 601.

Dois formatos de imagens são especificados. O primeiro é o CIF, onde a estrutura de amostragem da luminância é de 352 pels por linha, 288 linhas por imagem numa disposição ortogonal. A amostragem de cada uma das duas componentes de diferenciação de cor é realizada a 176 pels por linha, 144 linhas por imagem, no plano ortogonal. As amostras de diferenciação de cor estão situadas de tal maneira que as fronteiras de seus blocos coincidam com as fronteiras do bloco de luminância. A Figura 3 mostra esses limites. O segundo formato, quarter-CIF (QCIF), tem metade do número de pels e metade do número de linhas do caso CIF. O formato QCIF deve ser suportado por todos os codificadores, o que não é obrigatório para o formato CIF.

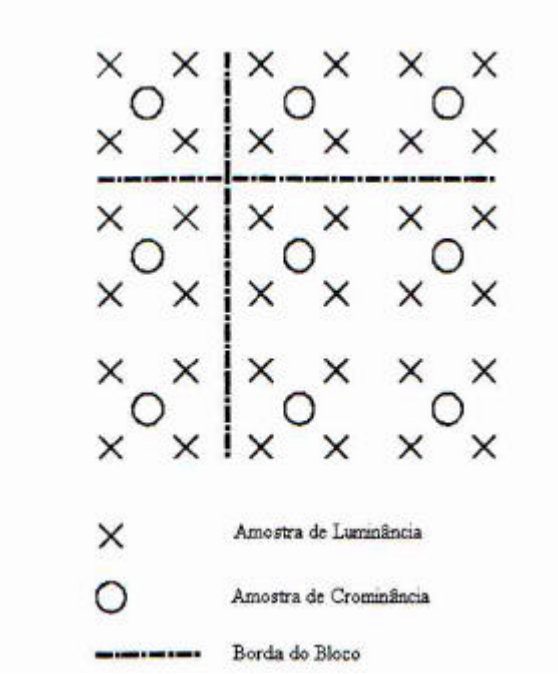


Figura 10 – Amostragem 4:1 – Espaço de Cor

ALGORITMO DE CODIFICAÇÃO DE VÍDEO

Como visto na Figura 2, os elementos principais do codificador fonte são a predição, o bloco de transformação e a quantização.

O erro de predição (modo INTER) ou a entrada de imagens (modo INTRA) é subdividido em blocos com 8 pels por 8 linhas no qual são segmentados e transmitidos ou não. Além disso, quatro blocos de luminância e dois correspondentes as cores são combinados para formar um macrobloco.

O critério de escolha do modo e transmissão de um bloco não está sujeito a recomendação e podem ser variados dinamicamente como parte duma estratégia de controle de codificação. Os blocos transmitidos são transformados e coeficientes resultantes são quantizados e codificados em comprimentos variáveis.

PREDIÇÃO

A predição é inter-picture e pode ser acrescida pela compensação de movimento, e por um filtro espacial.

A COMPENSAÇÃO DE MOVIMENTO

A compensação de movimento (MC) é opcional no codificador. E o decodificador aceitará um vetor por macrobloco. As componentes horizontais e verticais destes vetores de movimento têm valores inteiros que não excedem em ± 15. O vetor é utilizado para todos os quatro blocos de luminância presentes no macrobloco.

O FILTRO "LOOP"

O processo de predição pode ser modificado por um filtro especial bidimensional (FIL) que opera em pels com uma predição num bloco de 8 x 8.

A TRANSFORMADA

Os blocos transmitidos são primeiramente processados por uma transformada bidimensional cosseno discreta (DCT) de tamanho 8 x 8. A saída da transformada inversa pode variar de –256 a +255 depois de serem representadas com 9 bits. A função de transferência da transformada inversa é dada por:

$$f(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)\cos\left[\pi(2x+1)\frac{u}{16}\right]\cos\left[\pi(2y+1)\frac{v}{16}\right]$$

Com u, v, x, y = 0, 1, 2, ..., 7.

Onde:

x, y → coordenadas espaciais no domínio pel;

u, v → coordenadas no domínio da transformada;

C(u) = $1/\sqrt{2}$, para u = 0; e igual a 1, caso contrário;

C(v) = $1/\sqrt{2}$, para v = 0; e igual a 1, caso contrário.

Deve-se salientar que dentro do bloco onde está se aplicando a transformada, x=0 e y=0 referem-se ao pel mais próximo situado a esquerda e na borda superior da imagem, respectivamente.

QUANTIZAÇÃO

O número de quantizadores é 1 para o coeficiente INTRA dc, e 31 para todos os outros coeficientes. Dentro de um macrobloco o mesmo quantizador é utilizado para todos os coeficientes, exceto para o INTRA dc. Os níveis de decisão não estão definidos. O coeficiente INTRA dc é, normalmente, o valor transformado e quantizado linearmente com um tamanho de 8 e sem zona morta. Cada um dos 31 quantizadores é também normalmente linear, mas com uma zona morta central em torno do zero e com um tamanho de passo de valor par que varia entre 2 e 62.

Para prever distorção de quantização nos coeficientes de amplitude da transformada o que causaria um "overflow" nos laços do codificador e decodificador, funções que limitam esses valores são inseridas.

CODIFICADOR MULTIPLEX DE VÍDEO

ESTRUTURA DE DADOS

A menos que seja especificado de outra maneira, o bit mais significativo é transmitido primeiro. A menos que seja especificado de outra maneira, todos os bits sem função ou sobressalentes devem apresentar o valor "1". Os bits sobressalentes não devem ser utilizados até que suas funções sejam especificadas pelo CCITT.

DISPOSIÇÃO DO MULTIPLEXADOR DE VÍDEO

O multiplexador de vídeo é disposto numa estrutura hierárquica com quatro camadas, que são:

- Imagem
- Grupo de Blocos (GOB)
- Macrobloco (MB)
- Bloco

Um diagrama com a sintaxe do multiplexador de vídeo é mostrada na Figura 4.

Na seqüência, estão explicados de forma bem sucinta os componentes de cada camada, visto que está parte não é essencial para a consolidação do objetivo proposto. Para maiores detalhes consultar a recomendação completa H.261, [1].

CAMADA DE IMAGEM

Os dados para cada imagem consistem de um cabeçalho de imagem seguido pelos dados dos Grupos de Blocos (GOB's).

Picture start code (PSC) (20 bits): È uma palavra de 20 bits. Seu valor é 0000 0000 0000 0001 0000.

Temporal reference (TR) (5 bits): È um número de 5 bits que pode ter 32 valores possíveis.

Type information (PTYPE) (6bits): Indica as informações sobre a imagem completa:

Extra insertion information (PEI) (1 bit): É um bit que sinaliza a presença de um campo de dados opcional.

Spare information (PSPARE) (0/8/16...bits): Se o PEI estiver como valor "1", então os 9 bits seguintes consistem de 8 bits de dados (PSPARE) e um outro bit PEI para indicar se há mais 9 bits.

A CAMADA DE GRUPO DE BLOCOS

Cada imagem é dividida em grupos de blocos (GOB's). Um grupo de blocos compreende 1/12 do CIF ou 1/3 do QCIF das áreas das imagens. Um GOB corresponde a 176 pels por 48 linhas de Y e à correspondência espacial de 88 pels por 24 linhas de cada C_B e C_R.

Os dados de cada grupo de blocos consistem de um cabeçalho GOB seguido pelos dados dos macroblocos.

Group of start code (GBSC) (16 bits): É uma palavra de 16 bits, 0000 0000 0000 0001.

Group number (GN) (4 bits): São os quatro bits que indicam a posição do grupo de blocos.

Quantizer information (GQUANT) (5 bits) :É uma palavra-código de tamanho fixo de 5 bits que indica o quantizador a ser utilizado no grupo de blocos até ser anulado por qualquer MQUANT subseqüente. As palavras-código são as representações binárias naturais dos valores de QUANT (1 a 31).

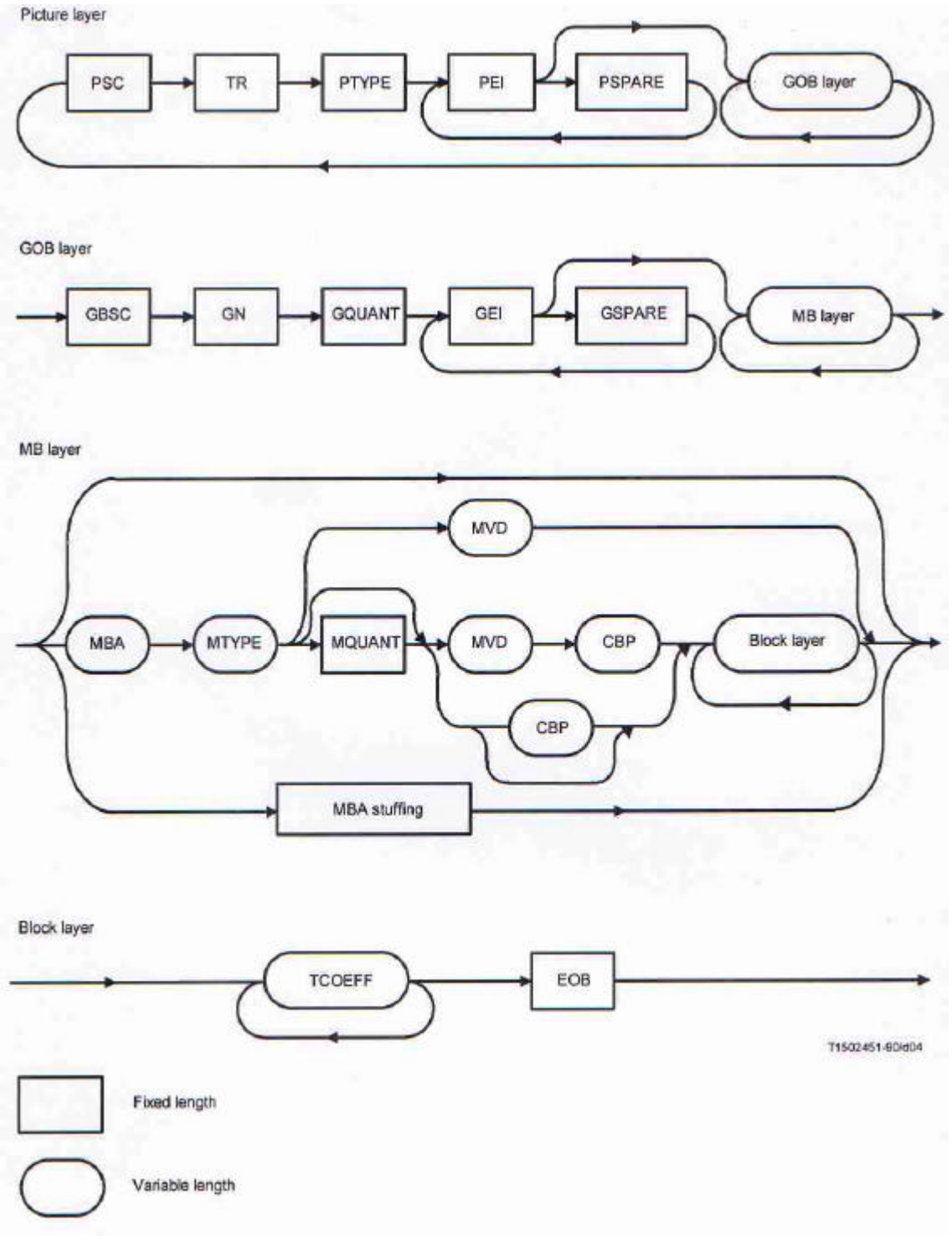


Figura 11 – Diagrama de sintaxe do codificador multiplex de vídeo

Extra insertion information (GEI) (1 bit): É um bit que sinaliza a presença de um campo de dados opcional.

Spare information (GSPARE) (0/8/16...bits): Se o GEI estiver como valor “1”, então os 9 bits seguintes consistem de 8 bits de dados (GSPARE) e um outro bit GEI para indicar se há mais 9 bits.

A CAMADA DE MACROBLOCO

Cada GOB é dividido em 33 macroblocos. Um macrobloco se relaciona a 16 pels por 16 linhas de Y a uma correspondência espacial de 8 pels por 8 linhas para cada C_B e C_R.

Os dados de um macrobloco consistem de um cabeçalho MB, seguidos por dados dos blocos. MQUNAT, MVD e CBP estão presentes quando indicados pelo MTYPE.

Macroblock adrress (MBA) (Tamanho variável): È uma palavra-código de comprimento variável que indica a posição do macrobloco dentro de um grupo de blocos. Para o primeiro macrobloco transmitido em um GOB, o MBA é o endereço absoluto. Para os macroblocos subseqüentes, o MBA é a diferença entre o endereço absoluto do macrobloco e o último macrobloco transmitido.

Type information (MTYPE) (Tamanho variável): Palavras-código de tamanho variável que informam a respeito do macrobloco e que elementos de dados estão presentes.

Quantizer (MQUANT) (5 bits): MQUANT está presente somente se indicado pelo MTYPE. Uma palavra-código de 5 bits significando o quantizador a ser usado nos blocos seguintes no grupo de blocos. As palavras-código para MQUANT são as mesmas utilizadas para GQUANT.

Motion vector data (MVD) (comprimento variável): Os dados do vetor de movimento são incluídos para todos os macroblocos (MC). O MVD é obtido do vetor de macrobloco pela subtração do vetor de macrobloco precedente.

Coded block pattern (CBP) (comprimento variável): O CBP está presente nos casos em que foi indicado pelo MTYPE. A palavra-código fornece um número padrão que significa que aqueles blocos no macrobloco para os quais ao menos um coeficiente da transformada é transmitido.

CAMADA DE BLOCOS

Um macrobloco compreende quatro blocos de luminância e um de cada par de blocos de diferenciação de cor.

Os dados de um bloco consistem de palavras-código dos coeficientes da transformada seguidos de um marcador de fim de bloco.

Transform coefficients (TCOEFF): Os dados do coeficiente da transformada sempre estão presentes em todos os seis blocos em um macrobloco quando MTYPE indica INTRA. Em outros casos o MTYPE e o sinal CBP cujos blocos possuem coeficientes de dados transmitidos para eles. Os coeficientes quantizados da transformada são seqüencialmente transmitidos.

End os block (EOB): Indica o término do bloco.

IV. ANEXO - CÓDIGO DO HORN & SCHUNCK

HORN2.C

```
/***********************************************************
   horn.c
   Travis Burkitt, modified by John Barron, 1992
   May  1988  -- written for MASSCOMP
    -- May 8, 1989  modified for SUN 3/60
    --- Implementation of Horn and Schunck's algorithm
     for calculating an optical flow field.
***********************************************************/
#include "lib2.h"
t_matrix *pic[FIVE];
t_matrix *floatpic[FIVE];
t_matrix *inpic[PIC_T];
extern t_matrix *Ix[1],*Iy[1],*It[1];
t_matrix *full_vels1[2],*full_vels[2],*ave[2];
extern int pic_x,pic_y;
extern int pic_t;
t_matrix *temp_vels[2];
int THRESHOLD,STANDARD,WRITE_SMOOTH;
char TYPE_IMG;
//int int_size_x,int_size_y;
//float actual_x,actual_y,size_x,size_y,offset_x,offset_y;
#include "horn2_read.c"
#include "horn2_sub.c"

/*****************************************************************/
/*   Main program                                        */
/*****************************************************************/
int main(int argc,char **argv)
{
int offset,size,start,end,middle,i,j;
int startx,starty,endx,endy,step,c,tt;
int numpass,time;
float sigma,tau;
unsigned char *path1,*path2,*path3;
float alpha;
//int fd_correct,no_bytes,num,fd;
//float ave_error,st_dev;
//float min_angle,max_angle,density;


if(argc < 7 || argc > 19)
     {
        printf("Usage: %s <filename stem> <alpha> <sigma> <central file
number> <number of iterations> <input path> <output path> [-S <smooth path>
-C <full correct filename> -B <cols> <rows> -T <tau> -H or -
MH]\n\n",argv[0]);
     printf("<filename stem> - stem of input filenames\n");
     printf("<alpha> - Lagrange multiplier value\n");
     printf("<sigma> - the standard deviation used in smoothing the
files\n");
```

```
     printf("   sigma==0.0 means no smoothing: use 2 or 5 unsmoothed
images\n");
     printf("<central file number> - the image file number for which\n");
     printf("   image velocity is to be computed\n");
     printf("<number of iterations> - number of iterations in the velocity
calculation\n");
     printf("<input path> - directory where input data resides\n");
     printf("<output path> - directory where computed flow fields put\n");
     printf("-S <smooth path> - directory where smoothed data is to be
put\n");
     printf("   if not present smoothed files are not written\n");
     printf("-B <cols> <rows> - use binary file of size <cols>*<rows>
characters\n");
        printf("-P - use ppm color binary file of size <cols>*<rows>
characters\n");
     printf("-G - use pgm grey scale binaryfile of size <cols>*<rows>
characters\n");
     printf("   instead of black and white rasterfiles as image input\n");
     printf("   image size read from rasterfile header if used\n");
     printf("-T <tau> - threshold the computed velocities on spatial
intensity gradient\n");
     printf("-H Standard Horn and Schunck\n");
     printf("   perform H and S differencing on 2 input images\n");
     printf(" sigma is ignored for -H\n");
     printf("-MH Non-standard Horn and Schunck - default\n");
     printf("    perform 4-point central differences on 5 input images\n");
     exit(1);
        }

printf("\n------------------------------\n");
printf("Command line: ");
for(i=0;i<argc;i++) printf("%s ",argv[i]);
printf("\n%d arguments\n",argc-1);

sscanf(argv[2],"%f",&alpha);
sscanf(argv[3],"%f",&sigma);
sscanf(argv[4],"%d",&middle);
sscanf(argv[5],"%d",&numpass);
printf("alpha=%f\n",alpha);
printf("sigma=%f\n",sigma);
printf("Central image: %d\n",middle);
printf("Number of iterations: %d\n",numpass);
 path1=i_create_name(MAX_PATH);
 if(path1==(int)NULL) {fprintf(stderr,"can't create scene file name\n");
                                        exit(0);
                                        }
 path2=i_create_name(MAX_PATH);
 if(path2==(int)NULL) {fprintf(stderr,"can't create scene file name\n");
                                        exit(0);
                                        }
 path3=i_create_name(MAX_PATH);
 if(path3==(int)NULL) {fprintf(stderr,"can't create scene file name\n");
                                        exit(0);
                                        }
strcpy(path1,argv[6]);
strcpy(path2,".");
strcpy(path3,argv[7]);

printf("Input directory: %s\n",path1);
printf("Output directory: %s\n",path3);
```

```
TYPE_IMG = FALSE;
STANDARD = FALSE;
THRESHOLD = FALSE;
WRITE_SMOOTH = FALSE;

i=8;

while(i<argc)
{
if(strcmp("_____________________________________________________________________
_

horn2_read.c
//#include "lib2.h"
extern t_matrix *full_vels[2];
extern t_matrix *inpic[PIC_T];
extern t_matrix *pic[FIVE];
int pic_x,pic_y;
int pic_t;
extern int THRESHOLD,STANDARD,WRITE_SMOOTH;
int xLen,yLen,iMax;
char ident[3];
extern char TYPE_IMG;

float *i_create_vector(int hei)
{
  float *ptr;

  if((ptr=(float *) calloc(hei,sizeof(float)))==NULL)
      return((int)NULL);
  return(ptr);
}

void i_destroy_vector(ptr)
float *ptr;
{
 if(ptr!=NULL)
      free((char *) ptr);
 ptr=NULL;
}

char *i_create_name(int wid)
{
 char *row;

 if((row=(char *) calloc(wid,sizeof(char)))==NULL)
     return((int)NULL);
 return(row);
}

void i_destroy_name(char *field)
{
  if(field!=NULL)
     {
      free((char *) field);
      field=NULL;
     }
}
void i_destroy_matrix_field(ptr)
t_matrix *ptr;
{
```

```
  void **field;
  int i;

  if(ptr!=NULL)
     {
          field=ptr->field;
          if(field!=NULL)
          {
           for(i=0;field[i]!=NULL;i++)
               free((char *) field[i]);
           free((char *) field);
           ptr->field=NULL;
          }
     }
}


Horn2_sub.c
//#include "lib2.h"
t_matrix *Ix[1],*Iy[1],*It[1];
extern t_matrix *full_vels1[2],*full_vels[2],*ave[2];
extern t_matrix *inpic[PIC_T];
//t_matrix *full_velocity[2];
extern t_matrix *pic[FIVE];
extern t_matrix *floatpic[FIVE];
extern int pic_x,pic_y;
extern int pic_t;
/*********************************************************************/
/* Perform 3D Gaussian smoothing by separable convolution          */
/*********************************************************************/
void convolve_Gaussian(sigma,start,frame,time)
int frame,time,start;
float sigma;
{
float *mask,term,sum,temp;
float minvel,maxvel;
int size,i,j,offset,a,b,ind,ind1,c;
t_matrix *pict[2];
it_float *pic0_ptr,*pic1_ptr,*floatpic_ptr;
it_byte *pic_ptr;


if(OUTPUT_SMOOTH)
{
printf("\nStart of 3D convolution\n");
printf("Time: %d Frame: %d\n",time,frame);
}

fflush(stdout);
size = (int) 6*sigma+1;
if(size%2==0) size = size+1;
offset = size/2;
if(pic_t < size)
     {
     printf("\nFatal error: not enough images\n");
     exit(1);
     }
sum = 0.0;


if(sigma != 0.0)
```

```
{
  mask=i_create_vector(size);
  if(mask==NULL) {fprintf(stderr,"can not create vector\n");
                  exit(0);
                  }
  for(i=0;i<size;i++)
     {
     mask[i] = (1.0/(sqrt(2.0*3.141592654)*sigma))*
          exp(-(i-offset)*(i-offset)/(2.0*sigma*sigma));
     sum = sum+mask[i];
     }

  if(OUTPUT_SMOOTH)
   {
   printf("Size: %d Offset: %d\nMask values: ",size,offset);
   for(i=0;i<size;i++)
     printf("%f ",mask[i]);
   printf("\nSum of mask values: %f\n",sum);
   }
  pict[0]=i_create_matrix(pic_x,pic_y,IT_FLOAT);
           if(pict[0]==NULL)
               {
               fprintf(stderr,"Can't allocate memory char inpic matrix\n");
               exit(0);
               }
pict[1]=i_create_matrix(pic_x,pic_y,IT_FLOAT);
           if(pict[1]==NULL)
               {
               fprintf(stderr,"Can't allocate memory char inpic matrix\n");
               exit(0);
               }

  for(i=0;i<pic_x;i++)
  {
  pic0_ptr=im_float_row(pict[0],i);
  for(j=0;j<pic_y;j++)
     {
     term = 0.0;
     for(a=-offset;a<=offset;a++)
         {
             ind=a+frame-start;
             ind1=a+offset;
             if(ind1<0 || ind1>(size-1)) exit(0);
             if(ind<0 || ind>(pic_t-1)) exit(0);
          term=term +(im_byte_value(inpic[ind],j,i)*(mask[ind1]));
          }
     *pic0_ptr=term;
//         if(j==pic_y/2 && i>3*pic_x/8 && i<5*pic_x/8)
fprintf(stderr,"inpic=%d pict0=%f ",im_byte_value(inpic[1],j,i),*pic0_ptr);
        pic0_ptr++;
     }
   }
  if(OUTPUT_SMOOTH) printf("\n Convolution in t direction completed\n");
//  c=getch();
  for(i=offset;i<pic_x-offset;i++)
  {
  pic1_ptr=im_float_row(pict[1],i);
  for(j=offset;j<pic_y-offset;j++)
     {
     term = 0.0;
     for(a=-offset;a<=offset;a++)
```

```
          {
                 ind1=a+offset;
                 if(ind1<0 || ind1>(size-1)) exit(0);
          term = term + (im_float_value(pict[0],j,i+a)*(mask[ind1]));
          }
     *pic1_ptr=term;
  //      if(j==pic_y/2 && i>3*pic_x/8 && i<5*pic_x/8)
fprintf(stderr,"pict1=%f ",*pic1_ptr);
        pic1_ptr++;
     }
  }
  i_destroy_matrix(pict[0]);
  if(OUTPUT_SMOOTH) printf("\n Convolution in x direction completed\n");
//  c=getch();
  maxvel=-30000;
  minvel=30000;
  for(i=offset;i<pic_x-offset;i++)
  {
  floatpic_ptr=im_float_row(floatpic[time],i);
  for(j=offset;j<pic_y-offset;j++)
     {
     term = 0.0;
     for(b=-offset;b<=offset;b++)
          {
                 ind1=b+offset;
                 if(ind1<0 || ind1>(size-1)) {exit(0);
                                              fprintf(stderr,"erro-índice do
vetor\n");
                                              }
          term = term + (im_float_value(pict[1],j+b,i)*(mask[ind1]));
          }
     *floatpic_ptr = term;
        if(term>maxvel) maxvel=term;
        if(term<minvel) minvel=term;
  //      if(j==pic_y/2 && i>3*pic_x/8 && i<5*pic_x/8)
fprintf(stderr,"floatpic=%f ",*floatpic_ptr);
        floatpic_ptr+_____________________________________ .
```