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

Relatório Final de Iniciação Científica
Sistemas de Processamento Digital de Imagens para Fins Didático/Científico: Estudo, Seleção e Implementação de Algoritmos de Segmentação

Orientador: João Marques de Carvalho

Bolsista: Sérgio Ferreira de Brito

Agosto — 1998

Sumário

Lista de Figuras

Capítulo 1

Introdução

1.1 Apresentação

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

Pretende-se, neste trabalho, construir uma biblioteca de algoritmos para Segmentação, que permita uma posterior sistematizção 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.

1.2 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 cientifica.[7][6]

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, que possibilitasse o seu uso em trabalhos futuros.

Desta maneira, seria garantida a preservação do *Know-how* prático 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.

1.3 Organização do Relatório

No Capítulo 2 daremos uma breve explanação dos conceitos básicos, estudados nesta bolsa, para termos uma boa familiarização dos termos e procedimentos relatados neste. No Capítulo 3 abordaremos todo o assunto estudado para que pudéssemos realizar a seleção e implementação dos algoritmos de segmentação. No Capítulo 4 descreveremos todos os algoritmos implementados e os resultados por estes obtidos. No Cap'ıtulo 5 daremos algumas conclusões do trabalho realizado nesta bolsa de iniciação científica. Em seguida, teremos um apêndice com o manual de utilização de todos os algoritmos implementados nesta a fim de facilitar o entedimento e posterior utilização por alunos de iniciação científica e pós graduação.

Capítulo 2

Conceitos Básicos

Neste capítulo será feita uma explanação geral a respeito do processamento digital de imagens (PDI), dando uma ênfase particular a filtragem espacial , que é uma operação largamente utilizada em vários algoritmos de PDI .[1]

2.1 Fundamentos de imagens digitais

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

2.1.1 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) diz 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 chamadas de componente de iluminação e refratância, e são denotadas por $i(x, y)$ e $r(x, y)$ 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.

2.1.2 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 $f(x, y)$ seja aproximada por amostras igualmente espaçadas arranjadas na forma de uma matriz N x M, como mostra a equação 2.4. Assim , uma imagem digital pode ser vista como uma matriz de pontos com N linhas e M colunas onde cada elemento da matriz representa uma quantidade discreta pertencente a um intervalo $[0, k-1]$:

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

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.

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 implicaria em 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 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 geralmente apresentam um efeito de *falso contorno*, mas geralmente imagens com grande quantidade de detalhes podem ser bem representadas utilizando poucos níveis de cinza.

2.2 Processamento Digital de Imagens

Os métodos de processamento digital de imagens tem 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 jornalisticas 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, fisica, 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 tipicos desta área são: reconhecimento automático de caracteres, máquinas com visão industrial para acompanhamento da montagem de produtos, reconhecimento militar, entre outros.

2.2.1 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.

2.2.2 Áreas fundamentais do processamento de imagens

Processamento digital de imagens envolve os seguintes tópicos 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 adequadas para o tipo de análise em questão.

2.2.3 Elementos de sistemas de processamento de imagens digitais

- **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 formas 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 de 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 a necessidade de velocidade de 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.

2.3 Realce de Imagens

O principal objetivo das técnicas de realce é processar uma imagem de 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 ruidoso. 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 de uma vizinhança 3x3 do pixel atual (ou vizinhança-8). 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 pela 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

Uma operação f é linear se, dadas duas imagens I e J e dois escalares a e b, se tiver:

$$f(aI + bJ) = af(I) + bf(J),$$

onde o produto de um escalar por uma imagem é definido como o produto de cada ponto pelo escalar. Operadores de filtragem que atuam sobre imagens para realizar funções de realce podem ser classificados como *pontuais* e *locais*.

- *Operadores pontuais* - o nível de cinza de um ponto na imagem transformada depende só do nível de cinza do ponto na imagem original (ou nas imagens originais se houver mais de uma imagem de entrada).
- *Operadores locais* - o novo nível de cinza de um ponto depende não só de seu nível de cinza antigo mas também dos níveis de cinza de seus vizinhos.

  Uma operação f é invariante ao deslocamento se:

$$f(I)(x + a, y + b) = f(I')(x, y)$$

onde $I'(x, y) = I(x + a, y + b)$.

Uma convolução é uma operação na qual o nível de cinza de um ponto da imagem transformada (processada) é obtido como uma combinação linear de toda a imagem de entrada e onde os coeficientes da combinação dependem só das posições relativas entre o ponto e os demais pontos da imagem. Pode-se mostrar que a convolução é uma operação linear e invariante ao deslocamento[1].

Sabemos que a convolução de duas funções unidimensionais, representada por $f(x) * g(x)$ é definida pela seguinte integral

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\alpha).g(x - \alpha).d\alpha$$

ou

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(x - \alpha).g(\alpha).d\alpha$$

onde α é uma variável utilizada para o cálculo da integração. Para o caso discreto, pode-se supor que $f(x)$ e $g(x)$ sejam discretizadas de modo que os intervalos de discretização sejam indênticos.

Assim, a convolução discreta de $f(x)$ e $g(x)$ pode então ser definida como o seguinte somatório:

$$f(x) * g(x) = \frac{1}{M}. \sum_{m=0}^{M-1} f(m).g(x - m)$$

ou

$$f(x) * g(x) = \frac{1}{M}. \sum_{m=0}^{M-1} f(x - m).g(m)$$

Quando temos funções bidimensionais , a convolução no caso contínuo pode ser definida de maneira análoga à convolução unidimensional da seguinte forma:

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

ou

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

onde α *e* β são variáveis auxiliares utilizadas para o cálculo da integração.

Na convolução bidimensional discreta, que é o caso da convolução de duas imagens digitais, dadas duas funções (imagens) $I(x,y)$ *e* $J(x,y)$, sendo x=0,1,...,M-1 e y=0,1,...,N-1, é uma outra função $H(x,y) = I(x,y) * J(x,y)$, onde , x=0,1,...,M-1 e y=0,1,...,N-1, dada por:

$$H(x,y) = \left(\frac{1}{M.N}\right) . \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(m,n).J(x-m, y-n),$$

ou equivalentemente por:

$$H(x,y) = \left(\frac{1}{M.N}\right) . \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(x-m, y-n).J(m,n)$$

O resultado da convolução de I por J num ponto p é na realidade uma média ponderada dos pontos de I , onde os pesos são dados pela imagem J. Ou seja , dado um ponto p de I, os pontos correspondentes que servem de peso na imagem J é uma vizinhança de p na imagem I, a qual chamamos de filtro ou máscara.

O método básico para a realização de filtragem 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. O funcionamento desta convolução de uma imagem com um filtro 3x3 será assunto da próxima secção.

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. Por questões de simetria usam-se, na definição dos núcleos dos filtros, vizinhaças nxn, onde n é um número ímpar. Por questões de eficiência computacional, preferem-se valores pequenos para n (no máximo 7). 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 filtragem por 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 corresponde ao pixel central da máscara (ou núcleo da máscara), definido pelas coordenadas (x,y), na imagem destino receberá um valor calculado em função de todos os elementos da imagem fonte cobertos pela 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.

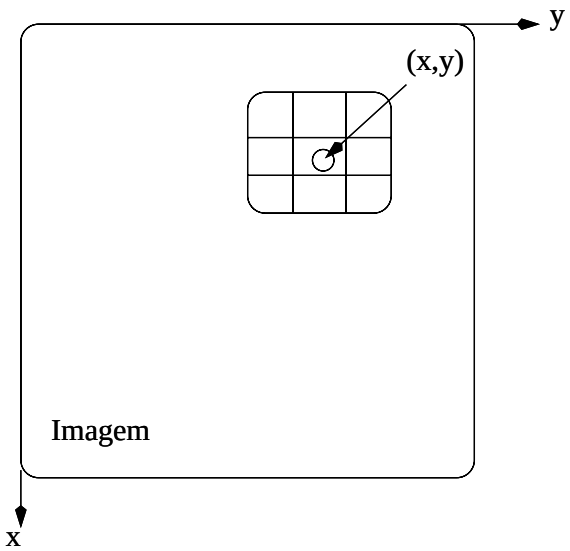


Figura 2.1: Máscara de Deslocamento

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. A máscara pode ter formatos diferentes, como cruz ou retângulo. Alguns algoritmos admitem a escolha do formato da máscara de deslocamento e outros são projetados para um tipo específicode 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 janela não terão valores definidos por estarem fora dos limites da imagem. Para contornar este problema, pode-se restringir o posicionamento da máscara de modo 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.

Capítulo 3

Segmentação

Por segmentação de uma imagem entende-se a extração ou identificação de objetos contidos na imagem, onde "objeto" é toda característica com conteúdo semântico relevante para a aplicação desejada. As descrições são constituídas basicamente de uma lista de objetos, seus rótulos e relações entre os objetos. Podem-se dintinguir dois tipos de objetos: objetos "complexos", que são formados por outros objetos, e objetos "simples", onde isto não acontece[2].

O processo de segmentação consiste em uma divisão ou separação de uma imagem em regiões de atributos similares. Os atributos básicos utilizados são a amplitude e a luminância, isto quando se trata de imagem monocromática, e componentes de cores (brilho,contraste,etc) para imagens coloridas. Contornos (Bordas) e textura também são atributos úteis à segmentação. Além disso, este processo não envolve classificar cada segmento. O segmentador apenas subdivide a imagem, não se dedicando a reconhecer os segmentos individuais ou relações (similaridades) de um pixel com uma certa região ou a outros pixels. Portanto, a segmentação faz parte, em geral, de um processo maior que é o de obter uma descrição da imagem[3][2].

Um dos passos na análise da imagem é a identificação dos objetos simples que correspondem, em geral, a linhas ou regiões (grupos de pontos conectados). A deteção das regiões pode ser feita de dois modos: através do agrupamento de pontos vizinhos com características semelhantes, ou através da determinação da fronteira (borda) da região. Neste capítulo serão vistos alguns métodos para a deteção de bordas e métodos para agrupamentos de pontos em regiões[2].

3.1 Realce de Bordas

Mudanças ou descontinuidades em algum atributo da amplitude de uma imagem, tal como a luminância, são características primitivas fundamentalmente importantes de uma imagem, pois elas frequentemente provêem indicações da extensão física de objetos na imagem[3]. Uma borda numa imagem monocromática, por exemplo, é uma mudança súbita do nível de cinza entre duas regiões relativamente homogêneas. Cada região é uniforme e homogênea com relação a alguma propriedade da amplitude, tais como tom ou textura, e o valor desta propriedade difere de maneira significante de acordo com a vizinhança de cada região[2][5]. Idealmente, a seção transversal de uma borda apresenta a forma de uma função degrau. Uma linha caracteriza-se por ter um nível de cinza relativamente constante ao longo de uma

faixa estreita e alongada. A seção transversal de uma linha tem a forma de um pico (ou depressão) estreito. Tanto as bordas quanto as linhas são descontinuidades da imagem.

Embora seja possível detectar bordas na imagem original, é conveniente descrever os algoritmos como se eles processassem dados de duas imagens derivadas: Uma que contém, para cada ponto, a magnitude da borda e outra que dá a direção e o sentido dela. Estas duas imagens podem ser obtidas pelos métodos que serão descritos a seguir, através de operadores diferenciais. Existem duas grandes classes de detectores de bordas:

- **Diferenciador de Primeira Ordem**
- **Diferenciador de Segunda Ordem**

Para a classe de Primeira Ordem, algumas formas de diferenciação espacial de primeira ordem são aplicadas, e o resultado do gradiente para detecção de bordas é comparado a um valor de limiar de nível de cinza. Uma borda é julgada existente se o gradiente excede este valor de limiar. Para a classe de Segunda Ordem, uma borda é julgada existente se há uma mudança significante na polaridade da segunda derivada[3].

3.1.1 Realce de Bordas aplicando Derivada de Primeira Ordem

Existem dois operadores gradiente fundamentais de Primeira Ordem (ou detetores de borda diferenciadores de primeira ordem). Um método envolve a geração de gradientes em duas direções ortogonais na imagem, enquanto o outro método utiliza um conjunto de derivadas direcionais. Dentre muitos operadores gradiente existentes na classe dos derivativos de primeira ordem os que mais se destacam são os operadores de *Roberts* e de *Sobel*.[3]

Operador de Roberts

Para a deteção de bordas, o método mais simples que existe talvez seja o *operador gradiente de Roberts* o qual pode ser descrito utilizando duas máscaras, resultando em um valor de gradiente para a primeira diagonal, gr1, e um valor para a segunda diagonal, gr2:

1	0
0	-1

gr1

0	-1
1	0

gr2

Figura 3.1: Máscaras para o cálculo do operador gradiente de Roberts

O valor exato do gradiente bidimensional na posição (x, y), correspondente ao ponto sup[erior esquerdo da janela 2x2, seria calculado por:

$$g_r(x, y) = \sqrt{g_{r1}^2 + g_{r2}^2} \tag{3.1}$$

Devido ao custo computacional, as operações de elevar ao quadrado e raiz quadrada são muitas vezes substituídas pela seguinte aproximação:

$$g_r(x, y) = \alpha.(|g_{r1}(x, y)| + |g_{r2}(x, y)|) \tag{3.2}$$

que é mais eficiente.

Uma desvantagem do operador de Roberts é a sua "anisotropia", ou seja , sua assimetria. Dependendo da direção, certas bordas são mais realçadas que outras, mesmo tendo igual magnitude.[2][12]

Operador de Sobel

Um operador gradiente mais sofisticado (3 X 3) é o *operador de Sobel*, o qual pode ser descrito utilizando duas máscaras, mostradas na Figura 3.2, resultando em um valor de gradiente na horizontal, g_{ox}, e outro valor de gradiente na vertical, g_{oy}, para o ponto central da janela 3x3.

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

gox

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

goy

Figura 3.2: Máscaras para o cálculo do gradiente de Sobel

O valor absoluto do gradiente bidimensional no ponto $g_o(x, y)$ é calculado por:

$$g_o(x, y) = \alpha(|g_{ox}| + |g_{oy}|) \tag{3.3}$$

Operador Spline

Este algoritmo implementa um novo operador gradiente , o qual é baseado em splines cúbicas para o realce de contornos em imagens digitais. A aproximação para o gradiente em um determinado ponto é obtida através da convolução da imagem com quatro máscaras. Cada máscara aproxima no ponto central a derivada da spline cúbica que interpola em uma das quatro direções possíveis: uma para a horizontal, uma para a vertical e duas para as diagonais. O desempenho do algoritmo proposto é comparado posteriormente com o dos dois outros métodos de cálculo do gradiente já mencionados.[12]

Assim como nos gradientes de Roberts e Sobel, o gradiente Spline é derivado de gradientes unidimensionais[12]. Este gradiente pode ser descrito usando quatro máscaras, como jámencionado, resultando em um gradiente horizontal, g_{px}, um gradiente vertical, g_{py} e dois gradientes nas diagonais, g_{p1} *e* g_{p2}.

O valor absoluto do gradiente Spline bidimensional é calculado por:

$$g_p(x, y) = \alpha(3(|g_{px}| + |g_{py}|) + 2(|g_{p1}| + |g_{p2}|)) \tag{3.4}$$

Observe que o peso dos valores de gradiente g_{px} *e* g_{py} são maiores que os pesos para g_{p1} *e* g_{p2} por um fator de $\frac{3}{2}$. Esta é uma aproximação de um fator de $\sqrt{2}$ que corresponde a relação das distâncias entre os pontos vizinhos nas direções vertical e horizontal.[12]

Os fatores de escala nas Eqs. 3.2, 3.3 e 3.4 são determinados de acordo com o máximo valor de todos os pixels da imagem gradiente considerando, primeiramente, o fator de escala igual a 1 (um). O fator de escala é calculado da seguinte forma: primeiramente convoluimos ambos os filtros do operador gradiente com a imagem de entrada e com o fator de escala

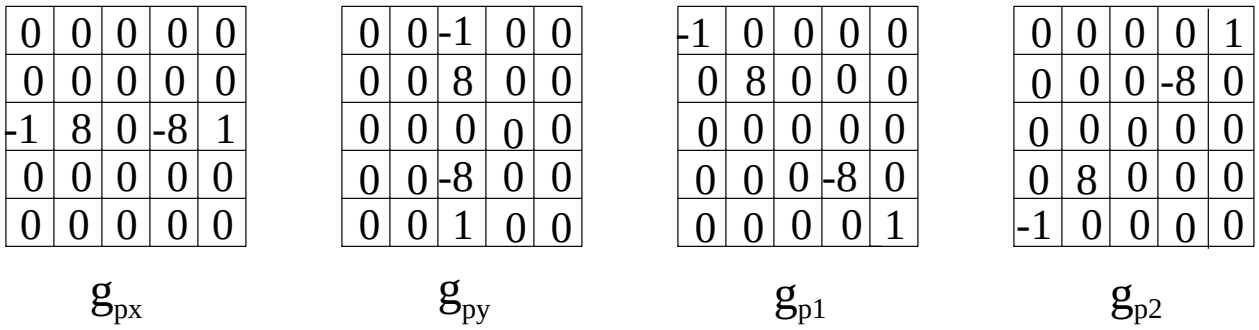


Figura 3.3: Máscaras para o cálculo do gradiente Spline

$\alpha = 1$, como já foi mencionado. Obtida a imagem gradiente, observamos qual o maior valor de nível de cinza existente nesta e consequentemente calculamos um fator de escala que multiplicado pelo maior valor de nível de cinza seja igual a $2^n - 1$ onde n é o número de bits por pixel da imagem. Por exemplo, se o maior valor de nível de cinza da imagem gradiente for 134 em uma imagem representada com 6 bits por pixel, ou seja, a faixa de valores de níveis de cinza da imagem original é $[0, 63]$, então o fator de escala pode ser calculado da seguinte forma:

$$134\alpha = 63 \iff \alpha = \frac{63}{134}$$

Como pudemos observar o fator de escala da fórmula do gradiente não depende de forma alguma da resolução da imagem original[2][12].

3.1.2 Realce de Bordas através de Derivadas de Segunda Ordem

Técnicas de detecção de bordas através de derivadas de segunda ordem empregam alguma forma de diferenciação espacial de segunda ordem para *"afinar"* (realçar) as bordas. Uma borda é dita existente se uma mudança significativa ocorre na segunda derivada. O mais famoso método diferenciador de segunda ordem é o *Laplaciano*[3].

Cálculo do Laplaciano

O operador Laplaciano de uma imagem $F(x, y)$ contínua é definido como :

$$G(x, y) = -\nabla^2 F(x, y)$$

Onde o Laplaciano é

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{3.5}$$

O Laplaciano $G(x, y)$ é zero se $F(x, y)$ é constante ou varia lineamente em amplitude. Se a taxa de variação (derivada) de $F(x, y)$ é maior que numa função linear, $G(x, y)$ exibe uma mudança de sinal no ponto de inflexão de $F(x, y)$. Quando a derivada da função $G(x, y)$ passa por zero significa dizer que há presença de borda.

No domínio discreto a mais simples aproximação para o Laplaciano contínuo é calcular as diferenças das inclinações (derivadas) em cada posição varrendo toda a imagem de acordo com a expressão abaixo:

$$G(j,k) = [F(j,k) - F(j,k-1)] - [F(j,k+1) - F(j,k)] + [F(j,k) - F(j+1,k)] - [F(j-1,k) - F(j,k)] \tag{3.6}$$

Este Laplaciano pode ser gerado pela convolução

$$G(j,k) = F(j,k) * H(j,k) \tag{3.7}$$

com

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.8}$$

ou

$$\mathbf{H} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.9}$$

Onde as duas matrizes de 3.8, correspondem às segundas derivadas ao longo de todas as linhas e colunas, respectivamente, assim como no Laplaciano contínuo expresso na Eq. 3.5. O Laplaciano discreto mostrado anteriormente na Eq. 3.6 é usualmente normalizado para unificar as médias de ganho dos pesos dos pixels positivos e negativos do filtro 3x3 $\mathbf{H}$. O filtro normalizado é definido como:

$$\mathbf{H} = \frac{1}{4} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.10}$$

3.2 Rastreamento de Contorno

Há, pelo menos, dois modos de fazer o rastreamento do contorno: segundo a varredura e por busca em todas as direções.[2] Ambos trabalham coma imagem gradiente, e não coma original.

Rastreamento segundo a varredura

No rastreamento segundo a varredura é varrida uma linha da imagem gradiente por vez e várias curvas são rastreadas de uma só vez. O processo é iniciado pela detecção de um ponto inicial do contorno, que é feito considerando valores de borda (gradiente) maiores que um limiar L. Para cada ponto (x, y) são examinados os vizinhos anteriores (já visitados e rotulados) e testado se este ponto corresponde à continuação de algum contorno já detectado. No teste de continuidade, o valor da borda em (x, y) é comparado com um limiar D (D $\leq$ L) e é verificado se a direção desta no ponto (x, y) está alinhada com a da borda num de seus vizinhos. Os passos do procedimento são mostrados no algoritmo descrito a seguir.

ALGORITMO 3.2.3.1 - *Rastreamento segundo a varredura (são dados D,L e ALFA)*:

1. $k \longleftarrow 0$
2. percorra a imagem no sentido da varedura;
3. para todo ponto g, tal que $A(g) > D$
   (a) se existir borda em p ($borda(p) \neq 0$) , p vizinho anterior de g, $|\phi(p) - \phi(g)| < ALFA$, então continue em g a borda cuja direção coincide melhor com g:
   $borda(g) \longleftarrow borda(p)$;
   (b) caso contrário, $seA(g) > L$, inicie nova borda: $k \longleftarrow k + 1$; $borda(g) \longleftarrow k$.

Notação:
g é (x, y); os vizinhos anteriores de g são $(x-1, y), (x, y-1), (x-1, y-1)$ e $(x-1, y+1)$; $A(g)$ é a magnitude da borda em g; $\phi(p)$ é a direção da borda em p; e $borda(p)$ indica qual a borda em p (se igual a zero então não há borda em p ou p ainda não foi visitado).

Deve-se incluir no algoritmo descrito anteriormente, uma linha e uma coluna (com bordas=0) antes da primeira linha e coluna da imagem. Usam-se limiares diferentes (L e D) para a detecção do contorno e sua continuação. Como $D \leq L$, o critério para a continuidade fica sendo (quanto à magnitude) menos rigoroso que para a deteção. Isto faz com que as bordas extraídas sejam diferentes, dependendo do sentido e direção da varredura.[2]

Rastreamento em todas as direções

No rastreamento por busca em todas as direções, segue-se um único contorno por vez. Inicialmente, detecta-se um ponto de contorno (por exemplo, escolhe-se o ponto da imagem gradiente com maior magnitude de borda) e examinam-se seus vizinhos procurando possíveis continuidades. Um vizinho é sua continuação se sua magnitude for maior que um limiar (d) e a borda estiver alinhada com a do ponto escolhido. Das possíveis continuidades (se houver) escolhe-se uma e repete-se o processo. As outras possibilidades são armazenadas para investigação posterior. Entre os pontos examinados são excluídos aqueles que pertencem a algum contorno. O algoritmo a seguir descreve os passos de rastreamento por busca uni-direcional.

ALGORITMO 3.2.3.2 - *Rastreamento do contorno por busca em todas as direções:*

1. $k \longleftarrow 0$
2. escolha um ponto g, tal que $A(g) > L$ e a $borda(g) = 0$; se não existe ponto nestas condições, então pare (fim).
3. inicie novo contorno: $k \longleftarrow k+1; borda(k) \longleftarrow k$;
4. ache todos os pontos p da vizinhança de g, tais que $A(p) > D$, a $borda(p) = 0$ e $|\phi(p) - \phi(g)| < ALFA$; para todos estes pontos p faça a $borda(p) \longleftarrow borda(g)$ e guarde estes pontos num conjunto X;
5. se X está vazio , vá para 2;
6. escolha e extraia um ponto g de X; vá para 4.

Notação:
g é (x, y); os vizinhos anteriores de g são $(x-1, y), (x, y-1), (x-1, y-1)$ *e* $(x-1, y+1)$; $A(g)$ é a magnitude da borda em g; $\phi(p)$ é a direção da borda em p; e $borda(p)$ indica qual a borda em p (se igual a zero então não há borda em p ou p ainda não foi visitado).

No algoritmo 3.2.3.2 ao invés de usar a direção no ponto g ($\phi(g)$) como referência, pode-se usar a "direção do contorno" definida como a média das direções dos últimos pontos do contorno (3 últimos, por exemplo). Isto faz o algoritmo menos sensível ao ruído. O mesmo pode ser feito com relação ao algoritmo 3.2.3.1. É fácil ver que, embora menos sensível que o algoritmo 3.2.3.1 à ordem em que as operações são feitas, o resultado obtido pelo algoritmo 3.2.3.2 depende, ainda , da ordem em que os pontos são escolhidos[2].

3.3 Seleção do Valor Limiar de Nível de Cinza ("*Threshold*")

Após a formação da imagem gradiente, através de métodos de deteção de bordas derivativos, é necessário que a imagem gradiente seja comparada a um valor de limiar (*threshold*) para determinar se existe contornos (bordas). O valor de limiar determina a sensibilidade do detetor de bordas.

Para imagens sem ruídos, este limiar pode ser escolhido de modo que todas as descontinuidades de amplitude a partir de um nível de contraste mínimo sejam reconhecidas como bordas.

Imagens com ruído, torna esta seleção difícil pois o valor de limiar tanto pode deixar despercebido bordas existentes quanto pode designar falsas bordas induzidas pelo ruído[3].

Vários métodos estão disponíveis para a escolha do limiar. Um deles é aproximar o histograma pela soma de duas normais. O limiar ótimo corresponde à interseção destas duas normais. Outro método bastante usado e conhecido consiste em suavizar o histograma e tomar o mínimo entre os dois picos.

Em vários problemas práticos, é interessante usar um limiar que varia de acordo com a posição na imagem para corrigir o efeito da iluminação desigual[2].

3.4 Deteção de Regiões

A deteção de regiões em imagens pode ser feita com um dos objetivos: extrair uma determinada região ou dividir (particionar) a imagem num conjunto de regiões disjuntas, cuja união seja a imagem inteira. Como os métodos para ambos os objetivos são semelhantes, far-se-á um tratamento unificado destes casos.

Uma região numa imagem é um conjunto de pontos "*ligados*" ou "conectados", isto é, de qualquer ponto da região pode-se chegar a qualquer outro ponto por um caminho inteiramente contido na região. Esta definição de região depende naturalmente do conceito de vizinhança usado - se vizinhança-4 ou vizinhança-8.

As regiões que se deseja detectar, em geral, são regiões "*homogêneas*", ou seja, apresentam alguma propriedade local aproximadamente constante em toda a sua extensão. Como mencionado anteriormente, algumas propriedades locais usadas comumente são nível de cinza e textura.

Os métodos estudados neste capítulo para particionamento são:

1. Crescimento de Regiões ("*region growing*");
2. Divisão e Fusão ("*split-and-merge*") de regiões;
3. Classificação de Pontos.

O primeiro método é semelhante ao do rastreamento de contorno. Tal como no caso da detecção de bordas, o crescimento pode ser feito segundo a varredura ou em todas as direções. Também as decisões são tomadas localmente no método de crescimento de regiões. Os métodos da divisão e fusão e o da classificação baseiam suas decisões em informações mais globais. Como o método da divisão e fusão considera uma partição inicial, presume-se que as informações de caráter exclusivamente local foram levadas em conta neste passo[2].

3.4.1 Crescimento de Regiões

É possível fazer o crescimento de regiões segundo a varredura ou por busca em todas as direções. No primeiro caso, a imagem é percorrida de cima para baixo e da esquerda para a direita. Cada ponto (x, y) é comparado com seus vizinhos anteriores (já rotulados) e adicionado (ou não) à região de um dos vizinhos. Ao contrário da detecção de contorno, todo ponto da imagem pertencerá a uma região; assim, se o ponto não for adicionado à região de um vizinho, ele deverá iniciar uma nova região. O método é descrito no Algoritmo 3.4.1.1.

ALGORITMO 3.4.1.1 - *Crescimento de Regiões Segundo a Varredura*

1. $k \longleftarrow 0$;
2. percorra a imagem segundo a varredura;
3. compare cada ponto g com seus vizinhos anteriores; dois casos podem acontecer.
   (a) g não pode ser adicionado à região de nenhum vizinho anterior por violar a condição de homogeneidade; neste caso inicie nova região: $k \longleftarrow k + 1$; $região(g) \longleftarrow k$;
   (b) g pode ser adicionado à região de p (vizinho anterior); neste caso adicione g à região de p : $região(g) \longleftarrow região(p)$.

Notação:
g é o ponto (x, y); os vizinhos anteriores de g são os pontos $(x-1, y)$, $(x, y-1)$, $(x-1, y-1)$, $(x-1, y+1)$; $região(p)$ indica a região do ponto p.

Ao contrário do crescimento de regiões segundo a varredura, onde várias regiões são "*crescidas*" simultaneamente, no crescimento por busca em todas as direções uma única região é crescida por vez. O método é semelhante ao da perseguição do contorno em todas as direções como mostrado no Algoritmo 3.4.1.2.

ALGORITMO 3.4.1.2 - *Crescimento de regiões por busca em todas as direções*

1. $k \longleftarrow 0$
2. escolha um ponto g, tal que $região(g) = 0$, se não existe ponto nesta condição, então pare: fim;
3. inicie nova região: $k \longleftarrow k + 1$; $região(g) \longleftarrow k$;
4. ache todos os pontos da vizinhança de g, tal que $região(p) = 0$ e p possa ser adicionado a região de g sem violar o critério de homogeneidade: faça $região(p) \longleftarrow região(g)$ e guarde estes pontos em um conjunto X;
5. se X está vazio, vá para 2; caso contrário , escolha e extraia um ponto g de X;
6. vá para 4;

Notação:
g é o ponto (x, y); os vizinhos anteriores de g são os pontos $(x-1, y)$, $(x, y-1)$, $(x-1, y-1)$, $(x-1, y+1)$; $região(p)$ indica a região do ponto p; inicialmente $região(p) = 0$ para todo ponto p da imagem.

A partição final obtida pelo algoritmo 3.4.1.2 é , em geral , dependente não só da ordem em que os pontos são escolhidos no passo 1 como da ordem em que os vizinhos de x são examinados, passo 3.[2]

3.4.2 Método da Divisão e Fusão

Dada uma partição inicial da imagem e uma função $H(.)$ [1] é uma função booleana tal que , para qualquer região R,*H(R)* seja verdadeira se e somente se R for homogênea, o método da divisão e fusão manipula a partição através de divisões e fusões de regiões até obter uma partição que satisfaça H. Uma versão do procedimento é apresentada no algoritmo 3.4.2.1.

ALGORÍTMO 3.4.2.1 - *Detecção de Regiões por Divisão e Fusão*

1. Escolha uma partição inicial;
2. escolha uma região R tal que *H(R)* seja falsa;
3. divida R em sub-regiões $R_1, R_2, \ldots$ tais que $H(R_i)$ seja verdadeira;
4. ache um par de regiões adjacentes R_i e R_j tal que $H(R_i \cup R_j)$ seja verdadeira;
5. una R_i e R_j criando uma nova região;

[1] *H(.)*

6. se não for possível achar nenhuma região para dividir, nem um par de regiões para unir, então pare; caso contrário vá para o passo 2.

É sempre possível no passo 3 decompor a região em sub-regiões que satifçam H, pois H é satisfeita para regiões pontuais. Contudo , deve-se procurar dividir a região num número mínimo de subregiões , o que nem sempre é fácil. Uma alternativa é não exigir que as sub-regiões, satisfaçam H. Mesmo assim o algoritmo converge para uma partição que satisfaça H, pois cada vez são obtidas regiões menores até atingir regiões pontuais. Deste modo pode-se reescrever o passo 3 na forma:

3) divida R em duas regiões, $R_1 e R_2$.

Um possível critério para dividir R é escolher duas regiões R_1 *e* R_2 de mesma área. A partição final depende da partição inicial escolhida. O número de passos dados até a convergência depende também da partição inicial escolhida, que tende a ser maior para regiões pouco homogêneas.

A ordem em que as regiões são consideradas influi também no resultado final. É necessário, por conseguinte, prover critérios para a ordem em que as regiões são examinadas. Por exemplo, pode-se dividir as regiões maiores e unir as regiões menores.

É possível enunciar duas versões simplificadas do algoritmo da divisão e fusão onde só ocorrem divisões ou só ocorrem fusões. No primeiro caso, começa-se uma divisão bem "*grossa*" da imagem, eventualmente a própria imagem. Fazem-se então divisões até chegar a uma partição satisfatória. A outra possibilidade é começar de uma partição "*fina*" (por exemplo, constituídas só de regiões pontuais) e através de fusões obter a partição desejada. Nos dois casos como são realizadas só divisões ou fusões, estas devem ser feitas criteriosamente.[2]

3.4.3 Segmentação por Classificação de Pontos

A classificação da imagem, ou seja, a atribuição de uma classe a cada ponto da imagem, pode ser vista como um método de segmentação. Em função de propriedades do ponto, escolhe-se uma classe para ele. Entre as propriedades que podem ser usadas estão o nível de cinza e as medidas de propriedades locais (por exemplo, textura). Para imagens multiespectrais, costumam-se usar os níveis de cinza nas várias bandas. Assim sendo, a cada ponto é associado um vetor de medidas.

Uma vez representados os pontos da imagem neste espaço bidimensional de características, há duas maneiras de rotular os pontos da imagem: usando as medidas de um conjunto de protótipos cujas classes são conhecidas (classificação supervisionada), ou agrupando numa mesma classe pontos com medidas semelhantes (classificação não-supervisionada, agregamento,"*clustering*").

Após a obtenção da imagem rotulada, o passo seguinte é a reunião dos pontos de uma mesma classe em regiões de pontos conectados. Isto pode ser feito em uma única varredura da imagem por um algoritmo de crescimento de regiões segundo a varredura.[2]

Capítulo 4

Descrição dos Algoritmos Implementados e Resultados Experimentais

Para atingirmos os objetivos desta fase inicial do projeto, que seria a construção de uma biblioteca de algoritmos para segmentação de imagens, foi necessário escolher uma linguagem de programação eficiente que facilitasse a implementação desses algoritmos. Optou-se então pela linguagem C, devido a 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. Lembrando também que C é totalmente compatível com os sistemas operacionais MS-DOS e UNIX, ambos bastante utilizados no meio acadêmico. A partir disso iniciou-se um estudo completo desta linguagem de programação [4]; em paralelo foi estudado os fundamentos do processamento digital de imagens, com destaque para a segmentação de imagens, discutidos no capítulo anterior.[8]

Aos poucos, foram sendo implementados processos e pré-processos da segmentação por extração de contornos, tais como: Operadores Gradiente (Roberts,Sobel,Spline) e rastreamento de contornos em todas as direções. Também foram implementados vários algoritmos de crescimento de regiões. 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*, um ambiente para pesquisa desenvolvido na Universidade do Novo México. 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 para PDI. 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 auxílio no desenvolvimento de sistemas e também como 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.[8]

Desta forma, devido a facilidade de leitura, definiu-se o formato *ASCII* como referência para o 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, com exceção do algoritmo de deteção de bordas que tem sua estrutura bem distinta das demais. Em seguida foi feito um conjunto de testes, todos realizados com a imagem *Bloco* , apresentando dimensões 84x87, com gradações de cinza abrangendo os 256 níveis possíveis. Esta imagem foi escolhida devido à sua grande utilização dentro das pesquisas desenvolvidas na área.[8]

Finalmente terminado os testes foi feita a descrição dos algoritmos selecionados e a análise dos resultados obtidos.

4.1 Descrição dos Algoritmos Implementados

4.1.1 Algoritmo básico de Filtragem

Inicialmente foi definido a implementação de um algoritmo de *covolução espacial* pois este é a base da maioria dos algoritmos usados em processamento digital de imagens.

O algoritmo inicial de convolução espacial utilizava uma máscara 3x3 , cujo processo já mencionamos no capítulo 2. Sua estrutura final resultou no seguinte:

1. Leitura do nome do arquivo da matriz (imagem) de entrada via teclado
2. Leitura da dimensão da matriz (imagem) de entrada via teclado
3. Leitura e armazenagem dos dados da matriz (imagem) de entrada
4. Leitura dos componentes do filtro (máscara) via teclado
5. Réplica das primeiras e últimas linhas e das primeiras e últimas colunas da matriz (imagem de entrada)
6. Convolução espacial da matriz da imagem de entrada com o filtro (máscara)
7. Armazenamento da matriz (imagem) convoluída em um arquivo de saída
8. Leitura do nome do arquivo de saída

As réplicas feitas no passo 5 da estrutura do programa são necessárias pois senão não teríamos como calcular os valores dos níveis de cinza dos pixels convoluídos da primeira e ultima linha, e primeira e ultima coluna da matriz da imagem de entrada.

Além da similaridade, este algoritmo apresenta uma estrutura muito parecida com os próximos dois programas que serão descritos, somente havendo uma pequena modificação nos passos 3 e 4 como poderemos perceber.

4.1.2 Algoritmos de Operadores Gradiente (Detetores de Bordas)

Após embasamento e estudos aprofundados sobre *segmentação* , surgiu a necessidade de que implementássemos algoritmos de operadores gradiente para "afinar" (realçar) bordas. Como já vimos no Capítulo 3,o realce das bordas ajuda na deteção delas, que é um dos pré-processos para segmentação de imagens.

Operador Gradiente de Roberts

Descrição do Algoritmo:

1. Definição dos filtros (máscaras) do operador gradiente de Roberts.
2. Leitura do nome do arquivo da matriz (imagem) de entrada via teclado.
3. Leitura da dimensão da matriz (imagem) de entrada via teclado.
4. Leitura e armazenagem dos dados da matriz (imagem) de entrada.
5. Réplica das primeiras e últimas linhas e das primeiras e últimas colunas da matriz (imagem) de entrada.
6. Convoluções espaciais da matriz da imagem de entrada com os filtros (máscaras) do operador gradiente de Roberts.
7. Armazenamento da matriz (imagem) convoluída em um arquivo de saída.
8. Leitura do nome do arquivo de saída.

Podemos perceber com clareza que neste não precisamos ler , via teclado , os coeficientes dos filtros (máscaras) a serem convoluídos com a imagem de entrada. Estes foram definidos dentro do próprio programa.

Vale salientar que , embora saibamos que os filtros convoluídos com a imagem de entrada sejam 2x2 , estes filtros aplicados neste algoritmo são equivalentes àqueles que outrora foram expostos.

Operador Gradiente de Sobel

Descrição do Algoritmo:

1. Definição dos filtros (máscaras) do operador gradiente de Sobel.
2. Leitura do nome do arquivo da matriz (imagem) de entrada via teclado.
3. Leitura da dimensão da matriz (imagem) de entrada via teclado.
4. Leitura e armazenagem dos dados da matriz (imagem) de entrada.
5. Réplica das primeiras e últimas linhas e das primeiras e últimas colunas da matriz (imagem) de entrada.
6. Convoluções espaciais da matriz da imagem de entrada com os filtros (máscaras) do operador gradiente de Sobel.
7. Armazenamento da matriz (imagem) convoluída em um arquivo de saída.

8. Leitura do nome do arquivo de saída.

Este programa é idêntico ao algoritmo anterior só que os filtros definidos no programa são diferentes,mas o processamento é o mesmo.

Operador Gradiente Spline

Descrição do Algoritmo:

1. Definição dos filtros (máscaras) do operador gradiente Spline.
2. Leitura do nome do arquivo da matriz (imagem) de entrada via teclado.
3. Leitura da dimensão da matriz (imagem) de entrada via teclado.
4. Leitura e armazenagem dos dados da matriz (imagem) de entrada.
5. Réplica das duas primeiras e últimas linhas e das duas primeiras e últimas colunas da matriz (imagem) de entrada.
6. Convoluções espaciais da matriz da imagem de entrada com os filtros (máscaras) do operador Spline.
7. Armazenamento da matriz (imagem) convoluída em um arquivo de saída.
8. Leitura do nome do arquivo de saída.

Podemos observar que este também tem os mesmos passos que os outros só que agora estamos convoluindo a imagem com máscaras 5x5 e, por isso , modificamos o processo de convolução devido ao aumento do número de pixels vizinhos ao centro da máscara. Uma outra coisa que foi modificada pelo uso de máscaras 5x5 é a replica das duas primeiras e duas últimas linhas e das duas primeiras e duas últimas colunas da matriz (imagem) de entrada devido aos mesmo motivos mencionados em 4.1.1.

Resultados Obtidos pelos algoritmos de gradiente

Como cada gradiente apresenta características específicas é difícil fazer uma avaliação completa sobre as suas capacidades , preferiu-se então fazer uma comparação visual dos resultados. A Tabela 4.1, apresenta os resultados produzidos pelos três algoritmos de gradiente implementados. Como podemos observar os gradientes de Sobel e Spline deixam os contornos da figura na imagem muito mais "*afinados*", ou mais bem defidos, que o de Roberts. Para se obter uma análise mais profunda deve-se consultar uma literatura espeífica[12].

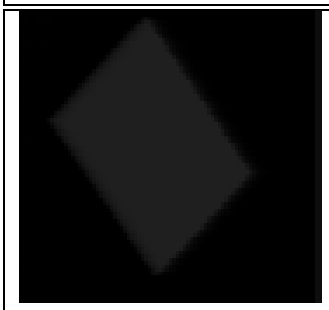
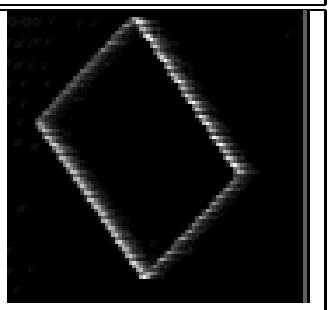
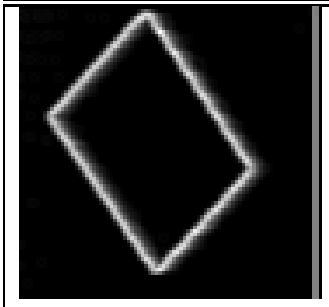
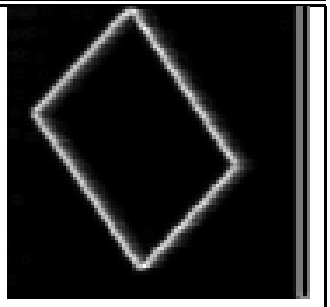
Original	Roberts
	
Sobel	Spline
	

Tabela 4.1: Resultados dos algoritmos de operadores gradiente

As performances destes gradientes também podem ser observadas utilizando o recurso de *"Zoom"*, como mostrado nas Tabelas 4.2 a 4.4. Nestas figuras são mostrados os valores de nível de cinza dos pixels em uma janela de cada imagem gradiente.

Percebemos,então, a vantagem do gradiente Spline, o qual além de realçar bem os contornos, não os deixa tão *borrados ou espalhados* quanto os contornos da imagem gradiente de Sobel. O gradiente Spline produz nos contornos da figura na imagem uma mudança muito abrupta de nível de cinza que, como já vimos, facilita o próximo passo de todo o processo de segmentação, que é o de extrair o contorno.

Algoritmo do Gradiente Laplaciano

Descrição do Algoritmo:

1. Definição dos filtros (máscaras) do operador gradiente de Laplaciano.
2. Leitura do nome do arquivo da matriz (imagem) de entrada via teclado.
3. Leitura da dimensão da matriz (imagem) de entrada via teclado.
4. Leitura e armazenagem dos dados da matriz (imagem) de entrada.
5. Réplica das primeiras e últimas linhas e das primeiras e últimas colunas da matriz (imagem) de entrada.
6. Convoluções espaciais da matriz da imagem de entrada com o filtro (máscara) do operador gradiente Laplaciano.
7. Armazenamento da matriz (imagem) convoluída em um arquivo de saída.

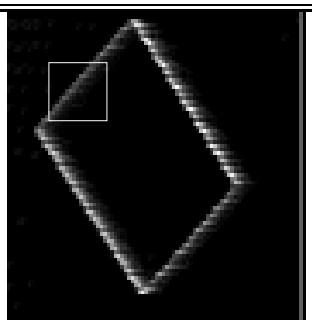
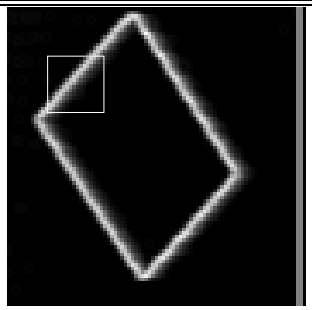
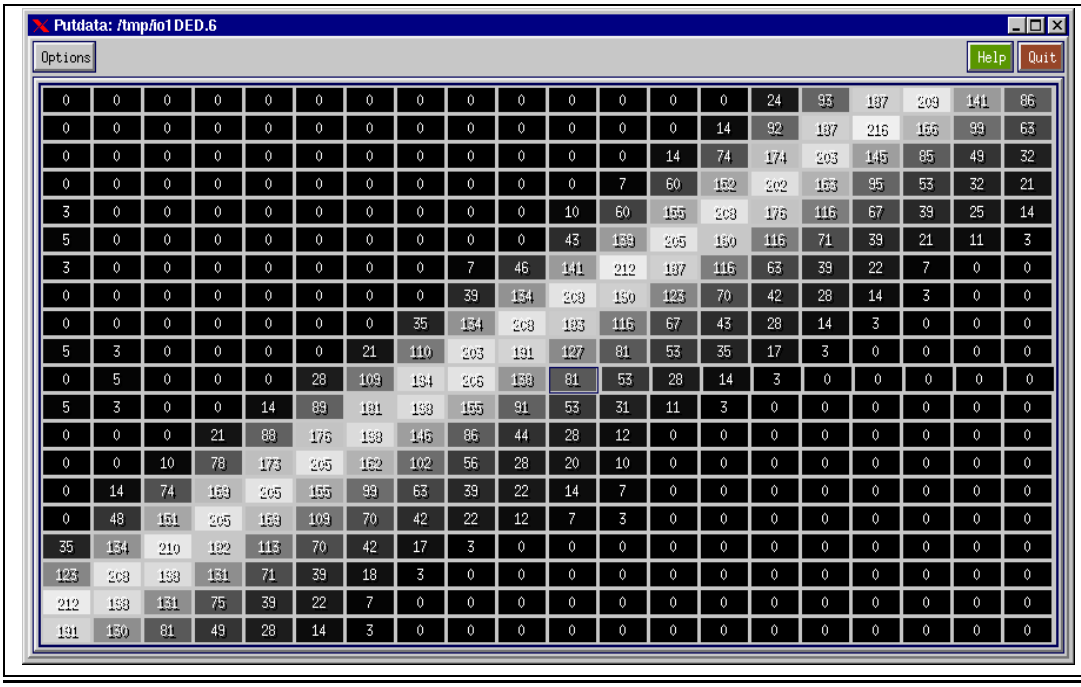
Roberts

Zoom
Putdata: /tmp/io1DED.6 Options Help Quit 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 80 50 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 50 123 80 50 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 29 80 50 29 14 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 29 65 58 29 21 7 7 0 0 0 0 0 0 0 0 0 0 0 0 0 14 65 58 43 21 7 7 7 7 0 0 0 0 0 0 0 0 0 0 0 21 80 80 58 21 14 21 14 0 0 0 0 0 0 0 0 0 0 0 0 0 65 65 43 21 14 7 7 0 0 0 0 0 0 0 0 0 0 0 14 94 94 65 36 21 7 7 0 0 0 0 0 0 0 0 0 0 0 0 65 65 36 21 14 14 7 0 0 0 0 0 0 0 0 0 0 0 0 72 80 50 36 29 21 7 0 0 0 0 0 7 7 0 0 0 0 0 43 65 58 21 21 7 7 0 0 0 0 0 0 14 0 0 0 0 0 58 94 72 36 36 14 14 0 0 0 0 0 0 0 0 0 0 0 0 29 65 58 36 14 14 7 0 0 0 0 0 0 0 0 0 0 0 0 43 94 80 36 21 7 7 7 0 0 0 0 0 0 0 0 0 0 0 21 72 58 29 21 7 7 0 7 0 0 0 0 0 0 0 0 0 0 29 87 65 29 21 14 21 14 14 0 0 0 0 0 0 0 0 0 0 65 72 43 21 14 7 0 0 0 0 0 0 0 0 0 0 0 0 0 72 80 43 21 14 7 0 0 0 0 0 0 0 0 0 0 0 0 0 58 65 43 29 14 7 7 0 0 0 0 0 0 0 0 0 0 0 0 0 101 65 36 14 7 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Tabela 4.2: Zoom na imagem gradiente de Roberts

Sobel



Zoom



0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	93	187	209	141	86
0	0	0	0	0	0	0	0	0	0	0	0	0	14	92	187	216	155	99	63
0	0	0	0	0	0	0	0	0	0	0	0	14	74	174	203	145	85	49	32
0	0	0	0	0	0	0	0	0	0	0	7	60	152	202	163	95	53	32	21
3	0	0	0	0	0	0	0	0	0	10	60	155	208	176	116	67	39	25	14
5	0	0	0	0	0	0	0	0	0	43	139	205	150	116	71	39	21	11	3
3	0	0	0	0	0	0	0	7	46	141	212	197	116	63	39	22	7	0	0
0	0	0	0	0	0	0	0	39	134	208	150	123	70	42	28	14	3	0	0
0	0	0	0	0	0	0	35	134	208	183	116	67	43	28	14	3	0	0	0
5	3	0	0	0	0	21	110	203	191	127	81	53	35	17	3	0	0	0	0
0	5	0	0	0	28	109	184	206	138	81	53	28	14	3	0	0	0	0	0
5	3	0	0	14	89	181	198	155	91	53	31	11	3	0	0	0	0	0	0
0	0	0	21	88	176	198	145	86	44	28	12	0	0	0	0	0	0	0	0
0	0	10	78	173	205	152	102	56	28	20	10	0	0	0	0	0	0	0	0
0	14	74	158	205	155	99	63	39	22	14	7	0	0	0	0	0	0	0	0
0	48	151	205	158	109	70	42	22	12	7	3	0	0	0	0	0	0	0	0
35	134	210	192	113	70	42	17	3	0	0	0	0	0	0	0	0	0	0	0
123	208	198	131	71	39	18	3	0	0	0	0	0	0	0	0	0	0	0	0
212	198	131	75	39	22	7	0	0	0	0	0	0	0	0	0	0	0	0	0
191	130	81	49	28	14	3	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabela 4.3: Zoom na imagem gradiente de Sobel

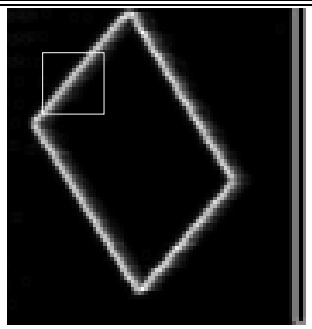
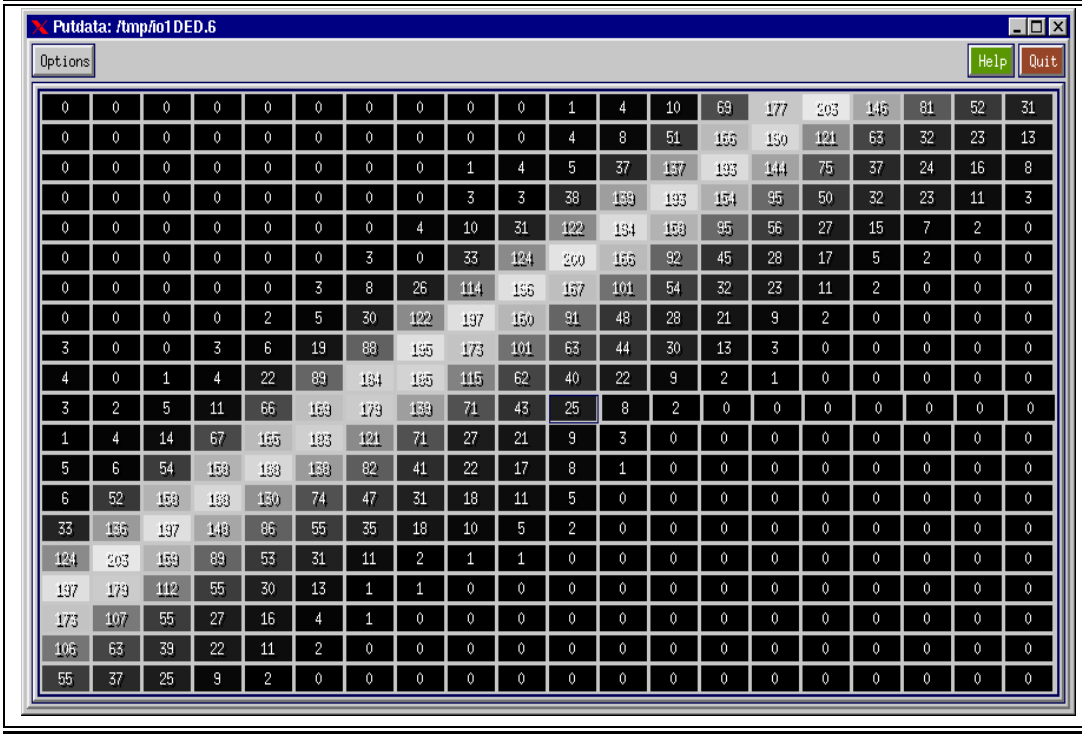
Spline

Zoom


Tabela 4.4: Zoom na imagem gradiente Spline

8. Leitura do nome do arquivo de saída.

Este programa algoritmo também é idêntico aos anteriores só que os filtro definido no programa é diferente, mas o processamento é o mesmo.

Vale salientar que este algoritmo, por não ter boa performance nos nossos propósitos, foi implementado apenas para enriquecimento de nosso trabalho.

4.1.3 Algoritmos de Segmentação

Segmentação por Rastreamento de Contorno em todas as Direções

O algoritmo que vamos descrever agora é um novo método de segmentação de imagens bidimensionais por rastreamento de contornos em todas as direções, proposto por Cortez e Carvalho, o qual compara cada pixel de uma determinada região com o pixel atual ou presente. Se um desses pixels da referida região é suficientemente similar ao pixel atual, então este pertence ao contorno. Neste novo método de extração de contornos é utilizada uma grande região nas vizinhaças do pixel que está sendo processado. Esta região depende da direção do contorno que está sendo seguido, como mostra a Figura 4.1 [13].

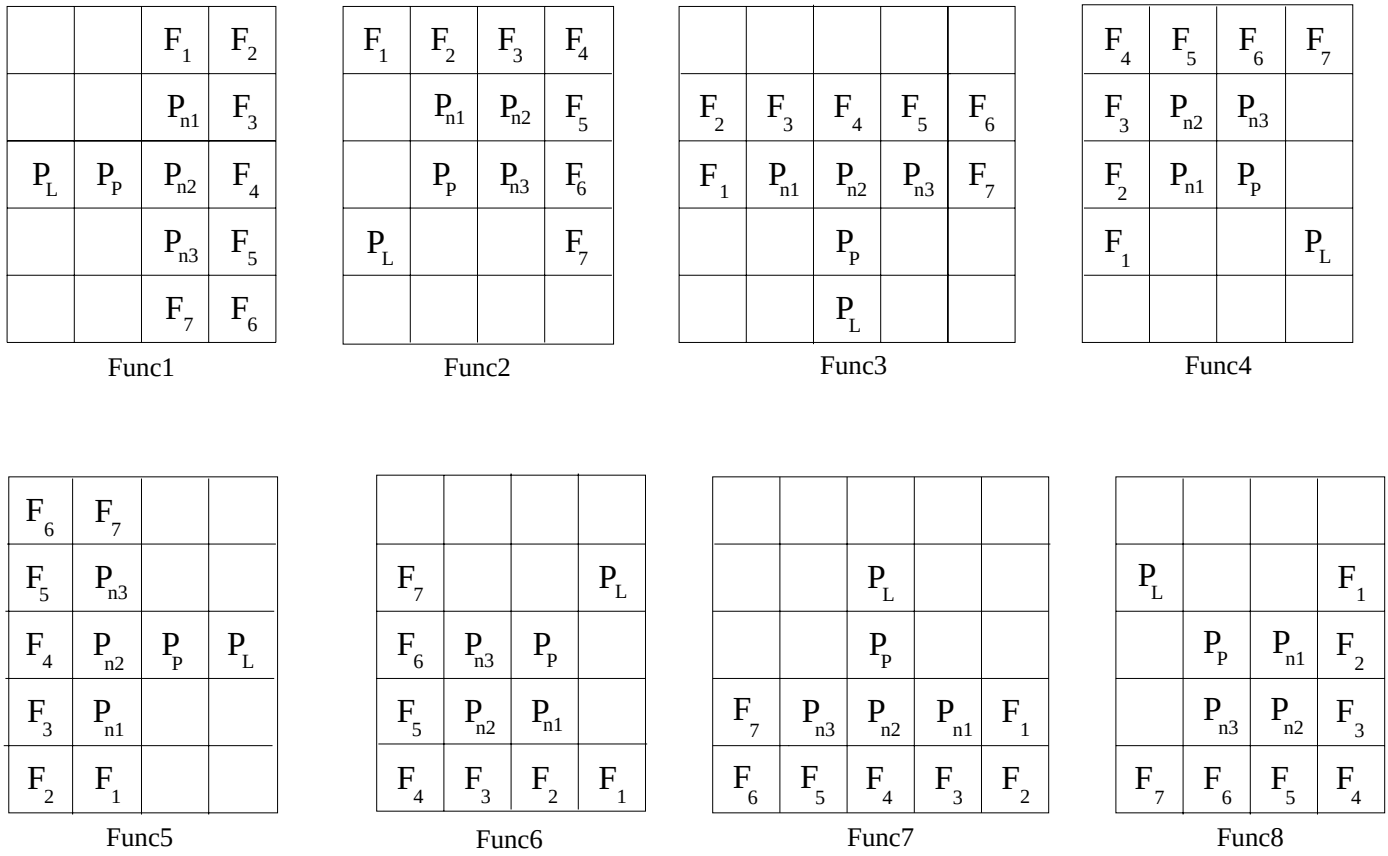


Figura 4.1: Regiões referentes a cada direção possível do contorno

Na Figura 4.1, temos que P_p *e* P_l são os atuais e últimos pixels processados, respecti-

vamente. A região de busca ao próximo pixel pertencente ao contorno é definida conforme for a direção de P_l *a* P_p. Os pixels candidatos ao contorno serão aqueles três vizinhos a P_p, que estão simetricamente distribuídos ao redor daquela direção. Podemos tomar como exemplo o caso Func1 da Figura 4.1, onde a direção de P_l *a* P_p é horizontal e o sentido da esquerda para a direita e ,consequentemente, o próximo pixel pertencente ao contorno pode ser tanto P_{n1} quanto P_{n2} , ou P_{n3}. Para cada um desses candidatos, considere agora os três pixels simetricamente vizinhos a estes, localizados na linha hipotética passando de P_p a P_{ni}, i=1,2,3. Na Figura 4.1 estes vizinhos são $N_1 = \{F_1, F_2, F_3\}$ para P_{n1}, $N_2 = \{F_3, F_4, F_5\}$ para P_{n2}, e $N_3 = \{F_5, F_6, F_7\}$ para P_{n3}. Seja S_i, i=1,2,3 , a soma dos níveis de cinza pertencentes a N_1, N_2 *e* N_3,respectivamente,e S_M seja o valor máximo das três somas.

Formalmente temos,

$$S_i = \sum_{j=0}^{2} F_{(2i+j-1)} \ , \ i = 1, 2, 3.$$

Dentre P_{n1}, P_{n2} *e* P_{n3} o pixel escolhido para ser o próximo pixel de contorno será aquele com o maior valor de nível de cinza, que também satisfaça a condição de que o valor correspondente S_i seja maior ou igual a 60% de S_M. São estas condições que garantem que o próximo pixel P_{n1}, P_{n2} *ou* P_{n3} pertença ao contorno atual. Se por um acaso a segunda das duas condições não for satisfeita, toma-se o pixel de valor de nível de cinza imediatamente menor e verifica-se a segunda condição. Se persistir de a segunda condição não ser satisfeita, toma-se o terceiro pixel candidato a pixel de contorno restante e verifica-se a segunda condição. Geralmente estas condições logo são satisfeitas, visto que este algoritmo já processa imagens gradiente e limiarizadas, mas se nenhum dos três P_{n1}, P_{n2} *e* P_{n3} satisfazem às condições o algoritmo termina.

Neste algoritmo, para cada direção possível existe uma região a para a qual utilizamos o mesmo processo descrito. Estas regiões estão ilustradas na Figura 4.1 com todos os referidos pixels também ilustrados ("*labels*") conforme for a direção.

Para o algoritmo descrito acima, usamos como ponto de partida o primeiro pixel com nível de cinza diferente de zero encontrado, utilizando o algoritmo de varredura da imagem de cima para baixo da esquerda para a direita. A condição de parada deste algoritmo acontece quando o mesmo processa algum dos pixels que já tenha sido processado, anteriormente.

Resultados Obtidos

Devido a este comportamento de "*previsão*" do próximo pixel de contorno na vizinhança imediata ao pixel atual, este processo de segmentação por extração de contornos apresentou boa operação, seguindo o contorno do objeto mesmo através de mudanças abruptas de direção.

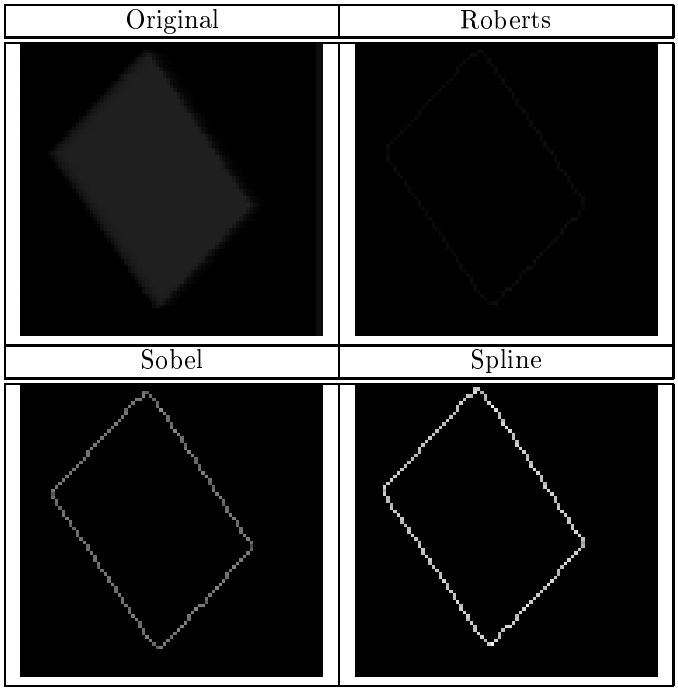


Tabela 4.5: Resultado do algoritmo para as imagens gradiente de Roberts, Sobel e Spline , respectivamente

O algoritmo descrito evita problemas como perder a direção do contorno (na busca dos pixels pertencentes ao contorno) ao processar imagens onde as mudanças de direção são abruptas e ao processar imagens com alto grau de ruído sem perder o caminho do contorno ou parar. O algoritmo está apto a reconhecer se o pixel diferente de zero (0) é pertencente ao contorno ou é apenas um pixel ruidoso isolado. Embora não tenhamos realizado nenhum teste com imagens ruidosas, é comprovado que existem estas características neste algoritmo[13].

Segmentação por Crescimento de Regiões segundo a Varredura

Como mencionado anteriormente, o crescimento de regiões pode ser feito segundo a varredura ou em todas as direções. No primeiro caso, a imagem é percorrida de cima para baixo e da esquerda para a direita. Cada ponto (x, y) é comparado com o pixel inicial escolhido pelo usuário. A finalidade deste tipo de crescimento de regiões é crescer todas as regiões da imagem que apresentem semelhança com o nível de cinza do pixel inicial. Diferentemente do método de crescimento segundo varredura descrito neste, que percorre a imagem comparando cada ponto (x, y) com os pontos que foram varridos. A comparação baseia-se na diferença entre o nível de cinza do pixel inicial com o nível de cinza do pixel que está sendo varrido no momento, (x, y). O módulo desta diferença tem que ser menor ou igual a um certo limiar L, para que o pixel que está sendo varrido pertença à região do pixel inicial. Aos pixels

pertencentes àquela região será atribuído um valor de nível de cinza particularizado, para que possamos distiguir bem dos pixels que estão nas demais regiões.

A desvantagem deste método é que ele só cresce as regiões com um tipo de característica, ou seja, regiões que tenham semelhança com o pixel inicial. Caso queira-se crescer regiões com outro tipo de característica temos a opção de executá-lo novamente.

Formalmente temos a seguinte condição para que o pixel atual pertença à referida região.

$$|I(x,y) - I(x_i,y_i)| \leq L \tag{4.1}$$

onde (x,y) é a posição do pixel atual; (x_i,y_i) é a posição do pixel inicial; L é o valor limiar das diferenças de níveis de cinza

Resultados Obtidos

Este algoritmo foi testado com as imagens mostradas na Tabela 4.6:

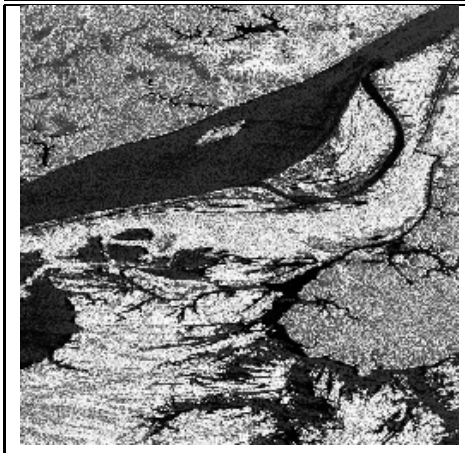
Original	Crescimento de regiões segundo a varredura
	
Original (Satélite)	Crescimento de Regiões segundo a varredura
	

Tabela 4.6: Resultados dos algoritmos de Crescimento de Regiões segundo a varredura

Como podemos observar, certas partes das imagens teste são detectadas como regiões de uma mesma característica, segundo os critérios apresentados.

Na primeira imagem teste não poderíamos tomar nenhuma conclusão objetiva, pois todas

as regiões destacáveis têm a mesma característica. Escolhemos, neste caso, um pixel inicial dentro de uma destas regiões para iniciar o processo. Mas, poderíamos também escolher um pixel localizado fora destas regiões, como mostrado na Tabela 4.7, o que resultaria numa inversão de níveis, como podemos verificar. Vale salientar que esta imagem, como veremos a seguir, será de grande importância para os testes com os algoritmos seguintes.

Na segunda imagem teste, temos uma separação de regiões de uma só característica, ou sejam, separação de regiões aquáticas. Esta é uma imagem satélite de parte de uma bacia, na qual também temos regiões de vegetação que foram detectadas por este algoritmo como mostra a tabela seguinte.

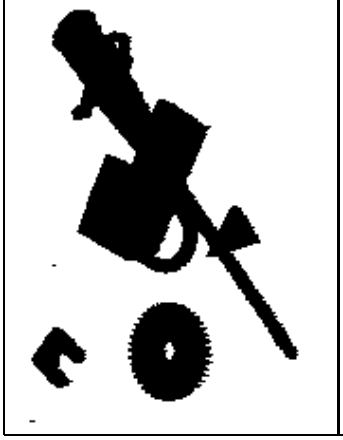
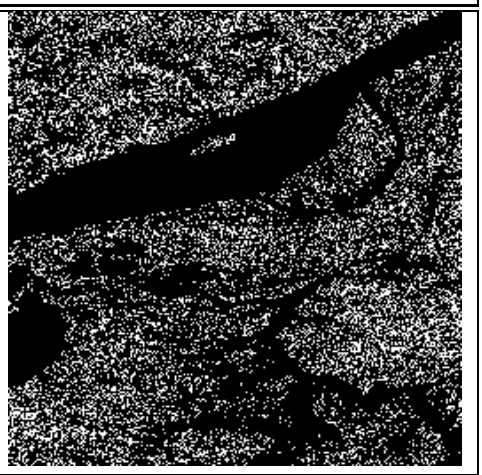
Teste-1	Teste-2
	

Tabela 4.7: Resultados dos algoritmos de Crescimento de Regiões segundo a varredura aplicado a outras regioes nas imagens de teste.

Como podemos observar esse algoritmo somente extrai regiões de uma caracterśtica. A seguir, descreveremos os algoritmos implementados para crescimento de regiões para outras aplicacões. Como exemplo, temos o problema de isolar uma região da imagem sem que as demais regiões com mesmas características desta sejam detectadas. Vale salientar que o algoritmo aqui descrito de crescimento de regiões é aplicado quando a finalidade é isolar todas as regiões com uma certa característica comum na imagem, enquanto os próximos algoritmos, a serem descritos, têm a finalidade de isolar somente uma região onde foi escolhido o pixel inicial, sem que outras regiões da mesma imagem que apresentem as mesmas características sejam crescidas.

Descrição do Algoritmo

1. Leitura do nome do arquivo da matriz (imagem) de entrada via teclado.
2. Leitura da dimensão da matriz (imagem) de entrada via teclado.
3. Leitura e armazenagem dos dados da matriz (imagem) de entrada.
4. Leitura das coordenadas do pixel inicial via teclado.
5. Leitura do valor de limiar para os testes (condições).
   - Optativo: Cálculo do valor médio de nível de cinza na janela 3x3 centralizada no pixel inicial.
6. Varredura da imagem fazendo os testes com cada pixel e decidindo se o pixel atual pertence ou não à referida região, além de testar a conectividade do pixel atual com algum dos pixels anexados a região.
7. Leitura do nome do arquivo de saída.
8. Armazenagem do arquivo de saída.

Dentre esses algoritmos, existe um que ao invés de comparar (testar) o nível de cinza do pixel atual com o nível de cinza do pixel inicial, compara com o valor médio de nível de cinza em uma janela 3x3 centralizada no pixel inicial (passo 5). Este também funciona muito bem. Foi implementado porque pode servir para uma aplicação específica, talvez com alguma modificação. Também foi implementado, a fim de enriquecer o nosso trabalho, um outro algoritmo que faz as comparações do valor de nível de cinza médio na janela 3x3 centralizada no pixel atual com o valor de nível de cinza médio da janela 3x3 centralizada no pixel inicial. Não obteve bons resultados para as nossas aplicações, mas pode ser útil para outras.

Crescimento de regiões em todas as direções

Quando falamos de crescimento de regiões automaticamente correlacionamos com o conceito de conectividade. Uma região em uma imagem nada mais é que um conjunto de pixels com uma mesma característica (semelhantes) conectados. Nesta secção vamos abordar algumas relações primitivas, mas ainda importantes, entre pixels em uma imagem digital. Como já foi mencionado, uma imagem é denotada por $f(x, y)$. Quando nos referimos a um pixel em particular, usamos letras minúsculas, assim como p e q. Um subconjunto de pixels de $f(x, y)$ é denotado por S. A seguir vamos descrever algumas relações que existem entre pixels de uma imagem digital.

Vizinhança de um pixel

Um pixel p nas coordenadas (x, y) tem quatro vizinhos na *horizontal* e *vertical* cujas coordenadas são dadas por

$$(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1)$$

Este conjunto de pixels, é denominado de vizinhança-4 de p, denotado por $N_4(p)$. Cada pixel está a uma unidade de distância de (x, y), e alguns dos vizinhos de p podem estar fora dos limites da imagem se (x, y) é um pixel pertencente a uma das bordas da imagem.

Os quatro pixels vizinhos de p nas *diagonais* têm coordenadas

$$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$$

e são denotados por $N_D(p)$. A união deste conjunto de pixels, com a vizinhança-4, é chamada de vizinhança-8 de p, denotado por $N_8(p)$. Como foi dito antes, estes pixels em $N_D(p)$ e $N_8(p)$ podem estar fora dos limites da imagem se (x, y) é um pixel pertencente a uma das bordas da imagem.

Conectividade

Conectividade entre pixels é um importante conceito usado para detectar contornos de objetos e componentes de uma região em uma imagem. Para estabelecer se dois pixels estão conectados, tem que ser determinado se eles, de alguma forma, são adjacentes (por exemplo, se eles estão numa vizinhança-4) e se seus valores de níveis de cinza satisfazem ao critério de similaridade (isto é, se eles são semelhantes). Como exemplo, em uma imagem binária na qual os valores dos pixels são 0 e 1, dois pixels podem ser adjacentes através de uma vizinhança-4, mas não são considerados conectados se não tiverem o mesmo valor.

Seja V um conjunto de valores de níveis de cinza usados para definir conectividade. Por exemplo, em uma imagem binária, $V = \{1\}$ para a conectividade de pixels com valor 1. Em uma imagem de níveis de cinza, para a conectividade de pixels com uma variação dos valores de intensidade de 32 a 64, isto significa que $V = \{32, 33..., 63, 64\}$. Consideramos três tipos de conectividade:

Conectividade-4 - Dois pixels p e q com valores de V são de conectividade-4 se q pertence ao conjunto $N_4(p)$.

Conectividade-8 - Dois pixels p e q com valores de V são de conectividade-8 se q pertence ao conjunto $N_8(p)$.

Conectividade-m (conectividade mista) - Dois pixels p e q com valores de V são de conectividade-m se

1. q pertence ao conjunto $N_4(p)$, ou
2. q pertence ao conjunto $N_D(p)$ e o conjunto $N_4(p) \cap N_4(q)$ é vazio. (Este é o conjunto de pixels que estão em uma vizinhança-4 de ambos e cujos valores pertencem a V.)

Conectividade mista é uma modificação da conectividade-8 e é introduzida com a finalidade de eliminar múltiplos caminhos de conecções que venham a ser feitas, as quais normalmente aumentam quando utilizamos a conectividade-8. Por exemplo, considere o arranjo de pixels mostrado na figura 4.2(a). Para $V = \{1\}$, os caminhos a serem seguidos entre a vizinhança-8 do pixel central são mostrados pelas linhas pontilhadas na Figura 4.2(b). Note a ambiguidade nos caminhos das conecções quando permitimos conectividade-8. Esta ambiguidade é eliminada usando conectividade-m, como mostra a Figura 4.2(c).

Um pixel p é *adjacente* a um pixel q se eles estão conectados. Podemos definir adjacência-4, -8, ou -m dependendo do tipo de conectividade especificada. Duas imagens subdivididas S_1 e S_2 são adjacentes se algum pixel de S_1 é adjacente a algum pixel em S_2.

Um *caminho* do pixel p com coordenadas (x, y) ao pixel q com coordenadas (s, t) é uma sequência de pixels distintos com coordenadas

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

onde $(x_0, y_0) = (x, y)$ e $(x_n, y_n) = (s, t)$, (x_i, y_i) é adjacente a $(x_{i-1}, y_{i-1}), 1 \leq i \leq n$, onde

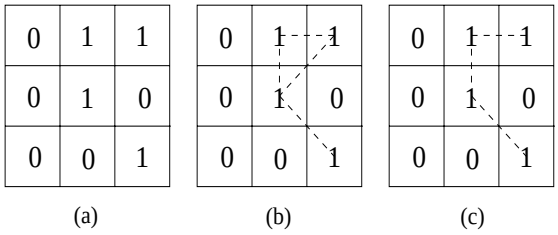


Figura 4.2: (a) Arranjo de pixels; (b) vizinhança-8 do pixel central; (c) vizinhança-m do mesmo pixel. As linhas pontilhadas são os caminhos a serem seguidos entre o pixel central e sua vizinhança.

n é o comprimento do caminho. Podemos definir caminhos-4, -8, ou -m dependendo do tipo de adjacência especificada. Se p e q são pixels de uma imagem subdividida S, então p é conectada a q em S se existe um caminho entre p e q para o qual todos os pixels deste pertençam a S.

Medidas de Distância

Para pixels p, q e z, com coordenadas (x, y), (s, t) e (u, v) respectivamente, D é a função distância ou métrica se

(a) $D(p, q) \geq 0$ $(D(p, q) = 0$ *se e somente se* $p = q)$,
(b) $D(p, q) = D(q, p)$, *e*
(c) $D(p, z) \leq D(p, q) + D(q, z)$.

A *distância Euclideana* entre p e q é definida como

$$D_e(p, q) = [(x - s)^2 + (y - t)^2]^{\frac{1}{2}} \tag{4.2}$$

Para esta medida de distância, os pixels que têm uma distância menor ou igual a um valor r de (x, y) são os pontos pertencentes a um círculo de raio r centralizado em (x, y).

A distância D_4 entre p e q é definida por

$$D_4(p, q) = |x - s| + |y - t| \tag{4.3}$$

Neste caso os pixels que têm uma distância D_4 de (x, y) menor ou igual a um valor r formam um lozango centralizado em (x, y). Por exemplo, os pixels com distância $D_4 \leq 2$ de (x, y) (ponto central) forma os seguintes contornos a uma distância constante:

$$\begin{array}{ccccc} & & 2 & & \\ & 2 & 1 & 2 & \\ 2 & 1 & 0 & 1 & 2 \\ & 2 & 1 & 2 & \\ & & 2 & & \end{array}$$

Os pixels com $D_4 = 1$ são da vizinhança-4 de (x, y).

A distância entre p e q é definida como

$$D_8 = máx(|x - s|, |y - t|) \tag{4.4}$$

Neste caso os pixels com distância D_8 de (x, y) menor ou igual a um valor r formam um quadrado centralizado em (x, y). Por exemplo, os pixels com distância $D_8 \leq 2$ de (x, y) (o ponto central) formam os seguintes contornos a uma distância constante. Os pixels com

$$\begin{array}{ccccc} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{array}$$

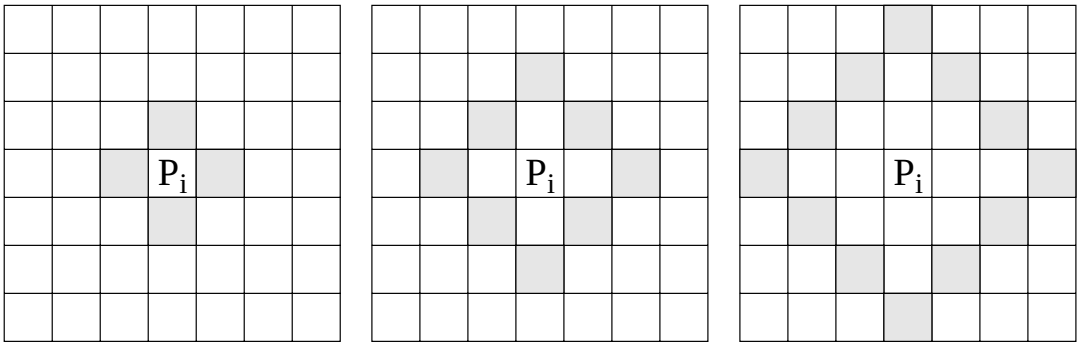
$D_8 = 1$ é a vizinhança-8 de (x, y).

A distância D_4 entre dois pontos p e q é igual ao comprimento do caminho mais curto entre esses dois pontos em uma vizinhança-4. O mesmo se aplica à D_8. De fato, nós podemos considerar ambas as distâncias D_4 e D_8 entre p e q se um caminho existe entre eles, pois as definições destas distâncias envolvem apenas as coordenadas destes pontos. Para a conectividade-m, entretanto, o valor da distância (comprimento do caminho) entre dois pixels depende dos valores dos pixels ao longo do caminho e daqueles de suas vizinhanças. Considere o seguinte arranjo de pixels e assuma que p, p_2, e p_4 têm valor 1 e que p_1 e p_3 podem ter valor 0 ou 1: se somente é permitida conectividade de pixels com valor 1, e p_1 e

$$\begin{array}{cc} & p_3 \quad p_4 \\ p_1 & p_2 \\ p & \end{array}$$

p_3 são 0, a distância-m entre p e p_4 é 2. Se tanto p_1 quanto p_2 têm valor 1, a distância é 3. Se p_1 e p_3 são 1, a distância é 4.

Esses conceitos aqui descritos de conectividade, vizinhança, caminhos, adjacência, e medição de distância, são de extrema importância para o entedimento análises automatizadas de imagens. Nas próximas secções nós vamos descrever algoritmos para os quais todos esses conceitos foram de muita importância durante nossas implementações.



(a) $D_4 = 1$ (b) $D_4 = 2$ (c) $D_4 = 2$

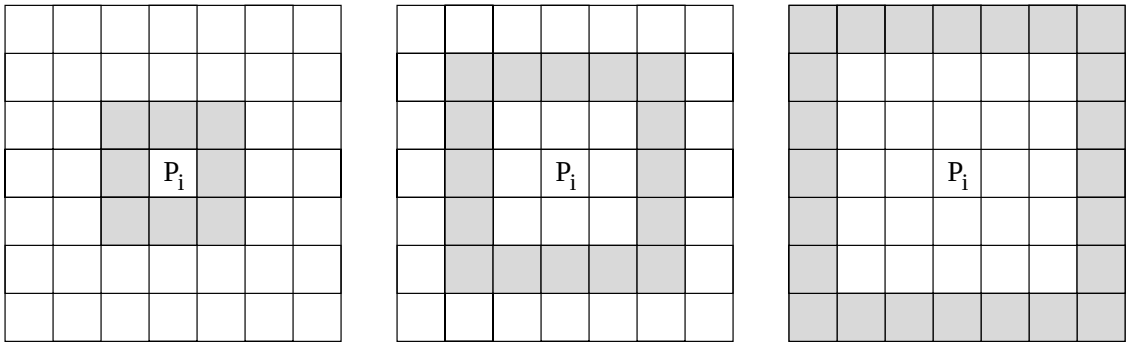
Figura 4.3: Crescimento de Regiões utilizando D_4, passo-a-passo.

Algoritmo de Crescimento de Regiões em todas as Direções utilizando distância D_4.

Esse algoritmo "*cresce*" regiões crescendo a distância D_4 a partir do pixel inicial. Primeiramente escolhemos e fornecemos as coordenadas do pixel inicial, ou seja, um ponto pertencente a uma região específica que deseja-se crescer. Como podemos ver na Figura 4.3, armazenado o valor de nível de cinza do pixel inicial, testamos se os pixels a uma distância $D_4 = 1$ obedecem à condição da diferença de níveis de cinza $|I(x, y) - I(x_i, y_i)| \leq L$. Na qual (x, y) é a posição do pixel atual (pertencente a vizinhança-4 do pixel inicial), (x_i, y_i) é a posição do pixel inicial, e L é o valor limiar das diferenças de níveis de cinza. Os pixels que obedecem à condição são adicionados a região. Feito isso, interativamente incrementamos a distância D_4, e novamente testamos se os pixels do lozango onde $D_4 = 2$ obedecem à condição, caso obedeçam são adicionados à região. E assim sucessivamente, até que chegue num ponto onde todos os pixels de uma certa distância constante ao pixel inicial, não pertençãm mais à região em questão, ou seja, até que todos os pixels a uma distância constante não obedeçam a condição.

Algoritmo de Crescimento de Regiões em todas as Direções utilizando distância D_8.

Este algoritmo tem todos os procedimentos e testes iguais aos do algoritmo utilizando D_4. A diferença está na maneira como é crescida a região. Ao invés de aumentarmos a D_4 aumentamos D_8. Como mostra a Figura 4.4.



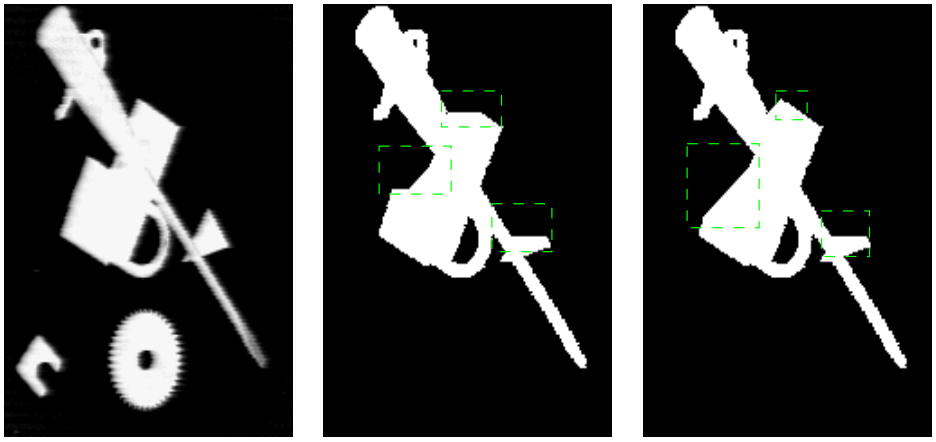
(a) $D_8 = 1$ (b) $D_8 = 2$ (c) $D_8 = 3$

Figura 4.4: Crescimento de Regiões utilizando D_8, passo-a-passo.

O propósito de se ter as duas maneiras de crescer a região é que, para algumas imagens, é melhor trabalhar com um tipo de crescimento do que com o outro. Ou seja a decisão de se usar ou um ou outro depende da região a ser crescida.

Resultados Obtidos

Como podemos observar, este algoritmo não cresce perfeitamente a região como nós desejamos. Nas imagens crescidas na Figura 4.5 tanto com o crescimento empregando D_8 quanto com D_4 ocorreram problemas no crescimento de certas partes da região (enfatizadas na Figura 4.5).



(a) Original (b) D_4 (c) D_8

Figura 4.5: Visualização dos problemas ocorridos nos algoritmos de crescimento de regiões em todas as direções empregando D_4 e D_8.

Isto se deve ao fato de que quando o lozango ou o quadrado (empregando D_4 ou D_8, respectivamente) começa a crescer, aumentando as respectivas distâncias, algumas partes dos mesmos tocam outras partes que fazem parte da mesma região a ser crescida. O programa testa se estes pixels, nesta parte da região, estão conectados com pixels que já estão anexados à região. A resposta é "FALSO" pois para o algoritmo esta parte é vista como pertencendo a outra região de mesma característica, mas desconectada da atual.

Vale salientar ainda, que a qualidade do crescimento de regiões com estes métodos depende muito da posição do pixel inicial na região a ser crescida. Quanto mais irregular a forma da região a ser crescida maiores serão as falhas do algoritmo. Por isso não realizamos os testes deste algoritmo na imagem teste 2 (satélite), pois os resultados não foram satisfatórios.

Na próxima secção vamos descrever um algoritmo de crescimento de regiões que diverge um pouco do estilo dos algoritmos de crescimento de regiões já descritos. Os testes e procedimentos são similares, mas os resultados são melhores.

Algoritmo de Crescimento de Regiões em todas as Direções.

Assim como em todos os algoritmos de crescimento de regiões, é escolhido primeiramente o pixel inicial, de modo que este esteja dentro da região que se deseja crescer. Fornecemos então as coordenadas deste pixel e o algoritmo começa a processar da seguinte maneira:

1. Testa-se com todos os pixels de sua vizinhaça-8 a condição de diferença de nível de cinza mencionada anteriormente.

2. Testa se os pixels na vizinhança-8 que satisfazem a condição, já estão na região. Como mostra a Figura 4.6, os pixels da vizinhança-8 do pixel que está sendo processado podem já estar anexados à região.

3. Se os pixels da vizinhança-8 do pixel que está sendo processado satisfizerem a condição e não estiverem anexado à região, então as coordenadas destes serão armazenadas para um teste posterior. Seguindo todos os passos, ou seja, passo 1, 2, e 3

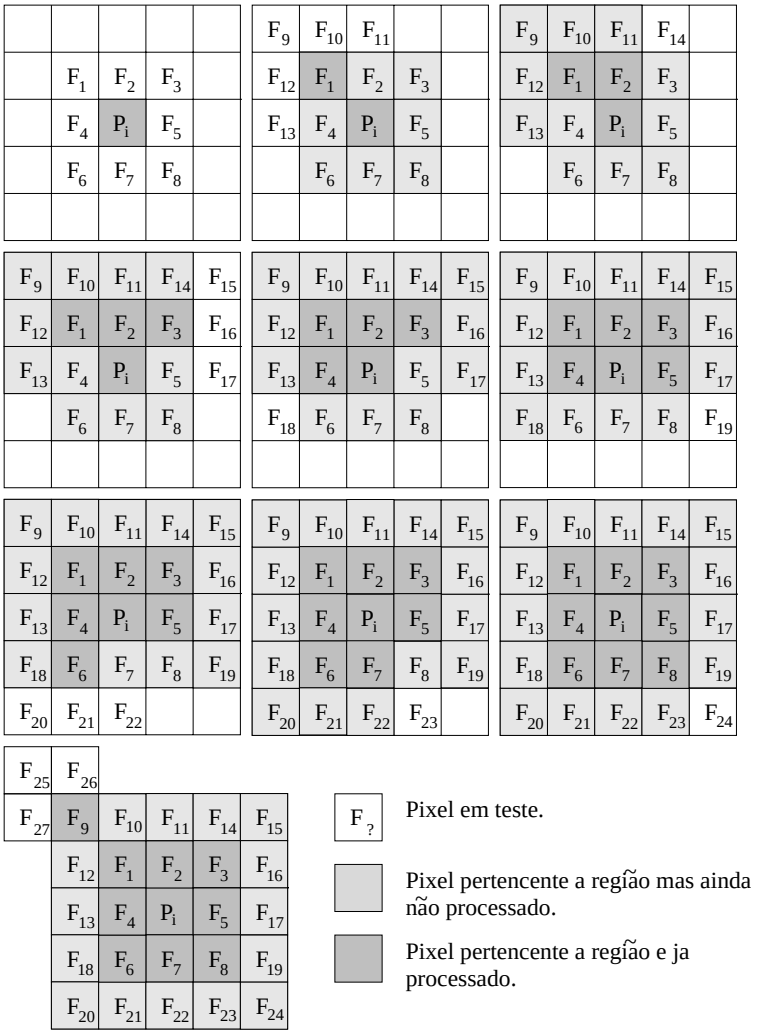


Figura 4.6: Crescimento de Regiões em todas as direções passo-a-passo.

O critério de parada deste algoritmo é a ausência de pixels a serem testados, isto é, quando todos os pixels de uma região específica já tiverem sido crescidos ou anexados á região em questão.

Além disso, este algoritmo tem a opção de crescer vária regiões numa mesma imagem, diferenciando-as com valores de nível de cinza distintos, como mostra a Figura 4.1.3.

Resultados Obtidos

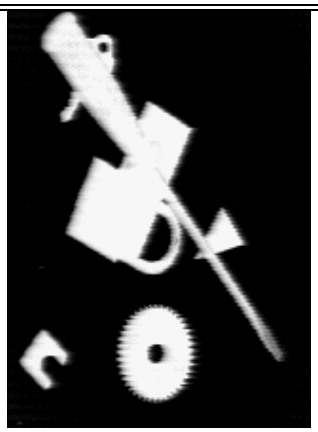
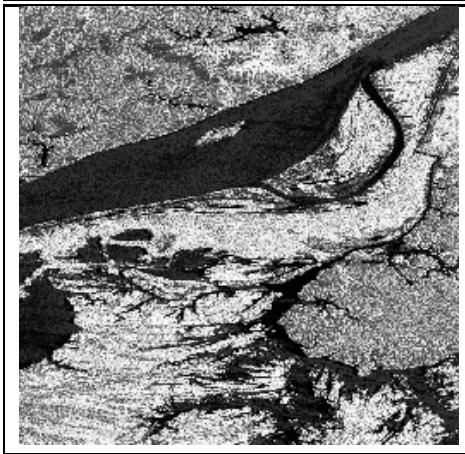
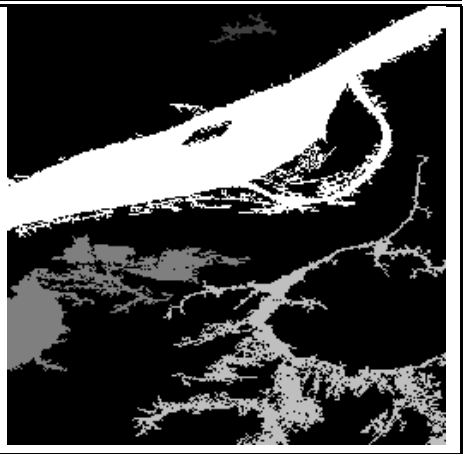
Original (teste 1)	Teste 1 processada
	
Original (Satélite - Teste 2)	Teste 2 processada
	

Tabela 4.8: Resultados do algoritmos de Crescimento de Regiões em todas as direções

Como podemos observar, este tipo de crescimento de regiões é muito mais robusto e obteve melhores resultados que os métodos de crescimento segundo a varredura e crescimento empregando D_4 e D_8 aqui descritos. Podemos observar que seus resultados independem da localização do ponto inicial escolhido dentro da região que se quer crescer, além de eliminarmos aqueles problemas mencionados nos algoritmos anteriores. Visível e teoricamente obtivemos bom êxito com todas as imagens testadas.

Além disso, esse algoritmo identifica cada região crescida com um nível de cinza diferente, facilitando, portanto, a visualização destas.

Capítulo 5

Conclusões

À luz do exposto, podemos afirmar que o estudo , seleção e implementação de algoritmos de segmentação de imagens foram feitos, conforme consta em nosso plano de trabalho. Não foram estudados e implementados os algoritmos de segmentação utilizando Classificadores Bayesianos, Redes Neurais e Operadores Morfológicos, pois o tempo de implementação e testes dos algoritmos estudados teve duração maior que a prevista inicialmente no plano de trabalho.

Além disso , todos os algoritmos implementados estão devidamente documentados e comentados, para uma posterior compreensão e utilização destes em trabalhos futuros. A seguir, temos um manual de utilização de todos os algoritmos desenvolvidos nesta bolsa de iniciação científica.

Apêndice A

Manual de Utilização dos Algoritmos de Detecção de Bordas e Segmentação

Antes de mais nada, comunicamos que todos estes algoritmos aqui comentados estão em *Linguagem C* padrão e foram testados e compilados em compiladores *cc e gcc* do Sistema Linux. Esses compiladores geram arquivos executáveis **.out** a partir dos arquivos de código fonte. Para executar qualquer destes arquivos temos que digitar na linha de comando o seguinte: `nome-do-arquivo.out`. Onde geralmente este nome-do-arquivo é o nome da aplicação do algoritmo, a fim de facilitar a utilização. Por exemplo, para executar o algoritmo de gradiente de Roberts na secção seguinte colocamos na linha de comando roberts.out e o programa é executado.

A.1 Algoritmo do Gradiente de Roberts.

Nome do arquivo fonte: roberts.c
Nome do arquivo executável: roberts.out

Ao executar o programa teremos:

Primeira Parte

```
Digite o nome do arquivo da imagem de entrada >>
```

Nesta parte, digite o caminho e nome do arquivo onde se encontra a matriz de dados (valores de níveis de cinza) da imagem de entrada da qual se deseja obter a imagem gradiente. Em seguida teremos:

Segunda Parte

```
Qual a dimensão MxN da matriz (imagem) de entrada >>
Digite o valor de M:
Digite o valor de N:
```

Nesta parte, teremos que informar ao programa a largura (M) e altura (N) da imagem de entrada, em pixels. Depois que o programa armazena os dados que estão no arquivo de entrada, automaticamente começa o processo de filtragem com os dois filtros de Roberts. Ao

final do processamento o programa vai solicitar o nome do arquivo em que o usuário deseja armazenar a imagem gradiente de Roberts, da seguinte forma:

Terceira Parte
```
Digite o nome do arquivo da imagem de saída >>
```

Daí então, digita-se o nome do arquivo de dados de saída e o programa termina.

A.2 Algoritmo do Gradiente de Sobel.

Nome do arquivo fonte: sobel.c
Nome do arquivo executável: sobel.out

Ao executar o programa teremos:

Primeira Parte
```
Digite o nome do arquivo da matriz da imagem de entrada >>
```

Nesta parte, digite o caminho e nome do arquivo onde se encontra a matriz de dados (valores de níveis de cinza) da imagem de entrada da qual se deseja obter a imagem gradiente. Em seguida teremos:

Segunda Parte
```
Qual a dimensão MxN da imagem de entrada >>
Digite o valor de M:
Digite o valor de N:
```

Nesta parte, teremos que informar ao programa a largura (M) e altura (N) da imagem de entrada, em pixels. Depois que o programa armazena os dados que estão no arquivo de entrada, automaticamente começa o processo de filtragem com os dois filtros de Sobel. Ao final do processamento o programa vai solicitar o nome do arquivo em que o usuário deseja armazenar a imagem gradiente de Sobel, da seguinte forma:

Terceira Parte
```
Digite o nome do arquivo da imagem de saída >>
```

Daí então, digita-se o nome do arquivo de dados de saída e o programa termina.

A.3 Algoritmo do Gradiente Spline.

Nome do arquivo fonte: spline.c
Nome do arquivo executável: spline.out

Ao executar o programa teremos:

Primeira Parte

```
Digite o nome do arquivo da matriz da imagem de entrada >>
```

Nesta parte, digite o caminho e nome do arquivo onde se encontra a matriz de dados (valores de níveis de cinza) da imagem de entrada da qual se deseja obter a imagem gradiente. Em seguida teremos:

Segunda Parte

```
Qual a dimensão MxN da imagem de entrada >>
Digite o valor de M:
Digite o valor de N:
```

Nesta parte, teremos que informar ao programa a largura (M) e altura (N) da imagem de entrada, em pixels. Depois que o programa armazena os dados que estão no arquivo de entrada, automaticamente começa o processo de filtragem com os quatro filtros de Spline. Ao final do processamento o programa vai solicitar o nome do arquivo em que o usuário deseja armazenar a imagem gradiente Spline, da seguinte forma:

Terceira Parte

```
Digite o nome do arquivo da imagem de saída >>
```

Daí então, digita-se o nome do arquivo de dados de saída e o programa termina.

A.4 Algoritmo do Gradiente Laplaciano.

Nome do arquivo fonte: laplaciano.c
Nome do arquivo executável: laplaciano.out

Ao executar o programa teremos:

Primeira Parte

```
Digite o nome do arquivo da matriz da imagem de entrada >>
```

Nesta parte, digite o caminho e nome do arquivo onde se encontra a matriz de dados (valores de níveis de cinza) da imagem de entrada da qual se deseja obter a imagem gradiente.

Em seguida teremos:

Segunda Parte
```
Qual a dimensão MxN da imagem de entrada >>
Digite o valor de M:
Digite o valor de N:
```

Neste caso, teremos que informar ao programa a largura (M) e altura (N) da imagem de entrada, em pixels. Depois que o programa armazena os dados que estão no arquivo de entrada, automaticamente começa o processo de filtragem com o filtro do Laplaciano. Ao final do processamento o programa vai solicitar o nome do arquivo em que o usuário deseja armazenar a imagem gradiente de Laplaciano, da seguinte forma:

Terceira Parte
```
Digite o nome do arquivo da imagem de saída >>
```

Daí então, digita-se o nome do arquivo de dados de saída e o programa termina.

A.5 Algoritmo de Limiarização ("Threshold").

Nome do arquivo fonte: limiar.c
Nome do arquivo executável: limiar.out

Ao executar o programa teremos:

Primeira Parte
```
Digite o nome do arquivo da matriz da imagem de entrada >>
```

Nesta parte, digite o caminho e nome do arquivo onde se encontra a matriz de dados (valores de níveis de cinza) da imagem de entrada da qual se deseja obter a imagem limiarizada. Em seguida teremos:

Segunda Parte
```
Qual a dimensão MxN da imagem de entrada >>
Digite o valor de M:
Digite o valor de N:
```

Neste caso, teremos que informar ao programa a largura (M) e altura (N) da imagem de entrada, em pixels. Depois que o programa armazena os dados que estão no arquivo de entrada, teremos:

Terceira parte
```
Digite o valor de limiar (ou ‘‘threshold’’) de nível de cinza >>
```

onde informamos o valor de nível de cinza limiar da imagem. Automaticamente começa o processo de limiarização da imagem (processo de "threshold"). Ao final do processamento

o programa vai solicitar o nome do arquivo em que o usuário deseja armazenar a imagem limiarizada, da seguinte forma:

Quarta Parte
```
Digite o nome do arquivo da imagem de saída >>
```

Daí então, digita-se o nome do arquivo de dados de saída e o programa termina.

A.6 Algoritmo do Rastreamento de Contorno em todas as direções.

Nome do arquivo fonte: tracking.c
Nome do arquivo executável: tracking.out

Ao executar o programa teremos:

Primeira Parte
```
Digite o nome do arquivo da matriz da imagem de entrada >>
```

Nesta parte, digite o caminho e nome do arquivo onde se encontra a matriz de dados (valores de níveis de cinza) da imagem de entrada da qual se deseja rastrear contornos. Em seguida teremos:

Segunda Parte
```
Qual a dimensão MxN da imagem de entrada >>
Digite o valor de M:
Digite o valor de N:
```

Neste caso, teremos que informar ao programa a largura (M) e altura (N) da imagem de entrada, em pixels. Depois que o programa armazena os dados que estão no arquivo de entrada, automaticamente começa o processo de rastreamento de contornos em todas as direções de objetos 2D. Ao final do processamento o programa vai solicitar o nome do arquivo em que o usuário deseja armazenar a imagem gradiente Spline, da seguinte forma:

Terceira Parte
```
Digite o nome do arquivo da imagem de saída >>
```

Daí então, digita-se o nome do arquivo de dados de saída e o programa termina.

A.7 Algoritmo do Crescimento de Regiões segundo a varredura.

Nome do arquivo fonte: regioes1.c
Nome do arquivo executável: regioes.out

Ao executar o programa teremos:

Primeira Parte

```
Digite o nome do arquivo da matriz da imagem de entrada >>
```

Nesta parte, digite o caminho e nome do arquivo onde se encontra a matriz de dados (valores de níveis de cinza) da imagem de entrada da qual se deseja crescer regiões. Em seguida teremos:

Segunda Parte

```
Qual a dimensão MxN da imagem de entrada >>
Digite o valor de M:
Digite o valor de N:
```

Neste caso, teremos que informar ao programa a largura (M) e altura (N) da imagem de entrada, em pixels. Em seguida temos:

Terceira Parte

```
Digite o ponto inicial do crescimento de regiao >>
x=
y=
```

Agora teremos que informar um ponto dentro de uma região na imagem, o qual tem características das regiões que queremos crescer.

Quarta Parte

```
Digite o valor de nivel de cinza limiar para o calculo das diferencas >>
```

Nesta parte, é preciso informar ao programa qual o valor de nível de cinza limiar para definirmos o crecimento de regiões. Este valor de limiar depende da resolução da imagem, ou seja, do número de níveis de cinza da imagem e da qualidade do crescimento de regiões que se quer obter. Depois que o programa armazena os dados que estão no arquivo de entrada, automaticamente começa o processo de crescimento de regiões segundo a varredura. Ao final do processamento o programa vai solicitar o nome do arquivo em que o usuário deseja armazenar a imagem processada, da seguinte forma:

Quinta Parte

```
Digite o nome do arquivo da imagem de saída >>
```

Daí então, digita-se o nome do arquivo de dados de saída e o programa termina.

Observação:

Para este tipo de algoritmo existem 3 versões diferentes que podem ser úteis para aplicações variadas.

Versão 1 - Esta versão simplesmente compara o valor de nível de cinza do pixel inicial com o valor de nível de cinza do pixel que está sendo processado.

Nome do arquivo fonte: regioes1.c
Nome do arquivo executável: regioes1.out

Versão 2 - Esta versão compara o valor médio de nível de cinza dos pixels da janela 3x3 centralizada no pixel inicial com o valor de nível e cinza do pixel que está sendo processado.

Nome do arquivo fonte: regioes1i.c
Nome do arquivo executável: regioes1i.out

Versão 3 - Esta versão compara o valor médio de nível de cinza dos pixels da janela 3x3 centralizada no pixel inicial com o valor médio de nível de cinza dos pixels da janela 3x3 centralizada no pixel que está sendo processado.

Nome do arquivo fonte: regioes1ii.c
Nome do arquivo executável: regioes1ii.out

A.8 Algoritmo do Crescimento de Regiões em todas as direções empregando Distâncias D_8 e D_4.

Nome dos respectivos arquivos fonte: regioes5.c, regioes5i.c
Nome dos respectivos arquivos executáveis: regioes5.out, regioes5i.out

Ao executar o programa teremos:

Primeira Parte

```
Digite o nome do arquivo da matriz da imagem de entrada >>
```

Nesta parte, digite o caminho e nome do arquivo onde se encontra a matriz de dados (valores de níveis de cinza) da imagem de entrada da qual se deseja crescer regiões. Em seguida teremos:

Segunda Parte

```
Qual a dimensão MxN da imagem de entrada >>
Digite o valor de M:
Digite o valor de N:
```

Neste caso, teremos que informar ao programa a largura (M) e altura (N) da imagem de entrada, em pixels.

Em seguida teremos:

Terceira Parte

```
Forneca o ponto inicial (x,y) do crescimento da regiao 1 >>
x=
y=
```

Agora teremos que informar um ponto dentro de uma região na imagem que queremos crescer.

Quarta Parte

```
Digite o valor de nivel de cinza limiar para o calculo das diferencas >>
```

Nesta parte, é preciso informar ao programa qual o valor de nível de cinza limiar para definirmos o crecimento de regiões. Este valor de limiar depende da resolução da imagem, ou seja, do número de níveis de cinza da imagem e da qualidade do crescimento de regiões que se quer obter. Depois que o programa armazena os dados que estão no arquivo de entrada, automaticamente começa o processo de crescimento de regiões segundo a varredura. Ao final do processamento o programa vai solicitar o nome do arquivo em que o usuário deseja armazenar a imagem processada, da seguinte forma:

Quinta Parte

```
Digite o nome do arquivo da imagem de saída >>
```

Daí então, digita-se o nome do arquivo de dados de saída e o programa termina. Depois que o programa termina o processo de crescimento de regiões, tem-se a opção de executá-lo novamente, como mostrado a seguir:

```
A regiao, cujo ponto inicial de crescimento foi escolhido anteriormente,ja foi
processada com o nome de arquivo de saida desejado.

Voce deseja crescimento de mais alguma regiao da imagem de entrada dada?(S/N)?
```

Então, se o usuário desejar crescer uma outra região que não seja aquela anteriormente crescida basta digitar **S**, caso contrário **N**.

A.9 Algoritmo do Crescimento de Regiões em todas as direções.

Nome do arquivo fonte: regioes10i.c
Nome do arquivo executável: regioes10i.out

Ao executar o programa teremos:

Primeira Parte

```
Digite o nome do arquivo da matriz da imagem de entrada >>
```

Nesta parte, digite o caminho e nome do arquivo onde se encontra a matriz de dados (valores de níveis de cinza) da imagem de entrada da qual se deseja crescer regiões. Em seguida teremos:

Segunda Parte

```
Qual a dimensão MxN da imagem de entrada >>
Digite o valor de M:
Digite o valor de N:
```

Neste caso, teremos que informar ao programa a largura (M) e altura (N) da imagem de entrada, em pixels.Em seguida teremos:

Terceira Parte

```
Quantas regioes existem na imagem de entrada ?  >>
```

Como este programa tem a opção de crescer várias regiões ao mesmo tempo ele perguntas quantas regiões desejamos crescer dentro da mesma imagem de entrada. Consequentemente, teremos que informar para cada região a ser crescida as coordenadas dos respectivos pontos iniciais. Portanto teremos:

Quarta Parte

```
Digite o ponto inicial do crescimento da regiao 1 >>
x=
y=
```

Agora teremos que informar as coordenadas de um ponto dentro da região 1 na imagem. Em seguida teremos:

Quinta Parte

```
Digite o valor de limiar para o calculo das diferencas dos niveis de cinza nesta
regiao >>
```

Nesta parte, é preciso informar ao programa qual o valor de nível de cinza limiar para definirmos o crecimento de região. Este valor de limiar depende da resolução da imagem, ou seja, do número de níveis de cinza da imagem e da qualidade do crescimento de região que se quer obter.

Quarta Parte

```
Digite o ponto inicial do crescimento da regiao 2 >>
x=
y=
```

Agora teremos que informar as coordenadas de um ponto dentro da região 2 na imagem. Em seguida teremos:

Quinta Parte

```
Digite o valor de limiar para o calculo das diferencas dos niveis de cinza nesta
regiao >>
```

A quarta e quinta parte do programa são repetidas até a quantidade de regiões especificadas no inicio se esgotar.

Depois que o programa armazena os dados que estão no arquivo de entrada, automaticamente começa o processo de crescimento de regiões em todas as direções. Ao final do processamento o programa vai solicitar o nome do arquivo em que o usuário deseja armazenar a imagem processada, da seguinte forma:

Sexta Parte

```
Digite o nome do arquivo da imagem de saída >>
```

Daí então, digita-se o nome do arquivo de dados de saída e o programa termina.

Referências Bibliográficas

[1] Raphael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Addison-Wesley, New York, 1992.

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

[3] William K. Pratt. *Digital Image Processing.* Addison-Wesley, New York, Julho 1989.

[4] V. V. Mizrahi. *Treinamento em linguagem C, módulo 1 e 2.* Makron Books, Brasil, 1990.

[5] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision,*Addison-Wesley, California, 1993.

[6] Fred Herique de Souza Paes. Circuito Integrado para Reconhecimento de Formas. *Relatório Final de Iniciação Científica,* 1996. Período: 08/95 a 07/96.

[7] Eurico Barreto Sprakel. Aplicações de Redes Neurais em Análise de Imagens. *Relatório Final de Iniciação Científica* 1996. Período: 08/95 a 08/96.

[8] José Josemar de Oliveira Jr. Sistema de Processamento de Imagens para Fins Didático Científico: Estudo, Seleção e Implementação de Algoritmos. Relatório Final de Iniciação Científica, 1997. Período: 02/97 a 08/97.

[9] Michel Goossens, Frank Mittelbach e Alexander Samarin. *The Latex Companion.* Addison-Wesley, Geneva-Suíça / Mainz-Alemanha, 1994.

[10] Department of Computing Services. *Introduction to Latex.* University of Waterloo, Ontario , 1991.

[11] Renato Fernandes Cantão, Ricardo Marchetti, Ricardo Biloti e Ricardo Akira Azana. *Latex Tex* , OTMMA Jr. Otimização e Modelagem Matemática, 1994.

[12] Elmar U. K. Melcher, João Marques de Carvalho, Habib Mehez, Nicolas Vaucher,Alain Hoele, Lírida Naviner, Jean François Naviner, Marcos de Morais, Ricardo A. S. Moreira, Fred Henrique Souza Paes, Valteir R. da Silva. *A Novel Gradient Operator Suited for VLSI Implementation of 2D Shape Recognation , I Workshop Brasileiro de HW/SW Codesign - IX SBCCI . .*

[13] Paulo Cézar Cotez e João Marques de Carvalho, *Look-ahead Boundary Tracking Segmentation of 2D Contours. ELECTRO'95 : XI Congresso Chileno de Ingenieria Electrica,* Univesidad de Magalannes, Novembro de 1995.

João Marques de Carvalho. (Professor orientador)

Sérgio Ferreira de Brito. (Bolsista)