Universidade Federal da Paraíba
Centro de Ciências e Tecnologia
Departamento de Engenharia Elétrica
Laboratório de Automação e Processamento de Sinais - LAPS

Relatório Final de Iniciação Científica

Título do Projeto:

Sistema de Processamento Digital de Imagens para Fins Didático/Científico: Estudo, Seleção e Implementação de Algoritmos de Filtragem Espacial.

Bolsista: José Josemar de Oliveira Júnior
Orientador: João Marques de Carvalho, Ph.D.

Período: Agosto de 1997 à Julho de 1998

Título do Projeto:
Sistema de Processamento Digital de Imagens para Fins Didático/Científico: Estudo, Seleção e Implementação de Algoritmos de Filtragem Espacial.

José Josemar de Oliveira Júnior (Aluno Bolsista)

João Marques de Carvalho, Ph.D. (Professor orientador)

Campina Grande, 28 de Agosto de 1998.

Período: Agosto de 1997 à Julho de 1998

Apresentação

Este relatório refere-se às atividades realizadas pelo bolsista de iniciação científica do PIBIC - CNPQ (Programa Institucional de Bolsas de Iniciação Científica), José Josemar de Oliveira Júnior, sob a orientação do professor João Marques de Carvalho, durante o período de Agosto de 1997 à Julho de 1998, no Laboratório de Automação e Processamento de Sinais - LAPS, Departamento de Engenharia Elétrica da Universidade Federal da Paraíba, Campus II, Campina Grande - PB.

Pretende-se, neste trabalho, construir uma biblioteca de algoritmos para filtragem espacial, visando criar uma infra-estrutura mínima de processamento de imagens para uso de alunos de iniciação científica e pós-graduação em seus projetos de pesquisa e também para uso como suporte didático em disciplinas de graduação e pós-graduação.

Índice

1 Introdução **1**

1.1 Motivação do Projeto 1

2 Fundamentação Teórica **3**

2.1 Processamento Digital de Imagens 3

2.1.1 Representação de imagens digitais 4

2.1.2 Campos fundamentais do processamento de imagens 4

2.1.3 Elementos de sistemas de processamento digital de imagens . . . 4

2.2 Fundamentos de imagens digitais 5

2.2.1 Um modelo simples de imagem 5

2.2.2 Amostragem e quantização 6

2.3 Realce de Imagens 8

2.3.1 Suavização espacial 8

2.3.2 Realce de bordas 9

2.4 Filtros espaciais 9

2.4.1 Teoria de filtros 9

2.4.2 Princípio de funcionamento 10

2.5 Morfologia Matemática 12

2.5.1 Considerações gerais 12

2.5.2 Operadores elementares binários 13

3 Metodologia Utilizada **18**

4 Algoritmos de filtragem **21**

4.1 Filtro da média 21

4.2 Filtro da mediana 21

4.3 Filtro da mediana adaptativo 22
4.4 Filtro da ordem adaptativo 22
4.5 Filtro da mediana cruzada 23
4.6 Filtro da média com os k vizinhos mais próximos 23
4.7 Filtro nopel 24
4.8 Filtro sigma 24

5 Operadores Morfológicos 26
5.1 Dilatação binária 26
5.2 Erosão binária 27
5.3 Abertura binária 28
5.4 Fechamento binário 29
5.5 Transformada de Hit e Miss 30
5.6 Suavização Morfológica 31
5.7 Extração de contornos 32
5.8 Gradiente Morfológico 33
5.9 Transformada TopHat 35

6 Resultados Experimentais 37
6.1 Algoritmos de filtragem 37
6.2 Operadores morfológicos 38

7 Conclusões e Trabalhos Futuros 46

A Xfilter-2D 1.0 - Manual de Utilização 49

B Biblioteca de Funções Básicas para Morfologia Matemática 50

Lista de Tabelas

6.1 Parâmetros utilizados nos testes. 39
6.2 Resultado dos algoritmos com ruído gaussiano (Parte I) 39
6.3 Resultado dos algoritmos com ruído gaussiano (Parte II) 40
6.4 Resultado dos algoritmos com ruído impulsivo (Parte I) 41
6.5 Resultado dos algoritmos com ruído impulsivo (Parte II) 42
6.6 Elementos estruturantes utilizados para cada operador. 43
6.7 Resultado dos operadores morfológicos para suavização. 44
6.8 Resultado dos operadores morfológicos para realce de bordas. 45

Lista de Figuras

2.1 Máscara de deslocamento. 11
2.2 Exemplo de dilatação 15
2.3 Exemplo de erosão 17

6.1 Exemplos de elementos estruturantes 43

Capítulo 1

Introdução

1.1 Motivação do Projeto

A atividade de pesquisa em Processamento Digital de Imagens e Visão Computacional no âmbito do DEE/LAPS tem resultado, ao longo dos últimos anos em uma considerável produção de artigos científicos publicados em congressos nacionais e internacionais, além de várias dissertações, teses e trabalhos de iniciação científica. Todos os trabalhos realizados nesta área, envolvem o processamento de imagens digitais, através de vários tipos de algoritmos. De acordo com a maneira como são utilizados, podemos classificar estes algoritmos em duas categorias:

- **Algoritmos de propósito geral** - Nesta categoria se enquadram os algoritmos básicos de filtragem, binarização, gradiente e segmentação. São algoritmos utilizados em quase todas as aplicações e tem como objetivo básico preparar a imagem para o processamento específico.
- **Algoritmos de propósito específico** - São algoritmos desenvolvidos para atender às necessidades de uma pesquisa ou problema específicos.

Deste modo, ao longo das pesquisas realizadas no LAPS, vários algoritmos foram desenvolvidos e implementados pelos vários alunos e pesquisadores. Entretanto, devido ao grande número de pessoas envolvidas e à própria dinâmica dos trabalhos de pesquisa, não foi possível realizar até o momento uma sistematização deste trabalho, como seria desejável. Esta sistematização teria por objetivo colocar os algoritmos desenvolvidos,

bem como vários outros algoritmos básicos de processamento, em uma forma padrão, devidamente implementados e documentados, que possibilitasse o seu uso em trabalhos futuros.

Desta maneira, seria garantida a preservação do *know-how* adquirido nos trabalhos já realizados, nem sempre possível através da publicação de artigos científicos. Como consequência seria evitada a repetição desnecessária de esforços, através do aproveitamento, por todos pesquisadores e alunos, do trabalho realizado pelos demais.

Capítulo 2

Fundamentação Teórica

Neste capítulo é feita uma explanação geral a respeito do processamento digital de imagens, ressaltando as técnicas de realce, dando uma ênfase particular a filtragem espacial [Ara89, Gon92, Mas84], além de noções gerais sobre morfologia matemática [Fac96, Par97].

2.1 Processamento Digital de Imagens

Os métodos de processamento digital de imagens têm 2 (duas) áreas de aplicações principais: melhoramento de imagens objetivando aumentar o nível de informação presente para posterior interpretação humana e processamento de dados de uma cena para percepção autonôma das máquinas. A primeira área surgiu no inicio dos anos 20, com a necessidade de se enviar fotos jornalísticas via cabo submarino de Nova York a Londres; e hoje, é aplicada para a resolução de uma variedade de problemas, em diversos campos, tais como: medicina, arqueologia, física, astronomia, entre outros. A segunda área, tem como focos de interesse procedimentos para extração de informações de imagens em uma forma adequada ao processamento computacional. Alguns problemas típicos desta área são: reconhecimento automático de caracteres, máquinas industriais com visão para acompanhamento da montagem de produtos, reconhecimento militar, entre outros.

2.1.1 Representação de imagens digitais

O termo imagem monocromática ou simplesmente imagem, refere-se a uma função bidimensional de intensidade luminosa $f(x, y)$, aonde x e y são coordenadas espaciais e o valor de f em qualquer ponto *(x,y)* é proporcional ao nível de cinza da imagem neste ponto. A imagem digital é uma imagem $f(x, y)$ na qual é feita uma discretização tanto das coordenadas espaciais quanto dos níveis de cinza. Uma imagem digital pode ser considerada uma matriz onde a posição de seus elementos identificam um ponto da imagem e o correspondente valor do elemento identifica o nível de cinza daquele ponto.

2.1.2 Campos fundamentais do processamento de imagens

Processamento digital de imagens envolve os seguintes campos principais: digitalização, codificação, realce, restauração, segmentação e descrição. *Digitalização* cobre os aspectos relacionados à conversão de imagens contínuas para uma forma discreta, ou seja, a amostragem das coordenadas espaciais e a quantificação dos níveis de cinza. O campo de *codificação* compreende as técnicas utilizadas para comprimir uma imagem digital, para diminuir o espaço requerido para armazenagem ou para um melhor aproveitamento de canais de transmissão. *Realce* procura acentuar certas características da imagem para posterior análise ou visualização. As técnicas de *restauração* procuram reverter o processo de degradação sofrido por uma imagem através do processo de modelagem de fenômenos que causaram tal degradação para que seja efetuada a operação inversa, restaurando dessa maneira a imagem à sua situação original. *Segmentação* é o processo de rotulação de regiões ou objetos em uma imagem para que se possam ser identificados e tratados separadamente. *Descrição* engloba as técnicas utilizadas para descrever os objetos, utilizando representações mais adequadas para o tipo de análise em questão.

2.1.3 Elementos de sistemas de processamento digital de imagens

- **Aquisição de imagem** - Dois elementos são necessários na aquisição de imagens. O primeiro é um dispositivo físico sensível a uma determinada faixa do espectro de energia eletromagnética e que produza um sinal elétrico na saída proporcional ao nível da energia sentida. O segundo é um digitalizador para converter a saída

elétrica do sensor físico em uma forma digital.

- **Armazenagem** - Existem três categorias principais para armazenamento digital de imagens: armazenagem de curto tempo para uso durante o processamento, armazenagem on-line que permita uma rápida chamada de seus dados internos e arquivos de armazenagem caracterizados pelo acesso não-frequente aos seus dados.

- **Processamento** - O processamento digital de imagens envolve procedimentos que são usualmente expressos em forma algorítmica. Com exceção dos dispositivos de entrada e saída, muitas funções do processamento de imagens podem ser implementadas por software, mas um hardware especializado em processamento de imagens frequentemente é necessário devido à necessidade de velocidade em certas aplicações.

- **Comunicação** - A comunicação entre os vários elementos do sistema de processamento é fundamental e deve permitir a transmissão de dados de imagens. Em alguns casos deve-se aplicar técnicas de compressão e descompressão de dados para facilitar o envio de informações a longas distâncias.

- **Dispositivos de saída** - Os dispositivos de saída mais usados nos sistemas de processamento de imagem modernos utilizam monitores de video, mas pode-se utilizar também sistemas com TRC (Tubos de Raios Catódicos) e com imagem impressa.

2.2 Fundamentos de imagens digitais

O propósito desta seção é introduzir alguns conceitos relacionados a imagens digitais e algumas notações utilizadas na sua descrição.

2.2.1 Um modelo simples de imagem

O termo imagem refere-se a uma função bidimensional da intensidade da luz, denotada por $f(x, y)$, onde o valor da amplitude de f no ponto de coordenadas espaciais (x, y) fornece a intensidade (brilho) da imagem naquele ponto. As imagens percebidas pelas

pessoas normalmente no seu dia-a-dia consistem da luz refletida pelos objetos. A natureza básica de $f(x, y)$ pode ser caracterizada por duas componentes: (1) a quantidade de luz incidente na cena vista e (2) a quantidade de luz refletida pelos objetos presentes na cena. Elas são denotadas por $i(x, y)$ e $r(x, y)$, e são chamadas de componente de iluminância e refratância, respectivamente. As funções anteriores se combinam como um produto para formar $f(x, y)$:

$$f(x, y) = i(x, y)r(x, y) \tag{2.1}$$

onde

$$0 < i(x, y) < \infty \tag{2.2}$$

e

$$0 < r(x, y) < 1. \tag{2.3}$$

A natureza de $i(x, y)$ é determinada pela fonte de luz, e a de $r(x, y)$ pelas características dos objetos presentes na cena. A partir desta seção chamaremos a intensidade de uma imagem monocromática f no ponto de coordenadas espaciais (x, y) como sendo o *nível de cinza (l)* da imagem naquele ponto; a faixa de variação de (l) em uma imagem é chamada de escala de cinza. Na prática esta variação corresponde ao intervalo [0, L], onde $l = 0$ é considerado preto e $l = \mathrm{L}$ é considerado branco na escala.

2.2.2 Amostragem e quantização

Para ser adequada ao processamento computacional, uma imagem representada pela função $f(x, y)$ precisa ser digitalizada tanto espacialmente como em amplitude. A digitalização das coordenadas espaciais é chamada de amostragem da imagem, e a digitalização da amplitude é chamada de quantização dos níveis de cinza. Suponha que uma imagem contínua $f(x, y)$ é aproximada por amostras igualmente espaçadas arranjadas na forma de uma matriz M x N, como mostra a Equação 2.4, onde cada elemento da matriz representa uma quantidade discreta:

$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2.4}$$

A matriz acima representa, o que normalmente é conhecido como uma imagem digital. Cada elemento da matriz é chamado de *pixel*. Os termos imagem e pixel serão utilizados nas discussões seguintes para denotar uma imagem digital e seus elementos.

Vamos definir a amostragem e a quantização através de termos matemáticos mais formais. Sendo Z o conjunto dos reais inteiros; podemos definir o processo de amostragem como sendo o particionamento do plano xy em uma grade, com as coordenadas de cada ponto da grade sendo um elemento do produto cartesiano Z x Z, que representa o conjunto de todos os pares ordenados (a,b), com a e b sendo elementos de Z. Portanto $f(x,y)$ é uma imagem digital se (x,y) são elementos de Z x Z e f é uma função que determina um valor de nível de cinza (que é um número inteiro pertencente a Z) para cada par distinto de coordenadas (x,y). Isto pode ser expresso matematicamente da seguinte forma:

$$(x,y) \in Z^2 \tag{2.5}$$

e

$$f(x,y) \in Z. \tag{2.6}$$

Esta determinação funcional representa o processo de quantização descrito anteriormente. A *resolução* (o grau de detalhes discerníveis) de uma imagem depende extremamente da quantidade de amostras e de níveis de cinza utilizados na digitalização da imagem. Se aumentarmos a quantidade desses parâmetros a matriz digitalizada se aproximará muito da imagem original, contudo isso pode requerer de uma grande quantidade de bits para representar a imagem, o que inviabiliza o armazenamento dessas informações.

É difícil definir o que seria uma *boa* imagem, pois isso varia de acordo com a aplicação requerida. Imagens adquiridas com baixa resolução espacial apresentam geralmente replicação de pixels, o que produz um efeito tipo *tabuleiro de damas*, enquanto que imagens com baixo número de níveis de cinza podem apresentar um efeito de *falso*

contorno. Geralmente imagens com grande quantidade de detalhes podem ser bem representadas utilizando poucos níveis de cinza.

2.3 Realce de Imagens

O principal objetivo das técnicas de realce é processar uma imagem de tal modo que o resultado final seja mais adequado que a imagem original para uma aplicação específica. Realce de imagens inclui expansão de contraste, suavização, realce de bordas e pseudocoloração. Estas técnicas envolvem 2 (duas) categorias principais: Métodos que operam no dominio espacial e métodos que operam no dominio da frequência. O *dominio espacial* refere-se ao próprio plano da imagem, e as técnicas nesta categoria são baseadas na manipulação direta dos pixels de uma imagem. As técnicas de processamento no *dominio da frequência* se baseam na modificação da transformada de Fourier de uma imagem. Daremos uma maior ênfase agora às técnicas de realce que constituem a classe dos filtros espaciais.

2.3.1 Suavização espacial

Suavização busca uma homogeneização dos *pixels* presentes nas diversas regiões das imagens, alterando *pixels* com níveis de cinza pouco semelhantes aos da vizinhança e que podem representar um ponto ruídoso. No domínio da freqüência, suavização é obtida através de filtros passa-baixa. No domínio espacial, através de ações realizadas dentro dos limites de uma máscara, que se desloca sobre toda a imagem efetuando operações lineares e não-lineares baseadas em informações locais. O nível de cinza do *pixel* central da janela de imagem definida pela máscara é substituído por um valor que é função do método empregado e dos níveis de cinza da vizinhança definida pelos *pixels* desta janela.

Estas técnicas, geralmente, incorporam características do ruído, o conhecimento *a priori* sobre bordas e propriedades do sistema visual humano para obter o efeito desejado. Técnicas de suavização têm como objetivos principais a remoção de ruído e a uniformização dos níveis de cinza dos *pixels* na imagem.

2.3.2 Realce de bordas

Bordas são características primitivas de uma imagem que são largamente utilizadas em sistemas de classificação e análise de imagens. Uma borda, é definida como sendo uma mudança ou descontinuidade local na luminosidade de uma imagem.

O realce de bordas é obtido no domínio da freqüência, através de filtros passa-alta, e no domínio espacial, por máscaras utilizando operadores diferenciais e direcionais. Seus objetivos principais consistem em realçar finos detalhes em uma imagem ou realçar detalhes que tenham sido borrados por erro ou por um efeito natural de um método particular de aquisição de imagens.

2.4 Filtros espaciais

Nesta seção será descrita a teoria geral dos filtros espaciais, e seu princípio de funcionamento.

2.4.1 Teoria de filtros

Um operador Ψ é dito linear caso satisfaça a seguinte equação:

$$\Psi\left(af + bg\right) = a \cdot \Psi(f) + b \cdot \Psi(g) \tag{2.7}$$

onde f e g são funções e a e b representam constantes.

Operadores que atuam sobre imagens para realizar funções de realce podem ser classificados como *pontuais* e *locais*, sendo que estas duas classes ainda podem ser divididas em operadores lineares e não-lineares.

Um operador Ψ é invariante ao deslocamento caso seja obtido o mesmo resultado aplicando Ψ a uma imagem e posteriormente deslocando o resultado ou quando Ψ é aplicado à imagem previamente deslocada. Deste modo, a translação da entrada causa uma translação equivalente na saída. Um exemplo de uma operação linear e invariante ao deslocamento é a convolução.

A convolução de duas funções, representada por $f(x) * g(x)$ é definida pela seguinte integral:

$$f\left(x\right) * g\left(x\right) = \int_{-\infty}^{\infty} f\left(\alpha\right) \cdot g\left(x - \alpha\right) d\alpha \tag{2.8}$$

onde α é uma variável auxiliar utilizada para integração.

Para o caso discreto, pode-se definir a convolução pelo seguinte somatório:

$$f[m] * g[n] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[n-k]. \tag{2.9}$$

Quando temos funções bidimensionais, a convolução no caso contínuo pode ser definida de maneira análoga á convolução unidimensional:

$$f(x,y) * g(x,y) = \int \int_{-\infty}^{\infty} f(\alpha,\beta) \cdot g(x-\alpha, y-\beta)\, d\alpha d\beta \tag{2.10}$$

onde α e β são variáveis auxiliares utilizadas para integração.

Da mesma forma pode-se definir a convolução bidimensional discreta pela seguinte equação:

$$f[m,n] * g[m,n] = \sum_{k,l=-\infty}^{\infty} f[k,l] \cdot g[m-k, n-l]. \tag{2.11}$$

O método básico para a realização de filtragens sobre uma imagem é a convolução de duas funções, uma representando a própria imagem e a outra representando o filtro a ser aplicado sobre a imagem. Desta maneira o nível de cinza resultante em um ponto é uma combinação linear dos níveis de cinza dessas funções, que pode ser expresso pela equação abaixo:

$$f[m,n] * g[m,n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k,l] \cdot g[m-k, n-l], \tag{2.12}$$

onde M e N representam a altura e a largura da imagem, respectivamente.

Esta última equação pode ser vista como uma representação matemática dos conceitos envolvidos na implementação do processo de máscaras de deslocamento, como será explicado a seguir.

2.4.2 Princípio de funcionamento

O princípio de funcionamento dos filtros que operam em domínio espacial baseia-se em relações de vizinhança entre os elementos de uma região de tamanho e formato predeterminado. Domínio espacial refere-se ao plano da própria imagem, sendo que nesta categoria trabalha-se diretamente com o valor dos *pixels* de uma imagem. No processo de filtragem são atribuídos valores aos elementos da imagem destino em função dos elementos presentes na imagem fonte.

A utilização de filtros com formatos diferentes e valores dependentes da posição na imagem, é conhecida como máscara de deslocamento ou janela móvel. Na Figura 2.1, é mostrada uma máscara com dimensões horizontais e verticais idênticas e iguais a três. O elemento central da máscara, definido pelas coordenadas (x, y), na imagem destino receberá um valor calculado em função de todos os elementos da imagem fonte pertencentes à máscara utilizada. No caso mostrado, estes elementos variam na horizontal de $(x-1)$ a $(x+1)$ e na vertical de $(y-1)$ a $(y+1)$, resultando num total de nove elementos por máscara.

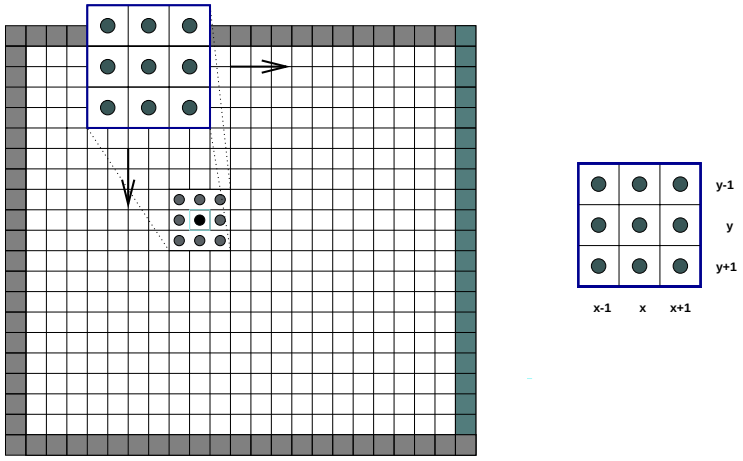


Figura 2.1: Máscara de deslocamento.

O formato da máscara de deslocamento mais comumente utilizado é o quadrado. Dependendo do algoritmo, pode-se ter submáscaras com outros formatos, como triangulares, pentagonais ou formatos irregulares. Até mesmo a máscara principal, que na maioria dos casos é a única, pode ter formatos diferentes, como a cruz ou retângulo. Alguns algoritmos admitem a escolha do formato da máscara de deslocamento e outros são projetados para um tipo específico de máscara, existindo também algoritmos onde até o tamanho da máscara é predeterminado.

Geralmente, o deslocamento da máscara é realizado da esquerda para a direita sobre cada linha da imagem. Finalizando esta linha, o centro da máscara é posicionado no início da linha seguinte, e o processo é repetido até que a máscara alcance o canto inferior direito da imagem.

Como pode ser visto, a utilização deste procedimento causa um problema nas bordas da imagem, pois a máscara deve estar com o seu centro sobre a posição (x, y) que se

deseja calcular. Com a máscara centralizada em um ponto da borda, alguns elementos da máscara não terão valores definidos por estarem fora dos limites da imagem. Para contornar este problema, pode-se restringir o posicionamento da máscara para que a mesma não seja sobreposta a pontos não pertencentes à imagem ou atribui-se aos pontos fora da imagem um valor predeterminado. No primeiro caso, após a execução do filtro, costuma-se preencher as bordas não calculadas com uma constante ou com o valor do ponto mais próximo. Este último procedimento é chamado de *repetição de bordas*. Outras abordagens podem ser seguidas, como definir os operadores de forma que tratem todos os casos especiais, assumir a imagem como sendo ciclicamente fechada ou atribuir o valor zero aos pontos fora da imagem.

2.5 Morfologia Matemática

A palavra morfologia vêm do grego (morphê=forma, logos=ciência); portanto é uma ciência que trata das formas que a matéria pode tomar. Como exemplo, a morfologia vegetal refere-se ao estudo da estrutura dos organismos vegetais. Seguindo esse exemplo, a morfologia matemática leva em consideração modelos matemáticos; estudando assim estruturas matemáticas.

A morfologia matemática surgiu em 1964 das pesquisas conjuntas de G. Matheron e J. Serra, que concentraram seus esforços no estudo da estrutura geométrica das entidades presentes numa imagem [Fac96]. Ela se aplica em várias áreas tais como: biologia, metalografia, medicina, visão robótica, controle de qualidade, reconhecimento de padrões, entre outras. Em termos de imagens, a morfologia matemática, que representa um ramo do processamento não linear, permite processar imagens com objetivos de realce, segmentação, detecção de bordas, esqueletização, afinamento, análise de formas e compressão, entre outras.

2.5.1 Considerações gerais

O princípio básico da morfologia matemática consiste em extrair informações relativas à geometria e à topologia de um conjunto desconhecido de uma imagem, utilizando uma entidade chamada elemento estruturante, que consiste num conjunto, completamente definido e conhecido (forma, tamanho), que é comparado, a partir de uma transformação, ao conjunto desconhecido cujo resultado permite avaliar este último. O

formato e o tamanho do elemento estruturante possibilitam testar e quantificar de que maneira este elemento *"está ou não contido na imagem"*.

De forma geral, existem dois tipos de morfologia matemática: a *morfologia binária* que se aplica sobre imagens binárias e a *morfologia cinzenta* que se aplica sobre imagens em níveis de cinza. Na morfologia binária, na vizinhança de cada pixel da imagem original, é procurada uma configuração de pontos pretos e brancos. Quando a configuração é encontrada, ao pixel correspondente da imagem resultante é dado o rótulo "verdadeiro"; senão, o pixel resultante recebe o rótulo "falso". Uma operação morfológica binária é portanto completamente determinada a partir da vizinhança examinada ao redor do ponto central, da configuração de pontos pretos e brancos nessa vizinhança e do algoritmo. Na morfologia cinzenta, na vizinhança de cada pixel ou numa parte da vizinhança da imagem original, é necessário conhecer o valor do pixel mais escuro MIN, e o valor do pixel mais claro MAX. O valor do pixel resultante corresponde a uma combinação particular de MIN e MAX. O tamanho e forma da vizinhança, as regiões de pesquisa de MIN e MAX e o algoritmo, determinam completamente uma operação de morfologia cinzenta.

2.5.2 Operadores elementares binários

Os pilares da morfologia são duas operações básicas, *dilatação e erosão*, a partir das quais, por composição, é possível realizar muitos outros operadores poderosos. Estas operações elementares são definidas a seguir.

Dilatação binária

Inicialmente, algumas definições básicas em relação a teoria de conjuntos são necessárias, com o intuito de definir a dilatação de uma forma genérica em termos de conjuntos.

A *translação* do conjunto A pelo ponto x é definida, em notação de conjuntos, como:

$$(A)_x = \{c | c = a + x, a \in A\}. \tag{2.13}$$

Por exemplo, se x for o ponto (1,2) então todos os pixels de A, serão deslocados uma linha para baixo e duas colunas para a direita. Esta é uma translação no mesmo sentido empregado pela computação gráfica, uma mudança na posição por uma quantidade específica.

A *reflexão* do conjunto A é definida como:

$$\hat{A} = \{c|c = -a, a \in A\}. \tag{2.14}$$

Isto corresponde a uma rotação do objeto A por 180° em relação a origem.

O *complemento* do conjunto A é o conjunto de pixels que não pertencem a A. Isto corresponde pela linguagem da teoria de conjuntos:

$$A^c = \{c|c \notin A\}. \tag{2.15}$$

A *interseção* de dois conjuntos A e B é o conjunto de pixels que pertencem a A e B simultaneamente:

$$A \cap B = \{c|\,(c \in A) \wedge (c \in B)\}. \tag{2.16}$$

A *união* de dois conjuntos A e B é o conjunto de pixels que pertencem a A, B ou a ambos:

$$A \cup B = \{c|\,(c \in A) \vee (c \in B)\}. \tag{2.17}$$

Finalmente, completando esta coleção de definições básicas, a *diferença* entre o conjunto A e o conjunto B é:

$$A - B = \{c|\,(c \in A) \wedge (c \notin B)\}. \tag{2.18}$$

que é o conjunto de pixels que pertencem exclusivamente a A. Isto é o mesmo que a interseção de A com o complemento de B ou $A \cap B^c$.

Agora é possível definir mais formalmente o significado da dilatação.

Uma *dilatação* de um conjunto A por um conjunto B é:

$$A \oplus B = \{c|c = a + b, a \in A, b \in B\} \tag{2.19}$$

onde A representa a imagem, e B o elemento estruturante.

Para explorar esta idéia, seja A um conjunto desconhecido, e B o elemento estruturante $\{(0,0)\,(0,1)\}$. Os pixels no conjunto $C = A + B$ são calculados usando a Equação 2.19, que pode ser reescrita neste caso, como:

$$A \oplus B = (A + \{(0,0)\}) \cup (A + \{(0,1)\}). \tag{2.20}$$

A Figura 2.2 ilustra esta operação, mostrando graficamente o efeito da dilatação.

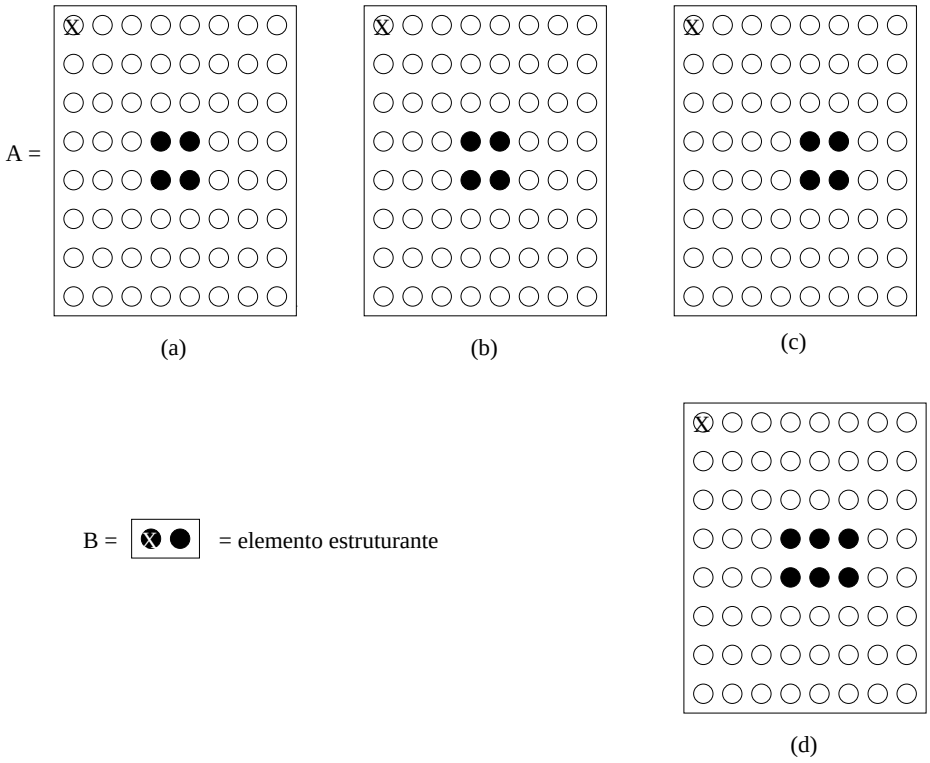


Figura 2.2: Exemplo de dilatação do conjunto A pelo conjunto B. (a) Os dois conjuntos. (b) O conjunto obtido adicionando (0,0) para todo elemento de A. (c) O conjunto obtido adicionando (0.1) para todo elemento de A. (d) A união dos dois conjuntos é o resultado da dilatação.

O pixel marcado com um "X", representa a origem de cada imagem, e pode ser branco ou preto. A localização da origem é importante. De acordo com a Figura 2.2, se a origem de B fosse o pixel à direita, neste caso o conjunto B seria {(0,-1)(0,0)}, e ao fazer a dilatação iriam ser adicionados pixels à esquerda de A, ao invés de à direita como mostrado.

Podemos generalizar o cálculo da dilatação como sendo a união de todas as translações especificadas pelo elemento estruturante. Igualmente, em vista da comutatividade, a dilatação pode ser considerada como a união de todas as translações do elemento estruturante por todos os pixels na imagem; isto é:

$$A \oplus B = \bigcup_{b \in B} (A)_b = \bigcup_{a \in A} (B)_a \,. \tag{2.21}$$

Concluindo, alguns dos efeitos obtidos aplicando a dilatação em uma imagem, são: o crescimento das regiões das formas originais e a união de conjuntos (regiões) antes separados.

Erosão binária

Se a dilatação adiciona pixels a um objeto, fazendo com que ele aumente, então a erosão tem efeito contrário, diminuindo o objeto. No caso mais simples, a erosão binária remove a camada externa de pixels de um objeto. Em geral, a erosão da imagem A pelo elemento estruturante B pode ser definida como:

$$A \ominus B = \{c|(B)_c \subseteq A\}. \tag{2.22}$$

Em outras palavras, a erosão é o conjunto de todos os pixels "c", tais que o elemento estruturante B transladado por "c" corresponda a um conjunto de pixels pretos em A.

Considere a imagem $A = \{(3,3)\,(3,4)\,(4,3)\,(4,4)\}$ e o elemento estruturante $B = \{(0,0)\,(0,1)\}$. O conjunto $A \ominus B$ é o conjunto de translações de B que o alinham sobre um conjunto de pixels pretos em A. Isto significa que nem todas as translações precisam ser consideradas, mas só aquelas que inicialmente coloquem a origem de B sobre algum membro de A.

Neste exemplo, há quatro translações deste tipo:

$$B_{(3,3)} = \{(3,3)\,(4,3)\}$$

$$B_{(3,4)} = \{(3,4)\,(4,4)\}$$

$$B_{(4,3)} = \{(4,3)\,(5,3)\}$$

$$B_{(4,4)} = \{(4,4)\,(5,4)\}$$

Em dois casos, $B_{(3,3)}$ e $B_{(3,4)}$, o conjunto resultante consiste de pixels que são todos membros de A, e estes pixels irão aparecer na erosão de A por B. Este resultado é ilustrado na Figura 2.3.

É importante notar que a erosão e a dilatação não são operações inversas; contudo, são *duais* entre si no seguinte sentido:

$$(A \ominus B)^c = A^c \oplus \hat{B} \tag{2.23}$$

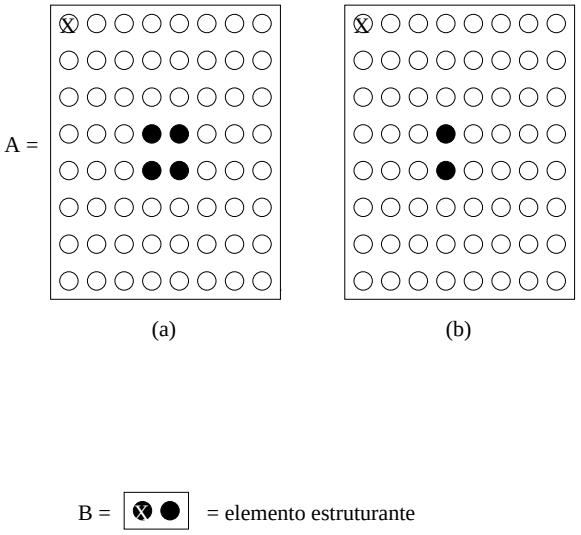


Figura 2.3: Exemplo de erosão do conjunto A pelo conjunto B. (a) Os dois conjuntos. (b) O resultado da erosão.

Isto diz que o complemento de uma erosão é o mesmo que a dilatação do complemento da imagem pelo elemento estruturante refletido.

Concluindo, alguns dos efeitos obtidos aplicando a erosão em uma imagem, são: o decrescimento das regiões definidas pelas formas originais, eliminação de conjuntos (regiões) de tamanho inferior ao tamanho do elemento estruturante e a separação de regiões adjacentes.

A seguir é descrita a metodologia utilizada no desenvolvimento deste projeto.

Capítulo 3

Metodologia Utilizada

Para atingir os objetivos desta fase inicial do projeto, que seria a construção de uma biblioteca de algoritmos para filtragem espacial de imagens, foi necessário escolher uma linguagem de programação eficiente, que facilitasse a implementação desses algoritmos. Optou-se então pela linguagem C, devido à sua forte utilização dentro das pesquisas desenvolvidas na área, além de que há vários aplicativos construídos sobre esta plataforma, que ajudaram no encaminhamento do projeto. Adicionalmente, C é totalmente compatível com os sistemas operacionais MS-DOS e UNIX, ambos bastante utilizados em meio acadêmico.

A partir disso iniciou-se um estudo completo desta linguagem de programação [Miz90]; em paralelo foram estudados os fundamentos do processamento digital de imagens, com destaque para a filtragem espacial, discutidos no capítulo anterior. Em seguida, surgiu a necessidade de se fazer uma revisão bibliográfica, para definir os algoritmos a serem inicialmente implementados. Nesta etapa procurou-se escolher um conjunto de algoritmos, que fossem suficientemente heterogêneos, de modo a permitir sua aplicação para diversos fins, tais como suavização de imagens, eliminação de pontos ruidosos, realce de bordas, entre outros. Assim foram selecionados 8 (oito) algoritmos para implementação, que serão descritos posteriormente.

Iniciada a implementação, sentiu-se a necessidade de uma ferramenta que auxiliasse na definição do formato de imagem a ser utilizado, e que posteriormente pudesse ajudar na avaliação dos testes a serem feitos, desta forma foi instalado o software *Khoros*.

Khoros é um ambiente para pesquisa em Processamento Digital de Imagens desenvolvido na Universidade do Novo México [Kho97]. O sistema *Khoros* integra múltiplos

modos de interface com o usuário, geradores de código, visualização de dados, computação distribuída e processamento de informações. O resultado é um ambiente único com ferramentas para pesquisa e desenvolvimento de sistemas computacionais. O sistema foi desenvolvido para ambientes *Unix* que utilizam *X Window System*, e portanto pode ser transportado para uma ampla faixa de estações de trabalho que utilizam sistema operacional semelhante. É composto de diversos aplicativos, que podem ser classificados como ferramentas para o desenvolvimento de sistemas e também aplicações para o usuário final. A utilização deste sistema para o desenvolvimento do trabalho se deve ao fato do mesmo ser de domínio público e de estar conseguindo uma boa aceitação, tanto no meio universitário quanto em outras instituições de pesquisa.

Desta forma, devido à facilidade de leitura e manipulação, definiu-se o formato para imagens *ASCII* como o padrão a ser usado no desenvolvimento dos programas, sendo absolutamente possível a sua conversão para outros formatos utilizando o *Khoros*.

Após a implementação e documentação dos algoritmos selecionados, foi feita uma padronização de suas estruturas, com o objetivo de facilitar a compreensão do código pelo usuário. Em seguida foi feito um conjunto de testes, todos realizados com a imagem *Lena*, apresentando dimensões 256x256, com gradações de cinza abrangendo 256 níveis. Esta imagem foi escolhida devido à sua grande utilização dentro das pesquisas desenvolvidas na área.

Como os filtros apresentam características específicas, é difícil fazer uma avaliação completa sobre as suas capacidades; como não é esse o objetivo principal deste projeto, preferiu-se então aplicar os filtros sobre um mesmo conjunto, fazendo-se comparações visuais sobre os resultados. Para se obter uma análise mais profunda deve-se consultar literatura específica [Are95a, Are95b]. Na realização dos testes corrompeu-se a imagem original com dois tipos de ruído, o ruído gaussiano com μ=0 e σ=15 e o ruído impulsivo a 10%; ambos utilizados nos artigos publicados na área, os resultados obtidos serão mostrados posteriormente.

Terminado os testes foi escrito um manual de utilização, com descrição dos algoritmos selecionados, detalhes dos filtros implementados, análise dos resultados obtidos e códigos fonte dos programas, consistindo assim num meio de consulta rápido e eficiente, que ficará a disposição dos usuários. Este manual encontra-se no Apêndice A.

Na sequência sentiu-se necessidade de pesquisar técnicas mais recentes para o realce de imagens. Com este objetivo iniciou-se o estudo dos fundamentos da morfologia

matemática [Fac96, Par97], apresentado no capítulo anterior. Esta disciplina tem se mostrado bastante útil no processamento e análise de imagens.

A partir do estudo realizado foram selecionadas as principais operações necessárias para a definição dos operadores morfológicos em geral, como: a união, a diferença, a erosão, dentre outras. Todas estas operações foram implementadas em linguagem de programação, resultando em uma biblioteca de funções, que se encontra no Apêndice B. Também foi selecionado um conjunto de 9 (nove) operadores básicos para suavização e realce de bordas, que foram implementados e documentados utilizando a biblioteca de funções anteriormente construída.

Testes foram realizados utilizando a imagem *Lena* na forma binária, visando avaliar subjetivamente o desempenho e a aplicabilidade destes operadores, sendo estes resultados mostrados no capítulo 6.

A seguir descrevemos os algoritmos de filtragem selecionados e os operadores morfológicos implementados, para em seguida mostrar os resultados experimentais obtidos.

Capítulo 4

Algoritmos de filtragem

Neste capítulo são descritos os algoritmos de filtragem tradicionais que foram implementados durante a realização deste projeto.

4.1 Filtro da média

Nome do programa: *Mean*

O filtro da média, descrito em [Pra78], é um dos filtros mais simples que podem ser definidos no domínio espacial. O valor da função $g(x, y)$, que representa a imagem filtrada, é definido pela média aritmética de todos os pontos pertencentes a janela ou máscara cujo centro é o ponto $f(x, y)$. O objetivo deste filtro é a atenuação do ruído através do cálculo da média, pois cada ponto na imagem resultante será função de todos os pontos dentro de uma janela. Apesar de conseguir atenuar o ruído, apresenta bons resultados apenas em regiões homogêneas, devido ao efeito de degradação de bordas. Este filtro pode ser implementado com janelas de formato e tamanhos diversos, embora sejam mais comumente utilizadas as janelas quadradas.

4.2 Filtro da mediana

Nome do programa: *Median*

Neste filtro, o procedimento adotado é a extração da mediana de um conjunto de valores correspondentes aos elementos da máscara de filtragem utilizada. Pode ser considerado como um caso especial do filtro da ordem, definido em [Hey82]. No filtro

da ordem, a saída do filtro é o valor que após a ordenação de todos os elementos da máscara, estiver ocupando a posição k, onde k é a ordem do filtro. No caso da mediana, o valor k é substituído por N/2, onde N é o número total de elementos presentes na janela. É um filtro que também permite a utilização de máscaras de diferentes formatos. Seu princípio de funcionamento baseia-se no fato de que elementos não pertencentes à imagem tendem a se localizar nos extremos do conjunto ordenado, devido à expectativa de que os níveis de cinza de pontos dentro de uma janela da imagem de dimensões relativamente pequenas apresentarem um certo grau de similaridade. Como os pontos ruidosos dificilmente serão retornados como o valor da mediana, este filtro tem um bom desempenho na eliminação de ruídos, embora em alguns casos possa degradar objetos com vértices muito agudos. Uma descrição mais completa pode ser encontrada em [Ahm87].

4.3 Filtro da mediana adaptativo

Nome do programa: *Adapmed*

O filtro da mediana adaptativo estudado em [Lin88] utiliza os mesmos conceitos apresentados no filtro da mediana tradicional. A diferença está na escolha da forma da máscara de deslocamento, que é feita de maneira adaptativa para cada ponto da imagem. A adaptação realizada, no caso deste filtro, é em relação ao comprimento dos eixos horizontal e vertical e das duas diagonais que definem a máscara, sendo que as diagonais podem ter o comprimento mínimo de um e máximo de cinco pontos. O processo de adaptação é realizado para cada um dos quatro eixos utilizados para compor a máscara e consiste na determinação de variações no valor do nível de cinza dentro de uma janela quadrada de dimensão horizontal e vertical iguais a cinco. Um procedimento de avaliação é aplicado envolvendo as variações encontradas e dois limiares fornecidos para o algoritmo, sendo então determinado o comprimento do eixo correspondente.

4.4 Filtro da ordem adaptativo

Nome do programa: *Adapord*

O filtro da ordem adaptativo [Lee87], foi projetado para realçar os contornos ao mesmo tempo que retira componentes de ruído, principalmente do tipo impulsivo. Sua

implementação tem como base a saída de dois outros sub-filtros, média e mediana. A idéia básica é aumentar a inclinação das bordas ao mesmo tempo que se tenta eliminar algum ruído existente. O filtro foi projetado para aplicação em janelas unidimensionais, podendo ser aplicado sequencialmente às linhas e posteriormente às colunas da imagem. A saída Y do filtro com parâmetro J sobre uma janela de comprimento 2N+1 é definida como sendo o elemento de ordem N+1-J, caso a média observada naquela janela seja maior ou igual à mediana da mesma, ou o elemento de ordem N+1+J, caso contrário. Como pode ser visto, o parâmetro J, que é um inteiro variando na faixa de um a N, definirá o deslocamento que será aplicado à posição mediana da janela, sendo a direção deste deslocamento definida pela relação entre a média e o valor da mediana na janela.

4.5 Filtro da mediana cruzada

Nome do programa: *Crossmed*

Este filtro foi implementado a partir do filtro da mediana tradicional para uma janela 3x3 [Are95a], consiste da aplicação da mediana para os elementos do par de diagonais, resultando num valor e no cálculo da mediana para os elementos dos eixos vertical e horizontal, resultando num segundo valor. A estes dois valores calculados é incluído o valor do elemento central da janela, sendo que a saída do filtro é a média ou a mediana destes últimos três elementos, dependendo da escolha do usuário. Estes conceitos podem ser estendidos a janelas de tamanho 5x5 ou 7x7, sendo que para tanto deverão ser incluídos mais pares de diagonais intermediárias, para que todos os elementos da janela possam participar do cálculo.

4.6 Filtro da média com os k vizinhos mais próximos

Nome do programa: *Knn*

O filtro da média com os K vizinhos mais próximos [Dav78] pode ser definido como um aprimoramento do tradicional filtro da média e tem seu funcionamento baseado no processo de seleção dos elementos que irão fazer parte da média. Como o que se deseja de um filtro é a sua capacidade de remoção de ruídos com preservação dos detalhes, o mesmo deveria operar de modo diferenciado em regiões homogêneas e em regiões com transições. E em regiões de bordas seria mais interessante que apenas os pixels

pertencentes ao mesmo lado da borda do pixel central fizesem parte da média. Este também seria o caso para as linhas, onde seriam escolhidos apenas os elementos que estivessem sobre determinada linha, a qual deveria conter o pixel central. No filtro da média com os K vizinhos mais próximos, utiliza-se a diferença absoluta do nível de cinza entre cada elemento da máscara de filtragem e o ponto central para selecionar os K elementos participantes da média. Desta maneira, com a escolha apropriada do tamanho da janela e do valor de K, a média será calculada geralmente com elementos pertencentes à mesma região, diminuindo a degradação da imagem que ocorreria com a aplicação de uma média com todos os elementos.

4.7 Filtro nopel

Nome do programa: *Nopel*

A idéia central deste algoritmo, descrito em [Imm91], é substituir o pixel da imagem sob o centro de uma janela 3x3 pelo seu vizinho mais próximo, caso o mesmo pixel central seja identificado como um ponto ruidoso. O conceito de proximidade neste caso refere-se à escala de níveis de cinza. O algoritmo deve ser aplicado iterativamente para conseguir o efeito de suavização. Como a determinação de um ponto ruidoso é realizada verificando se o mesmo é um máximo ou mínimo local, um ponto só pode ser alterado caso existam mais de dois níveis de cinza diferentes na sua janela. Esta restrição existe para evitar a perda de informação na imagem.

4.8 Filtro sigma

Nome do programa: *Sigma*

O filtro sigma, desenvolvido em [Lee83a], parte do princípio de que o ruído presente numa imagem segue distribuição normal. O valor do pixel sob o elemento central da janela é trocado pela média dos elementos pertencentes à janela que estiverem dentro de uma faixa de -2σ a 2σ em relação ao valor do elemento central, onde σ é um dado de entrada fornecido pelo usuário e representa o desvio padrão estimado para o ruído presente na imagem. Como podem ocorrer casos em que um ponto ruidoso esteja sozinho na faixa definida, ou seja, seu nível de cinza é muito diferente dos valores definidos para seus vizinhos, este ponto não seria suavizado, pois ficaria sozinho. Para

contornar tais casos, utiliza-se um parâmetro K, sendo que se não existirem pelo menos K elementos dentro da faixa de 2σ, a saída do filtro para este ponto vai ser definida como a média dos seus vizinhos imediatos. Para suavização do ruído multiplicativo [Lee83b], a faixa de intensidade é modificada para (-2σ x pixel central) a (+2σ x pixel central). Pode-se utilizar o filtro sigma polarizado, neste caso, os elementos que anteriormente estavam incluídos na faixa de 2σ, agora são processados em duas faixas distintas, tendo o elemento central como separador de faixas. A média dos elementos destas duas faixas (uma com intensidades maiores que o elemento central e outra com intensidades menores) é comparada com o valor do elemento central, sendo que a saída do filtro é definida como sendo a média da faixa que apresentar a menor diferença absoluta em relação ao elemento central. Deste modo, o filtro também passa a incorporar características de realce de bordas, além da remoção de ruídos.

Capítulo 5

Operadores Morfológicos

Neste capítulo são descritos os algoritmos dos operadores morfológicos implementados, bem como seus códigos fonte, que foram organizados em função de uma biblioteca básica que se encontra no Apêndice B deste relatório.

5.1 Dilatação binária

Nome do programa: *Bindil*

Descrição: Este operador aplica uma dilatação sobre uma imagem *im* através de um elemento estruturante p. Isto é feito movendo-se a origem de p para cada posição onde estão localizados pixels pretos na imagem *im*, copiando então os pixels pretos de p para as posições correspondentes na imagem de saída. Este é basicamente o processo especificado pela Equação 2.21, e que tem bastante semelhança com o processo de convolução descrito na seção 2.4.

Código fonte:

```
#include "bibm.h"

void main (int argc, char *argv[])
{
IMAGE orig, aux, se;
if (argc < 4)
{
  printf ("BinDil <Arq_entrada> <Elemento estruturante> <Arq_saida>\n");
```

```
  exit(1);
}
/* Le o elemento estruturante */
se = Input_ASC (argv[2]);
/* Le a imagem de entrada */
orig = Input_ASC (argv[1]);
/* Executa a dilatacao */
aux = Dilate(orig, se);
/* Escreve o resultado no arquivo de saida */
Output_ASC (aux, argv[3]);
}
```

5.2 Erosão binária

Nome do programa: *Binero*

Descrição: Este operador aplica uma erosão sobre uma imagem *im* através de um elemento estruturante *p*. Isto é feito movendo-se a origem de *p* para cada posição onde estão localizados pixels pretos na imagem *im*. Se todos os pixels pretos de *p* corresponderem a pixels pretos em *im*, será colocado pixel preto na imagem de saída nesta posição, caso contrário será colocado pixel branco, como explicado na seção 2.5.2.

Código fonte:

```
#include "bibm.h"

void main (int argc, char *argv[])
{
IMAGE orig, aux, se;
if (argc < 4)
{
  printf ("BinEro <Arq_entrada> <Elemento Estruturante> <Arq_saida>\n");
  exit(1);
}
/* Le o elemento estruturante */
se = Input_ASC (argv[2]);
```

```
/* Le a imagem de entrada */
orig = Input_ASC (argv[1]);
/* Executa a erosao */
aux = Erode(orig, se);
/* Escreve o resultado no arquivo de saida */
Output_ASC (aux, argv[3]);
}
```

5.3 Abertura binária

Nome do programa: *Abertura*

Descrição: A aplicação de uma erosão seguida imediatamente por uma dilatação usando o mesmo elemento estruturante é chamada de uma operação de *abertura*. Este nome descreve a observação de que esta operação tende a "abrir" pequenas lacunas ou espaços entre objetos próximos em uma imagem. Um outro uso bastante comum da abertura é na remoção de ruídos presentes em imagens binárias; isto decorre do fato que a etapa de erosão tende a remover pixels isolados bem como o contorno dos objetos, enquanto que a etapa de dilatação restaura a maior parte dos pixels pertencentes ao contorno dos objetos sem restaurar o ruído. Este operador pode ser matematicamente definido pela seguinte expressão:

$$A \text{ abe } B = \{(A \ominus B) \oplus B\}$$

Concluindo, o efeito da abertura é nivelar os contornos dos objetos pelo seu interior, o conjunto aberto é mais regular e menos rico em detalhes que o conjunto inicial.

Código fonte:

```
#include "max.h"


void main (int argc, char *argv[])
{
IMAGE orig, aux, aux1, se;
int k;
if (argc < 4)
```

```
{
  printf ("Abertura <Arq_entrada> <Elemento Estruturante> <Arq_saida>\n");
  exit(1);
}
/* Le o elemento estruturante */
se = Input_ASC (argv[2]);
/* Le a imagem de entrada */
orig = Input_ASC (argv[1]);
/* Executa a abertura binaria */
aux = Erode(orig, se);
aux1 = Dilate(aux, se);
/* Escreve o resultado no arquivo especificado */
Output_ASC (aux1, argv[3]);
}
```

5.4 Fechamento binário

Nome do programa: *Fechamento*

Descrição: Uma operação de *fechamento* é similar à abertura exceto pelo fato de se fazer primeiro a dilatação, seguida por uma erosão usando o mesmo elemento estruturante. Se uma abertura cria pequenas lacunas na imagem, o fechamento faz com que elas diminuam de tamanho, ou seja "fecha" estas lacunas. O fechamento remove grande parte do ruído "branco", deixando a imagem mais limpa. Esta operação também pode ser empregada para suavizar os contornos dos objetos presentes na imagem. Este operador pode ser matematicamente definido pela seguinte expressão:

$$A \text{ fec } B = \{(A \oplus B) \ominus B\}$$

Concluindo, o efeito do fechamento é suavizar as fronteiras pelo exterior, o conjunto fechado é mais regular e menos rico em detalhes que o conjunto inicial.

Código fonte:

```
#include "bibm.h"
```

```
void main (int argc, char *argv[])
{
IMAGE orig, aux, aux1, se;
if (argc < 4)
{
  printf ("Fechamento <Arq_entrada> <Elemento Estruturante> <Arq_saida>\n");
  exit(1);
}
/* Le o elemento estruturante */
se = Input_ASC (argv[2]);
/* Le a imagem de entrada */
orig = Input_ASC (argv[1]);
/* Executa o fechamento binario */
aux = Dilate(orig, se);
aux1 = Erode(aux, se);
/* Escreve o resultado no arquivo especificado */
Output_ASC (aux1, argv[3]);
}
```

5.5 Transformada de Hit e Miss

Nome do programa: *Hitmiss*

Descrição: Este operador morfológico é projetado para localizar simples formas presentes em uma imagem, sua operação é baseada na determinação das imagens *hit* e *miss*, definidas a seguir. O *hit* consiste no emparelhamento dos pixels internos do elemento estruturante S junto á imagem A, que é feito através da erosão $A \ominus S$. Um *miss* é obtido aplicando-se um *hit* no complemento da imagem, que é determinado por $A^c \ominus T$; onde A^c forma o conjunto de pixels externos de A, e T é definido como sendo o elemento estruturante S^c. De forma que a transformada de *hit* e *miss*, é obtida através da expressão:

$$A \otimes (S, T) = \{(A \ominus S) \cup (A^c \ominus T)\}$$

Código fonte:

```
#include "bibm.h"

void main (int argc, char *argv[])
{
IMAGE orig, orig1, compl, aux, aux1, aux2, se, se1;
if (argc < 5)
{
  printf ("HitMiss <Arq_entrada> <Elemento Estruturante1> <Elemento Estruturante2> <Ar
  exit(1);
}
/* Le os elementos estruturantes */
se = Input_ASC (argv[2]);
se1 = Input_ASC (argv[3]);
/* Le a imagem de entrada */
orig = Input_ASC (argv[1]);
/* Executa a transformada de hitmiss */
aux = Erode(orig, se);
compl = Complement(orig);
  aux1 = Erode(compl,se1);
aux2 = Intersection(aux,aux1);
/* Escreve o resultado no arquivo de saida */
Output_ASC(aux2, argv[4]);
}
```

5.6 Suavização Morfológica

Nome do programa: *Suaviza*

Descrição: A suavização utilizando operadores morfológicos pode ser obtida aplicando-se uma abertura seguida por um fechamento, como mostra a expressão a seguir:

$$A \text{ svz } B = \{[(((A \ominus B) \oplus B) \oplus B) \ominus B]\}$$

Este operador atua eliminando o ruído e ao mesmo tempo causando um borramento na imagem. A escolha do elemento estruturante a ser usado depende do tipo de ruído

a ser removido da imagem.

Código fonte:

```
#include "bibm.h"

void main (int argc, char *argv[])
{
IMAGE orig, orig1, interm, aux, aux1, aux2, aux3, se;
if (argc < 4)
{
  printf ("Suaviza <Arq_entrada1> <Arq_entrada2> <Arq_saida>\n");
  exit(1);
}
/* Le as imagens de entrada */
orig = Input_ASC (argv[1]);
orig1 = Input_ASC (argv[2]);
/* Executa a suavizacao entre as imagens */
aux = Erode(orig,orig1);
aux1 = Dilate(aux,orig1);
aux2 = Dilate(aux1,orig1);
aux3 = Erode(aux2,orig1);
/* Escreve o resultado no arquivo de saida */
Output_ASC (aux3, argv[3]);
}
```

5.7 Extração de contornos

Nome do programa: *Extcont*

Descrição: Os pixels que formam o contorno de um objeto são aqueles que tem pelo menos um vizinho que pertença ao fundo da imagem. Pode-se extrair o contorno de uma imagem usando uma erosão, seguida pela diferença entre a imagem original e a erodida. Este resultado irá fornecer os pixels que foram eliminados durante o processo de erosão; ou seja, o contorno do objeto. Pode-se escrever esta operação formalmente como:

$$A \text{ exc } B = \{A - (A \ominus B)\}$$

Código fonte:

```
#include "max.h"

void main (int argc, char *argv[])
{
IMAGE orig, aux, aux1, se;
if (argc < 4)
{
  printf ("ExtCont <Arq_entrada> <Elemento Estruturante> <Arq_saida>\n");
  exit(1);
}
/* Le o elemento estruturante */
se = Input_ASC (argv[2]);
/* Le a imagem de entrada */
orig = Input_ASC (argv[1]);
/* Executa a extracao do contorno */
        aux = Erode(orig, se);
aux1 = Difference(orig, aux);
/* Escreve o resultado no arquivo especificado */
Output_ASC (aux1, argv[3]);
}
```

5.8 Gradiente Morfológico

Nome do programa: *Gradiente*

Descrição: A informação do gradiente é muito usada no processamento de imagens para detectar bordas. De forma geral, a informação gradiente é uma versão digital da informação diferencial. Supondo o sinal f diferenciável no seu domínio, podemos mostrar que a derivada de f segundo a coordenada h na direção α na borda B é:

$$\frac{\delta f}{\delta h}(x) = \left(\frac{\delta f}{\delta h}\right)(x) + \sigma \cos\Theta - \alpha\delta_x(B)$$

onde a função $\left(\frac{\delta f}{\delta h}\right)$ representa a derivada do termo contínuo, σ representa a discontinuidade no ponto $x \in B$, o ângulo θ representa a reta perpendicular a B no ponto x orientada no sentido positivo da discontinuidade, e $\delta_x(B)$ é o pulso de Dirac no ponto x.

A definição do gradiente de f no ponto x é o vetor $\left(\frac{\delta f}{\delta x}\right)$, $\left(\frac{\delta f}{\delta y}\right)$ associado com a função diferencial df:

$$df = \frac{\delta f}{\delta x}dx + \frac{\delta f}{\delta y}dy$$

Uma aproximação desta definição utilizando operadores morfológicos pode ser obtida usando a seguinte expressão:

$$A \text{ gra } B = \{(A \oplus B) - (A \ominus B)\}$$

Esta expressão acima é a definição do gradiente morfológico, que detecta bordas de uma maneira bem menos dependente da direção que o operador gradiente normal.

Código fonte:

```
#include "max.h"

void main (int argc, char *argv[])
{
IMAGE orig, se, aux, aux1, aux2, aux3;
if (argc < 4)
{
  printf ("Gradiente <Arq_entrada1> <Elemento Estruturante> <Arq_saida>\n");
  exit(1);
}
/* Le o elemento estruturante */
se = Input_ASC (argv[2]);
/* Le as imagens de entrada */
orig = Input_ASC (argv[1]);
/* Execucao do operador gradiente */
aux = Erode(orig,se);
aux1 = Difference(orig,aux);
```

```
aux2 = Dilate(orig,se);
aux3 = Difference(aux2,aux);
/* Escreve o resultado no arquivo especificado */
Output_ASC (aux3, argv[3]);
}
```

5.9 Transformada TopHat

Nome do programa: *Tophat*

Descrição: Quando se tem uma imagem ruidosa e/ou heterogênea é difícil para os filtros clássicos de detecção de bordas (gradiente, limiarização) recuperar a informação relevante e ao mesmo tempo eliminar a heterogeneidade e o ruído. As técnicas de detecção de bordas colocarão em evidência além da informação relevante, bordas inúteis. Para contornar este problema se utiliza técnicas de detecção de picos; um modo de se fazer isto através de operadores morfológicos é a transformada de Tophat, que consiste em usar a combinação entre a imagem original e a imagem correspondente aberta, como mostra a equação abaixo:

$$A \text{ toh } B = \{A - ((A \ominus B) \ominus B)\}$$

Com o uso de um elemento estruturante adequado, o processo de abertura permite a eliminação dos picos. Fazer a diferença, entre os sinais original e resultado da abertura, permite tirar o ruído e eliminar a falta de homogeneidade, ou seja ressaltar a informação dos picos.

Código fonte:

```
#include "max.h"

void main (int argc, char *argv[])
{
IMAGE orig, aux, aux1, aux2, se;
if (argc < 4)
{
  printf ("TopHat <Arq_entrada1> <Elemento Estruturante> <Arq_saida>\n");
```

```
  exit(1);
}
/* Le o elemnto estruturante */
se = Input_ASC (argv[2]);
/* Le a imagem de entrada */
orig = Input_ASC (argv[1]);
/* Executa a transformada tophat */
aux = Erode(orig,se);
aux1 = Dilate(aux,se);
aux2 = Difference(orig,aux1);
/* Escreve o resultado no arquivo de saida */
Output_ASC (aux2, argv[3]);
}
```

No próximo capítulo é mostrado os resultados obtidos pelos algoritmos de filtragem e pelos operadores morfológicos implementados.

Capítulo 6

Resultados Experimentais

Neste capítulo são apresentados os resultados experimentais obtidos durante a realização deste projeto, inicialmente apresenta-se os resultados dos algoritmos de filtragem, e em seguida os resultados dos operadores morfológicos.

6.1 Algoritmos de filtragem

Os resultados a seguir foram obtidos utilizando-se os parâmetros mostrados na Tabela 6.1. Procurou-se usar os parâmetros indicados pelo autor de cada algoritmo, quando estes dados eram disponíveis, mas mesmo assim uma mudança destes valores em alguns casos pode melhorar o desempenho do filtro utilizado. Vale ressaltar que os resultados foram obtidos com apenas uma iteração para cada filtro, podendo haver modificações nos mesmos com um aumento no número de iterações, o que não foi realizado devido ao grande número de possibilidades envolvidas.

Na presença do ruído gaussiano, Tabelas 6.2 e 6.3, os filtros que apresentaram melhor desempenho na remoção de ruídos foram: *Knn*, *Mean* (com alto grau de borramento) e *Median*; enquanto que na presença do ruído impulsivo, Tabelas 6.4 e 6.5, os melhores foram *Median* e *Nopel* (removendo completamente o ruído da imagem).

Em ambos os casos o filtro *Adapord* não correspondeu ao esperado, embora sua principal característica seja o realce de contornos, o que não pode ser avaliado por estas imagens. Deve-se ressaltar o bom desempenho do filtro da mediana (*Median*), que em ambos os casos removeu a maior parte do ruído, mesmo tendo um algoritmo de implementação bastante simples.

6.2 Operadores morfológicos

Os resultados a seguir foram obtidos utilizando-se os elementos estruturantes mostrados na Figura 6.1, de acordo com a Tabela 6.6. A escolha dos elementos seguiu a indicação dos autores dos livros pesquisados [Fac96, Par97], estes elementos podem ser modificados de acordo com a aplicação a que se destina o operador. Os resultados são divididos em dois conjuntos: operadores morfológicos para suavização, mostrados na Tabela 6.7 e operadores morfológicos para realce de bordas, mostrados na Tabela 6.8.

Nos operadores de suavização, nota-se os efeitos da dilatação, onde ocorre um crescimento das regiões presentes na imagem, e da erosão com efeito contrário. O operador de abertura realça os contornos da imagem pelo lado interno, enquanto o fechamento realça pelo lado externo; já o operador de suavização ocasiona um alto borramento na imagem original.

Nos operadores para realce de bordas pode-se comparar a diferença entre o resultado do operador de extração de contornos e o operador gradiente, este último consegue realçar com mais fidelidade as bordas existentes na imagem original. O operador de Tophat por sua vez, detecta as variações mais bruscas (picos) presentes na imagem. O operador de Hitmiss merece uma atenção especial, ele realçou as bordas mais á direita da imagem, isto deve-se aos elementos estruturantes utilizados, mudando-se estes elementos corretamente, se pode realçar bordas em outras direções.

Estes resultados permitem avaliar a forma pela qual os operadores implementados modificam a imagem. No próximo capítulo são feitas as conclusões finais à respeito deste projeto.

Algoritmo	**Janela**	**Outros**
Adapmed	5x5	T1=10, T2=20
Adapord	5x5	J=1
Cross	3x3	mediana
Knn	3x3	K=7
Mean	5x5	-
Median	3x3	-
Nopel	3x3	-
Sigma	3x3	σ=20, Imp., N-pol., K=3

Tabela 6.1: Parâmetros utilizados nos testes.

Original	Ruido gaussiano	Adapmed
		
Adapord	**Cross**	**Knn**
		

Tabela 6.2: Resultado dos algoritmos para a imagem corrompida com ruído gaussiano (Parte I).

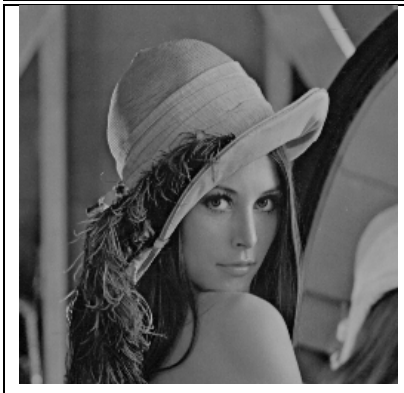
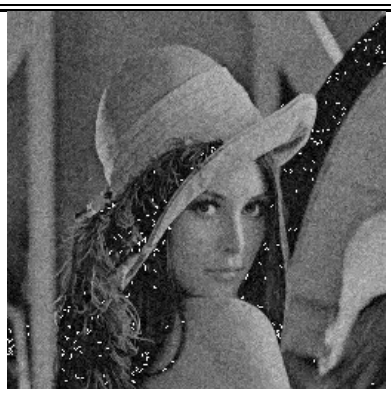
Original	Ruido gaussiano	Mean
		
Median	Nopel	Sigma
		

Tabela 6.3: Resultado dos algoritmos para a imagem corrompida com ruído gaussiano (Parte II).

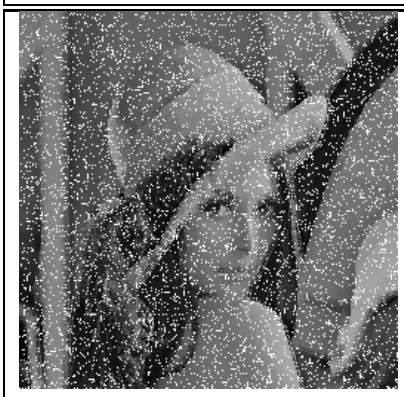
Original	Ruido impulsivo	Adapmed
		
Adapord	Cross	Knn
		

Tabela 6.4: Resultado dos algoritmos para a imagem corrompida com ruído impulsivo (Parte I).

Original	Ruido impulsivo	Mean
		
Median	Nopel	Sigma
		

Tabela 6.5: Resultado dos algoritmos para a imagem corrompida com ruído impulsivo (Parte II).

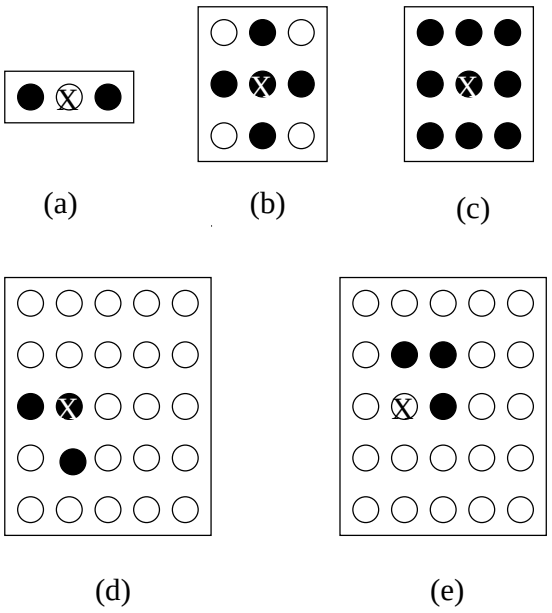


Figura 6.1: Exemplos de elementos estruturantes. (a) Horizontal. (b) Cruz. (c) Simples. (d) Hit. (e) Miss.

Operador	**Elemento Estruturante**
Bindil	Simples
Binero	Simples
Abertura	Simples
Fechamento	Simples
Hitmiss	Hit e Miss
Suaviza	Horizontal
Extcont	Simples
Gradiente	Horizontal
Tophat	Horizontal

Tabela 6.6: Elementos estruturantes utilizados para cada operador.

Original	Bindil	Binero
		
Abertura	Fechamento	Suaviza
		

Tabela 6.7: Resultado dos operadores morfológicos para suavização.

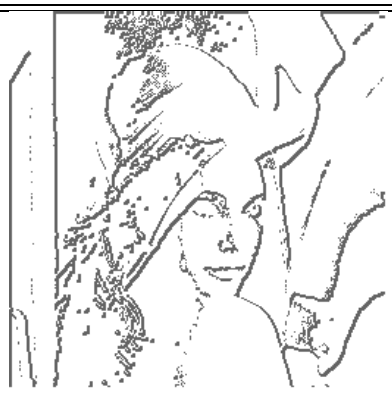
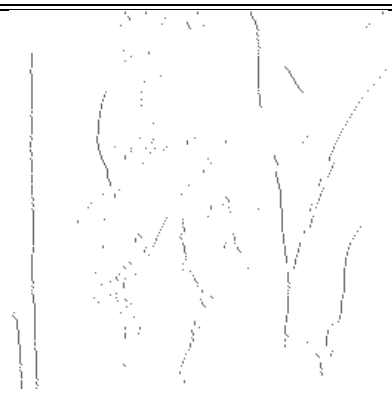
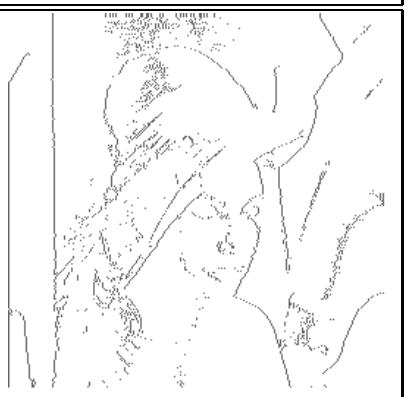
Original	Extcont	Gradiente
		
Original	Hitmiss	Tophat
		

Tabela 6.8: Resultado dos operadores morfológicos para realce de bordas.

Capítulo 7

Conclusões e Trabalhos Futuros

A partir do que foi mostrado no capítulo anterior, em relação aos algoritmos de filtragem, nota-se a diversidade de maneiras através das quais podemos fazer o pré-processamento de imagens realçando detalhes através da filtragem espacial. A utilização de determinado algoritmo depende muito da caracteristica que se deseja realçar e da imagem a ser utilizada, portanto fica difícil atender a todas as necessidades, mas se espera que a biblioteca contruída seja de bastante utilidade.

Em relação aos operadores morfológicos, percebe-se a potencialidade de aplicações possíveis utilizando os conceitos de morfologia matemática. Nota-se que o pilar central desta técnica tem por base o elemento estruturante, de forma que dependendo do objetivo do processamento, é necessário um estudo mais aprofundado para se fazer a escolha correta do elemento.

Dando continuidade ao projeto, sugere-se compatibilizar a biblioteca de algoritmos de filtragem com o *Khoros*, de forma que se contribua para um melhoramento do software. Quanto aos operadores morfológicos, será dado continuidade ao seu estudo, visando aplicá-los no pré-processamento de imagens para análise de documentos.

Desta forma, conclue-se este relatório, esperando-se que as atividades desenvolvidas possam ajudar outros alunos e pesquisadores no andamento de suas disciplinas e de seus projetos de pesquisa, respectivamente.

Bibliografia

[Ahm87] Ahmad, M. O. e Sundararajan, D. A fast algorithm for two-dimensional median filtering. *IEEE Trans. on Circuits and Systems*, 34:1364–1373, 1987.

[Ara89] Araújo, A. A. de. *Técnicas de Realce para Imagem.* IX Congresso da SBC - VIII Jornada de Atualização em Informática, Uberlândia - MG, Julho 1989.

[Are95a] Areco, E. R. Comparação de algoritmos de suavização espacial utilizando o ambiente khoros. Master's thesis, Universidade Federal de Minas Gerais - Instituto de Ciências Exatas - Departamento de Ciências da Computação, 1995.

[Are95b] Areco, E. R. e Araújo, A. A. de e Campos, M. F. M. Comparação de algoritmos de suavização espacial. In *Anais do 2o. SBAI*, Curitiba - PR, Setembro 1995. Simpósio Brasileiro de Automação Inteligente.

[Car97] Carvalho, J. M. de. Sistema de processamento digital de imagens para fins didático/científico: Estudo, seleção e implementação de algoritmos. Projeto de Pesquisa, 1997.

[Dav78] Davis, L. S. e Rosenfeld, A. Noise cleaning by iterated averaging. *IEEE Trans. on Systems, Man and Cybernetics*, 8(9):705–710, 1978.

[Dep88] Department of Computing Services. *Introduction to LaTeX.* University of Waterloo, 1988.

[Fac96] Facon, J. *Morfologia Matemática: Teoria e Exemplos.* PUC-PR, 1996.

[Gon92] Gonzalez, R. C. e Woods, R. E. *Digital Image Processing.* Addison-Wesley, 1992.

[Goo97] Goosens, M. e Mittelbach, F. e Samarin, A. *The LaTeX Companion*. Addison-Wesley, 1997.

[Hey82] Heygster, G. Rank filters in digital image processing. *Computer Graphics and Image Processing*, 19:148–164, 1982.

[Imm91] Imme, M. A noise peak elimination filter. *CVGIP: Graphic Models and Image Processing*, 53(2):204–211, 1991.

[Kho97] Khoral Research Inc. *Advanced Khoros Manuals*, 1997.

[Lee83a] Lee, J. S. Digital image smoothing and the sigma filter. *Computer Graphics and Image Processing*, 24:255–269, 1983.

[Lee83b] Lee, J. S. A simple speckle smoothing algorithm for synthetic aperture radar images. *IEEE Trans. on Systems, Man and Cybernetics*, 13:85–89, 1983.

[Lee87] Lee, Y. H. e Fam, A. T. An edge gradient enhacing adaptative order statistic filter. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 35(5):680–695, 1987.

[Lin88] Lin, H. e Wilson Jr., A. N. Median filters with adaptative lenght. *IEEE Trans. on Circuits and Systems*, 35(6):675–689, 1988.

[Mas84] Mascarenhas, N. D. A. e Velasco, F. R. D. *Processamento Digital de Imagens*. IV Escola de Computação, São Paulo - SP, Julho 1984.

[Miz90] Mizrahi, V. V. *Treinamento em Linguagem C - Módulos 1 e 2*. Makron Books, 1990.

[Par97] Parker, J. R. *Algorithms For Image Processing and Computer Vision*. Jonh Wiley & Sons, 1997.

[Pra78] Pratt, W. K. *Digital Image Processing*. Jonh Wiley & Sons, 1978.

Apêndice A

Xfilter-2D 1.0 - Manual de Utilização

Apêndice B

Biblioteca de Funções Básicas para Morfologia Matemática