unesp UNIVERSIDADE ESTADUAL PAULISTA
Faculdade de Ciências e Tecnologia
Programa de Pós-Graduação em Ciências Cartográficas

TÉCNICAS PARA A EXTRAÇÃO AUTOMÁTICA DE FEIÇÕES RETAS EM IMAGENS DIGITAIS

Almir Olivette Artero

Orientador: Prof. Dr. Antonio Maria Garcia Tommaselli

Dissertação apresentada ao Programa de Pós-Graduação em Ciências Cartográficas para a obtenção do Título de Mestre em Ciências pela Faculdade de Ciências e Tecnologia da Universidade Estadual Paulista.

Presidente Prudente
1999

unesp
UNIVERSIDADE ESTADUAL PAULISTA
CAMPUS DE PRESIDENTE PRUDENTE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

ALMIR OLIVETTE ARTERO

TÉCNICAS PARA A EXTRAÇÃO AUTOMÁTICA DE FEIÇÕES RETAS EM IMAGENS DIGITAIS

Dissertação apresentada ao Curso de Pós-Graduação em Ciências Cartográficas da Faculdade de Ciências e Tecnologia da UNESP, como requisito parcial para a obtenção do título de Mestre em Ciências Cartográficas (Área de Concentração: Aquisição, Análise e Representação de Informações Espaciais).

Orientador: Prof. Dr. *Antonio Maria Garcia Tommaselli*

PRESIDENTE PRUDENTE

1999

Ficha catalográfica

Artero, Almir Olivette
A825t Técnicas para a extração automática de feições retas em imagens digitais / Almir Olivette Artero.-- Presidente Prudente:UNESP/FCT, 1999.
xvi, 117p. : il., 30cm.
Dissertação (Mestrado).-- UNESP, Faculdade de Ciências e Tecnologia, Presidente Prudente, 1999.

1. Processamento digital de imagens. 2. Detecção automática de linhas retas. 3. Segmentação. 4. Extração de feições. 5. Fotogrametria Digital. I.Título.
CDD(18 ed.) 519.7

TERMO DE APROVAÇÃO

ALMIR OLIVETTE ARTERO

TÉCNICAS PARA A EXTRAÇÃO AUTOMÁTICA DE FEIÇÕES RETAS EM IMAGENS DIGITAIS

Comissão Julgadora

Dissertação para a obtenção do título de Mestre

Presidente e Orientador. . . . . .

2° Examinador. . . . . .

3° Examinador. . . . . .

Ao meu pai Francisco Artero, minha mãe Helena e minha irmã Angélica, por estarem sempre ao meu lado e também ao Arquimedes por me distrair.

AGRADECIMENTOS

Agradeço ao meu orientador Professor Antonio M. G. Tommaselli, por compartilhar suas idéias e pesquisas, me orientando em todas as etapas deste trabalho, e me recebendo em sua casa, abrindo mão de suas horas de descanso.

- Aos meus professores Aluir Porfirio Dal Poz, Arlete A. C. Meneguette, Erivaldo Antonio da Silva, João Fernando C. da Silva, João Francisco Galera Monico, Júlio K. Hasegawa, Maurício Galo, Messias Meneguete Júnior, Nilton Nobuhiro Imai, por me transmitirem o conhecimento e pela amizade;
- Ao professor Neri Alves, pela sua amizade, sempre me incentivando a continuar;
- À Faculdade de Ciências e Tecnologia – Unesp;
- Aos colegas de curso;
- Aos funcionários do Campus;
- Aos amigos da Telefonica;

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

ARTERO, A. O. Técnicas para a extração automática de feições retas em imagens digitais. Presidente Prudente, 1999. 117p. Dissertação de Mestrado – Faculdade de Ciências e Tecnologia, Campus de Presidente Prudente, Universidade Estadual Paulista.

RESUMO

Este trabalho apresenta um estudo e a implementação de algumas técnicas utilizadas para a extração de feições retas em imagens digitais. Os resultados obtidos com a aplicação de algumas das técnicas estudadas são avaliados por meio de uma imagem-padrão. Além da análise individual dos resultados obtidos pelas técnicas utilizadas em cada etapa, foram avaliados também os resultados obtidos com diferentes combinações das técnicas nas diversas etapas do processo. O mesmo processamento foi também aplicado a um detalhe de uma imagem aérea. O sistema desenvolvido foi implementado na linguagem de programação C++, sendo utilizado o ambiente de programação C++ Builder.

Palavras-chaves: Detecção Automática de Linhas Retas; Extração de Feições; Segmentação; Processamento Digital de Imagens; Fotogrametria Digital.

ABSTRACT

This work presents a study and the implementation of some techniques used for the extraction of straight features in digital images. The results obtained with the application of some of the studied techniques are assessed by means of an standard image. Besides the individual analysis of the results obtained by the techniques used in each stage, the results obtained with different combinations of the techniques in the several stages of the process were also appraised. The same processing was also applied to a detail of an aerial image. The developed system was implemented in the programming language C++, being used the programming environment C++ Builder.

Keywords: Automatic Detection of Straight Lines; Features Extraction; Segmentation; Digital Image Processing; Digital Photogrammetry.

CAPITULO 1

INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

Acredita-se que um terço do processamento cerebral seja utilizado no processo visual, e a forma pela qual as pessoas analisam as cenas e conseguem identificar objetos presentes nelas, mesmo quando tais objetos se apresentam parcialmente obstruídos, ou ainda com algumas diferenças de escala (devido a um afastamento do observador), rotação ou translação, não é um processo de fácil compreensão. A disponibilidade de *hardware* (computadores para o processamento e também dos meios para a captura das imagens - câmaras digitais) não tem sido suficiente para resolver o problema, cuja solução, com certeza está na forma como o processamento precisa ser realizado.

Uma série de aplicações, que atualmente precisam ser executadas por pessoas, continuam ainda sem uma solução automatizada, devido à impossibilidade de uma análise confiável das imagens obtidas. Entre as diversas áreas que buscam esta solução estão:

Cartografia – Embora grande parte dos processos desta área tenham sido automatizados (câmaras digitais capturando imagens e enviando-as diretamente a sistemas computacionais, com apoio simultâneo de sistemas de posicionamento global – GPS, e outros dispositivos), a identificação dos objetos presentes nas imagens, continua ainda

sendo uma tarefa difícil de ser totalmente resolvida computacionalmente. Apesar do grande avanço tecnológico do *hardware*, uma grande quantidade de informações, presente nas imagens, continua ainda dependendo de um operador humano para serem extraídas, e verifica-se, nesta etapa, o grande gargalo do processo.

Militar – Nesta área também se busca uma automação para o processo de localização de alvos. Devido aos grandes investimentos em pesquisas, direcionados à esta área, é possível afirmar que muitos avanços importantes em análise de imagens se devem a esta área.

Medicina – A obtenção de um processo automático que auxilie o diagnóstico baseado em imagens de órgãos do corpo humano é de grande importância, e novamente as barreiras existentes estão na forma de análise das imagens, que são obtidas pelos meios mais comuns nesta área tais como a tomografia computadorizada, ressonância magnética e também pelo ainda extremamente utilizado processo de radiografia baseado em Raios-X.

Visão Computacional – Esta área tem como objetivo a obtenção de máquinas dotadas do sentido de visão (isto inclui a compreensão das imagens obtidas), e neste caso, as demais áreas citadas anteriormente, e muitas tantas outras, se tornam imediatamente apenas aplicações específicas desta área.

A análise de imagens continua sendo uma tarefa difícil de ser implementada em computadores, e o que há de fato disponível até o momento é um conjunto de técnicas de processamento de imagens, que podem ser utilizadas para retirar algumas informações sobre os objetos presentes na cena. Algumas das feições mais importantes são as linhas retas, bordas e pontos isolados.

Devido à dificuldade de se encontrar linhas retas isoladas em imagens digitais, uma melhor alternativa é a utilização das bordas dos objetos (linhas retas formadas pelas bordas da imagem).

Alguns dos motivos que justificam a preferência pela utilização das linhas retas (Tommaselli e Tozzi, 1993) são:

- apresentam-se em grande abundância em ambientes modificados pelo homem (formato de peças em linhas de montagens, construções prediais a serem identificadas em imagens aéreas etc.);

- facilidade de detecção em imagens digitais;
- simplicidade no processo posterior de vetorização;
- e menor probabilidade de erros grosseiros no estabelecimento de suas homólogas no espaço objeto (em relação aos pontos de apoio).

É possível ainda afirmar que as bordas presentes em uma imagem são importantes porque definem os limites dos objetos, e consequentemente, o cálculo do perímetro e área dos objetos analisados. Estas grandezas são importantes porque permitem uma primeira indicação da forma do objeto (Parker, 1996), (Castleman, 1996) de forma a determinar o seu grau de pertinência em relação a um conjunto de objetos conhecidos (quadrados, círculos etc.). Na área de mapeamento especificamente, as linhas retas existentes nas imagens oferecem informações importantes aos processos de orientação das imagens, e também auxiliam durante a etapa de registro de imagens (Tommaselli e Tozzi,1996).

Não somente a escolha das técnicas mais adequadas a cada tipo de aplicação, mas também a forma como tais técnicas podem trabalhar em conjunto são objeto de estudo neste trabalho.

1.2 ESTRUTURA DO TRABALHO

O capítulo 2 deste trabalho apresenta uma revisão de algumas técnicas utilizadas no processo de extração de feições retas em imagens digitais. Como os processos tradicionais para a execução desta tarefa seguem normalmente uma seqüência (Paine e Lodwick, 1988), composta por cinco etapas: Suavização (*Smoothing*), Detecção de Bordas (*Edge Detection*), Limiarização (*Thresholding*), Afinamento de bordas (*Thinning*) e Conexão (*Linking*), a apresentação das técnicas neste capítulo segue a mesma seqüência.

No capítulo 3 são descritos os detalhes do sistema desenvolvido, bem como as correções e melhorias que foram feitas nas técnicas utilizadas.

No capítulo 4 são apresentados os resultados que foram obtidos com a utilização das técnicas implementadas, quando aplicadas sobre uma imagem padrão (simulada) e também sobre o detalhe de uma imagem aérea.

Finalmente, no capítulo 5 são apresentadas algumas conclusões que foram obtidas com a realização deste trabalho, além de algumas recomendações para trabalhos futuros.

CAPITULO 2

ETAPAS DO PROCESSO DE EXTRAÇÃO DE FEIÇÕES

2.1 CONSIDERAÇÕES INICIAIS

A primeira etapa de um processo de extração de informações em uma imagem é conhecida por segmentação, e consiste no processo de separar os objetos presentes na imagem. Dependendo da aplicação, uma operação simples como a limiarização (*thresholding*) pode ser utilizada para segmentar uma imagem em tons de cinza e, desta forma, separar os objetos presentes, do fundo da imagem. Uma outra possibilidade para a segmentação é a que faz uso da determinação das bordas dos objetos presentes na cena, e como tais regiões são caracterizadas por uma variação brusca na imagem, a utilização de detectores de descontinuidades são as ferramentas mais utilizadas. As descontinuidades básicas em imagens são os pontos, as linhas e as bordas, porém, em processos fotogramétricos e também em muitos outros, as de maior importância são as linhas retas (definidas pelas bordas dos objetos).

Os processos tradicionais para a execução da tarefa de conversão de imagens digitais em vetores seguem normalmente uma seqüência tal qual apresentam Paine e Lodwick(1988), que sugerem as cinco seguintes etapas:

1 – Suavização (*Smoothing*) – Inicialmente deve ser executada uma suavização da imagem, com o objetivo de reduzir algumas variações exageradas, que produzem

bordas falsas na imagem (nesta etapa a utilização de filtros capazes de suavizar a imagem, porém preservando as bordas verdadeiras precisa ser investigada). Filtros comuns do tipo passa-baixa são de fácil implementação, porém, não atendem à necessidade de preservação de bordas, e alguma variação se torna necessária;

2 – Detecção de Bordas (*Edge Detection*) – Consiste na aplicação de um detector de bordas, que, normalmente, é baseado na aplicação de operações de detecção de variações de brilho na imagem (normalmente uma diferenciação na região é a solução adotada). O operador de Sobel (Gonzalez, 1993), (Pratt, 1991), (Ekestron, 1983), (Tommaselli e Tozzi, 1993) e também o operador de Nevatia e Babu (Pratt, 1991) merecem bastante atenção nesta etapa;

3 – Limiarização (*Thresholding*) – Esta operação é normalmente utilizada para eliminar algumas bordas detectadas, mas que, por apresentarem uma baixa magnitude, devem ser desconsideradas, a fim de simplificar o processamento posterior;

4 – Afinamento de bordas (*Thinning*) – Etapa em que as bordas com uma espessura de mais de um pixel precisam ser afinadas, para uma melhor definição de sua verdadeira localização. O resultado deve ser uma borda com a largura de um pixel.

5 – Conexão (*Linking*) – Esta etapa deve ser capaz de definir quais os pixels de borda devem ser agrupados, de forma a comporem cada uma das linhas de bordas obtidas.

Este conjunto de etapas é também denominado como um processo único de vetorização. Neste trabalho, em algumas situações é utilizada esta denominação.

A grande quantidade de métodos desenvolvidos em cada uma das fases do processo impossibilita um estudo completo sobre todos eles, e uma verificação dos resultados que podem ser obtidos com a aplicação de alguns métodos diferentes em cada uma das etapas apresentadas anteriormente se mostra mais viável. Outra questão que precisa ser investigada é a definição da própria seqüência apresentada anteriormente, que pode ser modificada, no sentido de melhorar os resultados obtidos (por exemplo a aplicação da etapa de limiarização antes da etapa de detecção de bordas). Outra consideração que merece atenção é a forma como certos detectores agrupam algumas destas etapas do processo, em uma única, tal como ocorre com o Operador de Canny (Parker, 1996), que considera as etapas de suavização, detecção e afinamento, como uma única etapa (detecção).

Os resultados obtidos em cada uma das etapas precisam ser avaliados por meio da utilização de imagens padrão, onde são conhecidos os verdadeiros valores das incógnitas (coordenadas dos pixels de borda, ou ainda os valores dos elementos que definem as linhas de borda – de acordo com a parametrização adotada) no processamento dos algoritmos. Algumas destas formas de avaliação são sugeridas em Pratt (1991) e Parker (1996), e normalmente são baseadas em uma análise estatística da qualidade dos resultados obtidos (em relação à imagens-padrão).

2.1.1 Imagens analógicas e imagens digitais

Uma imagem analógica pode ser definida como uma função bidimensional *f,* que associa para cada *x*,*y*, uma informação de cor *c*.

$$\begin{array}{c} f : \hat{A}^2 \rightarrow \hat{A} \\ x,y \rightarrow c \end{array} \qquad (1)$$

As imagens analógicas não podem ser manipuladas por um computador, e desta forma, um modelo adequado é a representação por meio de uma matriz bidimensional, onde cada elemento representa uma pequena área da imagem (*pixel – picture element*), constituindo-se em uma discretização da imagem sobre uma grade regular (matriz quadrada ou retangular).

2.1.2 Vizinhança em imagens digitais

Um importante conceito em imagens digitais é o de vizinhança de um pixel. Assim, um pixel *p* com coordenadas (*x*,*y*) possui quatro vizinhos, localizados nas coordenadas (*x*-1,*y*), (*x*+1,*y*), (*x*,*y*-1) e (*x*,*y*+1). A este conjunto de pixels é dado o nome de vizinhança-4 do pixel *p*. Da forma semelhante é definida a vizinhança-8 de um pixel *p*, dada pelo conjunto dos pixels que estão na vizinhança-4, além dos pixels localizados nas coordenadas (*x*-1,*y*-1), (*x*-1,*y*+1), (*x*+1,*y*-1) e (*x*+1,*y*+1). Neste trabalho é também

utilizada em várias rotinas uma vizinhança-24, definida pelos mesmos critérios que a vizinhança-4 e vizinhança-8, como mostra a figura 1(c).

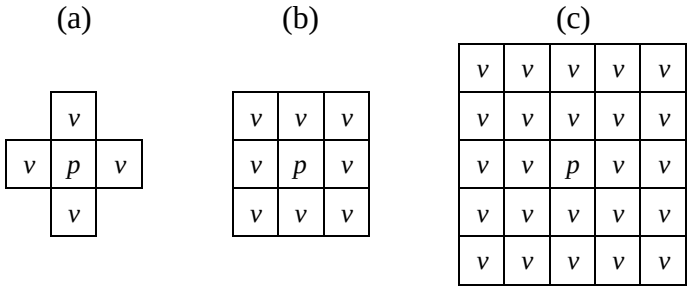


Figura 1 – (a) Vizinhança-4, (b) Vizinhança-8 e (c) Vizinhança-24.

2.1.3 Operações pontuais e a convolução de imagens

Quando o processamento ocorre individualmente sobre os pixels da imagem, tem-se as chamadas operações pontuais. Algumas das operações pontuais mais utilizadas são a obtenção de imagens negativas (invertidas), manipulação de contraste, subtração de imagens, etc.

Uma outra importante forma de processar uma imagem é conhecida como convolução entre uma imagem *i* e uma máscara *h*, representada por $h*i$ é dada por:

$$h*i(x,y)=\sum_{a=0}^{n-1}\sum_{b=0}^{m-1}h(a,b).i(x-a,y-b) \qquad (2)$$

com x,y : dimensões da imagem *i*;

m,n : dimensões do filtro *h*;

Na prática, uma pequena janela (imagem – filtro) é deslocada sobre a imagem original, e o valor do pixel da imagem é obtido pela soma do produto dos elementos da janela, pelos elementos correspondentes na imagem. Em seguida a janela é deslocada, e a operação se repete. A figura 2 mostra o processo de convolução de um filtro *h* e uma imagem *i*, resultando na imagem *k*.

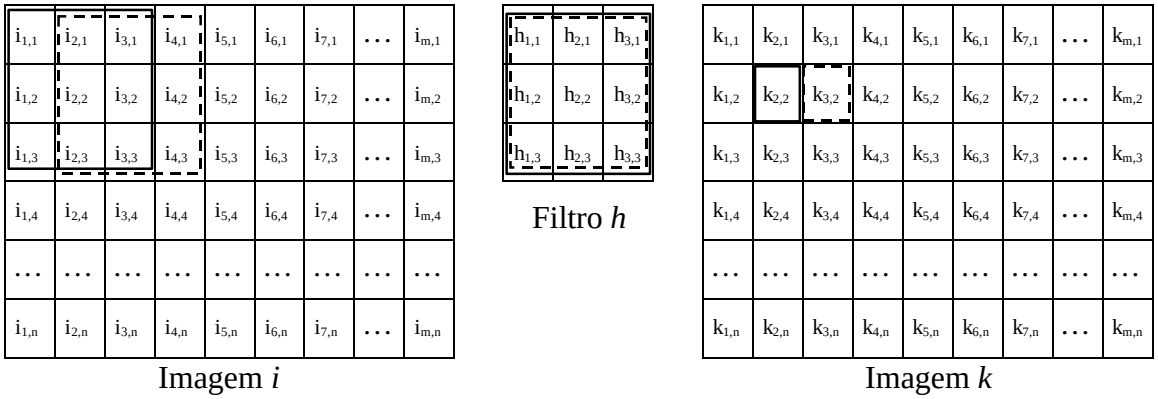


Figura 2 - Operação de convolução sobre imagens digitais.

2.2 ETAPA DE SUAVIZAÇÃO

Conforme apresentado anteriormente, a primeira etapa em um processo de vetorização consiste em uma suavização da imagem, que tem por objetivo eliminar o ruído na imagem e também o excesso de detalhes, que não são importantes (devem ser preservados apenas as feições mais significativas).

Cabe, ainda, a consideração de que os filtros de suavização provocam um borramento da imagem, e assim, diminuem as definições de bordas da imagem. Quando se tem por objetivo fazer a vetorização da imagem, a preservação da localização das bordas é fundamental; portanto, é desejável, nesta etapa, a utilização de filtros de suavização que apresentem uma máxima preservação das bordas.

A presença de ruídos e bordas em certas regiões da imagem caracteriza estas regiões como áreas de altas-freqüências. Por este motivo os filtros projetados para a redução de ruídos são também conhecidos por filtros passa-baixa, por serem projetados de forma a atenuar as altas-freqüências, não modificando as baixas freqüências (áreas homogêneas da imagem).

2.2.1 Filtro passa-baixa

Um filtro passa-baixa no domínio do espaço pressupõe a aplicação de uma máscara como a que aparece na figura 3. Este tipo de filtro é normalmente utilizado para suavizar imagens, principalmente para eliminar (atenuar) ruídos.

$\frac{1}{5}$ | 0 | 1 | 0 |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 1 | 0 |

(a)

$\frac{1}{9}$ | 1 | 1 | 1 |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |

(b)

Figura 3 – Exemplos de filtros passa baixa no domínio do espaço.

É fácil observar que a aplicação das máscaras da figura 3 eqüivalem à execução de uma operação de cálculo da média dos elementos em cada pixel da imagem, no caso da figura 3, (a) e (b) em uma vizinhança 4 e 8 respectivamente.

A figura 4 mostra o resultado obtido com a aplicação de um filtro passa-baixa$_{3x1}$ sobre uma linha de imagem

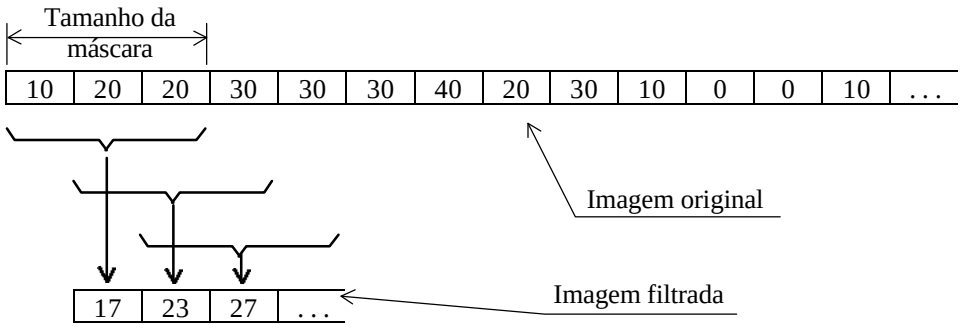


Figura 4 – Aplicação de um Filtro da média$_{3x1}$ sobre uma linha da imagem.

O filtro da média produz uma boa suavização na imagem, porém, apesar de ser ótimo para a remoção de ruídos, tem a desvantagem de atenuar muito a informação de borda.

2.2.2 Filtro da mediana

Este filtro tem como grande vantagem o fato de permitir uma boa remoção dos ruídos, não alterando muito as informações de borda. A mediana de uma seqüência é dada pelo elemento que divide os valores da seqüência em duas partes iguais (a seqüência precisa estar ordenada). A figura 5 mostra um exemplo de aplicação deste

filtro sobre uma linha da imagem. O filtro da mediana não pode ser obtido por meio de uma máscara de convolução.

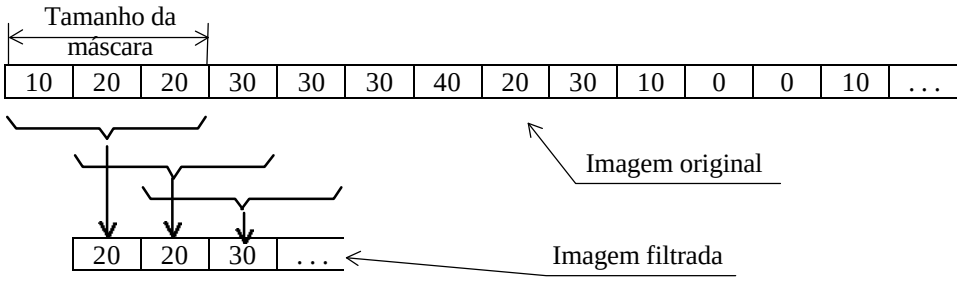


Figura 5 – Aplicação de um Filtro da mediana$_{3x1}$ sobre uma linha da imagem.

A figura 6 mostra graficamente a diferença entre os resultados obtidos com a aplicação dos filtros da média e mediana. Em (a) e (b) verifica-se os resultados obtidos com a aplicação destes filtros em uma região de borda.

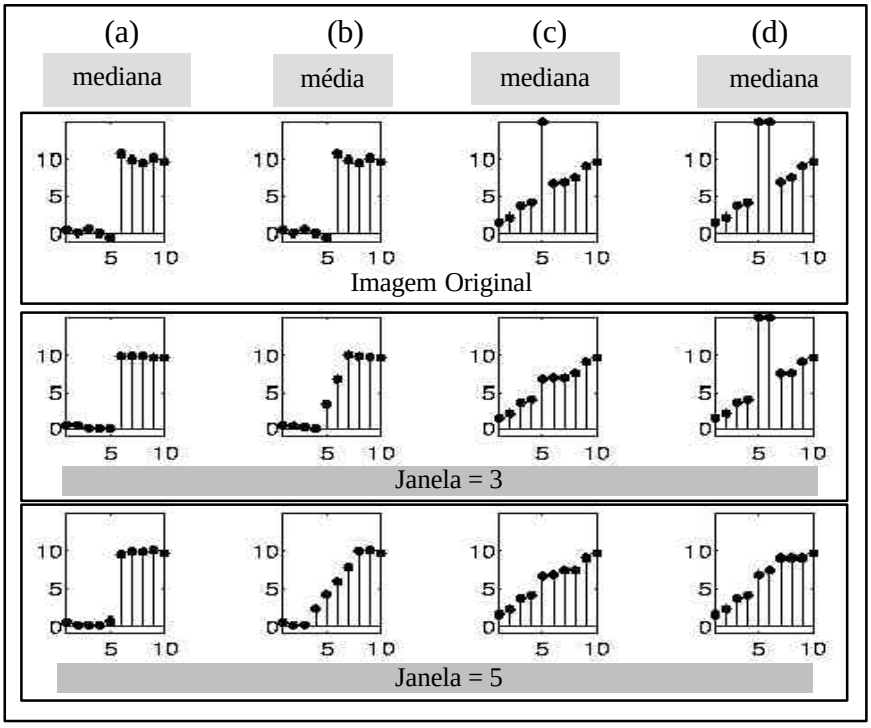


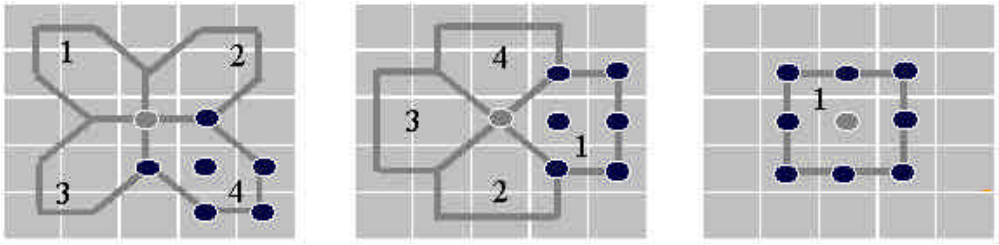
Figura 6 – Comparação de resultados obtidos com os filtros da média e mediana usando janelas de tamanho 3 e 5.

No caso do filtro da média, percebe-se uma grande modificação na região de borda (suavização), enquanto que com o filtro da mediana isto não ocorre. Em (c) tem-se a aplicação do filtro da mediana com tamanho 3 e 5 e observa-se a remoção de um

pixel muito diferente de seus vizinhos (ruído). Em (d) a imagem original possui dois pixels muito diferentes dos demais, porém iguais entre si; neste caso, enquanto que a utilização de uma janela de tamanho 3 age como na presença de uma borda, um filtro com tamanho 5 elimina estes pixels, caracterizando-os como ruído.

2.2.3 Filtro da mediana com análise de variância

Este filtro (Newton, 1993) sugere que o valor a ser atribuído ao pixel seja dado pela mediana dos valores da vizinhança que apresentar a menor variância. Na figura 7, em (a) são construídas quatro regiões (Nordeste, Sudeste, Sudoeste e Noroeste – sete pixels cada), em (b) são construídas quatro regiões (Norte, Leste, Sul e Oeste – sete pixels cada) e em (c) é construída apenas a região central (nove pixels). Ao pixel central é atribuído o valor da mediana da região de menor variância.



(4 regiões-NO,ND,SD,SO) (4 regiões – L,S,O,N) (1 região - CENTRAL)

Figura 7 –Filtro da mediana com a análise da variância.

Este filtro também é chamado de filtro de suavização com preservação de bordas e cantos.

2.3 - ETAPA DE DETECÇÃO DE FEIÇÕES

Na análise de imagens, algumas das feições mais importantes são pixels isolados, linhas retas e as bordas. Enquanto que no domínio da freqüência, estas feições são caracterizadas pela presença de altas freqüências, no domínio do espaço, tais feições são representadas por variações bruscas no brilho dos pixels da imagem. No domínio

espacial, a técnica mais comum para a detecção de feições é baseada na aplicação de operadores locais (máscaras de convolução) de diferenças em cada pixel da imagem. Tais operadores apresentam uma resposta (magnitude) mais elevada nas bordas da imagem. Além da preocupação com a localização dos pixels de borda, existe também a necessidade de se conhecer a inclinação da borda nestes pixels, o que também pode ser obtido pelos mesmos operadores.

2.3.1 Detecção de pixels isolados

O elemento mais simples que pode ocorrer em uma imagem é o pixel isolado, que pode ser detectado com a aplicação de uma máscara de convolução tal qual a que aparece na figura 8.

-1	-1	-1
-1	8	-1
-1	-1	-1

Figura 8 – Máscara utilizada para detectar pixels isolados.

2.3.2 DETECÇÃO DE LINHAS ISOLADAS

As linhas que aparecem em uma imagem oferecem grandes possibilidades de identificação dos objetos nela presentes. A figura 9 mostra um conjunto de máscaras que pode ser utilizado para detectar linhas isoladas em imagens digitais.

-1	-1	-1
2	2	2
-1	-1	-1

(a)

-1	-1	2
-1	2	1
2	1	1

(b)

-1	2	1
-1	2	1
-1	2	1

(c)

2	1	1
-1	2	1
-1	-1	2

(d)

Figura 9 – Máscaras que podem ser utilizadas para detectar linhas isoladas nas direções (a) 0^o , (b) 45^o , (c) 90^o e (d) -45^o(Gonzalez, 1993).

Operadores de detecção de linhas ou pixels isolados não apresentam muitas aplicações, pois, não é comum a ocorrência destes elementos em imagens digitais. Uma melhor alternativa para a extração de feições é a utilização das bordas presentes nas imagens.

2.3.3 Bordas em imagens

As bordas dos elementos presentes em uma imagem são fundamentais no processo de análise de imagens. Isto ocorre porque as bordas definem o contorno dos objetos presentes na imagem (Gonzalez, 1993).

A maioria dos processos de detecção de descontinuidades baseia-se no fato de que tais descontinuidades são, na verdade, uma modificação do nível de cinza no pixel em estudo em relação a seus vizinhos, e assim, tal modificação pode ser determinada pela derivada do sinal no pixel.

A figura 10 mostra uma descontinuidade em uma imagem, e o gráfico associado a estas descontinuidades. Também é apresentado o gráfico da primeira derivada no local, que apresenta um máximo ou mínimo neste local. O gráfico da segunda derivada também é apresentado, e neste caso verifica-se uma característica ainda mais interessante para a detecção da descontinuidades. Como se observa, a segunda derivada apresenta uma passagem por zero (*zero-crossing*) exatamente na localização da descontinuidade. A segunda derivada é conhecida também como Laplaciano (Gonzalez, 1993).

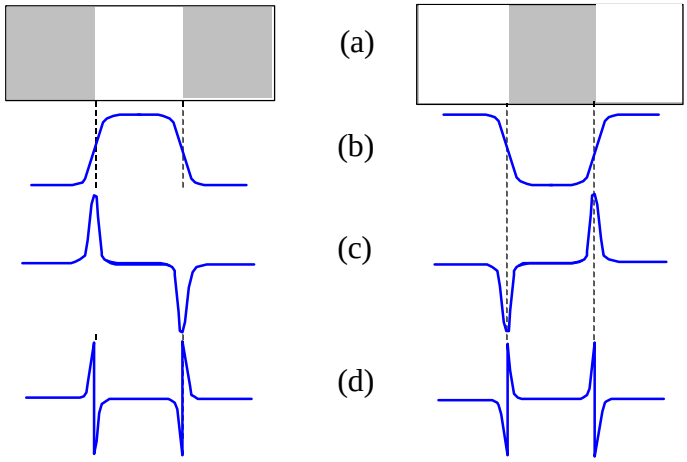


Figura 10 – (a) Imagem com descontinuidade, (b) Sinal da descontinuidade, (c) primeira derivada do sinal e (d) segunda derivada do sinal (Gonzalez, 1993).

Na prática uma máscara de convolução pode ser construída de forma a se obter uma aproximação da derivada (utilizando diferenças). Este é o caso dos operadores de Sobel e Prewitt, que serão descritos a seguir.

2.3.4 Operador de Prewitt

Um operador bastante utilizado na detecção de bordas é o detector de Prewitt, que apresenta as máscaras de convolução que aparecem na figura 11, definindo os gradientes em *x* e *y*, respectivamente como *Gx* e *Gy*.

-1	0	1
-1	0	1
-1	0	1

Gx

1	1	1
0	0	0
-1	-1	-1

Gy

Figura 11 – Máscaras utilizadas no detector de Prewitt.

2.3.5 Operador de Sobel

Considerando que os pixels mais próximos do centro devem apresentar uma maior influência sobre o mesmo, o operador de Sobel é definido com valores maiores na região central, e então as máscaras são as que aparecem na figura 12 (Gonzalez, 1993).

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

Gx

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

Gy

Figura 12 – Máscaras utilizadas no detector de Sobel.

As aplicações dos dois operadores *Gx* e *Gy* resultam nos gradientes da borda na direção *x* e em *y*, respectivamente e, por meio destes, é possível a obtenção da magnitude e da direção (ângulo) da borda em cada pixel.

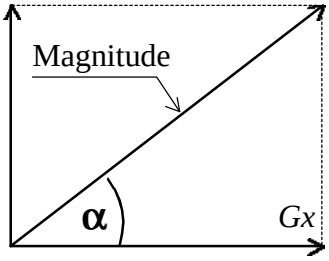


Figura 13 – Determinação da Magnitude dos gradientes e também do ângulo.

A magnitude e a direção da bordas são dadas por (3) e (4).

$$Magnitude_{Gx,Gy} = \sqrt{Gx^2 + Gy^2} \tag{3}$$

$$a = arctan\left(\frac{Gy}{Gx}\right) \tag{4}$$

É importante notar que, pela forma como as máscaras são construídas (baseadas no operador diferencial), as mesmas apresentam uma resposta nula em regiões homogêneas (onde a derivada é nula). Magnitudes de pequeno valor indicam que o pixel em estudo não pertence a uma borda bem definida (área mais homogênea).

2.3.6 Operadores de Nevatia e Babu

Este operador faz a detecção das bordas através da aplicação de um conjunto de máscaras construídas de tal forma a apresentarem uma maior resposta para linhas com determinados ângulos de inclinação. A figura 14 mostra um conjunto de 12 máscaras (Pratt, 1991) que são utilizadas para a detecção de linhas com 0, 30, 60, 90, 120, 150, 180, 210, 240, 270, 300 e 330 graus.

Observa - se que as máscaras utilizadas para detectar bordas com direção α e $(\alpha + \pi)$ são similares, sendo a única diferença definida pela troca dos sinais.

Esta observação permite uma melhora na aplicação do algoritmo que sugere a aplicação das 12 máscaras. Com a utilização de apenas 6 máscaras, e a verificação do sinal para a determinação do ângulo (neste caso a maior magnitude deve ser definida por meio do valor absoluto).

$\frac{1}{1000}$

100	100	0	-100	-100
100	100	0	-100	-100
100	100	0	-100	-100
100	100	0	-100	-100
100	100	0	-100	-100

0 graus

$\frac{1}{1000}$

-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100

180 graus

$\frac{1}{1102}$

100	-32	-100	-100	-100
100	78	-92	-100	-100
100	100	0	-100	-100
100	100	92	-78	-100
100	100	100	32	-100

30 graus

$\frac{1}{1102}$

-100	32	100	100	100
-100	-78	92	100	100
-100	-100	0	100	100
-100	-100	-92	78	100
-100	-100	-100	-32	100

210 graus

$\frac{1}{1102}$

-100	-100	-100	-100	-100
32	-78	-100	-100	-100
100	92	0	-92	-100
100	100	100	78	-32
100	100	100	100	100

60 graus

$\frac{1}{1102}$

100	100	100	100	100
-32	78	100	100	100
-100	-92	0	92	100
-100	-100	-100	-78	32
-100	-100	-100	-100	-100

240 graus

$\frac{1}{1102}$

-100	-100	-100	-100	-100
-100	-100	-100	-100	-100
0	0	0	0	0
100	100	100	100	100
100	100	100	100	100

90 graus

$\frac{1}{1102}$

100	100	100	100	100
100	100	100	100	100
0	0	0	0	0
-100	-100	-100	-100	-100
-100	-100	-100	-100	-100

270 graus

$\frac{1}{1102}$

-100	-100	-100	-100	-100
-100	-100	-100	-78	32
-100	-92	0	92	100
-32	78	100	100	100
100	100	100	100	100

120 graus

$\frac{1}{1102}$

100	100	100	100	100
100	100	100	78	-32
100	92	0	-92	-100
32	-78	-100	-100	-100
-100	-100	-100	-100	-100

300 graus

$\frac{1}{1102}$

-100	-100	-100	-32	-100
-100	-100	-92	78	-100
-100	-100	0	100	100
100	-78	92	100	100
100	32	100	100	100

150 graus

$\frac{1}{1102}$

100	100	100	32	100
100	100	92	-78	100
100	100	0	-100	-100
-100	78	-92	-100	-100
-100	-32	-100	-100	-100

330 graus

Figura 14 – Máscaras utilizadas pelo operador de Nevatia e Babu.

2.3.7 Comparação da eficiência de operadores

Além dos operadores mostrados anteriormente, vários outros têm sido definidos e apresentados na literatura. A tabela 1 mostra um conjunto dos operadores mais utilizados na detecção de bordas (Pratt, 1991).

Tabela 1 – Operadores detectores de gradiente com respostas máximas em bordas na horizontal e vertical (Pratt, 1991).

OPERADOR	GRADIENTE HORIZONTAL	GRADIENTE VERTICAL
DIFERENÇA DE PONTO	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
DIFERENÇA DE PONTO SEPARADA	$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
ROBERTS	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
PREWITT	$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
SOBEL	$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
FREI-CHEN	$\frac{1}{2+\sqrt{2}}\begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{2+\sqrt{2}}\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$

Uma grande limitação destes operadores é a ineficiência que apresentam quando são aplicados em áreas com um alto nível de ruído. Na prática, além da suavização da imagem, este problema pode ser resolvido com um aumento do tamanho da máscara utilizada no processo (Artero e Tommaselli, 1999), o que implica na consideração de uma maior vizinhança do pixel. Por outro lado, o aumento exagerado diminui a sensibilidade do detector, quanto à pequenas variações de direção das bordas e, desta forma, é preciso utilizar um tamanho de máscara mais adequado aos diversos casos (uma possibilidade interessante seria a modificação dos tamanhos de acordo com o nível de ruído presente na região, constituindo-se assim uma espécie de detector adaptativo).

Quando da avaliação da qualidade de um detector de bordas, procura se comparar os resultados obtidos pelo operador, quando o mesmo é aplicado sobre uma imagem em que se conhece as verdadeiras localizações dos pixels da borda. Uma

abordagem baseada nesta idéia é apresentada em Parker (1996), que define a seguinte função:

$$E_1 = \frac{\sum_{i=1}^{I_A} \left(\frac{1}{1 + a d(i)^2} \right)}{max(I_A, I_I)} \tag{5}$$

onde I_A : número de pixels detectados como pertencentes a borda;

I_I : número de pixels na imagem teste;

$d(i)$: distância entre os pixels detectados e verdadeiros;

a : fator de escala, e pode ser mantido constante em um conjunto de testes.

2.4 LIMIARIZAÇÃO (*THRESHOLDING*)

Um método eficiente de limiarização deve ser capaz de fornecer automaticamente um valor (limiar), para o qual todos os pixels com valor de brilho inferior a este limiar devam ser eliminados (magnitude igual a zero, e sem direção). Neste trabalho a etapa de limiarização é utilizada para eliminar as bordas insignificantes da imagem, após o processo de detecção de bordas. De uma maneira geral, é possível afirmar que as bordas menos importantes (e que devem ser eliminadas) são aquelas que possuem uma magnitude muito pequena.

Existe um conjunto muito grande de técnicas para a obtenção de um valor limiar, e observa-se que cada uma delas utiliza algum critério que considera importante para obter o valor do melhor valor limiar (Sahoo, 1988). A seguir são descritos alguns métodos utilizados para a obtenção deste limiar procurado.

2.4.1 P-Tile

O método mais simples para a obtenção do limiar é conhecido por Método *p-tile* (Sahoo, 1988), sendo baseado na condição de que se conhece a área de ocupação do objeto na imagem. Por este método, sabendo-se que o objeto ocupa *p%* da imagem,

então o limiar é definido pelo valor que resulta em *p%* da imagem acima deste valor (supõe-se que o objeto possui valores altos enquanto que o fundo possui valores baixos). Este método é adequado apenas para situações restritas, quando o objeto a ser isolado possui cor distinta do fundo da imagem, ou seja, possui um histograma bimodal, como o que aparece na figura 15(b).

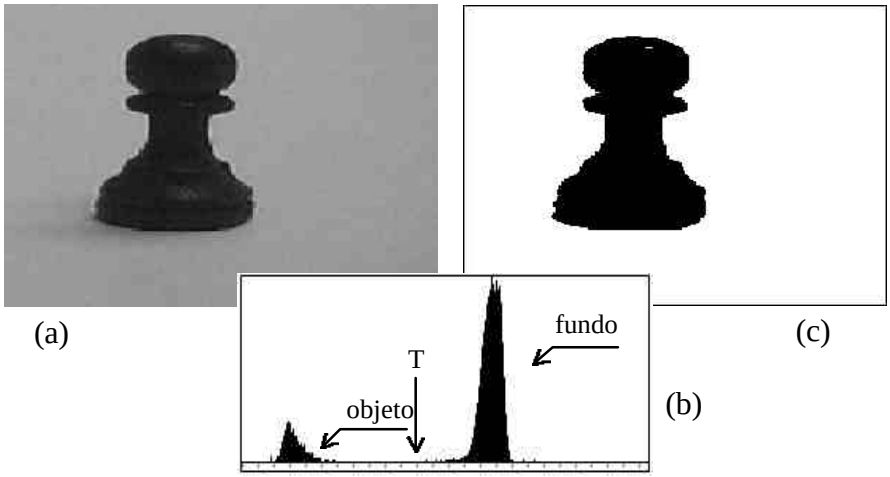


Figura 15 – (a) Imagem original, (b) Histograma obtido e (c) Imagem obtida após aplicação do limiar T.

A figura 15 mostra uma situação em que o método pode ser empregado com sucesso, enquanto que a figura 16 mostra uma outra situação em que não existe uma boa definição de objeto e fundo (o histograma não é bimodal).

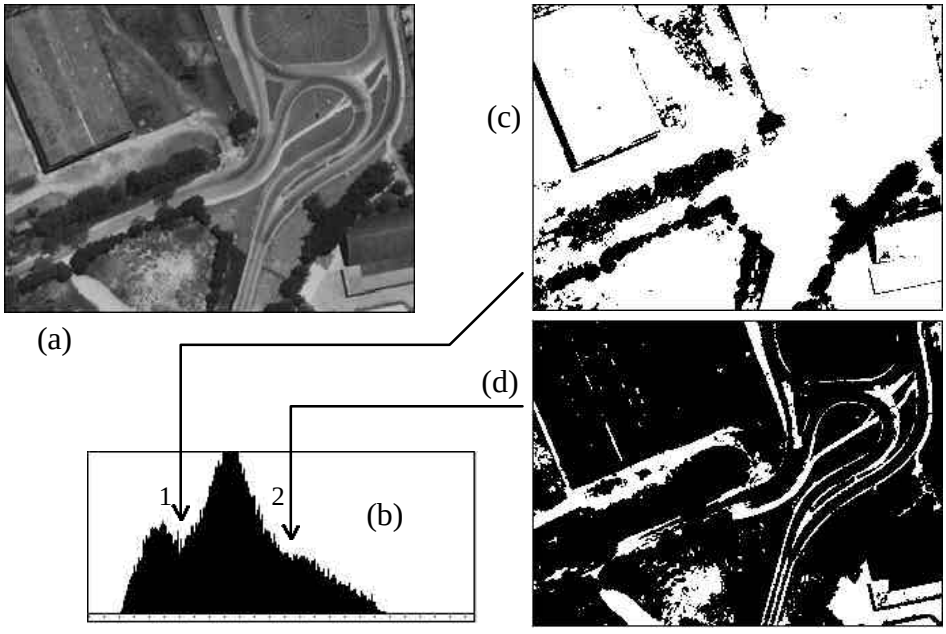


Figura 16 – (a) Imagem original, (b) Histograma, (c) Imagem obtida com a aplicação do limiar 1 e (d) Imagem obtida com a aplicação do limiar 2.

Neste trabalho, a etapa de limiarização deverá ser aplicada após o processo de detecção de bordas, e assim, diferente do que se apresenta nas figuras 15 e 16, existe uma certa definição na tonalidade dos elementos presentes na imagem, ou seja, as bordas sempre aparecem em tons claros sobre um fundo escuro (fgura 17).

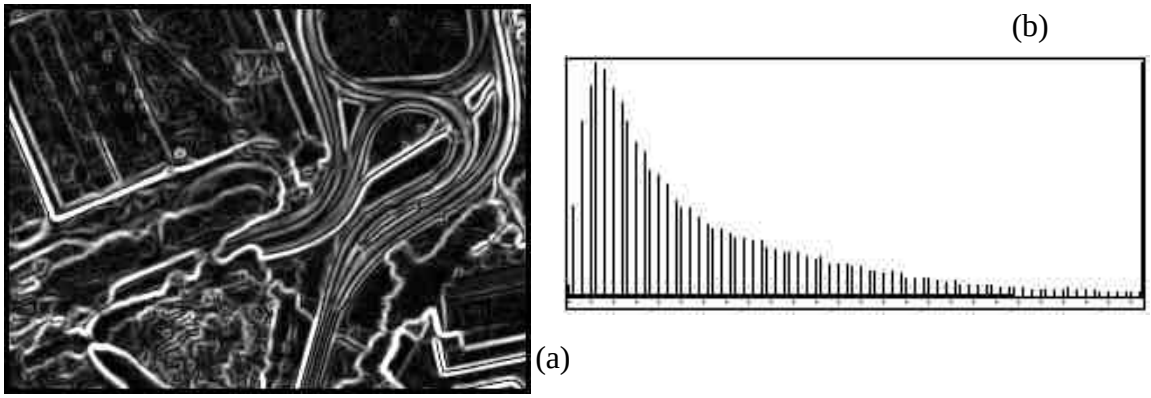


Figura 17 – (a) Imagem de bordas e (b) Histograma da imagem.

Outro fato que precisa ser considerado, é que neste trabalho a etapa de limiarização deve eliminar todos os pixels que possuam a magnitude menor que o valor de limiar. Por outro lado, os pixels que possuem uma magnitude maior não devem ter suas magnitudes alteradas de forma alguma. Apesar de existir o conhecimento do comportamento dos tons de cinza na imagem, ainda assim não existe um valor óbvio para o limiar, pois o histograma não é bimodal, como aquele que aparece na figura 15(b), porém, apresenta um comportamento bem mais previsível que aquele da figura 16 (b).

2.4.2 Método de Otsu

Este método é baseado na análise discriminante (Sahoo, 1988), e o valor do limiar é obtido supondo que os pixels da imagem podem ser classificados em duas classes (C_0 e C_1) que são o objeto e o fundo. Tomando s_B^2 e s_T^2 as variâncias entre as classes e total respectivamente.

A variância entre as classes (Parker, 1996), (Sahoo, 1988) é dada por :

$$s_B^2 = w_0 w_1 (m_0 m_1)^2 \tag{6}$$

onde:

$$w_0 = \sum_{i=0}^{t} p_i \tag{7}$$

$$w_1 = 1 - w_0 \tag{8}$$

$$m_0 = \frac{m_t}{w_0} \tag{9}$$

com

$$m_t = \sum_{i=0}^{t} i.\, p_i \tag{10}$$

e

$$p_i = \frac{n_i}{n} \tag{11}$$

enquanto que a variância total entre as classes(Parker, 1996), (Sahoo, 1988) é dada por:

$$s_T^2 = \sum_{i=0}^{l-1} (i - m_T)^2 . p_i \tag{12}$$

onde:

$$m_T = \sum_{i=0}^{l-1} i.\, p_i \tag{13}$$

e n_i é a freqüência que o valor i ocorre na imagem e n o valor total de pixels na imagem. O valor ótimo para o limiar, segundo este método, é dado pelo valor de t, tal que n seja mínimo em (14).

$$n = \frac{s_B^2}{s_T^2} \tag{14}$$

2.4.3 Método de Pun

Este método é baseado na teoria da informação, que se baseia na premissa de que a geração de informação pode ser modelada como um processo probabilístico

(Gonzalez, 1993). Por esta teoria define-se a entropia (quantidade de código necessária para representar um símbolo) de cada pixel da imagem por:

$$Entropia(x) = x \,.\, log(x) \tag{15}$$

Pelo método, para cada possível valor de limiar são definidas duas entropias à posteriori (do objeto e do fundo da imagem), dadas por:

$$H_b = -\sum_{i=0}^{t} p_i \log_e p_i \tag{16}$$

$$H_w = -\sum_{i=t+1}^{l-1} p_i \log_e p_i \tag{17}$$

e o valor do limiar ótimo é dado por

$$T = Arg\ máximo\ \{\ H_b(t) + H_w(t)\ \} \tag{18}$$

2.4.4 Método de Kapur, Sahoo e Wong

Como se trata de um outro método baseado na teoria da informação, este método também faz uso da entropia, porém, definindo H_b e H_w conforme (19) e (20)

$$H_b = -\sum_{i=0}^{t} \frac{p_i}{p_t} \log_e \left(\frac{p_i}{p_t} \right) \tag{19}$$

$$H_w = -\sum_{i=t+1}^{l-1} \frac{p_i}{1-p_t} \log_e \left(\frac{p_i}{1-p_t} \right) \tag{20}$$

e o valor do limiar ótimo é dado por

$$T = Arg\ máximo\ \{\ H_b(t) + H_w(t)\ \} \tag{21}$$

2.4.5 Método de Johannsen e Bille

Um outro método baseado na teoria da informação, neste caso, o valor do limiar é dado por :

$$T = Arg\ máximo\ \{\ S_1(t) + S_2(t)\ \} \tag{22}$$

Onde

$$S_1 = \log_e\left(\sum_{i=0}^{t} p_i\right) - \frac{1}{\sum_{i=0}^{t} p_i}\left[p_t \log_e p_t + \left(\sum_{i=0}^{t-1} p_i\right)\log_e\left(\sum_{i=0}^{t-1} p_i\right)\right] \tag{23}$$

$$S_2 = \log_e\left(\sum_{i=t}^{l-1} p_i\right) - \frac{1}{\sum_{i=t}^{l-1} p_i}\left[p_t \log_e p_t + \left(\sum_{i=t+1}^{l-1} p_i\right)\log_e\left(\sum_{i=t+1}^{l-1} p_i\right)\right] \tag{24}$$

2.4.6 Método do Triângulo

Este método sugere o traçado de uma linha entre os valores de máximo e mínimo no histograma da imagem, e utiliza como limiar o valor de brilho que apresenta a maior valor de *d* (distância entre a reta e os valores de freqüência(brilho) – não nulos).

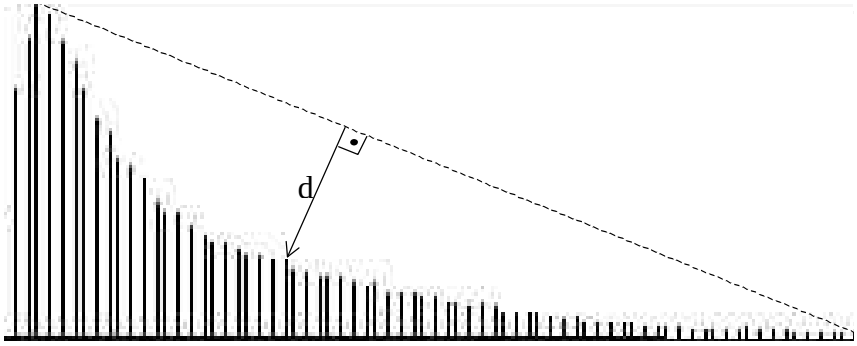


Figura 18 – Critério utilizado para a determinação do valor de limiar, pelo Método do Triângulo.

A forma obtida nos histogramas de imagens de bordas se mostram muito atraentes para a aplicação deste método de limiarização.

2.4.7 Eficiência dos métodos globais

De maneira geral (não se tratando especificamente de imagens de bordas), os resultados obtidos com o uso de cada um dos métodos descritos não apontam para um método mais adequado que outro (Sahoo, 1988), e o que se observa na prática é que os resultados obtidos variam de acordo com a imagem processada.

Observa-se ainda que, devido a problemas de reflectância, sombras e falta de iluminação regular nas imagens, um valor de limiar global geralmente não apresenta bons resultados (Hussain, 1991). No caso do processamento de imagens aéreas estes problemas também precisam ser levados em consideração.

2.5 ETAPA DE AFINAMENTO DE BORDAS (*THINNING*)

Quando na obtenção da borda, normalmente não se obtém uma linha única definindo a mesma, e sim um conjunto de pixels. A etapa seguinte no processo deve realizar um afinamento da linha de borda. Existem vários métodos propostos com o objetivo de resolver este problema, sendo que alguns dos mais conhecidos fazem parte das operações morfológicas afinamento (*Thinning*) e geração de esqueleto (*Skeletonization*); porém, tais operações são geralmente empregadas em imagens binárias, onde todos os pixels possuem apenas dois valores de brilho, que se resumem em cor de fundo e cor de frente (do objeto). Neste caso, o afinamento consiste em uma eliminação gradual dos pixels da região (pixels de uma borda), de tal modo que no final do processo sobram apenas os pixels centrais da região.

Quando as linhas a serem afinadas não são binárias, ou seja, são compostas por pixels de brilho variado (diferenças de magnitude), e ainda se verifica que o valor do brilho é maior de acordo com a proximidade do mesmo em relação a borda, tais métodos não são adequados. A figura 19 mostra o sinal de uma borda obtida por meio

de algum dos operadores descritos anteriormente, e também os resultados que podem ser obtidos por diferentes técnicas de afinamento.

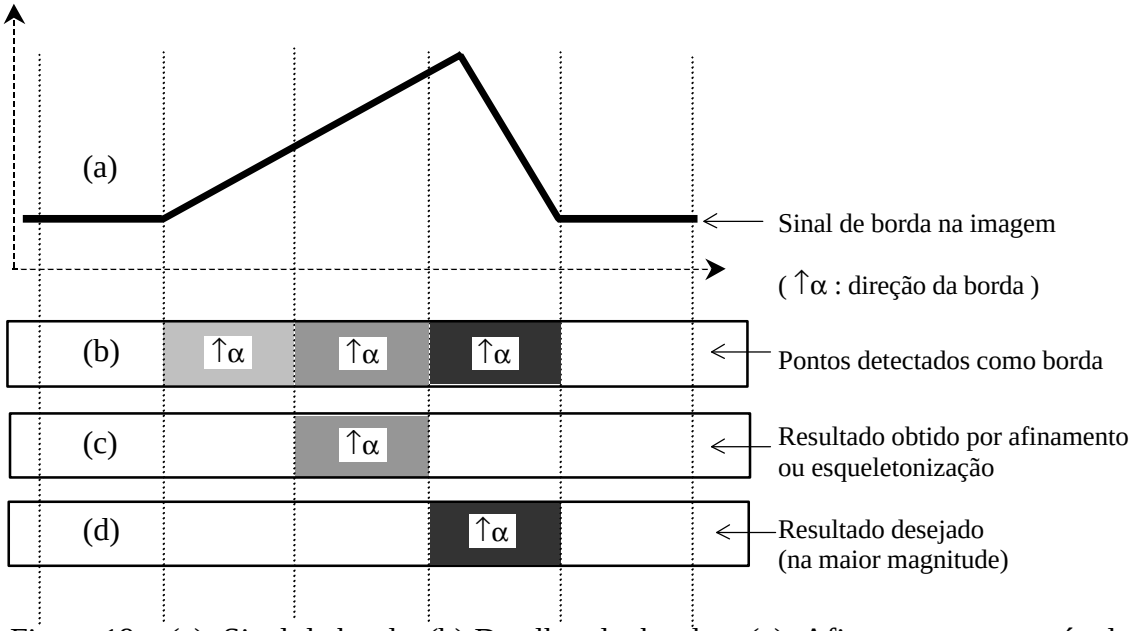


Figura 19 – (a) Sinal de borda, (b) Detalhe da borda, (c) Afinamento por métodos de afinamento binário e (d) Considerando o pixel de maior magnitude que melhor representa a localização da borda.

2.5.1 Método da supressão não máxima

Por este método a eliminação dos pixels é feita perpendicularmente à direção da borda. Com o objetivo de simplificar as rotinas de busca na vizinhança podem ser utilizadas as regiões apresentadas na figura 20(a), (Zhou et al, 1989) e (Venkateswar-Chellapa, 1992) para cada direção de pixel de borda, ou ainda as direções apresentadas na figura 20(b) (Tommaselli, 1999).

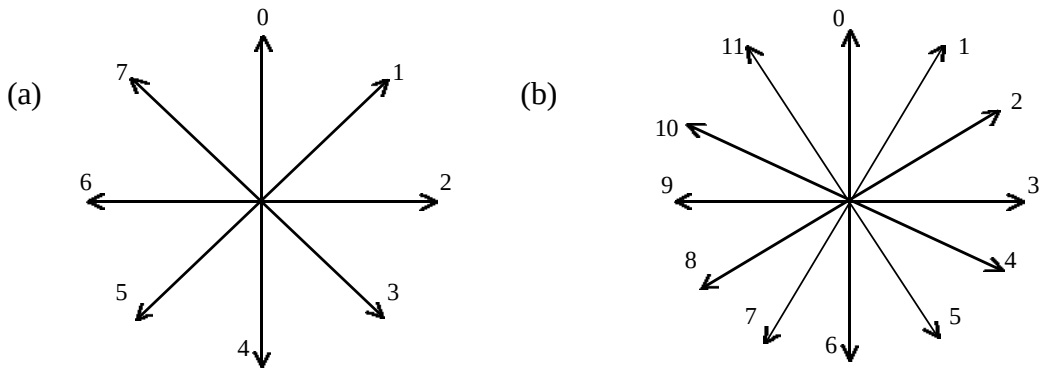


Figura 20 – Direções consideradas no método de afinamento por supressão não máxima (a) (Zhou Et al., 1989) e (Venkateswar-Chellapa, 1992), (b) (Tommaselli, 1999).

Visualmente, esta discretização não apresenta problemas nos resultados obtidos, e o afinamento se desenvolve com uma comparação entre os pixels, de forma a manter apenas o de maior magnitude. A figura 21 mostra detalhes dos pixels pertencentes a duas bordas.

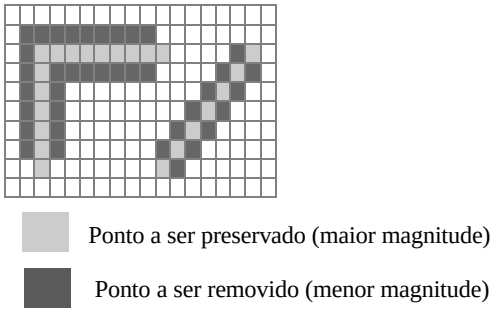


Figura 21 – Processo de afinamento de bordas.

Durante o processo, um pixel com direção 90^o ou 270^o é comparado com seus dois vizinhos horizontais (à direita e à esquerda), enquanto que um pixel com direção horizontal (0^o ou 180^o) deverá ser comparado com seus dois vizinhos verticais (pixels acima e abaixo). No caso de bordas inclinadas, tal como aparece na figura 21, os pixels devem ser comparados com seus vizinhos na diagonal pela borda. Porém, neste caso o processo não é tão simples como nos dois casos anteriores, pois os pixels a serem comparados precisam estar também à uma distância unitária do pixel em análise. Como se trabalha sobre uma malha regular, não existem pixels nestes locais, tornando-se necessária a realização de uma reamostragem na região, o que normalmente pode ser feito através de uma interpolação bilinear. A figura 22 mostra esta situação em que a borda possui uma direção não horizontal ou vertical

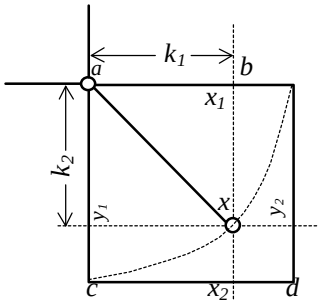


Figura 22 – Detalhe da localização dos pontos a serem interpolados.

No caso da figura 22, deseja-se conhecer a magnitude no ponto *x*, sendo conhecidas apenas as magnitudes nos pontos pertencentes à grade (*a*, *b*, *c* e *d*). Inicialmente, as magnitudes nos pontos x_1 e x_2 são obtidas por duas interpolações lineares, utilizando as equações paramétricas dos segmentos de retas *ab* e *cd*, respectivamente dadas por:

$$x_1 = a + (b - a).k_1 \tag{25}$$

$$x_2 = c + (d - c).k_1 \tag{26}$$

e, finalmente, a magnitude no ponto *x* é obtida por uma terceira interpolação linear, desta vez utilizando a equação paramétrica da reta x_1x_2, dada por:

$$x = x_1 + (x_2 - x_1).k_2 \tag{27}$$

onde *x* é a magnitude interpolada procurada. Aplicando (25) e (26) em (27) tem-se:

$$x = a + (b - a)k_1 + [(c + (d - c)k_1) - (a + (b - a).k_1)]k_2 \tag{28}$$

assim:

$$x = a + (b - a)k1 + (c -.a)k2) + (d -.c -.b +.a)k_1k_2 \tag{29}$$

e para o caso particular em que $k_1 = k_2$ pode ser obtida a seguinte simplificação

$$x = a + (b - 2a + c)k1 + (d - c - b + a)k1^2 \tag{30}$$

Observamos ainda que k_1 é o coseno do ângulo enquanto que k_2 é o seno do mesmo (ambos em valor absoluto).

A detecção de bordas utilizando os operadores de Sobel resulta em um conjunto contínuo de ângulos, e a discretização em apenas oito direções (figura 20) representa uma simplificação muito drástica.

A discretização em doze direções (a cada 30 graus) apresentada em Tommaselli (1999), constitui-se em um melhor aproveitamento dos valores de direção, quando na

etapa de detecção de bordas é utilizado o operador de Nevatia e Babu (doze direções discretas entre 0 e 330^{o}).

Finalmente, com o objetivo de se avaliar possíveis melhoras nos resultados desta etapa de afinamento de bordas, é ainda implementada neste trabalho uma modificação no método, em que não é necessária a discretização das direções de bordas, consistindo em um método de supressão não máxima generalizada (Artero e Tommaselli, 1999).

2.6 ETAPA DE CONEXÃO (L*INKING*)

A etapa de conexão de pixels (*linking*) deve localizar todos os pixels pertencentes a uma borda e agrupá-los, de forma a constituírem uma única feição, dotada de seus atributos peculiares, ou seja, no caso de uma linha reta, um pixel inicial, um pixel final, uma direção etc. O método mais conhecido que pode ser utilizado nesta etapa é a Transformada de Hough, que permite a localização de objetos (linhas, círculos etc.) em imagens digitais. Apesar desta operação ser de fácil compreensão, o seu algoritmo apresenta grandes restrições quanto a sua implementação. Neste trabalho é apresentado ainda um outro método, que apresenta como vantagem uma maior eficiência da estrutura utilizada.

2.6.1 A Transformada de Hough

Uma das técnicas mais conhecidas para a determinação de formas geométricas que passam por um dado conjunto de pixels é a Transformada de Hough. Trata-se de uma técnica robusta, capaz de identificar linhas retas e também outros tipos de feições.

A figura 23 mostra o princípio da técnica (Low, 1991), que pode ser utilizada para localizar a posição de um círculo de raio conhecido, passando por um dado conjunto de pixels.

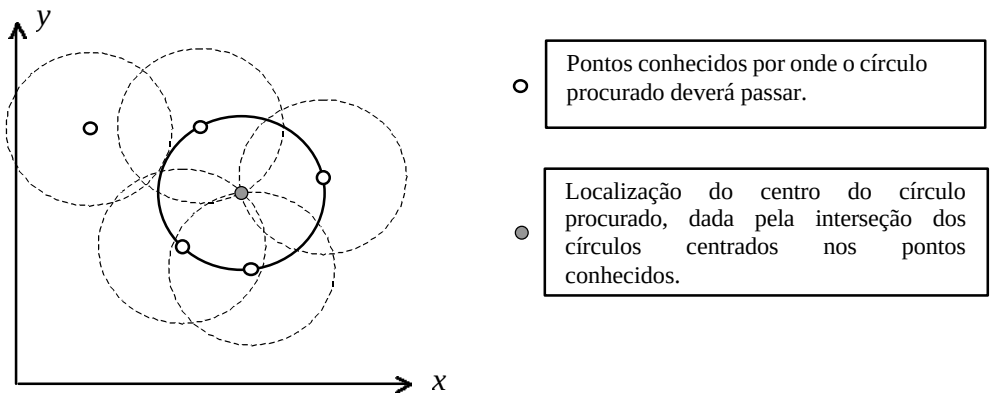


Figura 23 – Localização da posição do circulo de raio conhecido passando por um conjunto de pixels (Low, 1991).

No caso de linhas retas, a mesma idéia é repetida, porém desta vez, para cada pixel da imagem traça-se todo o conjunto de retas que passam pelo mesmo. Novamente a reta que passar simultaneamente pelo maior número de pixels deverá ser a reta procurada.

Assim dado o conjunto de pixels p_1, p_2,...,p_n, de coordenadas (x_1,y_1), (x_2,y_2)...(x_n,y_n), são traçadas infinitas retas, que passam por cada um deles, conforme mostra a figura 24.

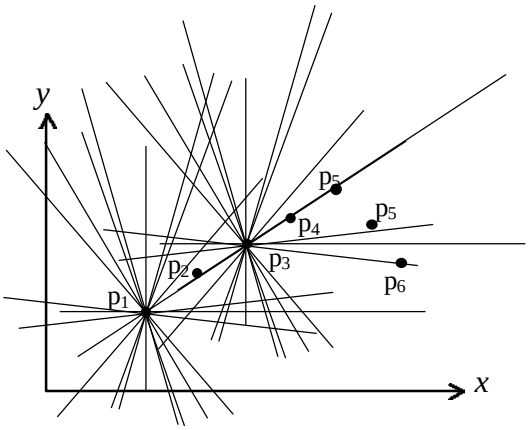


Figura 24 – Posição dos pixels no espaço *x*-*y* e das retas possíveis sobre estes.

Utilizando a Transformada de Hough não é necessário o traçado de tantas retas como mostra a figura 24. Assim, partindo da equação da reta dada por:

$$y = ax + b \tag{31}$$

cada pixel p_i, de coordenadas (x_i,y_i) é utilizado para gerar uma equação de reta dada por:

$$b = y_i - ax_i \quad (32)$$

Enquanto que na expressão (31) os parâmetros são *x* e *y*, em (32) os parâmetros da reta são *a* e *b*. A figura 25(a) mostra a localização de cinco pixels, no sistema *x*-*y* e em (b) a localização das cinco retas geradas, no sistema *a*-*b*. O ponto de interseção das cinco retas define os valores dos parâmetros *a* e *b*, da reta procurada no sistema *x*-*y*.

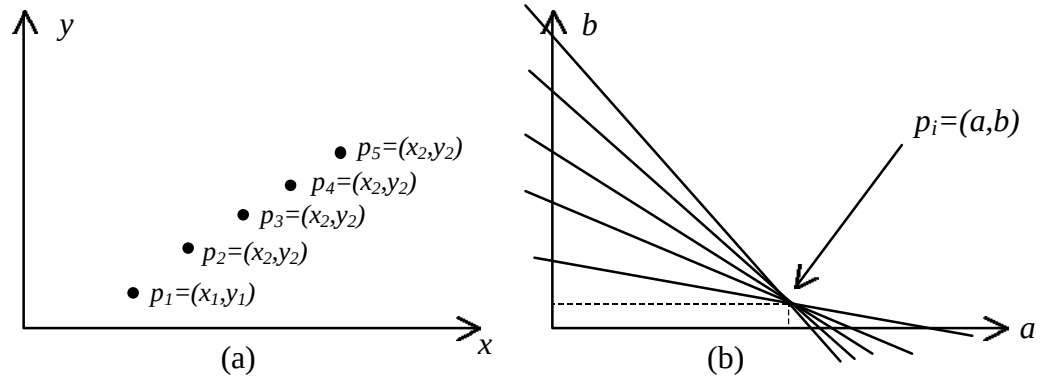


Figura 25 – Transformação entre os espaços de parâmetros *x*-*y* e *a*-*b* (pela Transformada de Hough). (a)Pixels no espaço *x*-*y*, e (b) Linhas no espaço *a*-*b*.

Utilizando esta parametrização, é preciso alocar na memória do computador, um vetor de dimensão *m* x *n*, onde *m* é o número de valores que *a* pode assumir e *n* é o número de valores que *b* pode resultar a partir de (32) (*m* e *n* constituem a discretização do parâmetro *a* e *b*). A figura 26 mostra o algoritmo para a implementação da Transformada de Hough utilizando esta parametrização.

Algoritmo → Transformada de Hough - (espaço *a*,*b*)

1 – Faça acumuladores $A(a,b)=0$; (Quantize o espaço do parâmetros (*a*-*b*) adequadamente;)
2 – Diferencie a imagem utilizando o Operador de Sobel
(a) gx = gradiente-x (b) gy = gradiente-y
3 – Se a magnitude da borda é maior que um threshold, calcule $a = (gy/gx)$;
4 – Calcule $b = -ax_i+y_i$;
5 – Incremente o acumulador $A(a,b) = A(a,b) + 1$;
6 – Repita (3) até (5) para todos os pixels de borda;
7 – Os picos no acumulador $A(a,b)$ fornecem os gradientes da linha e as interseções;

Figura 26 – Algoritmo para a Transformada de Hough no espaço *a*-*b* (Hussain, 1991).

No algoritmo apresentado na figura 26 uma grande simplificação do processo é realizada, sendo calculado o valor de *b* apenas para um valor de *a* igual a direção da borda. Na prática, esta simplificação não se mostra totalmente confiável, pois a direção (obtida na etapa de detecção de bordas) dos pixels de borda, não é muito precisa, devido, principalmente, aos problemas de *aliasing* que podem ser visualizados na figura 27.

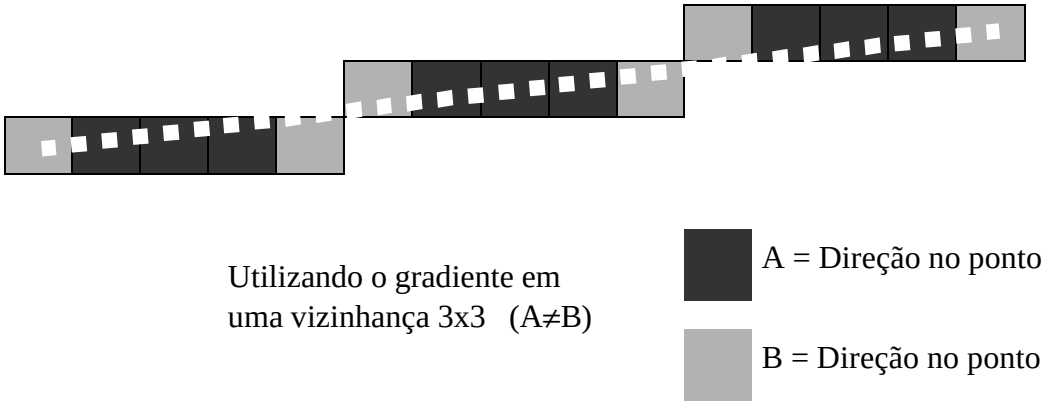


Figura 27 –Influência do aliasing na determinação da direção dos gradientes.

De qualquer forma, o conhecimento da direção dos pixels permite que sejam traçadas apenas linhas retas que possuem uma direção com valor próximo às direções dos pixels, ou seja, para alguns valores acima e abaixo de *a*, o que deverá tornar os resultados mais confiáveis (figura 28) .

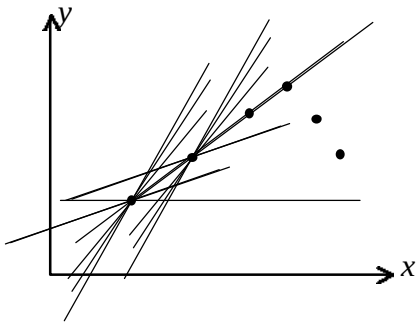


Figura 28 –Utilizando de um conjunto de linhas com inclinações próximas a direção do pixel.

A parametrização apresentada na expressão (31) possui o inconveniente de não ser capaz de trabalhar com retas verticais (quando a $\rightarrow \pm\infty$). Neste caso, a utilização da parametrização (33) se mostra capaz de contornar tal problema.

$$\rho = x\,cosq + y\,senq \qquad (33)$$

Na implementação da Transformada de Hough com a representação q - ρ, para cada pixel da imagem, com coordenadas x e y, faz-se variar o valor de q em (33) e obtém-se, assim, o valor de r. O resultado obtido pode ser visualizado no gráfico apresentado na figura 29(b), desta vez no espaço q e r.

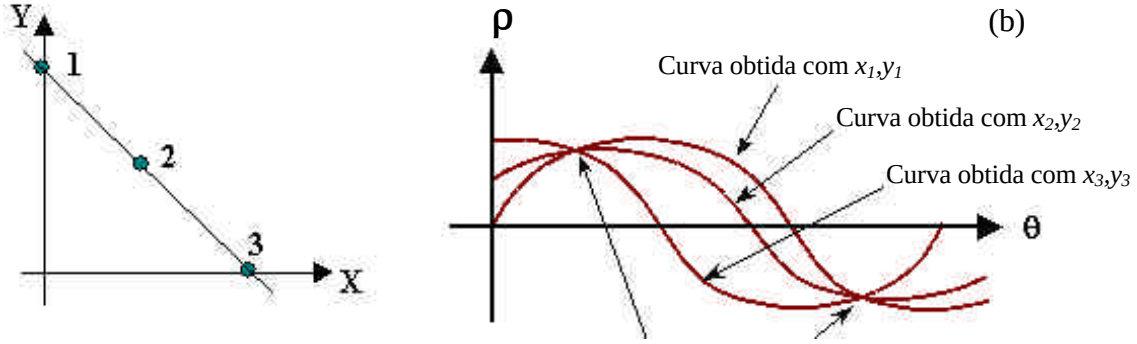


Figura 29 – (a) Pixels no espaço x-y e (b) Transformada de Hough utilizando a parametrização r-q.

Utilizando esta nova parametrização, é preciso alocar na memória do computador um array de dimensão m x n, onde m é o número de valores que q pode assumir e n é o número de valores que r pode resultar em (33) (m e n constituem a discretização dos parâmetros q e r).

Novamente, uma grande simplificação do processo pode ser feita se for conhecida a direção da borda em cada pixel, desta vez, limitando os valores de q em um pequeno intervalo em torno do valor da direção.

A figura 30 mostra o algoritmo usado para implementar a Transformada de Hough (Jain et al., 1995) usando esta parametrização.

Algoritmo → Transformada de Hough - (espaço q-r)

1 - Quantize o espaço do parâmetros (q-r) adequadamente;
2 - Assuma que cada célula no espaço dos parâmetros é um acumulador; inicialize todas as células com zero;
3 - Para cada pixel (x,y) no espaço imagem, incremente em 1 cada um dos acumuladores, que satisfaçam a equação (33) - (fazendo q variar e calculando ρ);
4 - Os máximos nos acumuladores correspondem aos valores de r e q procurados;

Figura 30 – Algoritmo para a Transformada de Hough no espaço q-r.

Além de ser uma técnica bem conhecida, a Transformada de Hough tem ainda como grande vantagem, ser capaz de identificar linhas retas em bordas que tenham sofrido algum tipo de quebra (tais situações são muito comuns no processamento de imagens aéreas, onde construções ou mesmo árvores ocultam bordas do tipo divisa de terrenos ou guias e sarjetas - oclusão). Um problema que dificulta a utilização da Transformada de Hough é que uma discretização muito densificada dos parâmetros q e r implica na necessidade de uma estrutura (array de acumuladores) muito grande. Por outro lado, a utilização de uma quantização insuficiente pode não apresentar a precisão desejada nos resultados. Um outro problema que ocorre é o aparecimento de falsos picos nos acumuladores, provocados pelas interseções das curvas geradas por retas muito grandes, que superam os valores de acumuladores devidos a pequenas retas. Um exemplo desta situação é apresentada na figura 31; no caso, observa-se que o alinhamento dos pixels da forma em que se encontram induz a Transformada de Hough a identificar as três horizontais, de quatro pixels cada, quando o que se deveria encontrar são as quatro retas verticais de apenas três pixels cada. Este problema pode ser evitado, fazendo o valor de q variar apenas em uma faixa próxima ao valor da direção da borda.

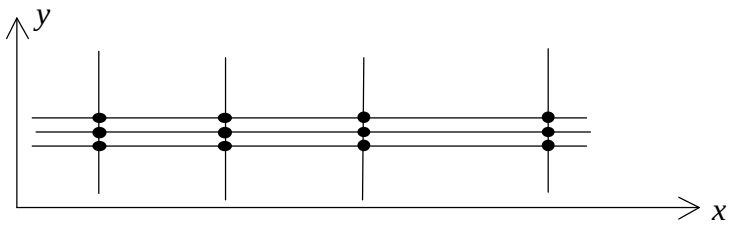


Figura 31 – Problema de falsos picos na Transformada de Hough.

2.6.2 Agrupamento q-r

Este método pode ser visto como uma otimização da Transformada de Hough, também se utilizando da parametrização q e r. A importância do método está em apresentar uma organização na seqüência de operações, que permite uma implementação computacional mais eficiente, devido à otimização da estrutura necessária, pois, neste caso, não há necessidade de uma estrutura de acumuladores regular, e geralmente, muito esparsa (Dudani e Luk, 1978).

O método pode ser descrito através dos seguintes passos:

1- Para cada pixel da imagem cujas coordenadas são x_i e y_i, e onde se conhece a direção q_i calcula-se r_i a partir deste três elementos, utilizando a expressão (33);

2- O conjunto de pixels assim obtidos (x_i,y_i,q_i,r_i) é dividido em grupos, classificados inicialmente de acordo com q. Nesta etapa é comum a utilização de uma histograma de freqüências para q, onde os picos do histograma definem o número e os valores de classes a serem utilizadas;

3- Em uma próxima etapa, os grupos ou classes definidos em 2 são também repartidos em subgrupos, desta vez usando como critério o elemento r. Novamente um histograma de freqüências (de r) pode ser utilizado, e desta vez os picos do histograma definem as subclasses a serem utilizadas;

4- Cada subclasse de r define a localização das retas (de acordo com a posição em relação a origem - translação), enquanto que as classes em q definem as inclinações (rotação);

5- Verifica-se a continuidade dos segmentos dentro dos subgrupos.

A figura 32 mostra o fluxo de dados utilizado neste método.

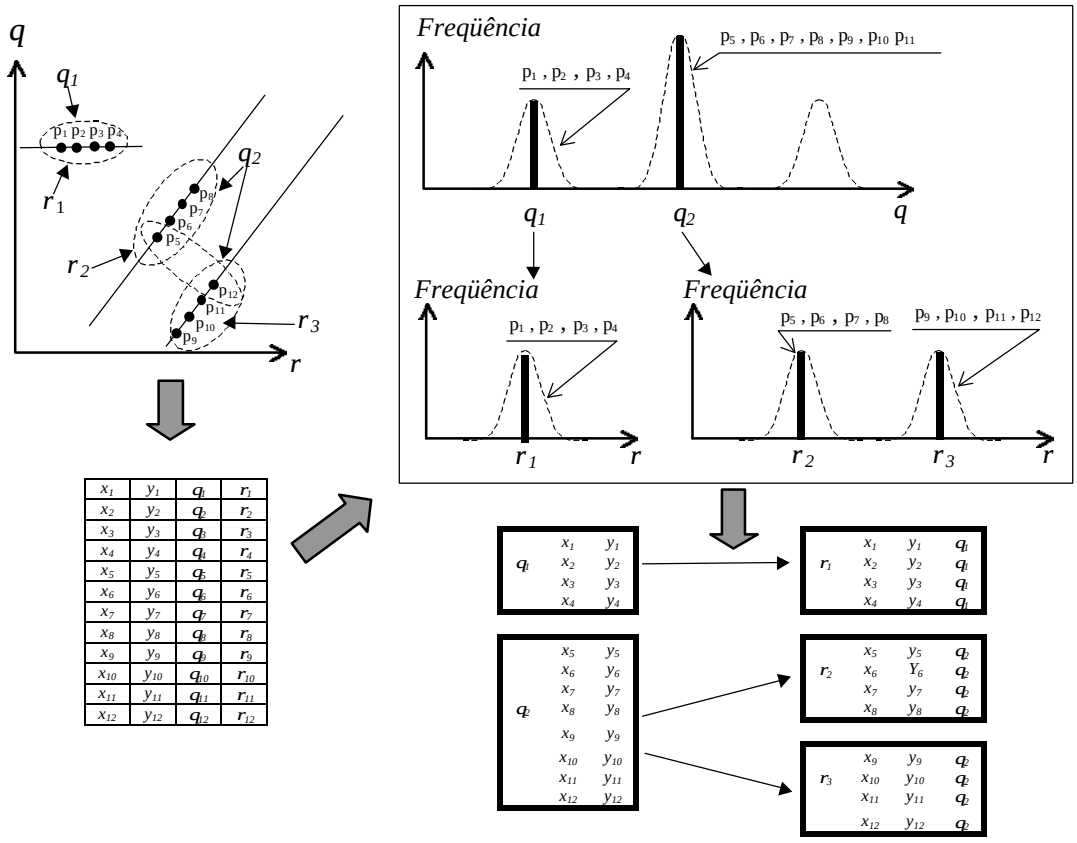


Figura 32 – Fluxo dos dados no método de agrupamento q-r

A grande vantagem desta forma de organizar os pixels, está em permitir a utilização de uma estrutura computacional bem mais simples, do que a Transformada de Hough.

Enfim, apesar da Transformada de Hough ser uma técnica robusta e muito conhecida, apresenta grandes dificuldades de implementação (necessidade de uma estrutura muito grande para apresentar resultados adequados). Além disso, não consegue definir o início e o fim das retas determinadas (o que pode ser realizado em uma etapa posterior à identificação das linhas retas). Por outro lado, os problemas de oclusão que ocorrem com freqüência em imagens aéreas podem ser resolvidos com o apoio deste método. A forma de implementação utilizando o agrupamento q -r representa uma excelente melhora no processo, dispensando a necessidade de uma estrutura muito grande, porém, devido ao fato de apoiar-se também na informação de direção dos pixels de borda, a mesma deve apresentar problemas, uma vez que não se observa na prática uma total confiança nesta informação (conforme mostra a figura 27).

2.6.3 Método da varredura e rotulação (*scan & label*)

Este método faz uma varredura e rotulação dos pixels da imagem (Zhou et al., 1989), (Venkasteswar, 1994) e (Tommaselli, 1999), de forma que pixels com uma mesma direção e vizinhos recebem um mesmo rótulo. Ao mesmo tempo, uma estrutura de dados é utilizada para guardar as características das retas que estão sendo definidas (pixel inicial(x_i,y_i), pixel final(x_f,y_f), direção(dir), número de pixels). Neste método a imagem é percorrida da esquerda para a direita e de cima para baixo e, para cada pixel, de acordo com a sua direção são verificados os vizinhos (a vizinhança a ser considerada depende da direção do pixel examinado – pixel atual). Caso seja encontrado nas proximidades do pixel atual algum vizinho já rotulado, então o pixel atual recebe o mesmo rótulo. Operando desta forma, um segmento começa a ser definido. No momento da análise dos pixels, é utilizada a seguinte convenção.

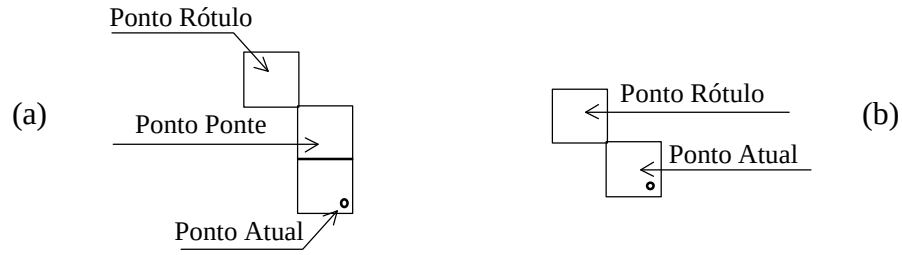


Figura 33 – Classificação dos pixels envolvidos no processo de conexão.

Entende-se por pixel vizinho do pixel atual, o pixel que se localiza no máximo a uma distância igual a dois, e que possua a mesma direção que o pixel atual. No caso da distância ser igual a dois, significa que existe um pixel entre o pixel atual e o seu vizinho (pixel rótulo), que também não possui um rótulo; neste caso, deve tratar-se de um pixel sem magnitude e sem direção e tais informações podem ter sido eliminadas na etapa de limiarização, ou ainda, nunca terem existido, devido a uma pequena quebra do segmento). Este pixel representa um buraco na linha que está sendo traçada, e para evitar este problema, ele deve também ser incluído nesta linha. Este pixel recebe a denominação de pixel ponte, por fazer a ligação entre o pixel atual e o pixel rótulo. O pixel atual deve receber o rótulo do pixel vizinho. O pixel ponte deve receber o mesmo rótulo, a mesma direção e uma magnitude, que pode ser a média entre as magnitudes dos pixels atual e vizinho.

Observa-se na implementação a necessidade de uma estrutura, que deverá receber os dados das retas conectadas no processo. Uma possível estrutura a ser utilizada é formada pelos campos apresentados na figura 34.

```
struct linha
  {
   int   xi;      // coordenada x do pixel inicial da reta i
   int   yi;      // coordenada y do pixel inicial da reta i
   int   xf;      // coordenada x do pixel final da reta i
   int   yf;      // coordenada y do pixel final da reta i
   int   dir;     // direção dos pixels da reta i
   int   n;       // número de pixels na reta i
   float a;       // parâmetro a da reta
   float b;       // parâmetro b da reta
   bool  horiz;   // horizontal = true  e  vertical = false
  };
```

Figura 34 – Estrutura de dados utilizada para armazenar os atributos dos segmentos.

A figura 35 mostra as vizinhanças a serem verificadas no processo de conexão dos pixels, de acordo com as direções de seus gradientes.

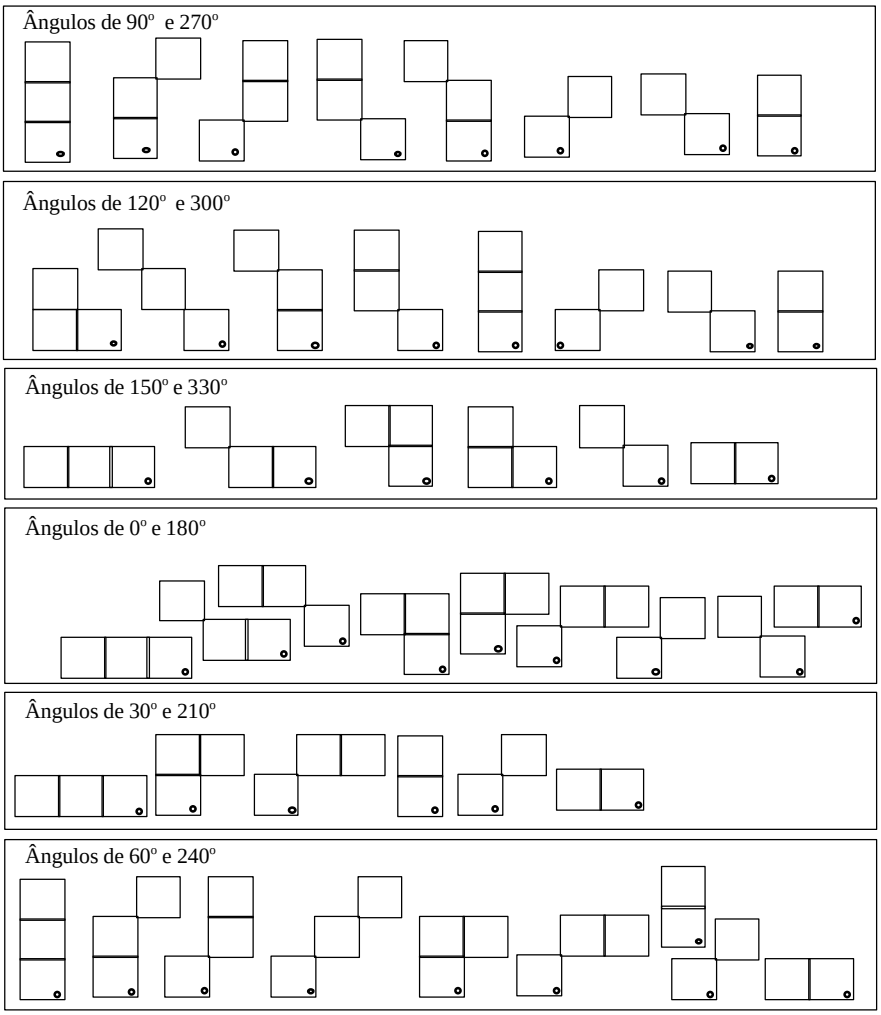


Figura 35 – Vizinhanças a serem verificadas no processo de conexão (Zhou et al., 1989) (Venkateswar-Chellapa, 1992),(Tommaselli, 1999).

2.6.3.1 Eliminação de linhas insignificantes

Após a realização da rotina de varredura e rotulação, conforme descrita anteriormente, verifica-se a presença de uma grande quantidade de linhas que apresentam um tamanho insignificante (em muitos casos, apenas um pixel).

Por outro lado, pode ocorrer também que estes pequenos segmentos, podem ser reconectados uns aos outros, formando segmentos maiores. Assim, parece mais interessante que antes de se fazer a eliminação dos segmentos com poucos pixels, se faça a conexão dos segmentos colineares, descrita na próxima seção.

2.6.3.2 Conexão de segmentos colineares

Uma estratégia que pode ser utilizada para fazer este processamento é investigar as vizinhanças de cada pixel inicial e final de todos os segmentos obtidos, e verificar se existe nas proximidades destes, o início ou fim de um outro segmento. Caso ocorra esta situação é preciso verificar ainda se eles possuem uma mesma direção, e neste caso, deve ser feita a união dos dois segmentos, formando um único. Por fim os atributos do novo segmento precisam ser atualizados, sendo feitas as devidas alterações na estrutura de dados que mantém o controle da localização do início e fim deste segmento, bem como eliminação dos segmentos que foram unidos. Na prática, o primeiro segmento pode ser atualizado para conter as características do novo segmento, e o segundo segmento deve ser eliminado, ou seja, ter seu rótulo desativado.

Todo este processamento pode ser realizado diretamente na estrutura de dados gerada na etapa anterior, permitindo uma boa performance no processamento, se comparada a uma varredura na imagem digital, pois, enquanto a imagem digital pode possuir alguns milhões de pixels, a estrutura deverá ter alguns milhares de vetores.

Novamente a informação da direção das linhas pode ser utilizada para uma melhora na forma como as investigações na vizinhança de um pixel final ou inicial devem ser feitas. A figura 36 mostra as vizinhanças que devem ser investigadas, para o caso de uma linha reta com direção α.

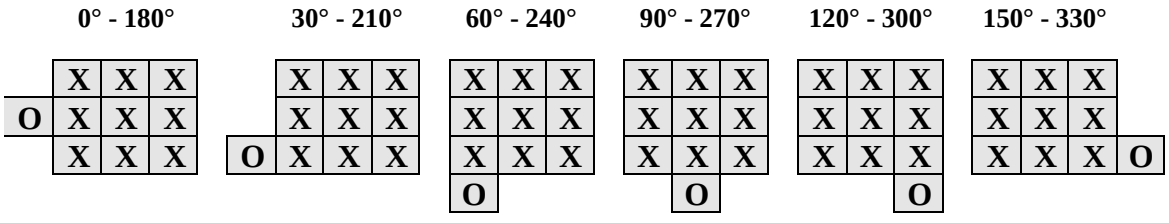


Figura 36 – Detalhe das vizinhanças a serem investigadas no processo de conexão de segmentos colineares.

Para cada segmento i, obtém-se na estrutura a posição dos pixels inicial e final da reta x_i,y_i e x_f,y_f; verifica-se no espaço da imagem se na vizinhança existe algum pixel, que pertença a um outro segmento j (isto ocorre se o pixel vizinho possui rótulo

diferente do pixel atual). É importante também verificar se a reta *j* possui uma direção próxima à direção da reta *i* (uma diferença mínima é aceitável nesta etapa). No caso de se encontrar algum pixel nesta situação, o segmento *j* terá todos os seus pixels rotulados com o valor do rótulo do segmento *i*.

A troca de rótulos é realizada no espaço imagem, e pode ser resolvida de uma forma suficientemente satisfatória, utilizando uma janela limite, que contenha os pixels início e fim do segmento *j*. Por este método todos os pixels pertencentes à menor janela que contém o segmento serão processados. Uma outra alternativa mais eficiente é fazer a troca dos rótulos utilizando um algoritmo recursivo que se desloca dentro do segmento por um processo de inundação (*seed-fill*). Apesar da recursão ser mais difícil de ser construída, apenas os pixels com o mesmo rótulo são processados.

Conforme já observado anteriormente (figura 27), a etapa de detecção de bordas nem sempre consegue os mesmos valores de direção para todos os pixels pertencentes a uma mesma borda, e assim a etapa de conexão de segmentos colineares precisa considerar pequenas diferenças de direção, no momento em que analisa a junção de dois segmentos.

Em muitos casos, dois segmentos próximos, que possuem pequenas diferenças de direção podem ser conectados em um único segmento, sem problema algum. Por outro lado, existem situações em que tais segmentos não deverão ser unidos. Um critério que pode ser utilizado para definir em quais situações dois segmentos podem ser unidos é mostrado na figura 37.

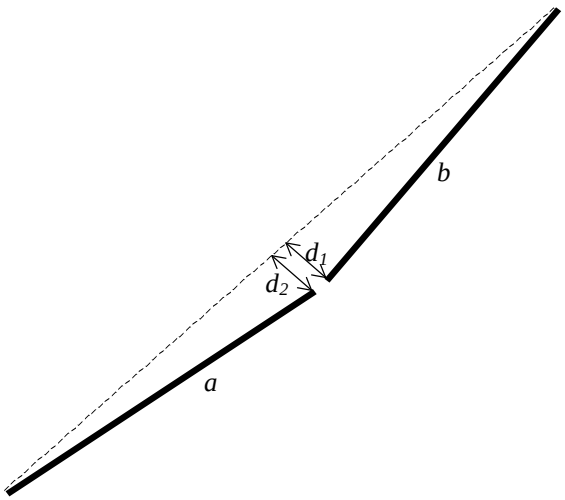


Figura 37 – Critério adotado para realizar conexão dos segmentos colineares.

Na figura 37, os segmentos *a* e *b* devem ser unidos caso as distâncias d_1 e d_2 sejam menores que um fator de qualidade previamente definido. Os valores de d_1 e d_2 são definidos pela distância entre os pixels finais dos segmentos *a* e *b*, e a reta definida pelos extremos opostos dos segmentos, sendo estas distâncias obtidas conforme mostra a figura 38.

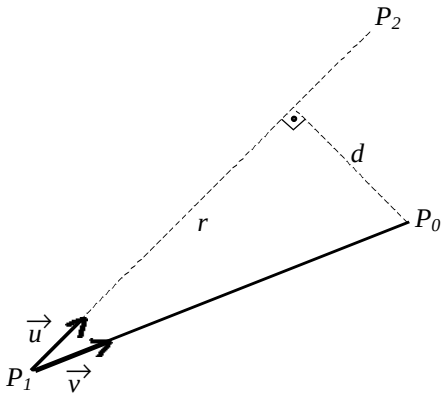


Figura 38 – Geometria utilizada para definir as distâncias d_1 e d_2.

Na figura 38, a distância entre o ponto P_o e a reta *r* é dada por

$$d(P_0, r) = \frac{|\ \vec{u} \wedge \vec{v}\ |}{|\ \vec{u}\ |} \tag{34}$$

Finalmente, é preciso considerar que os *templates* apresentados na figura 36, utilizados para verificar a vizinhança dos pixels inicial e final dos segmentos, em busca de outros segmentos nas proximidades, implicam novamente no aparecimento de pixels ponte (pixels que devem ser ligados entre os dois segmentos que estão sendo unidos). No caso da direção igual a 0° e 180° verifica-se a situação apresentada na figura 39.

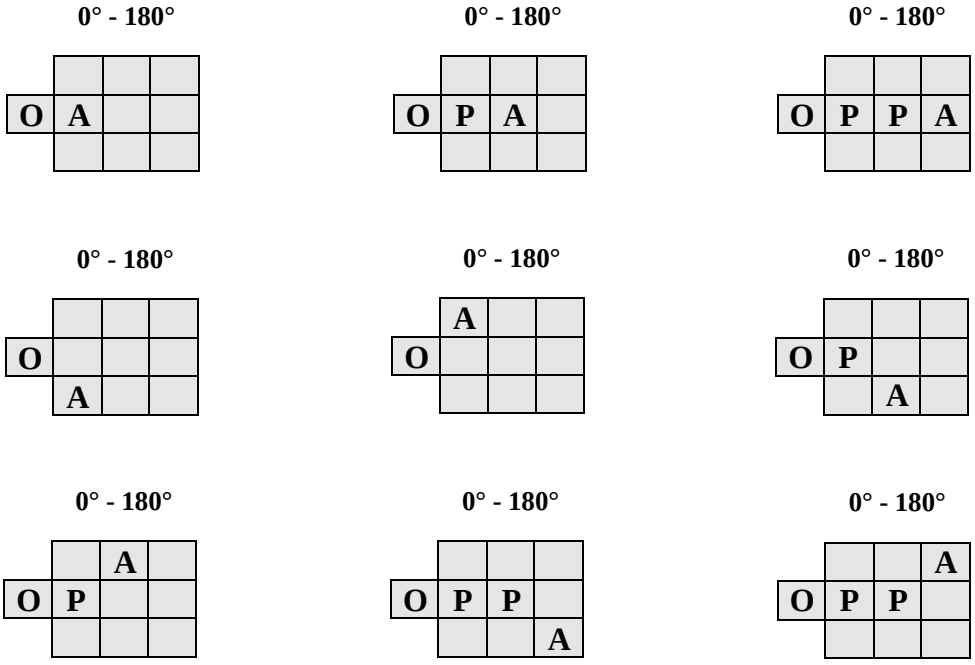


Figura 39 – Localização de alguns pixels pontes no processo de conexão de segmentos.

Conforme mencionado anteriormente, após a junção de dois segmentos, as informações sobre o segmento obtido devem ser atualizadas na estrutura de linhas. A quantidade de pixels pertencentes ao mesmo, é dada pela soma dos totais de pixels pertencentes aos dois segmentos que foram unidos, e ainda dos pixels ponte (caso ocorram).

A informação de direção é um pouco mais difícil de ser obtida, pois segmentos com direções próximas (porém diferentes) podem ser unidos, caso seja satisfeita a condição imposta na figura 37, e neste caso, o segmento obtido pela união dos dois segmentos originais precisa receber alguma direção, que deve ser próxima das direções dos segmentos unidos.

2.7 AJUSTAMENTO DE RETAS

A próxima etapa a ser realizada após a identificação dos pixels que pertencem a um dado segmento, é a determinação da equação da reta que melhor se ajusta a estes pixels (ajustamento). O consagrado Método dos Mínimos Quadrados (M.M.Q.) pode ser

utilizado nesta etapa, apresentando resultados adequados. O método é bem simples e consiste em minimizar o somatório dos quadrados das distâncias entre os pixels e a reta em que são ajustados. O método permite ainda que, para o caso de se obter uma minimização insuficiente, a reta possa ser quebrada em outras menores, até se sejam obtidas retas adequadas aos pixels detectados, atendendo a um fator de qualidade previamente estabelecido. Desta forma é estabelecido um controle de qualidade dos resultados obtidos.

2.7.1 Introdução ao Método dos Mínimos Quadrados (M.M.Q.)

Dada uma seqüência de pontos $S_{i,j} = ((x_1,y_1) , (x_2,y_2) ,..., (x_n,y_n))$, que são obtidos por meio de uma função *f*, desconhecida, deseja-se determinar uma função *g*, que melhor se aproxima a *f*.

A tentativa de aproximar uma função *f*, por uma função *g*, introduz um erro em cada um dos pontos conhecidos, que normalmente é chamado resíduo v_i. Como o que se procura é obter *g*, de forma que a mesma seja a melhor aproximação de *f*, o objetivo a ser alcançado então é fazer com que estes resíduos sejam minimizados. Uma primeira abordagem leva então a concluir que o que se deve fazer é:

$$\sum_{i=1}^{n} v_i = mínimo \tag{35}$$

Esta simplificação possibilita que erros positivos eliminem erros negativos, não apresentando bons resultados. Uma segunda tentativa pode levar então a utilização dos valores absolutos de v_i,

$$\sum_{i=1}^{n} | v_i | = mínimo \tag{36}$$

Porém, isto também apresenta um inconveniente (Humes et al., 1984), quando se tenta obter a minimização, o que é obtido pela primeira derivada igual a zero (a função valor absoluto não possui derivada no ponto zero).

Finalmente uma forma simples e que se mostra viável é a utilização dos quadrados dos v_i. Assim procura-se obter:

$$\sum_{i=1}^{n} v_i^2 = \text{mínimo} \tag{37}$$

Este critério para a aproximação de duas funções é conhecido como Método dos Mínimos Quadrados M.M.Q., sendo intensamente utilizado nos problemas de ajustamento, nas mais diversas áreas.

Neste trabalho, deseja-se obter as equações das retas que melhor se ajustam aos conjuntos de pixels rotulados na etapa anterior como pertencentes às bordas. Assim, dada a equação da reta:

$$y = ax + b \tag{38}$$

onde, são conhecidos os valores de x e y, deseja-se obter os valores de a e b. Embora o modelo sugira a aplicação do Método Combinado (Gemael, 1994), uma simplificação pode ser feita, quando se assume que o parâmetro x não possui erros e, assim, é possível a aplicação do Método paramétrico, com a seguinte construção:

Sejam:

L_b : vetor dos valores observados;

V : vetor dos resíduos;

L_a : vetor dos valores observados ajustados;

então

$$L_a = L_b + V \tag{39}$$

e X_0 : Vetor dos valores aproximados dos parâmetros;

X : Vetor correção;

X_a : Vetor dos parâmetros ajustados;

$$X_a = X_0 + X \tag{40}$$

Por este método, o modelo matemático a ser usado deve ser:

$$L_a = F(X_a) \tag{41}$$

Aplicando (39) e (40) em (41), resulta

$$L_b + V = F(X_0 + X) \tag{42}$$

Para o caso de sistemas não lineares, é conveniente uma linearização por Taylor e, assim a expressão (42) fica:

$$L_b + V = F(X_0 + X) = F(X_0) + \left.\frac{\partial F}{\partial X_a}\right|_{X_a = X_0} X \tag{43}$$

Em seguida, designando a função dos parâmetros aproximados por L_0 obtém-se:

$$L_0 = F(X_0) \tag{44}$$

e chamando a matriz das derivadas parciais de A, é possível definir o vetor de correções X, dado por:

$$X = -(A^t PA)^{-1}.A^t PL \tag{45}$$

com $L = L_0 - L_b$.

Neste trabalho, ajustando a seqüência de pixels $X(i,j)$ = ((x_1,y_1) , (x_2,y_2) ,..., (x_n,y_n)), em uma função da forma:

$$y = ax + b \tag{46}$$

o sistema de equações fica:

$$\begin{aligned} y_1 &= a.x_1 + b + v_1 \\ y_1 &= a.x_1 + b + v_2 \\ &\cdots\cdots\cdots\cdots \\ y_1 &= a.x_1 + b + v_n \end{aligned} \tag{47}$$

ou

$$\begin{aligned} v_1 &= a.x_1 + b - y_1 \\ v_1 &= a.x_1 + b - y_2 \\ &\cdots\cdots\cdots\cdots \\ v_1 &= a.x_1 + b - y_n \end{aligned} \tag{48}$$

que utilizando a notação matricial resulta em:

$$A = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ ... & 1 \\ xn & 1 \end{pmatrix} \qquad L = \begin{pmatrix} -y_1 \\ -y_2 \\ .... \\ -y_n \end{pmatrix} \tag{49}$$

e assim

$$\begin{pmatrix} v_1 \\ v_2 \\ .... \\ v_n \end{pmatrix} = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ ... & 1 \\ x_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} -y_1 \\ -y_2 \\ .... \\ -y_n \end{pmatrix} \tag{50}$$

Para o caso de sistemas lineares, a solução obtida em (45) é imediata, pois o sistema não precisa ser linearizado (usando uma aproximação por Taylor).

Novamente verifica-se a ocorrência de problemas com relação a parametrização adotada ($y=ax+b$), pois, utilizando a seqüência $S_{i,j}$ = ((x_1,y_1) , (x_2,y_2) ,..., (x_n,y_n)), para os casos em que todos os valores y_i são iguais (retas verticais), o sistema não pode ser resolvido, pois a matriz A^tPA não possui uma inversa. Devido a este problema que deve ocorrer em várias situações, antes de tentar fazer a inversão da matriz é preciso verificar se a mesma possui tal inversa.

Uma condição necessária e suficiente para que uma matriz quadrada seja inversível é que o seu determinante seja diferente de zero (Caroli, 1984).

A abordagem genérica apresentada neste tópico pode ser particularizada, para o caso em que os pixels pertencentes às seqüências são sempre ajustados por linhas retas. Este caso particular do ajustamento é conhecido como Regressão Linear (uma terminologia muito antiga, originada nas Ciências Sociais (Press, 1992)). Esta particularização permite a construção de um modelo mais simples de ser visualizado (Humes et al., 1984), e ainda dispensa o uso de operações com matrizes e vetores (subtrações, produtos e inversões).

Este modelo toma a seqüência de pixels $S_{i,j}$ = ((x_1,y_1) , (x_2,y_2) ,..., (x_n,y_n)), e a função g, escolhida como a função que mais se aproxima de f (função desconhecida que gerou a seqüência $S_{i,j}$). O resíduo em cada pixel (x_i,y_i) é determinado por:

$$v_i = y_i - g(x_i) = y_i - a - bx_i \tag{51}$$

em seguida determinam-se a e b, que minimizam os quadrados dos resíduos em (51), ou seja, minimiza

$$\sum_{i=1}^{n} v_i^2 = \sum_{i=1}^{n} (y_i - ax_i - b)^2 \tag{52}$$

o que pode ser feito tomando as derivadas parciais de (52) em relação à a e b iguais a zero, ficando então:

$$2\sum_{i=1}^{n} (y_i - ax_i - b)(-x_i) = 0 \tag{53}$$

e

$$2\sum_{i=1}^{n} (y_i - ax_i - b)(-1) = 0 \tag{54}$$

que podem ser escritas como:

$$a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i x_i \tag{55}$$

$$a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} 1 = \sum_{i=1}^{n} y_i \tag{56}$$

ou utilizando a notação matricial, chega-se finalmente ao sistema (57).

$$\begin{pmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{pmatrix} \tag{57}$$

cuja solução é dada por:

$$a = \frac{\left(\sum_{i=1}^{n} 1\right)\left(\sum_{i=1}^{n} x_i y_i\right) - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{\left(\sum_{i=1}^{n} 1\right)\left(\sum_{i=1}^{n} x_i^2\right) - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{58}$$

e

$$b = \frac{-\left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} x_i y_i\right) + \left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} x_i^2\right)}{\left(\sum_{i=1}^{n} 1\right)\left(\sum_{i=1}^{n} x_i^2\right) - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{59}$$

Computacionalmente, a forma que (58) e (59) aparecem em Boratto (1987) é mais conveniente para ser utilizada nos algoritmos (equações (60) e (61)).

$$a = \frac{\sum_{i=1}^{n} x_i y_i - \frac{\sum_{i=1}^{n} x_i y_i}{N}}{\sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{N}} \tag{60}$$

e

$$b = \frac{\sum_{i=1}^{n} y_i}{N} - a\frac{\sum_{i=1}^{n} x_i}{N} \tag{61}$$

A determinação dos parâmetros *a* e *b* também pode ser feita em função do determinante da matriz, ficando da seguinte forma (Tommaselli e Tozzi, 1993):

$$a = \frac{1}{\det(A^t A)}(N\sum x_i y_i - \sum x_i \sum y_i) \tag{62}$$

e

$$b = \frac{1}{\det(A^t A)}(-\sum x_i \sum x_i y_i + \sum {x_i}^2 \sum y_i) \tag{63}$$

Uma solução que pode ser adotada para evitar o problema com as retas verticais é utilizar duas parametrizações diferentes, cada uma adequada às retas apresentadas. A sugestão é utilizar para segmentos aproximadamente horizontais (onde ocorre uma boa variação na coordenada *x* dos pixels) ou com pequena inclinação, a equação (64).

$$y = ax + b \tag{64}$$

No caso de segmentos aproximadamente verticais (onde as coordenadas *x* dos pixels não variam) ou com grande inclinação, uma modificação na parametrização (64) pode ser feita de forma a resolver o problema (Tommaselli e Tozzi, 1993).

$$x = a^*y + b^* \tag{65}$$

A figura 40 mostra as regiões onde utilizar cada uma destas formas paramétricas para representar as retas.

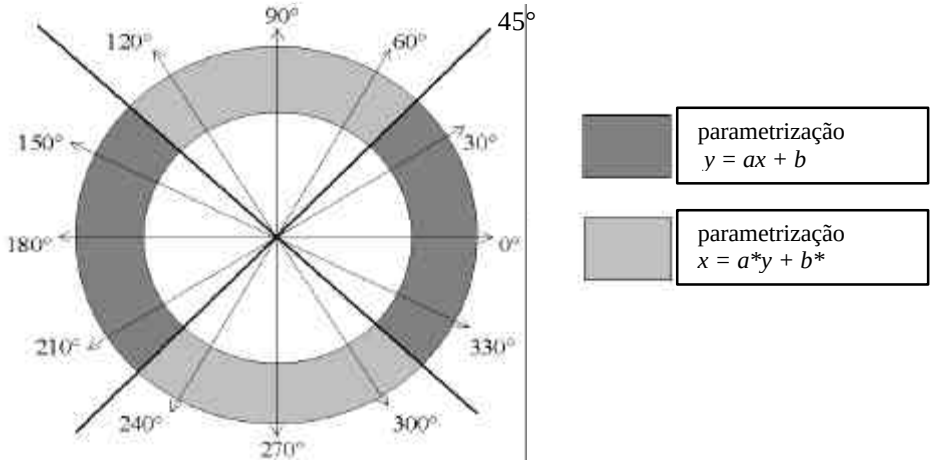


Figura 40 – Quantização utilizada no processo de detecção de bordas, e demais etapas do processo de vetorização, e as parametrizações mais convenientes usadas em cada região.

2.7.2 Problema com o M.M.Q.

Os filtros para detecção de bordas implementados (Sobel e Nevatia e Babu) não apresentam um bom comportamento nos cantos das feições, e isto precisa ser levado em consideração no processo de ajustamento da reta sobre o conjunto de pixels da borda. A figura 41 mostra a reta que será ajustada sobre um dado conjunto de pixels pertencentes a uma borda.

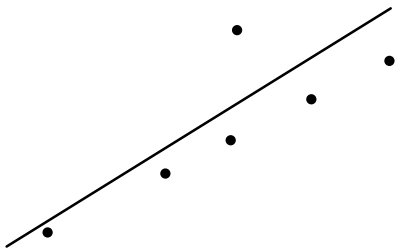


Figura 41 - Linha reta obtida com o Método dos Mínimos Quadrados.

A figura 42(a) mostra o detalhe de um canto de uma feição; em (b) tem-se a borda obtida no processo de detecção (usando o detetor de Nevatia e Babu – após a etapa de afinamento de borda), onde é possível notar que os pixels da borda horizontal sofreram um deslocamento.

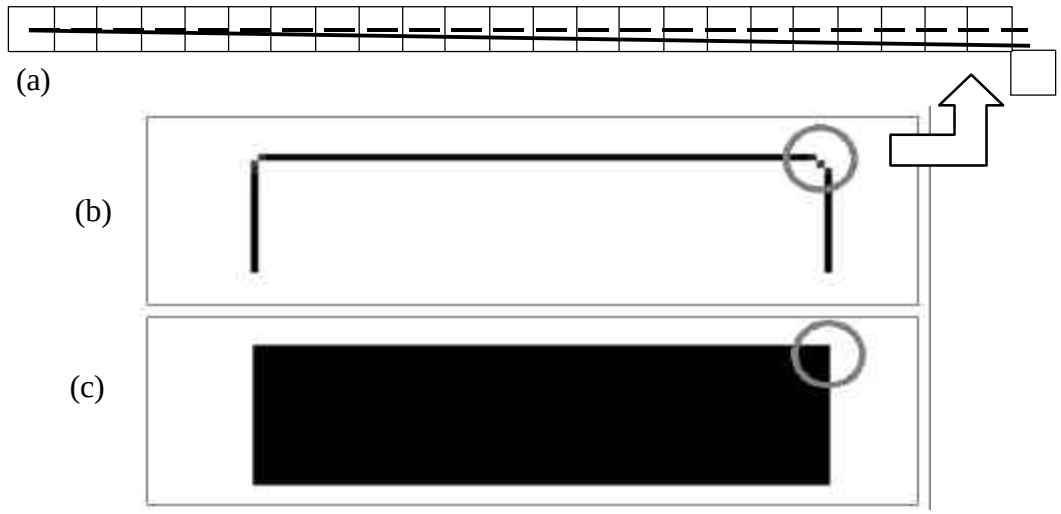


Figura 42 – Problema apresentado pelo detetor de bordas (Nevatia e Babu) nos cantos das feições.

Na figura 42, os pixels identificados como pertencentes à borda horizontal (b) em destaque apresentam o posicionamento que aparece em detalhe em (a). A linha tracejada é a linha da borda procurada, e a linha contínua é a que será obtida utilizando o Método dos Mínimos Quadrados.

Neste caso, como os erros não possuem uma distribuição normal, (Jain et al., 1995) o método dos mínimos quadrados tal como definido anteriormente não é uma boa opção. Uma possível solução para este caso é utilizar a mediana mínima quadrada (a mediana é um estimador mais robusto que a média, para definir um valor central (Press, 1992). A figura 43 permite uma comparação entre os resultados obtidos com a utilização destes dois métodos, para o mesmo conjunto de pixels apresentados na figura 42.

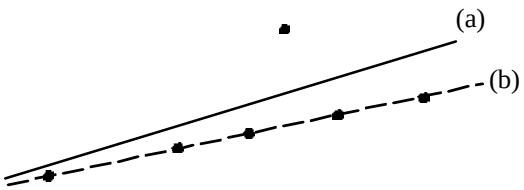


Figura 43 – (a) reta obtida com o método dos mínimos quadrados e (b) reta obtida utilizando a condição da mediana mínima quadrada.

Enquanto o método dos mínimos quadrados é amplamente conhecido e largamente utilizado, o método que utiliza como condição a ser satisfeita a mediana mínima quadrada (também conhecido como método dos mínimos quadrados com detecção de erros) não é tão conhecido. A figura 44 mostra o algoritmo para este método.

Algoritmo → Regressão Mediana Mínima Quadrada.

Assuma que existem n pixels e p parâmetros no modelo linear
1 – Escolha p pixels de forma aleatória do conjunto de n pixels;
2 – Calcule a distância do modelo até os p pixels;
3 – Calcule a mediana dos quadrados das distâncias;
Repita os passos 1-2-3 até que seja encontrada uma medida com a mediana suficientemente pequena, ou um número pré definido de reamostragens tenha sido atingido.

Figura 44 – Algoritmo para a regressão utilizando a mediana mínima quadrada.

2.7.3 Aproximação de poligonais

Devido a problemas de *Aliasing*, já apresentados anteriormente, ocorre a necessidade de se conectar pixels de borda de direções diferentes. Nos processos posteriores, dependendo da curvatura dos segmentos detectados, o ajustamento não deverá funcionar adequadamente. Uma solução que pode ser adotada para resolver este problema é a quebra do segmento (*splitting*), em dois menores. Entre os extremos dos segmento é traçada uma linha reta, e o pixel do segmento que apresentar a maior distância (euclidiana) a esta linha será o pixel de quebra. Para os dois novos segmentos também pode ser aplicado o mesmo procedimento, caso necessário, e o processo todo pode se repetir, até que os segmentos obtidos atendam as necessidades de linearidade exigidas (após o ajustamento apresentem um pequeno resíduo). A figura 45 mostra este método sendo aplicado em um segmento de borda. Após a quebra de um segmento, utilizando este método, os segmentos resultantes podem ser reconectados com outros segmentos nas proximidades, pela repetição da etapa de conexão de segmentos colineares, o que caracteriza a operação de reconexão (*merge*), caso seja atendida a condição apresentada na figura 36.

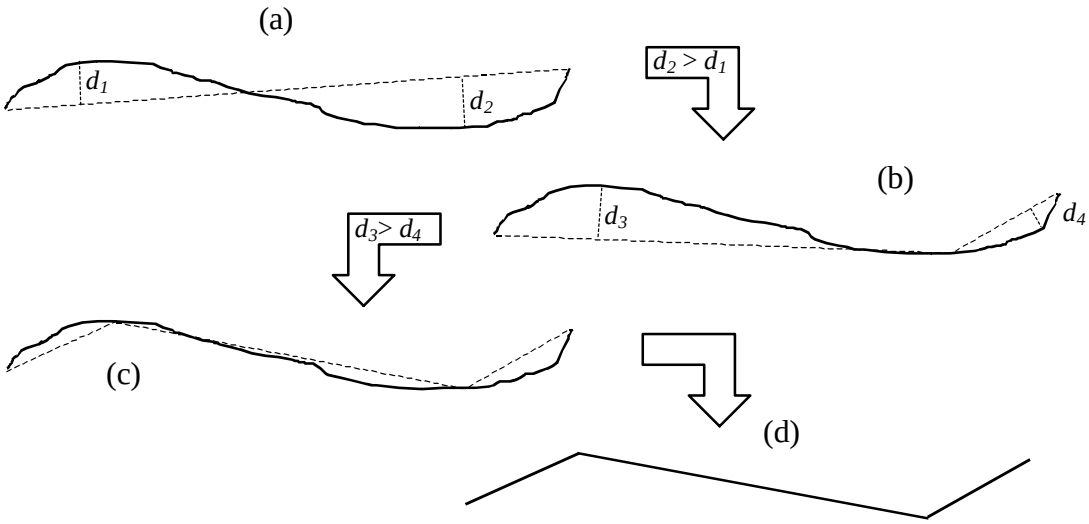


Figura 45 – (a) Segmento original, (b) dois segmentos obtidos após quebra no pixel de d_2, (c) três segmentos obtidos após quebra no pixel de d_3, (d) três segmentos lineares obtidos.

CAPITULO 3

IMPLEMENTAÇÃO E TESTE DE UM AMBIENTE PARA A EXTRAÇÃO DE LINHAS RETAS

3.1 O SISTEMA DESENVOLVIDO

O sistema desenvolvido para a realização dos experimentos foi implementado utilizando a linguagem C++, sendo utilizado o ambiente de programação C^{++}Builder, da Borland International, Inc. A escolha por tal ferramenta se justificou por ser um ambiente de alta produtividade, que tem incorporados os mais avançados recursos de programação (programação orientada a objetos e eventos) e também por oferecer facilidades para a manipulação de imagens (leitura e escrita do arquivo, e também acesso aos elementos da imagem – *pixels*) de uma forma bem simples.

Com o objetivo de permitir um acompanhamento dos processamentos realizados (análise visual dos resultados), o sistema foi projetado com três áreas de imagens, cada uma com uma resolução de 300 por 220 pixels. A primeira área é destinada a entrada de imagens, sendo referenciada neste trabalho como Imagem de Entrada (*ImE*). Nesta área pode ser carregada uma imagem diretamente do disco, ou ainda, uma cópia da imagem que no momento está na área três ou dois. A segunda área de imagem é destinada ao armazenamento temporário de imagens (*swap*), e a terceira área de imagem é utilizada para apresentar os resultados obtidos com o processamento da imagem de entrada. A figura 46 mostra graficamente a estrutura adotada, e também o menu de operações construído no programa desenvolvido.

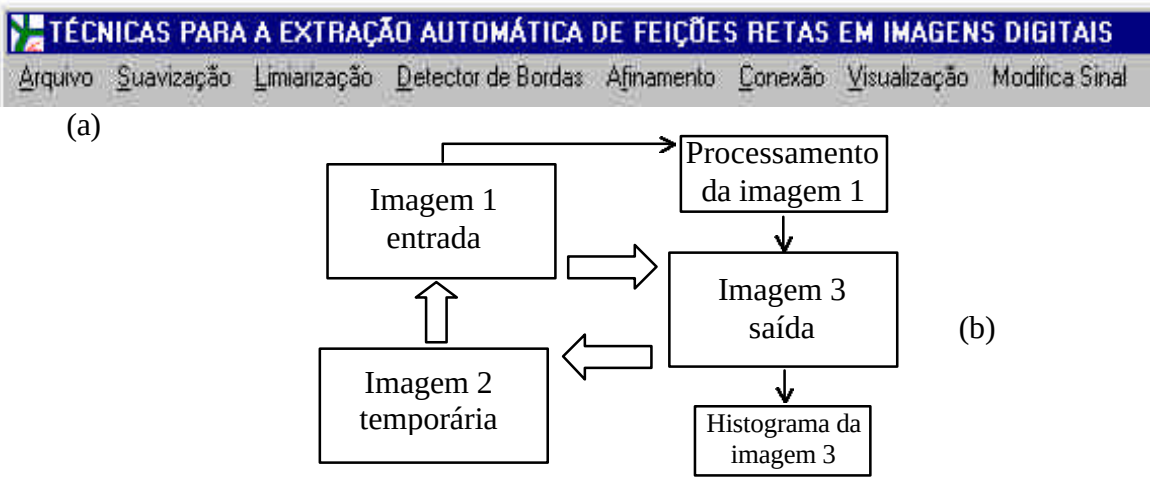


Figura 46 – (a) Menu de operações do sistema desenvolvido e (b) detalhe do fluxo de processamento das imagens.

Todas as áreas de imagens apresentadas na figura 46 apresentam recursos para ampliação (*zoom*), pelos fatores dois, quatro e seis. Este recurso se mostra muito importante na etapa de análise visual dos resultados.

As imagens nas três áreas podem ser salvas em disco, e também podem ser carregadas a partir do disco, embora o fluxo mais comum seja fazer a carga na imagem de entrada. O carregamento da imagem de saída é interessante quando se deseja fazer uma comparação visual dos resultados obtidos com a vetorização e, neste caso, carrega-se na imagem de saída a imagem original e em seguida projeta-se os vetores sobre ela. A figura 47 mostra a utilização deste recurso.

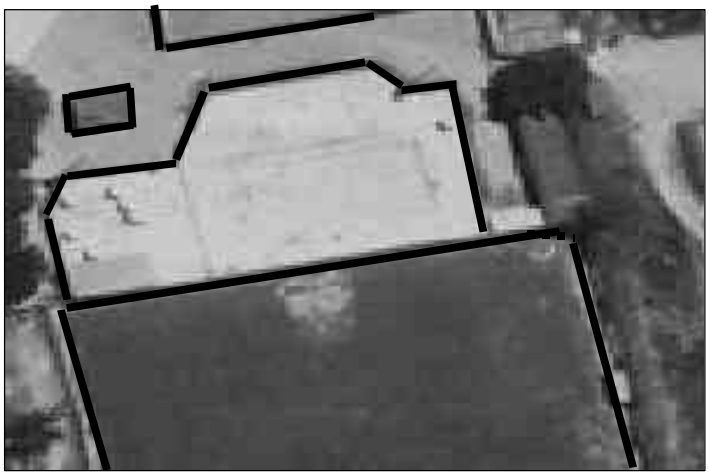


Figura 47 – Imagem original sobreposta pelos vetores obtidos no processamento (linha tracejada).

Conforme mostra a figura 46(a), o sistema desenvolvido tem implementado as rotinas para efetuar os processamentos que são mostrados na figura 48, a qual mostra também a seqüência de experimentos que podem ser realizados. Observando esta figura verifica-se que é possível a obtenção de 144 diferentes combinações das etapas no processamento.

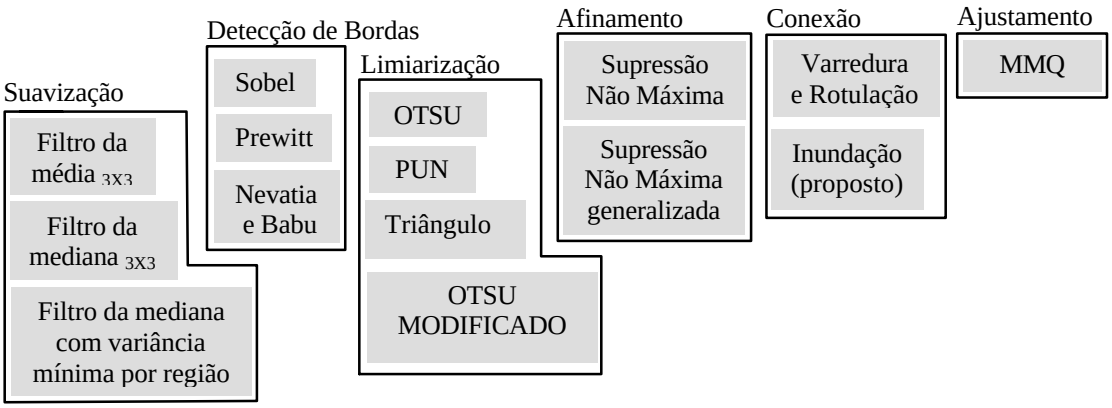


Figura 48 – Etapas a serem verificadas no processo de detecção de feições lineares.

3.2 CONSTRUÇÃO DA IMAGEM PADRÃO

O sistema oferece a opção para gerar a imagem padrão utilizada neste trabalho, desenhando o contorno de três polígonos que podem ou não serem preenchidos. A opção de manter os polígonos sem o preenchimento é útil para a análise de resultados, no momento de se verificar quantos pixels de borda foram realmente detectados, pois são conhecidas as verdadeiras posições dos pixels de borda e também as posições obtidas após o processamento.

3.3 MODIFICAÇÃO DO SINAL

Nesta opção estão disponíveis algumas operações que podem ser utilizadas para melhorar a visualização dos resultados obtidos, ou mesmo para melhorar o resultado final, quando tais aplicações são utilizadas entre uma etapa e outra.

Aumenta – Esta operação aumenta o nível do sinal, multiplicando por dois todas as magnitudes dos pixels da imagem de entrada (não permitindo que os valores

ultrapassem o valor máximo – 255), ou seja, dada a imagem de entrada *ImE*, os valores dos pixels da imagem de saída *ImS* são dados por:

$$\mathrm{Im}S = \begin{cases} 2.\mathrm{Im}E & se\ \mathrm{Im}E < 128 \\ 255 & caso\ contrário \end{cases} \tag{66}$$

Diminui – Esta operação diminui o nível do sinal, dividindo por dois todas as magnitudes dos pixels da imagem de entrada, assim, dada a imagem de entrada *ImE*, os valores dos pixels da imagem de saída *ImS* são dados por:

$$\mathrm{Im}S = \begin{cases} \dfrac{\mathrm{Im}E}{2} & se\ \mathrm{Im}E > 1 \\ 0 & caso\ contrário \end{cases} \tag{67}$$

Auto-escala – Esta operação maximiza a distribuição das magnitudes dos pixels na faixa de 0 à 255 (*stretch*). Neste caso, dada a imagem de entrada *ImE*, os valores dos pixels da imagem de saída *ImS* são definidos por:

$$\mathrm{Im}S = \frac{\text{Valor máximo da escala}}{\mathrm{Im}E_{\max} - \mathrm{Im}E_{\min}}(\mathrm{Im}E - \mathrm{Im}E_{\min}) \tag{68}$$

Equalização do Histograma – Trata-se de uma operação que tenta melhorar a distribuição dos tons de cinza da imagem. Nos cálculos são utilizadas a freqüência de cada tom de cinza na imagem (geralmente utiliza-se um histograma) e também o valor da freqüência acumulada, que consiste na soma de todas as freqüências até o valor do brilho atual. Os novos valores dos pixels da imagem equalizada são dados por (69) (Low, 1991).

$$\text{valor equalizado} = \max\left[0, round\left(\frac{\text{Nº Tons de cinza total x freq. acumulada}}{\text{Nº de Colunas x Nº de Linhas}}\right) - 1\right] \tag{69}$$

Conversão para tons de cinza – Esta operação faz a conversão de imagens coloridas, com suas componentes de cor R, G e B, para imagens de tons de cinza, através do cálculo da luminância dos pixels (Gonzalez, 1993), dada por (70).

$$Luminância = 0,299.R + 0,587.G + 0,114.B \quad (70)$$

Inverte Imagem – Resulta no negativo da imagem, sendo algumas vezes utilizada para uma melhor visualização de alguns detalhes. Dada a imagem de entrada *ImE*, os valores dos pixels da imagem de saída *ImS* são dados por (71).

$$ImS = 255 – ImE \quad (71)$$

3.4 ETAPA DE SUAVIZAÇÃO

Nesta etapa estão implementados os filtros da média (máscaras 3x3 e 5x5), mediana e mediana com variância mínima por região. O filtro da média é o que apresenta uma maior suavização, porém atenuando muito a informação de borda (principalmente quando se utiliza a máscara 5x5). O filtro da mediana apresenta um bom resultado, eliminando as altas freqüências (ruídos e excesso de detalhes que dificultam a detecção das feições mais importantes) e ainda preservando as bordas. O terceiro tipo é uma variação do filtro da mediana, e apresenta como vantagem a utilização de uma vizinhança maior (ver seção 2.2.3). Este filtro define nove regiões ao redor do pixel sob análise, verificando a sua semelhança com os demais em cada região; através da análise da variância define-se a provável região à qual o mesmo deve pertencer (região de menor variância), e o valor assumido para o pixel é a mediana dos pixels da região selecionada. Visualmente percebe-se uma boa suavização, e ainda uma excelente preservação das bordas. A figura 49 mostra os resultados obtidos com a aplicação destes filtros.

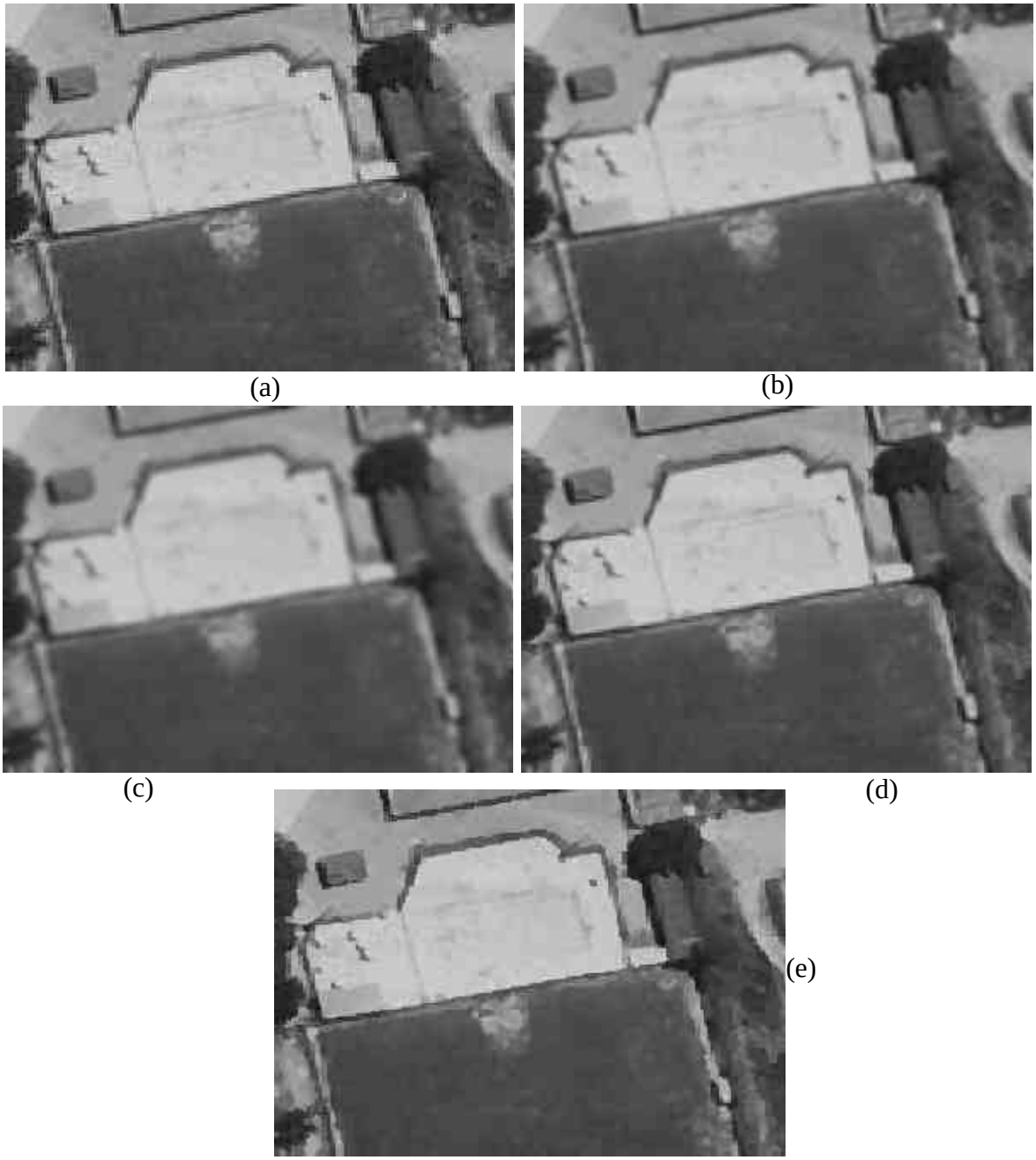
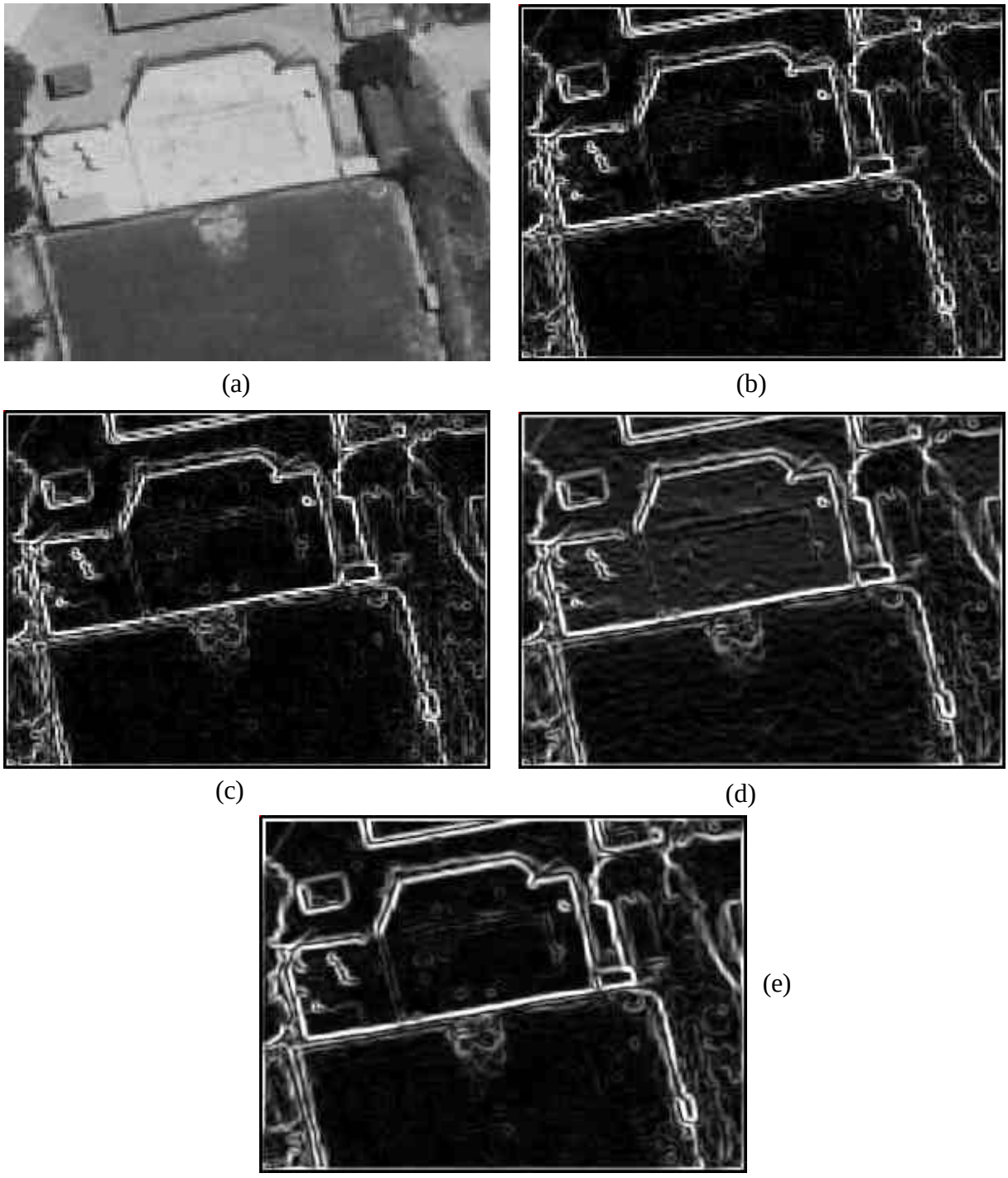


Figura 49 – (a) Imagem original, (b) resultado com a aplicação do filtro da média$_{3x3}$, (c) filtro da média$_{5x5}$, (d) Filtro da mediana e (e) Filtro da mediana com variância mínima por região.

3.5 ETAPA DE DETECÇÃO DE FEIÇÕES

Com o objetivo de detectar os pixels de borda foram implementados os operadores de Prewitt$_{3x3}$, Sobel$_{3x3}$, Sobel$_{5x5}$ e Nevatia e Babu. Os resultados com a aplicação destes quatro operadores podem ser visualizados na figura 50.



(a) (b)

(c) (d)

(e)

Figura 50 – (a) Imagem original, (b) resultado com a aplicação do operador de $Prewitt_{3x3}$, (c) Operador de $Sobel_{3x3}$, (d) Operador de $Sobel_{5x5}$, e (e) Operador de Nevatia e Babu.

Em áreas de muito ruído nota-se uma sensível diferença nos resultados obtidos pelos operadores. A figura 51 mostra os resultados obtidos pelos operadores de Sobel e Nevatia e Babu em uma imagem sintética com a presença de forte ruído.

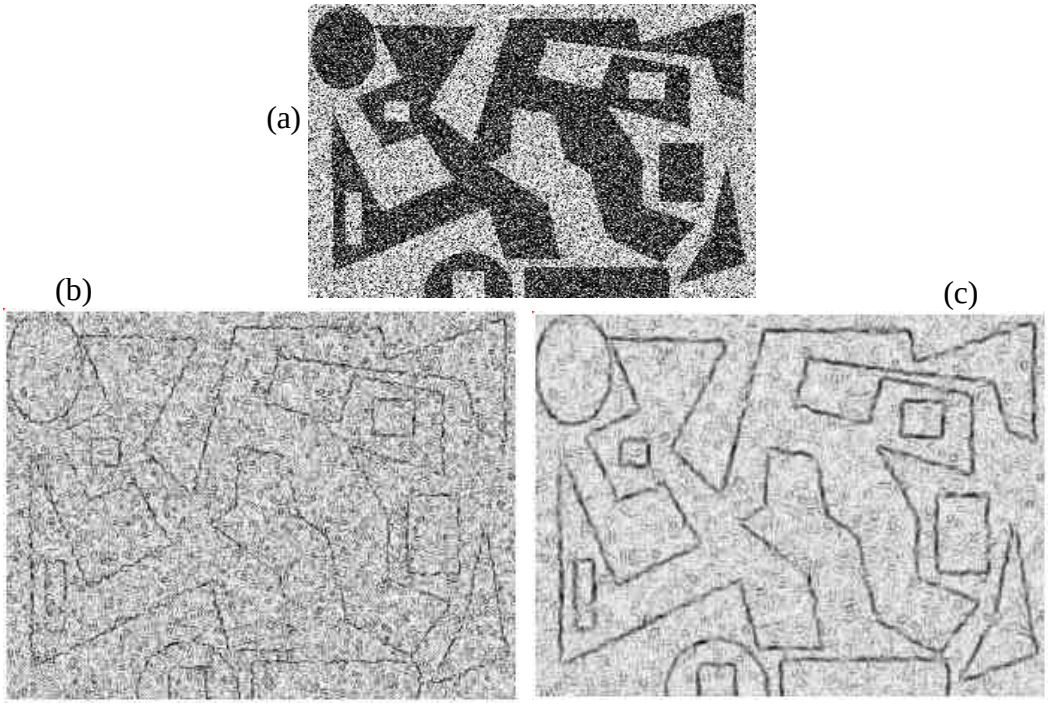


Figura 51 – (a) imagem original, com muito ruído, (b) resultado da detecção utilizando o Operador de Sobel e (c) Operador de Nevatia e Babu. (Artero e Tommaselli, 1999).

Os bons resultados obtidos com o operador de Nevatia e Babu motivaram o desenvolvimento de uma técnica para a construção das máscaras em tempo de execução. Assim, torna-se possível a obtenção de máscaras específicas para a detecção de direções estabelecidas durante o processamento. No caso de um processo de detecção assistida, a partir do conhecimento da direção de alguma borda, outras podem ser detectadas utilizando-se de uma máscara específica para a direção desejada (extração seletiva). A seguir é descrito um método para a construção das máscaras de Nevatia e Babu, que apresenta bons resultados.

3.5.1 Método para a construção das máscaras de Nevatia e Babu

O processo descrito a seguir supõe que as máscaras são definidas de forma que uma célula tem o valor dado pela expressão:

$$Valor = Área\ sob\ a\ reta - Área\ \ sobre\ a\ reta \tag{72}$$

sendo que a reta considerada deve ser posicionada de forma a passar pelo centro da máscara, e possuir inclinação α a ser detectada.

A figura 52 mostra, graficamente, a proposta de implementação das máscaras.

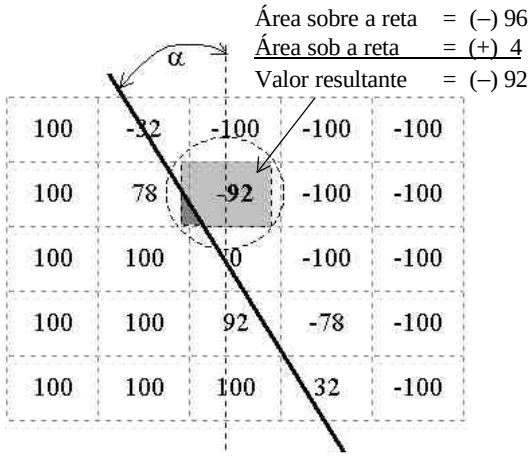


Figura 52 – Método implementado para a construção de máscaras de Nevatia e Babu.

Por este método, o cálculo das máscaras fica totalmente reduzido ao cálculo das áreas das células (acima e abaixo da linha reta - que é definida pelo ângulo da borda a ser detectada). Neste trabalho, para a obtenção de uma máscara *m* x *n*, foi utilizada a seguinte geometria (após um giro de 90 graus), mostrada na figura 53.

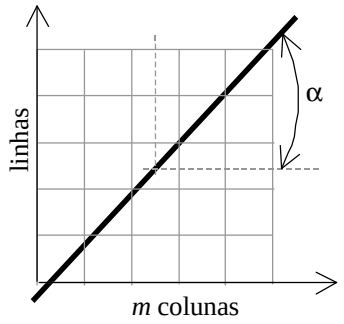


Figura 53 – Geometria utilizada no método.

Lembrando que a equação da reta é dada por:

$$y = ax + b \tag{73}$$

com a e b dados por :

$$a = arctan(a) \tag{74}$$

e

$$b = -ax_c + y_c \tag{75}$$

onde x_c e y_c são as coordenadas do centro da máscara a ser obtida, dadas por:

$$x_c = \frac{m}{2}k - 0.5 \tag{76}$$

e

$$y_c = \frac{n}{2}k - 0.5 \tag{77}$$

Em (76) e (77), m e n são as dimensões da máscara, e para $m \neq n$ são produzídas máscaras retangulares. A constante k define a quantidade de pontos que estão dentro de cada célula, e serve para controlar a qualidade dos resultados obtidos por este método, sendo que os testes executados mostraram que valores adequados são obtidos com k>100.

O método sugerido neste trabalho, para o cálculo das áreas, é bastante simples e assemelha-se à forma como as pessoas fariam para obtê-lo, caso dispusessem de uma grade (pixels), onde fosse possível contar o número de elementos que se situam acima e abaixo da reta, cuja equação é conhecida em (73).

A figura 54 mostra como os elementos de cada célula são classificados como acima e abaixo da reta.

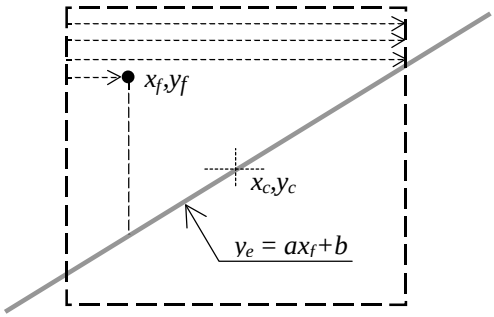


Figura 54– Determinação da posição dos elementos acima e abaixo da reta.

Na figura 54, para todo elemento x_f,y_f da máscara, calcula-se $y_e = ax_f + b$, e se $y_e > y_f$ então o elemento está abaixo da reta, e deve ser somado à área total da célula. Caso $y_e < y_f$ então o elemento está acima da reta, e precisa ser subtraído da área calculada. O valor total a ser atribuído à célula é dado por:

$$Célula = Área\ total\ acumulada\ /\ k^2 \tag{78}$$

sendo *k* o mesmo valor definido anteriormente.

Uma boa melhora no processamento pode ser obtida testando inicialmente todos os pontos limites de uma célula (canto superior esquerdo e direito, e canto inferior esquerdo e direito) e, caso todos estes se encontrem acima ou abaixo da reta, a célula receberá imediatamente o valor –100 e 100 respectivamente..

A figura 55 mostra as máscaras obtidas por este algoritmo, permitindo uma comparação com as máscaras da figura 14 (seção 2.3.6).

$\frac{1}{1000}$

100	100	0	-100	-100
100	100	0	-100	-100
100	100	0	-100	-100
100	100	0	-100	-100
100	100	0	-100	-100

0 graus

$\frac{1}{1000}$

-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100

180 graus

$\frac{1}{1098}$

100	-30	-100	-100	-100
100	76	-92	-100	-100
100	100	0	-100	-100
100	100	92	-76	-100
100	100	100	30	-100

30 graus

$\frac{1}{1098}$

-100	30	100	100	100
-100	-76	92	100	100
-100	-100	0	100	100
-100	-100	-92	76	100
-100	-100	-100	-30	100

210 graus

$\frac{1}{1098}$

-100	-100	-100	-100	-100
30	-76	-100	-100	-100
100	92	0	-92	-100
100	100	100	76	-30
100	100	100	100	100

60 graus

$\frac{1}{1098}$

100	100	100	100	100
-30	76	100	100	100
-100	-92	0	92	100
-100	-100	-100	-76	30
-100	-100	-100	-100	-100

240 graus

$\frac{1}{1098}$

-100	-100	-100	-100	-100
-100	-100	-100	-100	-100
0	0	0	0	0
100	100	100	100	100
100	100	100	100	100

90 graus

$\frac{1}{1098}$

100	100	100	100	100
100	100	100	100	100
0	0	0	0	0
-100	-100	-100	-100	-100
-100	-100	-100	-100	-100

270 graus

$\frac{1}{1098}$

-100	-100	-100	-100	-100
-100	-100	-100	-76	30
-100	-92	0	92	100
-30	76	100	100	100
100	100	100	100	100

120 graus

$\frac{1}{1098}$

100	100	100	100	100
100	100	100	76	-30
100	92	0	-92	-100
30	-76	-100	-100	-100
-100	-100	-100	-100	-100

300 graus

$\frac{1}{1098}$

-100	-100	-100	-30	-100
-100	-100	-92	76	-100
-100	-100	0	100	100
100	-76	92	100	100
100	30	100	100	100

150 graus

$\frac{1}{1098}$

100	100	100	30	100
100	100	92	-76	100
100	100	0	-100	-100
-100	76	-92	-100	-100
-100	-30	-100	-100	-100

330 graus

Figura 55 – Máscaras geradas pelo método proposto.

A seguir, são apresentadas na figura 56 algumas máscaras construídas por este método para a detecção de retas nas direções 10 e 20 graus (diferentes daquelas apresentadas por Nevatia e Babu). O denominador do fator multiplicativo é definido pela soma dos elementos positivos em cada uma das máscaras obtidas.

$\frac{1}{1105}$

100	100	-70	-100	-100
100	100	-35	-100	-100
100	100	0	-100	-100
100	100	35	-100	-100
100	100	70	-100	-100

10 graus

$\frac{1}{1125}$

100	54	-100	-100	-100
100	99	-72	-100	-100
100	100	0	-100	-100
100	100	72	-99	-100
100	100	100	-54	-100

20 graus

Figura 56 – Máscaras geradas pelo software desenvolvido.

O método também permite a obtenção de máscaras com diferentes dimensões, o que pode ser útil para a detecção de linhas maiores, e consequentemente facilitar o processamento na etapa posterior (conexão).

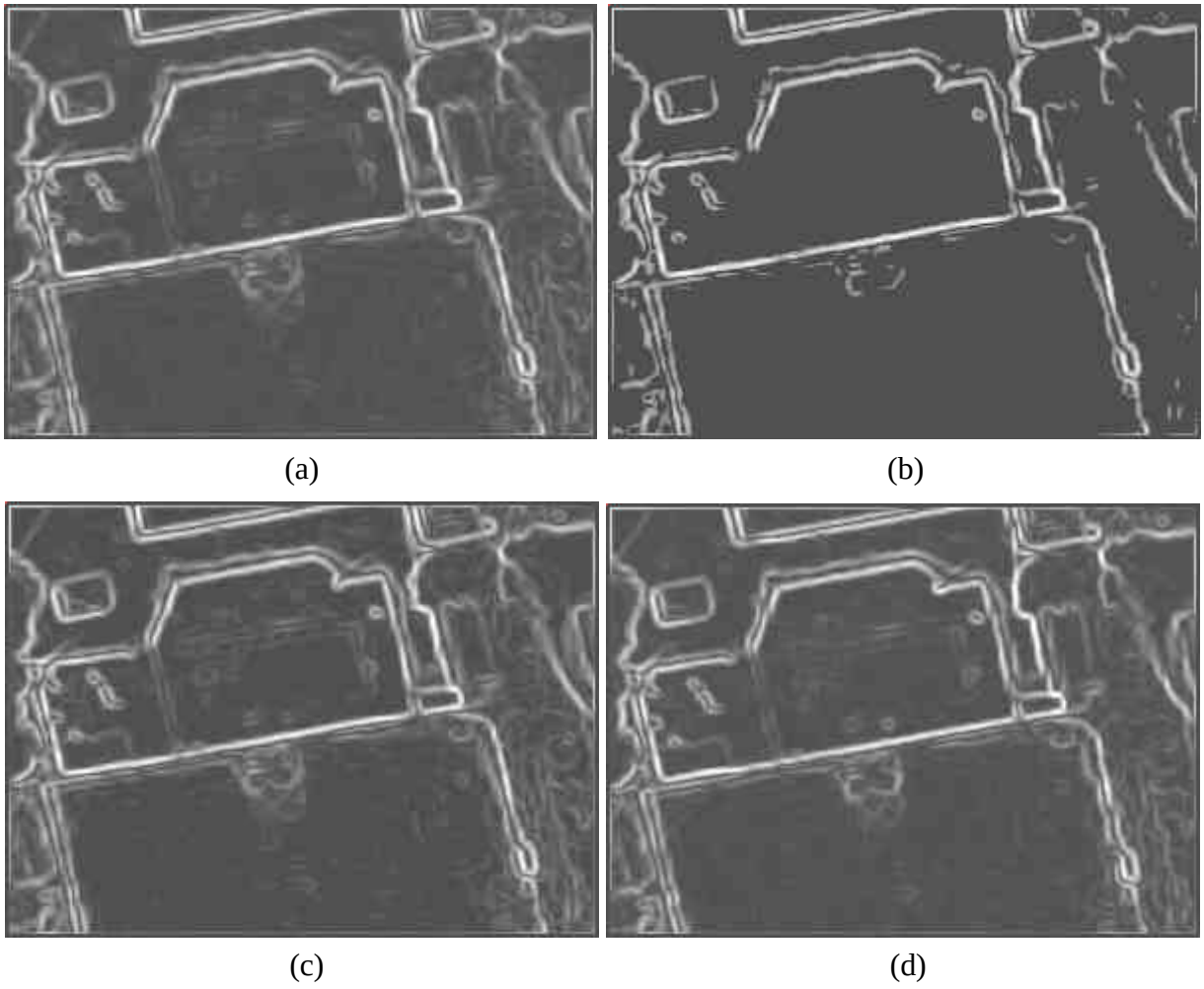
3.6 ETAPA DE LIMIARIZAÇÃO

Para a realização desta etapa o sistema tem implementado os métodos de Otsu, Pun (Parker, 1996) e o Método do Triângulo. Tais métodos de limiarização são caracterizados como do tipo global, ou seja, para todos os pixels da imagem é definido um único limiar.

A figura 57 mostra os resultados obtidos com a aplicação dos métodos de Otsu, Pun e do Triângulo. Para o caso destas imagens, o método de Otsu apresenta bons resultados. O algoritmo de Otsu e Pun utilizados neste trabalho foram implementados a partir de Parker (1996), que inclui o código fonte em um CD-Rom.

Devido a presença de áreas escuras e claras nas imagens, verifica-se que é uma tarefa bastante difícil a determinação de um único limiar, que apresente bons resultados (Gonzalez, 1993). Com o objetivo de trabalhar melhor com tais peculiaridades, foram desenvolvidos vários métodos para a determinação de um valor de limiar, que agem localmente. Nestes métodos os valores obtidos são utilizados apenas no centro da região utilizada para a obtenção do valor. Um método que se baseia nesta idéia (Parker, 1996) é o método de Chow & Kaneko, que sugere a divisão da imagem em 49 regiões (com

sobreposição), cada uma das quais com 64 x 64 pixels (para uma imagem de 256 x 256 pixels).



(a) (b)

(c) (d)

Figura 57 – (a) Imagem original de bordas, (b) resultado obtido com o método de Otsu, (c) Método de Pun e (d) Método do Triângulo.

Com o objetivo de apresentar os resultados que podem ser obtidos com a aplicação de um método local, neste trabalho está sendo implementada uma variação do método de Otsu, de forma a trabalhar em regiões.

3.6.1 Método de Otsu modificado

O objetivo inicial da proposta seria para cada pixel da imagem calcular o valor de um limiar, obtido apenas para uma pequena região ao redor do mesmo.

Devido ao grande processamento que isto implicaria, uma nova estratégia foi desenvolvida, e assim, o método determina o valor do limiar utilizando os pixels pertencentes a uma janela 60 x 60, e o valor obtido é utilizado para todos os pixels localizados na região central desta janela (20 x 20). A figura 58 mostra a construção destas regiões, e observa-se que na implementação, algumas regiões da borda da imagem precisam ser diferenciadas, pois não possuem todos os blocos vizinhos. Na rotina desenvolvida foram observadas as 9 situações indicadas na figura 58.

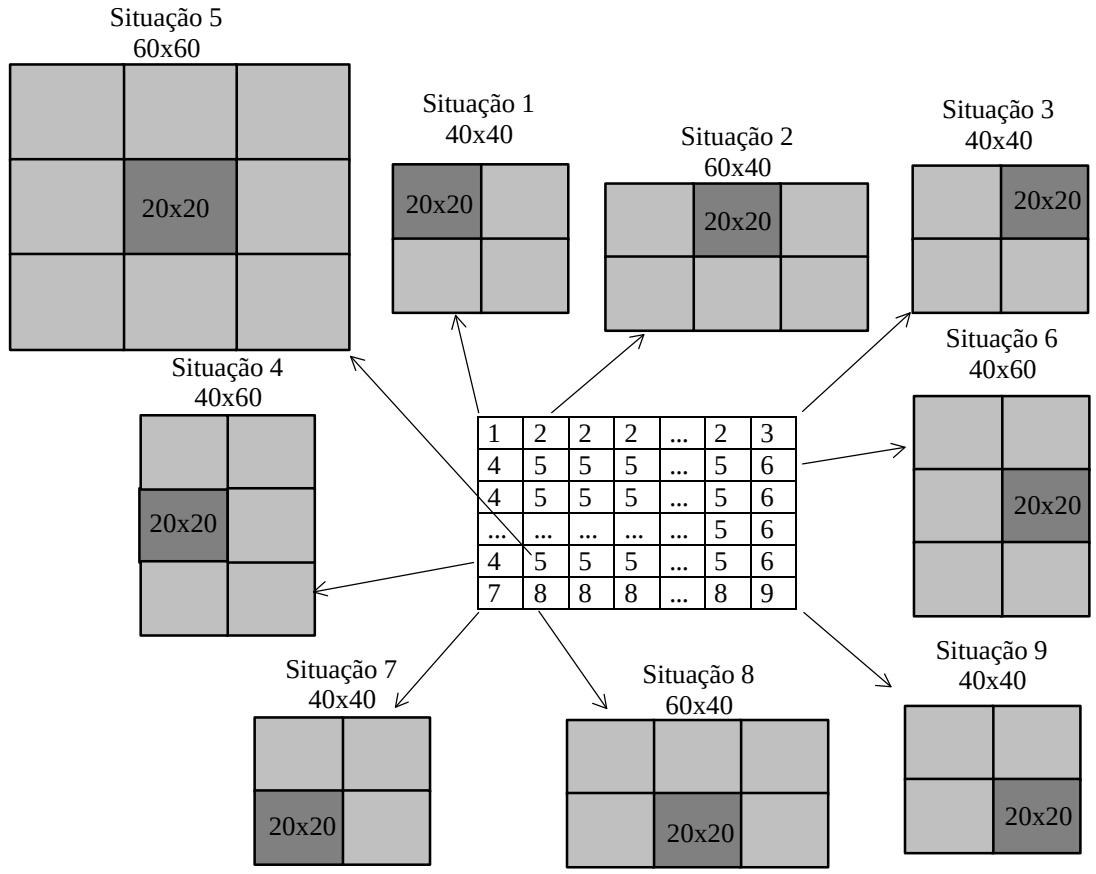
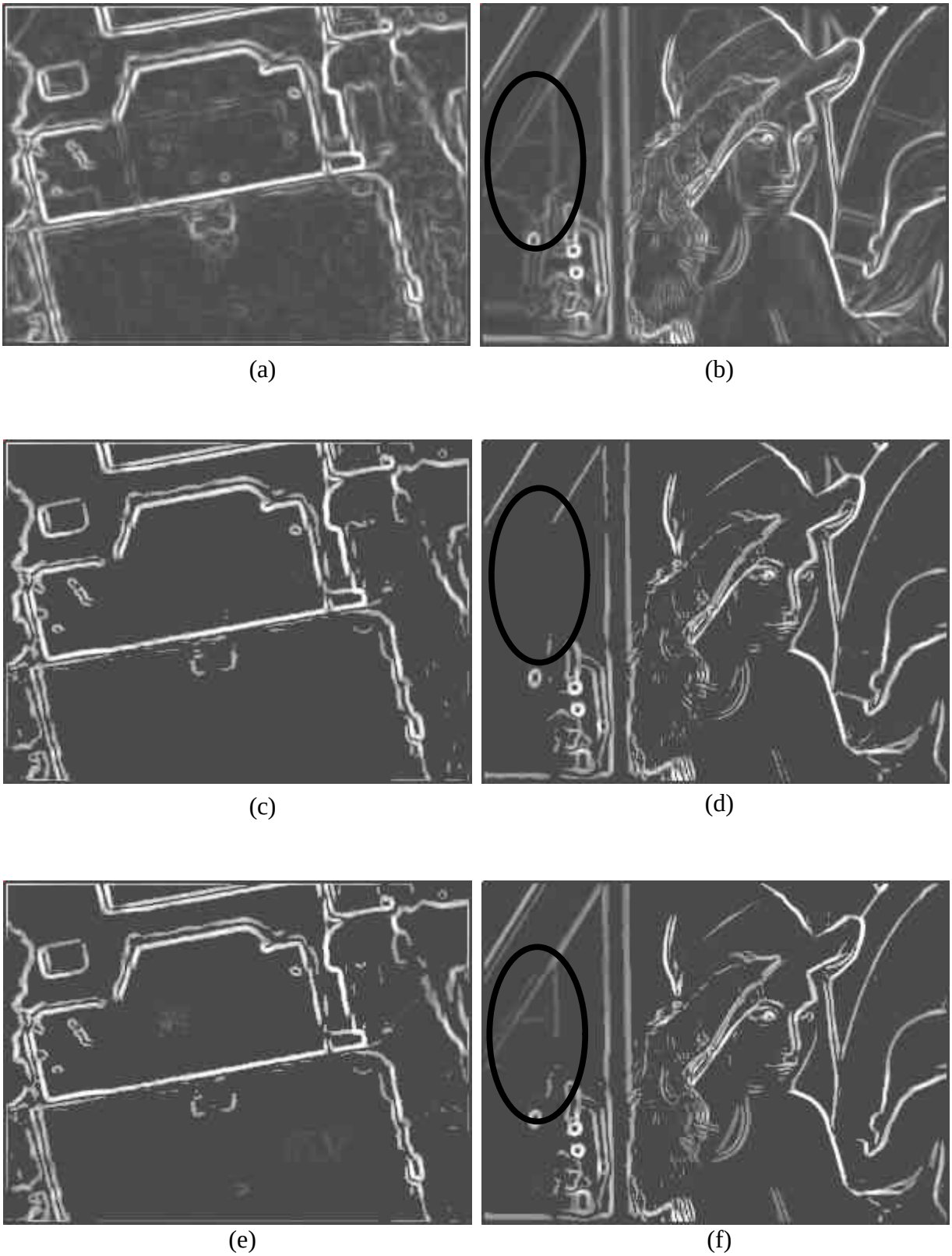


Figura 58 – Formação das regiões a serem utilizados no processo de limiarização local (Otsu modificado).

A figura 59 mostra a diferença entre os resultados obtidos com a aplicação dos métodos de Otsu e do Método de Otsu Modificado (proposto).

A figura 59 (d) mostra que algumas linhas importantes são removidas da imagem original (b) com a utilização do Método de Otsu (Global), sendo, no entanto, preservadas, conforme mostra (f) com a aplicação do Método de Otsu modificado, que atua de forma local.



(a) (b)

(c) (d)

(e) (f)

Figura 59 – (a),(b) Imagens originais, (c),(d) resultados obtidos com a aplicação do Método de Otsu e (e) e (f) Resultados obtidos com o Método de Otsu modificado (proposto).

3.7 ETAPA DE AFINAMENTO DE BORDAS

Após a eliminação dos pixels de borda com pequenas magnitudes pela etapa de limiarização, as bordas precisam passar por uma etapa de afinamento, devendo, após isto, restarem apenas segmentos com uma largura mínima (apenas a largura de um pixel). Além do método de afinamento por supressão não máxima, neste trabalho também está implementada uma generalização desta técnica. Por este novo método, não existe a necessidade de discretizar as direções dos pixels de bordas, conforme ocorre com o método da supressão não máxima original (em 8 ou 12 direções), conforme apresentado na seção 2.5.1 (figura 20).

3.7.1 Supressão não máxima generalizada

Conforme apresentado na seção 2.5.1, o método de afinamento de linhas por supressão não máxima trabalha com apenas oito direções, caracterizando uma grande discretização dos valores das direções de bordas. Com o objetivo de investigar possíveis melhoras que podem ser obtidas dispensando esta discretização dos ângulos, está sendo proposto um método onde a rotina de afinamento trabalha com ângulos de qualquer valor, determinando em tempo real quais pixels devem ser investigados. Nos casos em que se torna necessária uma reamostragem, para a obtenção dos valores das magnitudes dos pontos a serem utilizados na comparação (realizada por meio de uma interpolação bilinear), os pontos que deverão ser utilizados também são identificados pela rotina.

A figura 60 mostra o detalhe de uma borda e também a geometria utilizada para determinar automaticamente, a partir do conhecimento da direção da borda, a localização dos pontos m_1 e m_2 que devem ter suas magnitudes comparadas com a magnitude do pixel m (pixel investigado). A figura mostra também as coordenadas destes pontos, que são obtidas por meio das coordenadas do pixel m e da informação de direção da bordas, conforme definem as expressões (79) a (82).

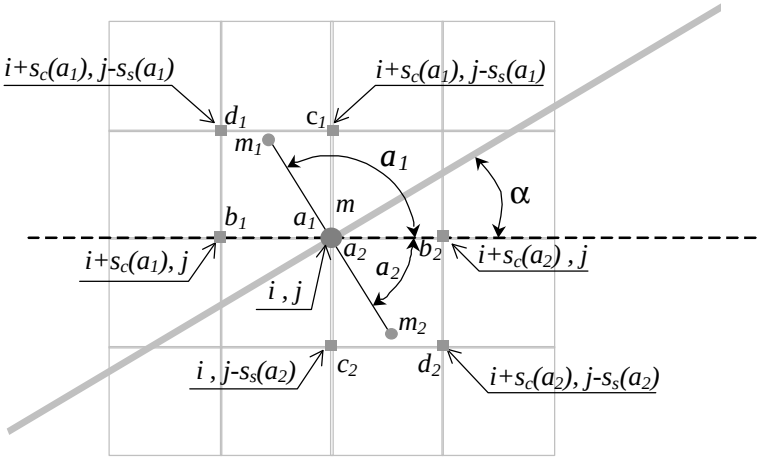


Figura 60 – Determinação dos pontos a serem verificados no processo de afinamento.

Na figura os ângulos a_1 e a_2 são dados por:

$$a_1 = a + \frac{p}{2} \tag{79}$$

e

$$a_2 = a - \frac{p}{2} \tag{80}$$

S_s e S_c representam o sinal dos senos e cosenos destes ângulos, tem-se:

$$s_c(x) = \begin{cases} 0 \ \ se \ \ \cos(x) = 0 \\ 1 \ \ se \ \ \cos(x) > 0 \\ -1 \ \ se \ \ \cos(x) < 0 \end{cases} \tag{81}$$

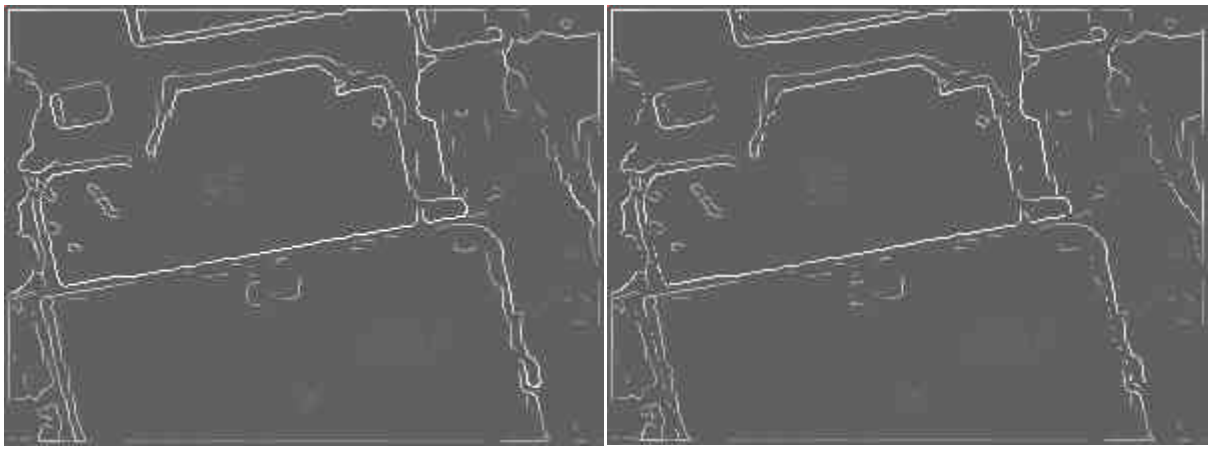
e

$$s_s(x) = \begin{cases} 0 \ \ se \ \ \text{sen}(x) = 0 \\ 1 \ \ se \ \ \text{sen}(x) > 0 \\ -1 \ \ se \ \ \text{sen}(x) < 0 \end{cases} \tag{82}$$

Após a determinação dos pixels a_1, b_1, c_1 e d_1 (conforme figura 60), a magnitude na posição do ponto m_1 pode ser determinada por meio de uma interpolação bilinear,

conforme apresentado na seção 2.5.1. O mesmo ocorre para a determinação da magnitude na posição do ponto m_2, a partir das magnitudes dos pixels a_2, b_2, c_2 e d_2.

A figura 61 mostra os resultados obtidos com a utilização do método da supressão não máxima e da supressão não máxima generalizada (proposta).



(a) (b)

Figura 61 – Resultados obtidos com a aplicação da (a) supressão não máxima e (b) supressão não máxima generalizada (proposta).

A grande vantagem deste método é que o mesmo pode ser aplicado em pixels de borda que possuem direções numa faixa de 0 à 360 graus, enquanto que no método da supressão não máxima, era preciso fazer uma discretização destes valores em apenas oito ou doze direções.

3.8 ETAPA DE CONEXÃO

Após a etapa de afinamento, os pixels de borda podem ser agrupados, formando os segmentos. Devido aos problemas que a Transformada de Hough apresenta (necessidade de grandes estruturas, e problemas de falsos picos), neste trabalho foi implementado apenas o método de Varredura e Rotulação (*Scan & Label*) (Zhou et al., 1989), (Venkasteswar, 1994) e (Tommaselli, 1999) e ainda um método proposto, que também faz uma rotulação dos pixels, porém, com a diferença de que, ao

encontrar um pixel de borda, localiza imediatamente todos os seus vizinhos por um processo de inundação (pode ser utilizada uma vizinhança 4 ou 8).

Pelo primeiro método, é realizada uma varredura na imagem da direita para a esquerda, e de cima para baixo, sendo atribuído ao primeiro pixel de borda encontrado, um rótulo de valor 1 (um). Para este mesmo pixel (pixel atual) são verificadas as vizinhanças que aparecem na figura 34 (seção 2.6.3). Conforme mostra tal figura, pixels pertencentes a uma vizinhança 8, com a mesma direção que o pixel atual recebem também o mesmo rótulo que o pixel atual. Caso não existam pixels na vizinhança 8 com a direção específica, a busca é feita em uma vizinhança maior (25). Neste caso, existindo algum pixel com a mesma direção que o pixel atual, ele recebe o mesmo rótulo que o pixel atual. Também recebe o mesmo rótulo o pixel que se localiza entre estes dois pixels, sendo considerado um pixel de ligação ou ponte.

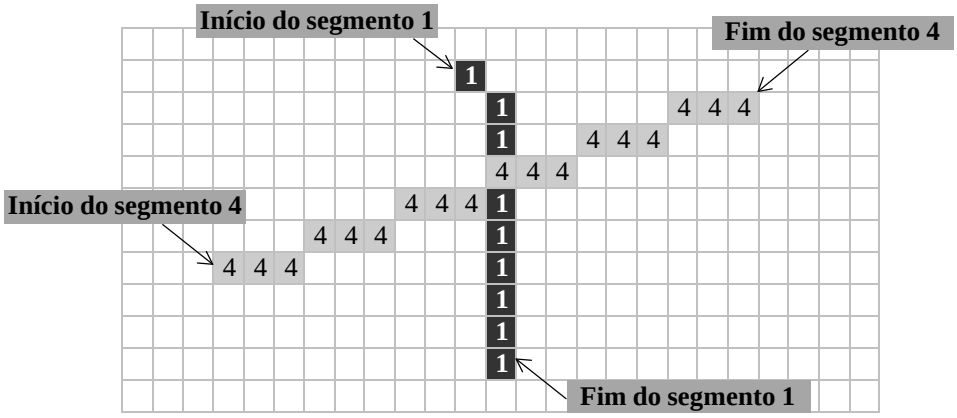


Figura 62 – Os números representam rótulos obtidos, e nota-se que o segmento de rótulos 4 cruza o segmento de rótulos 1.

Uma informação importante que precisa ser levada em consideração no desenvolvimento das rotinas seguintes é que devido ao método de varredura e rotulação trabalhar em uma vizinhança 8 e 24, podem aparecer segmentos com algumas quebras, conforme mostra a figura 62. Portanto, as demais rotinas para os processamentos posteriores, que tiverem a necessidade de percorrer os pixels de um segmento, precisam então ser projetadas para trabalhar em uma vizinhança 24, pois, caso contrário, partindo do pixel de início não se consegue atingir o pixel final.

3.8.1 Rotulação por inundação

Uma nova proposta para a conexão (*linkagem*) está sendo apresentada neste trabalho e consiste em rotular por inundação (*seed-fill*), todos os pixels vizinhos ao pixel atual (obtido por varredura) com um mesmo rótulo. Esta parece ser a forma como as pessoas fazem visualmente a identificação das linhas retas presentes em uma imagem, ou seja, identificam a reta toda no momento em que se dirige a atenção para uma parte da mesma. Diferente do Método de varredura e rotulação (Scan & Label), que vai formando as retas a partir de uma sucessão de varreduras nas linhas da imagem.

Neste método a varredura é feita no espaço imagem, apenas para impedir que algum pixel presente na imagem deixe de ser analisado. Porém, quando um pixel pertencente à uma reta é detectado, todos os pixels pertencentes a mesma são rotulados imediatamente (por inundação). Uma grande vantagem deste método é que o mesmo não apresenta o problema mostrado na figura 64 (seção 3.8.2), que o Método de Varredura e Rotulação apresenta.

Este método apresenta bons resultados na rotulação, porém a identificação dos pixels inicial e final da reta que está sendo rotulada é uma tarefa não resolvida durante o processo de rotulação, uma vez que a inundação não apresenta uma direção bem definida de caminhamento, espalhando-se praticamente aleatoriamente em uma vizinhança-8. Este problema pode ser resolvido em uma etapa posterior do processamento, como é o caso do sistema desenvolvido neste trabalho.

3.8.2 Determinação dos pixels extremos dos segmentos

Um dos resultados indesejáveis deste método está em não haver uma forma robusta para estabelecer os pixels início e fim de cada segmento. Uma possível forma de auxiliar esta tarefa no momento de construção dos segmentos, é utilizar medidas de distância para verificar se o pixel atual deve ser mesmo classificado como final ou inicial da reta (na estrutura). A figura 61 mostra o esquema utilizado para resolver este problema.

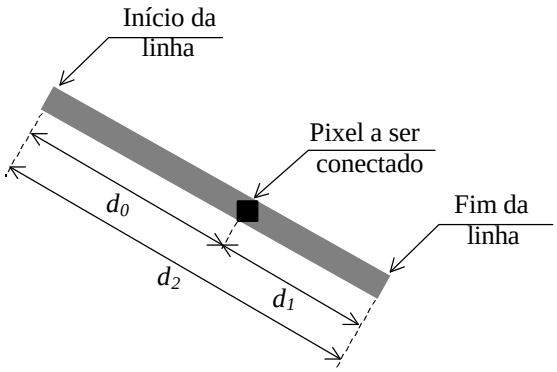


Figura 63 – Esquema utilizado para evitar problemas na definição dos pixels extremos da linha durante o Método de Varredura e rotulação ou Inundação (proposto).

Pelo esquema sugerido, d_0 é a distância entre o pixel a ser rotulado (atual) e o pixel início da reta, d_1 é a distância entre o pixel a ser rotulado (atual) e o pixel final da reta, d_2 é a distância entre os pixels início e fim da reta (comprimento da reta). Caso $d_0 > d_2$, então o pixel é rotulado e a estrutura da linha recebe suas coordenadas como o fim da linha. Caso $d_1 > d_2$ o pixel atual é rotulado e a estrutura da linha recebe suas coordenadas como o início da linha. Se $d_2 > d_1$ e $d_2 > d_0$, então o pixel atual é rotulado, porém os pixels início e fim da reta são mantidos.

Um segundo problema que ocorre no método de varredura e rotulação, pode ser visualizado na figura 64. Percebe-se que, para linhas com um inclinação entre 0^o e 30^o, apesar dos pixels constituintes da mesma terem sido detectados com uma mesma direção (na etapa de detecção), e estarem devidamente conectados em uma vizinhança 8, ainda assim o resultado da conexão resulta em um conjunto de pequenas retas, quando a linha deveria ser única.

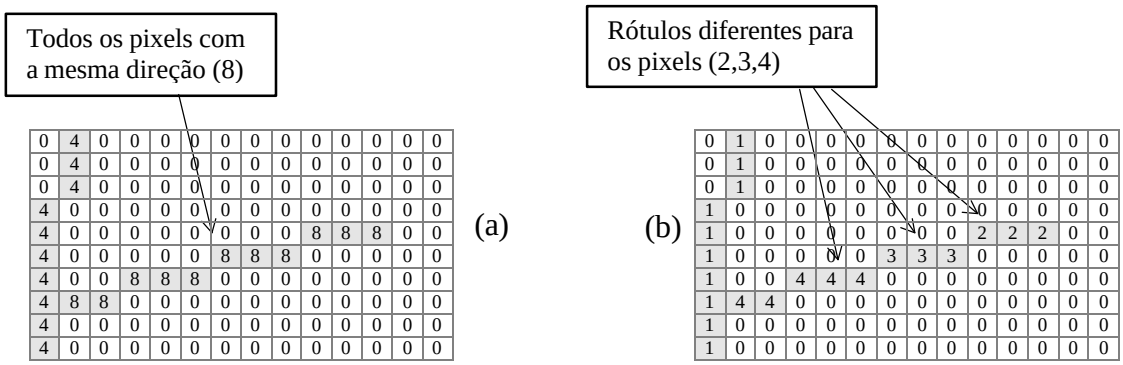


Figura 64 – (a) Direção dos pixels de uma dada reta e (b) resultado da conexão utilizando o método da varredura e rotulação (rótulos diferentes para pixels com uma mesma direção).

Conforme mostra a figura 64 (a), todos os pixels da borda apresentam a mesma direção, porém, conforme mostra-se na mesma figura em (b) verifica-se que eles não recebem o mesmo rótulo, provocando uma fragmentação da reta. A figura 65 mostra em detalhe dos pixels, seus rótulos e a causa do problema.

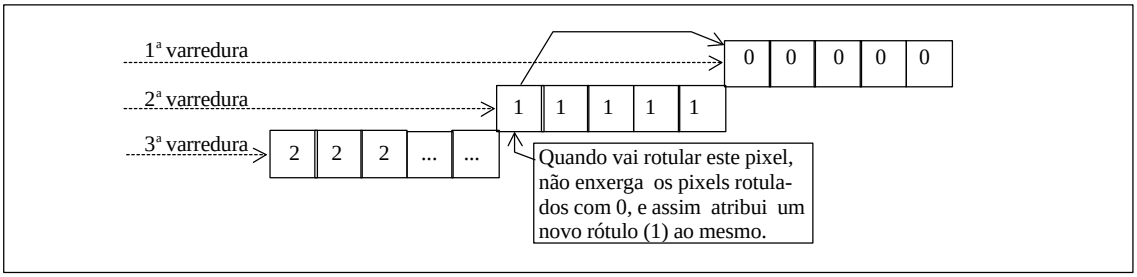


Figura 65 – Detalhe do problema que ocorre no método apresentado (Varredura e Rotulação).

Uma sugestão que pode corrigir o problema é sempre verificar se um pixel está sob a ação de dois rótulos vizinhos, e neste caso, herdar o menor deles (anterior – na figura acima, o rótulo 0), fazendo uma propagação em todos os demais pixels já rotulados com o rótulo atual (na figura 65, o rótulo 1), fazendo com que os mesmos recebam também o rótulo anterior (na figura 65, o rótulo 0). Desta forma tem-se o procedimento apresentado na figura 66.

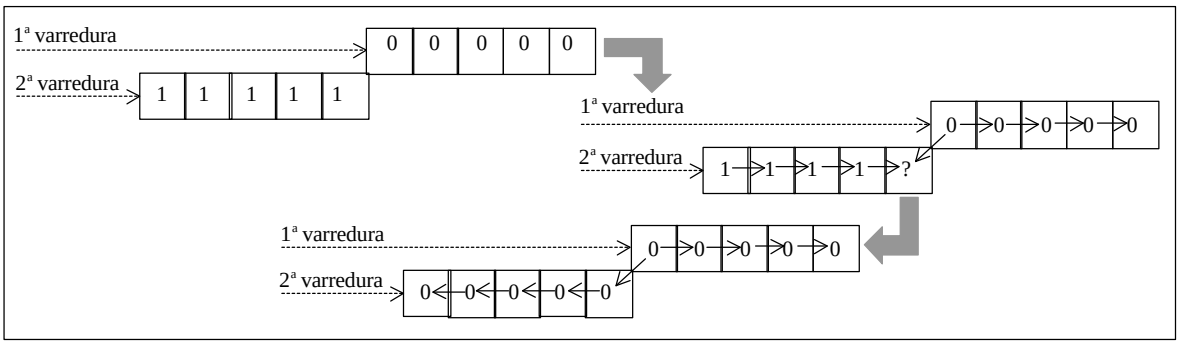


Figura 66 – Sugestão para a correção do problema no método Scan & Label.

Este esquema tem apresentado bons resultados em diversas situações, mas a sua utilização não é imprescindível, pois, a etapa de reconexão de segmentos colineares (seção 3.8.3) também pode resolver o problema.

Mesmo utilizando o esquema apresentado na figura 63, alguns segmentos continuam apresentando resultados insatisfatórios na definição de seus pixels extremos. A figura 67 mostra o detalhe de um segmento, e dos seus pixels extremos obtidos na etapa de rotulação implementada (mesmo usando o esquema exibido na figura 63). Nota-se em (a) que os pixels início e fim não se encontram realmente nos extremos do segmento, pois, para tal segmento o resultado esperado seria aquele exibido em (b).

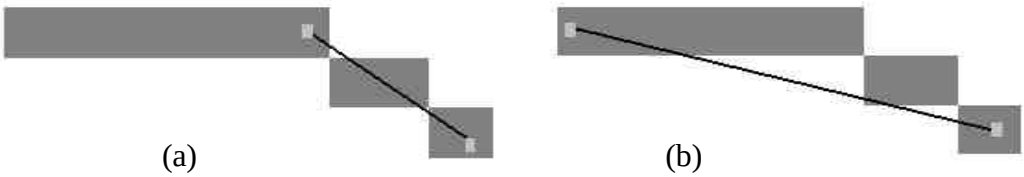


Figura 67 – Localização dos pixels extremos (a) antes e (b) depois da correção.

A técnica proposta para a rotulação (inundação) das linhas da forma como está implementada, não retorna na estrutura de linhas, os pixels inicial e final da reta, e assim, esta informação precisa ser obtida em uma etapa adicional. A etapa implementada parte do pixel inicial (atual) do segmento e, por inundação, verifica quais são os pixels extremos (pela esquerda, direita, por cima e por baixo).

Uma possível forma de se avaliar a necessidade de aplicar a rotina para a determinação dos pixels extremos das linhas pode ser dada pela seguinte condição:

$$\exists \ \textit{necessidade} \text{ se } \ L < \frac{N}{2} \tag{83}$$

onde L é o comprimento do segmento e N é o número de pixels que o mesmo possui. Em (83) observa-se que fica evidenciada a necessidade quando, para um grande número de pixels tem-se um pequeno comprimento de segmento.

O algoritmo implementado no sistema que obtém os pixels extremos em *x* e também em *y* com boa eficiência é também apoiada na condição de distância entre pixels, sendo então definidos os quatro pixels extremos em *x* e *y*, conforme mostra a figura 68.

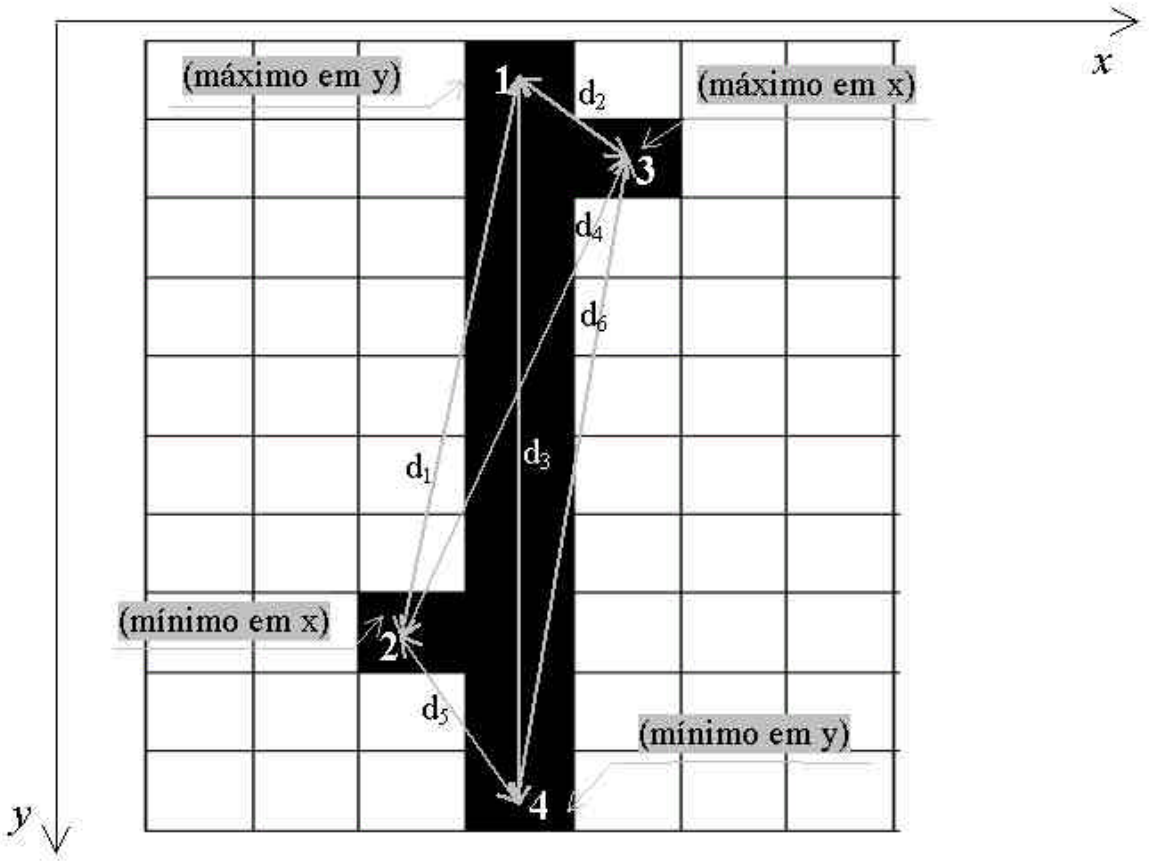


Figura 68 – Detalhe de um segmento onde os extremos precisam ser verificados, para se definir os pixels extremos da reta

Em seguida o algoritmo determina os dois pixels mais distantes ente os quatro possíveis (1, 2, 3 e 4 respectivamente mínimo em *y*, mínimo em *x*, máximo em *x* e máximo em *y*). Neste caso, especificamente, serão escolhidos os pixels 1 e 4 – mínimo e máximo em *y* respectivamente. Na mesma figura, d_1, d_2, d_3, d_4, d_5 e d_6 são respectivamente, as distâncias entre os pixels 1-2, 1-3, 1-4, 2-3, 2-4 e 3-4. Este método tem resolvido de forma adequada a determinação dos pixels extremos dos segmentos, de qualquer direção.

Como a rotina implementada precisa percorrer todos os pixels sob um mesmo rótulo, esta etapa pode também ser utilizada para se fazer uma recontagem do número de pixels que possuem o mesmo rótulo, informação importante a ser mantida na

estrutura de linhas. A recontagem é importante pois permite manter atualizado o número de pixels que cada segmento possui, e que por sua vez é utilizado na etapa posterior (ajustamento).

3.8.3 Conexão dos segmentos colineares

No processo de conexão de segmentos colineares, descrito na seção 2.6.3.2, verifica-se que, em muitas situações, pequenos segmentos com uma inclinação aproximadamente vertical (por exemplo 60 graus), são conectados com segmentos aproximadamente horizontais (por exemplo 30 graus), e esta operação mascara a verdadeira direção do segmento formado (no caso aproximadamente horizontal e vertical ao mesmo tempo). Os testes realizados na prática mostram que este algoritmo não atende tais situações, e assim uma nova metodologia está sendo utilizada.

No sistema desenvolvido foi implementada a rotina que conecta os segmentos colineares, conforme apresentado na seção 2.6.3.2 (máscaras de direção - figura 36). Na implementação da rotina foram definidas as prioridades apresentadas na figura 69 para a conexão dos segmentos. Nesta figura, o pixel *O* é o pixel final ou inicial do segmento que poderá ser conectado a algum outro nas proximidades. O pixel *A* é o local onde será analisado a existência de algum outro segmento com um ângulo próximo (diferença menor ou igual a trinta graus) ao do primeiro segmento. No caso de ocorrer a conexão (o que é possível se for atendida a condição apresentada na seção 2.6.3.2 (figura 37), os pixels *P* são então anexados ao novo segmento, como pixels ponte, recebendo também a direção dos demais pixels, e a magnitude é dada pela média das magnitudes do pixel *O* e *A*.

Conforme também mostra a figura 69, a seqüência de operações realizadas, procura inicialmente em uma vizinhança mais próxima, e em seguida, caso não encontre segmentos nas proximidades, se afasta um pouco, até uma distância máxima de três pixels. Este procedimento permite a conexão de segmentos separados no máximo por três pixels.

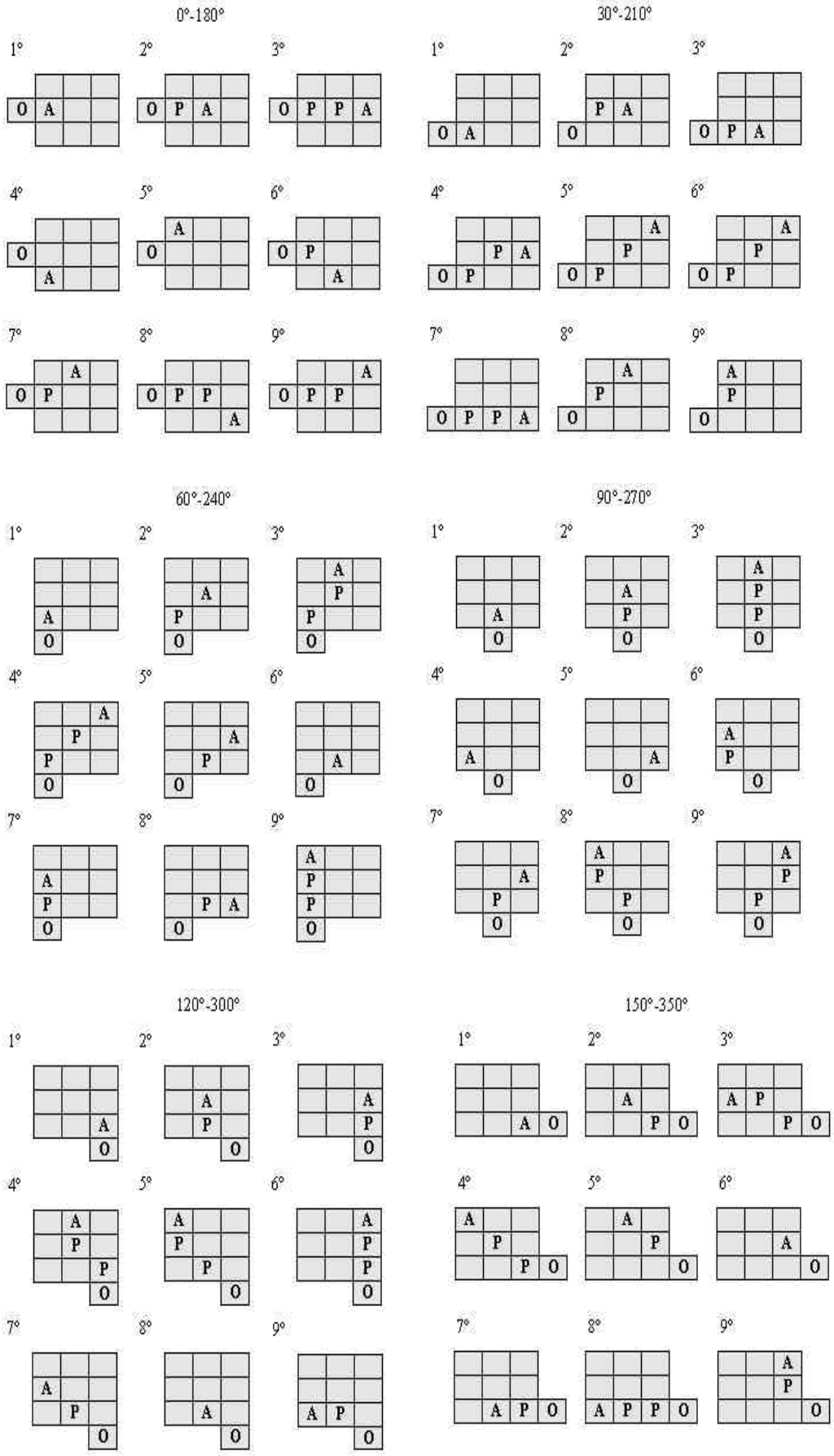


Figura 69 – Regiões utilizadas na rotina de conexão de segmentos colineares.

A figura 70 mostra como estas máscaras são utilizadas com os respectivos segmentos.

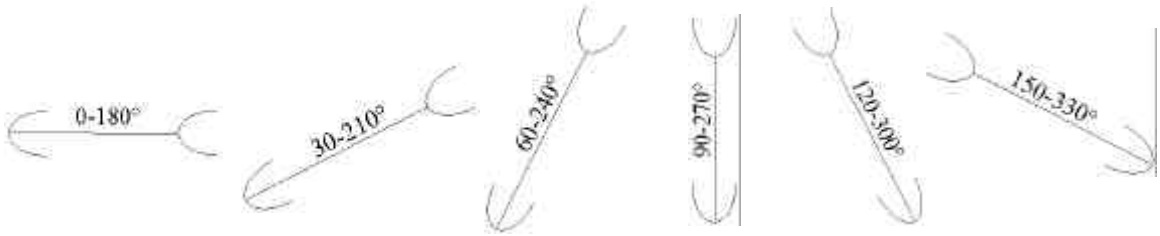


Figura 70 – Relacionamento entre os segmentos na rotina de conexão de segmentos colineares.

Conforme apresentado anteriormente, a conexão deve ser verificada para segmentos de mesma direção, e também segmentos com direção próxima.

Um estudo dos segmentos apresentados na figura 69 mostra que o segmento de direção 0°-180° consegue detectar em suas extremidades segmentos de 150°-330°, 0°-180° e 30°-210° (sempre a sua direita - caso existam). O segmento de 30°-210° consegue detectar nas suas extremidades segmentos de 0°-180°, 30°-210° e 60°-240° (sempre a sua direita - caso existam). Da mesma forma, um segmento de 60°-210° consegue detectar nas suas extremidades segmentos de 30°-210°, 60°-240° e 90°-270° (sempre a sua direita - caso existam). Um segmento de 90°-270° consegue detectar nas suas extremidades segmentos de 60°-240°, 90°-270° e - 120°-300°, (sempre na sua parte superior - caso existam). Um segmento de 120°-300° consegue detectar nas suas extremidades segmentos de 90°-270°, 120°-300° e - 150°-330°, (à esquerda e acima - caso existam), e finalmente um segmento de 150°-330° consegue detectar nas suas extremidades segmentos de 0°-180°, 150°-330° e 120°-300° (à esquerda e acima - caso existam). Entretanto, um segmento de 150°-330° não consegue detectar na sua extremidade mais a direita segmentos de 0°-180°, apesar de ser detectado pelos segmentos de 150°-330° e 120°-300° (caso existam).

Uma possível solução para esta situação é utilizar especificamente para os segmentos com direção 0-180°, a máscara que aparece na figura 71.

X	X	X		X	X	X
X	X	X	O	X	X	X
X	X	X		X	X	X

Figura 71 – Máscara para a conexão de segmentos colineares.

A figura 72(a) mostra a ocorrência deste problema, e a figura 72(b) mostra o resultado obtido com a correção da máscara para segmentos com direção 0°-180° (figura 71).

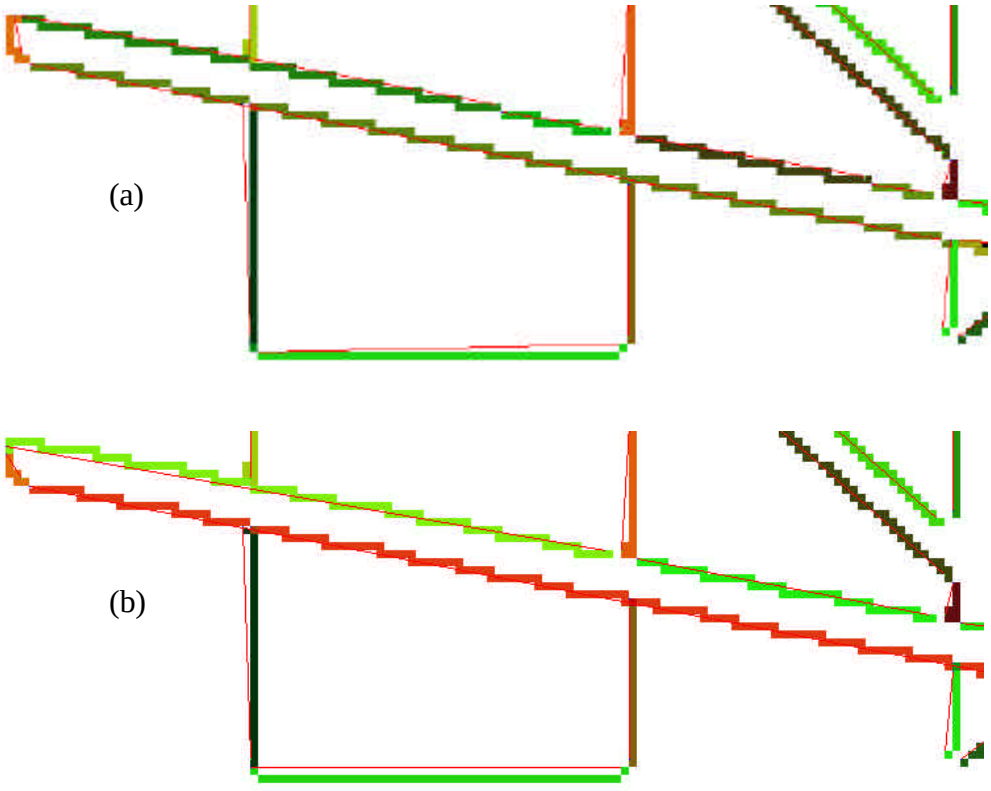


Figura 72 – (a) Vários segmentos colineares não conectados e (b) após a utilização da máscara da figura 71.

Após a conexão dos segmentos colineares, um segmento pode ter sido obtido pela união de diversos segmentos menores com diferentes direções (direções próximas); assim, após esta etapa, a direção precisa ser novamente calculada, ou não se tem a informação da verdadeira localização dos pixels início e fim de cada segmento.

Devido ao problema de *aliasing* já mencionado, torna-se necessário fazer um relaxamento no momento de verificar se dois segmentos são colineares. Na implementação foram considerados segmentos colineares aqueles que possuem direções próximas (diferença máxima de 30°), sendo a confirmação de colinearidade obtida pelo critério apresentado na figura 37 (seção 2.6.3.2).

Após a conexão de dois segmentos colineares os pixels início e fim do novo segmento precisam ser definidos. As diferentes formas como os pixels início e fim de cada segmento envolvido na união pode aparecer tem tornado complicada a tarefa de definir os extremos do novo segmento. Uma solução que tem se mostrado eficiente é, após a troca dos rótulos de um segmento pelos rótulos do outro segmento (todos os pixels dos dois segmentos passam a ter o mesmo rótulo), executa-se a rotina de determinação dos pixels extremos das linhas já abordada.

Finalmente, no processo de conexão de segmentos pode ocorrer que um segmento possua uma direção, e o outro possua uma direção diferente. Entretanto, o novo segmento obtido pela união dos dois deve receber uma direção, ainda a ser definida.

3.8.4 Correção das direções dos segmentos

Após a determinação de todos os pixels que pertencem temporariamente a um dado segmento (todos os pixels possuem o mesmo rótulo), verifica-se que alguns segmentos são constituídos por pixels com direções diferentes, (não existindo mais a informação correta da verdadeira direção do segmento todo). Nota-se, ainda, que, apesar de alguns segmentos possuírem pixels com direção 0 , 30 e até 60 graus, o conjunto todo (segmento formado) pode se mostrar claramente com uma direção próxima de 30 graus.

Neste caso é possível adotar para todos os pixels do segmento, a direção global do segmento, que pode ser definida por meio dos pixels extremos definidos na etapa anterior.

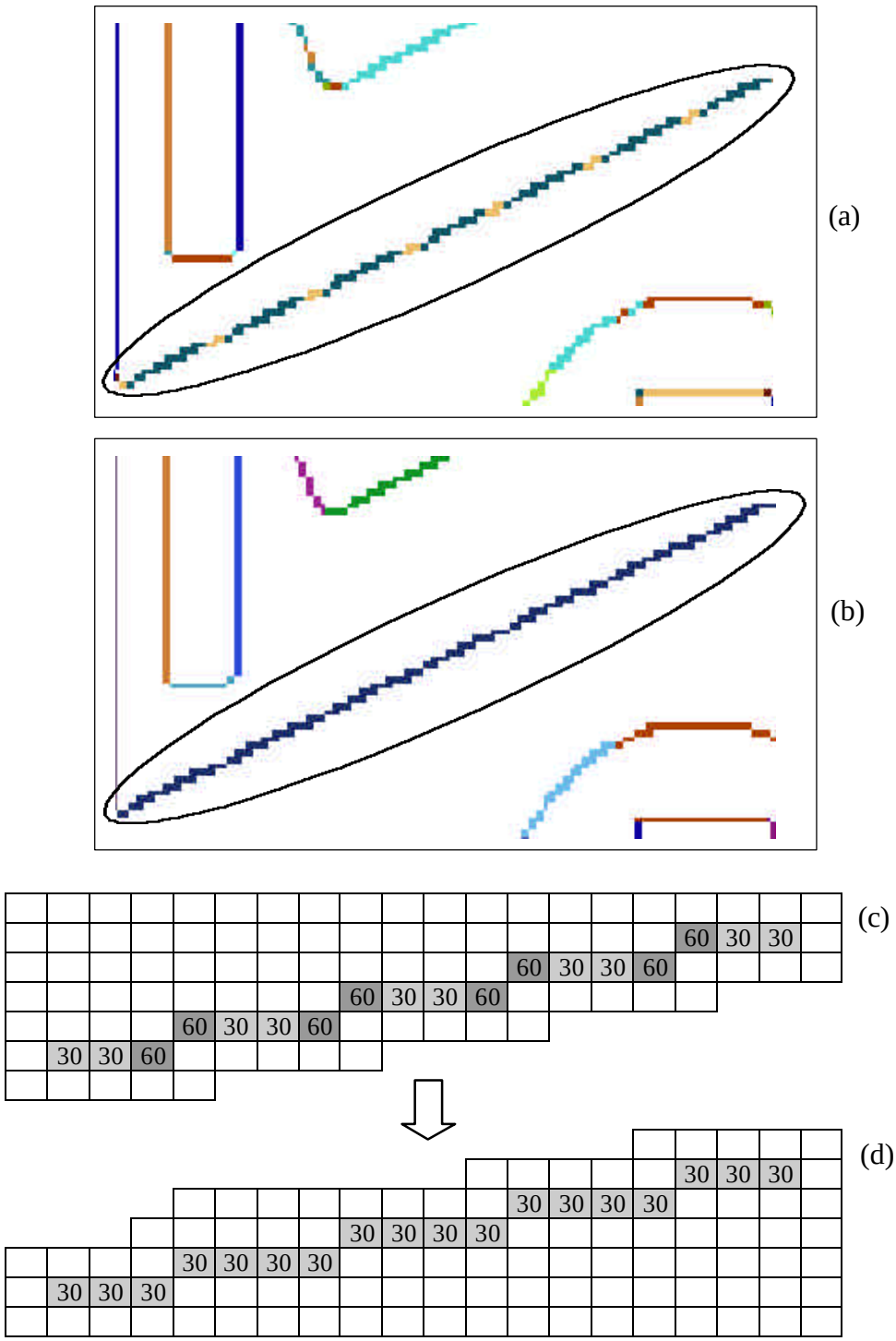


Figura 73 – (a) Imagem de direções obtida pela etapa de detecção de bordas, (b) imagem de direções após a correção sugerida, (c) detalhe do segmento destacado em (a) e (d) detalhe do segmento destacado em (b)

A rotina toma apenas os pixels início e fim do segmento, e assim determina o ângulo aproximado do segmento (direção aproximada). O ângulo do segmento é dado por 84).

$$a = \begin{cases} \arctan\left(\dfrac{y_f - y_i}{x_f - x_i}\right) & \text{se } x_f > x_i \\ \arctan\left(\dfrac{y_i - y_f}{x_i - x_f}\right) & \text{se } x_i > x_f \end{cases} \tag{84}$$

Em (84) não está sendo mais levado em consideração a direção da borda (considerando a direção do gradiente - borda), mas apenas a direção do segmento, obtida conforme descrito anteriormente, exclusivamente através da localização dos pixels extremos deste. A informação de direção obtida nesta etapa é importante pois define a parametrização mais adequada a ser utilizada no processo de ajustamento de retas (modelo $y=ax+b$ para retas horizontais ou $x=a*y+b*$ para retas verticais).

Isto é importante para prevenir o problema de tentar ajustar uma reta vertical com o modelo $y=ax+b$, o que resultaria em $a=\pm¥$, ou por outro lado, tentar ajustar uma reta horizontal com o modelo $x=a*y+b*$.

Após a determinação dos pixels que pertencem a cada segmento de reta (todos os pixels possuem um mesmo rótulo), pode ser realizado o ajustamento das retas sobre eles.

Neste trabalho, está implementada uma rotina de ajustamento, que trabalha apenas com linhas retas (regressão linear). Durante o processamento também é realizada uma totalização dos valores dos resíduos, e caso seja obtido um número acima de um valor previamente estabelecido, um processo de quebra do segmento (*splitting*) é executado. A figura 74 mostra um exemplo obtido nos experimentos, onde se verifica tal situação.

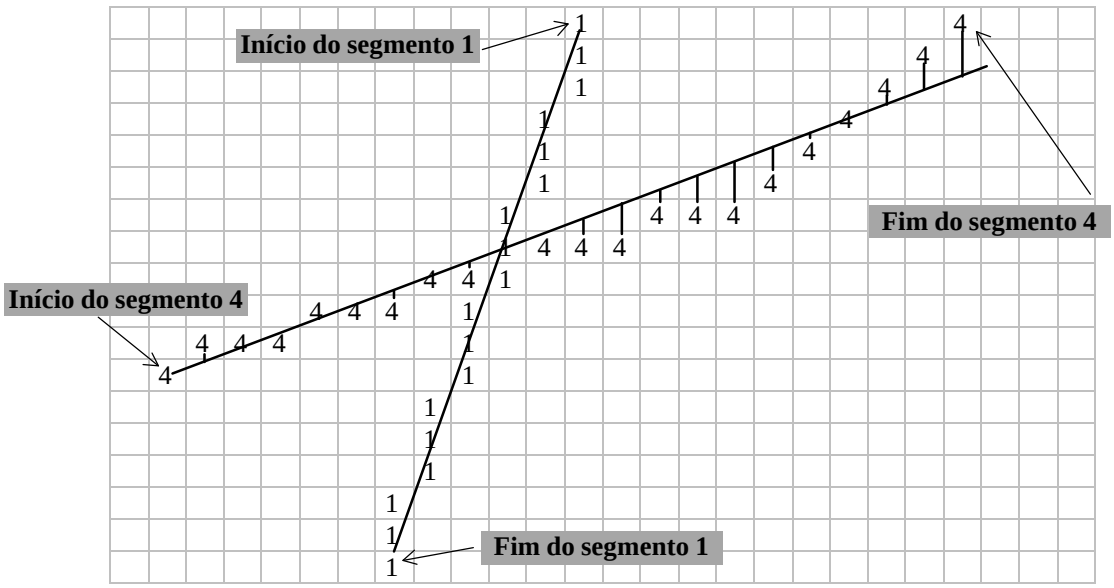


Figura 74 – Ajustamento de retas sobre dois segmentos (1 e 4).

Neste caso, o valor do resíduo está acima do limite previamente estabelecido, e portanto deve ser iniciado o processo de divisão do segmento (*splitting*), conforme mostra a figura 75, segmento número 4. No caso da situação apresentada nesta figura, o método de quebra implementado localiza a maior distância entre os pixels do segmento 4 e a reta que liga os extremos do segmento.

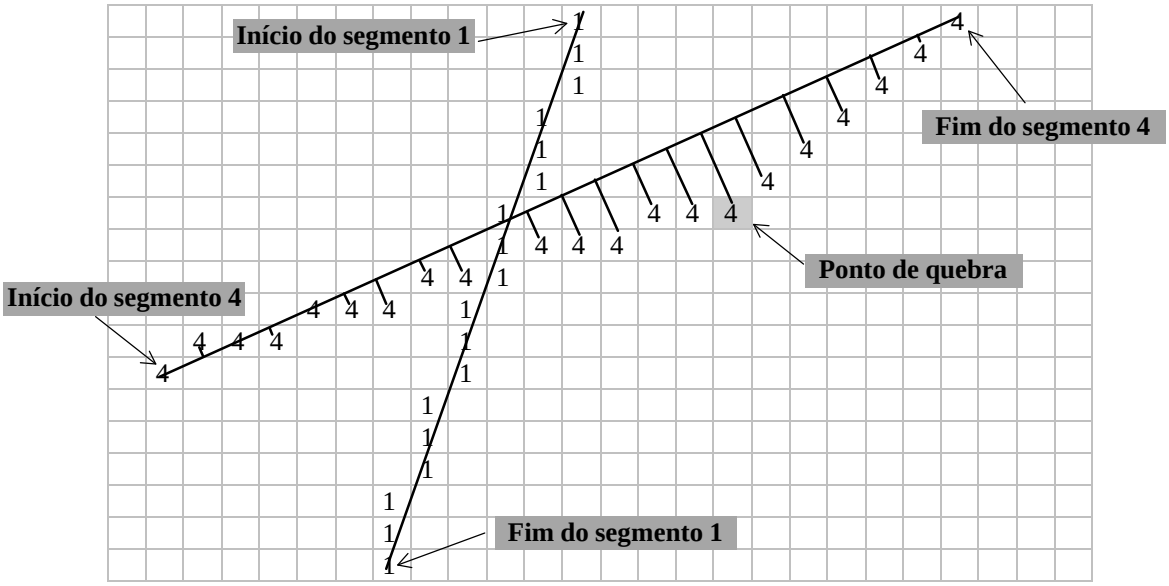


Figura 75 – Configuração dos segmentos 1 e 4, e a identificação dos pixels de quebra no processo de ajustamento.

O pixel destacado na figura 75, e indicado como pixel de quebra, deve se tornar então o pixel final do segmento que se inicia no pixel de início do segmento 4. Em seguida, todos os pixels a partir do pixel de início do segmento 4 devem ser substituídos por um novo valor de rótulo (um rótulo ainda não utilizado - no caso 9), constituindo-se assim um novo segmento de rótulo 9, conforme mostra a figura 76.

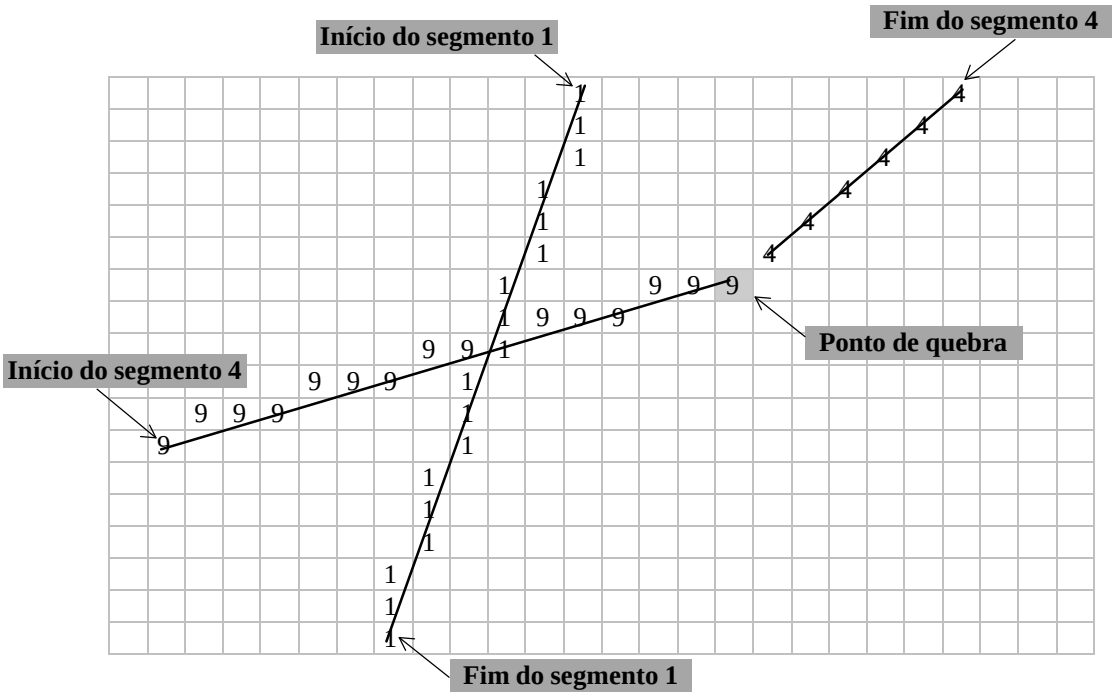


Figura 76 – Configuração dos segmentos após a quebra do segmento de rótulo 4 em dois segmentos (rótulos 4 e 9).

Na rotina implementada, o pixel início do segmento novo (rótulo 9) coincide com o início do segmento de rótulo 4, enquanto que o fim do segmento novo coincide com o pixel de quebra do segmento antigo (rótulo 4). Como o pixel fim do segmento 4 não é alterado, o novo pixel de início do mesmo pode ser determinado pela rotina de determinação de pixels extremos já apresentada anteriormente (que também atua por um processo de inundação – perseguição de pixels), sendo inicializada neste pixel (pixel início do segmento 4).

Neste processo observa-se que, para que a rotina (baseada em um processo de inundação) funcione adequadamente, conseguindo percorrer o segmento do início até o

fim, e transpondo os cruzamentos (segmento 1 na figura 74), é preciso que seja verificada uma vizinhança 24 (a rotina de inundação de ser capaz de realizar saltos de um pixel, para continuar a percorrer o segmento com quebras). Por outro lado, para que o pixel de quebra não seja ultrapassado, apenas a vizinhança 8 deve ser utilizada. Com o objetivo de resolver esta situação o sistema utiliza algoritmo apresentado na figura 77.

```
quebra_proxima = Quebra_Na_Vizinhanca(i,j,1);                  // verifica se quebra está na vizinhança 8
se (quebra_proxima)                                            // se estiver
 troca_local_rotulos(i,j,1);                                   // troca os rótulos na vizinhança 8
senão                                                          // se a quebra não está na vizinhança 8
  {
   possivel_caminhar = Possivel_deslocar_vizinhanca(i,j,1);    // verifica se consegue caminhar na vizinhança 8
   se (possivel_caminhar)                                      // se for
    trocar_rotulo_vizinhanca_1(i,j);                           // caminha na vizinhança 8 (trocando rótulos)
  senão                                                        // se não é possível caminhar na vizinhança 8
        {
        quebra_proxima = Quebra_Na_Vizinhanca(i,j,2);          // verifica se a quebra está na vizinhança 24
        se (quebra_proxima)                                    // se estiver
         troca_local_rotulos(i,j,2);                           // troca os rótulos na vizinhança 24
        senão                                                  // se a quebra não está na vizinhança 24
           trocar_rotulo_vizinhanca_2(i,j);                    // caminha na vizinhança 24 (trocando rótulos)
        }
```

Figura 77 – Algoritmo utilizado para fazer a troca de rótulos no processo de quebra de segmentos.

Conforme se observa na rotina, inicialmente é verificado se o pixel de quebra está nas proximidades (vizinhança 8) e, caso isto ocorra, apenas os pixels dentro desta vizinhança terão seu rótulos verificados e trocados pelo novo valor, pela rotina *troca_local_rotulos(i,j,1)*. Caso o pixel de quebra não esteja nas proximidades (vizinhança 8), a rotina verifica se é possível caminhar utilizando uma vizinhança 8 (isto será possível se não houver um pixel de interrupção no segmento). Neste caso, percorre o segmento utilizando uma vizinhança 8. Não havendo a possibilidade de caminhar na vizinhança 8 (devido a alguma interrupção do segmento – devido a uma interseção com um outro segmento) a rotina então verifica se o pixel de quebra está na vizinhança 24. Neste caso, executa a rotina *troca_local_rotulos(i,j,2)*, que finaliza o processo verificando uma possível troca apenas nos pixels da vizinhança 24. Caso a quebra não esteja ainda na vizinhança 24, é verificada a possibilidade de se caminhar em uma vizinhança 24, e desta forma, é possível transpor o pixel de interrupção e continuar o processo até que o pixel de quebra seja atingido.

As rotinas *trocar_rotulo_vizinhanca_2(i,j)* e *trocar_rotulo_vizinhanca_1(i,j)* também são recursivas e participam do processo, conforme descrito.

Após a quebra do segmento em dois, os mesmos também precisam passar pelo processo de ajustamento, e caso ocorra dos resíduos continuarem acima de um limite aceitável, os segmentos precisam ser quebrados novamente. A figura 78 mostra o algoritmo utilizado para resolver esta etapa do processamento.

```
se linha_horizontal(i);                                  // se o segmento i é aproximadamente horizontal
 {
  ajusta_horizontal(i);                                  // ajusta o segmento i usando a parametrização y=ax+b
  Q = (∑ residuos) / N      onde N : Número de pontos;   // Define um fator de qualidade do ajustamento
                            do segmento i
  se Q > limiar
    Quebre_o_segmento(i);                                // Quebra o segmento em dois (i e novo)
  }
senão                                                    // se o segmento não é aproximadamente horizontal
{
  ajusta_vertical(i);                                    // ajusta o segmento i usando a parametrização x=ay+b
  Q = (∑ residuos) / N      onde N : Número de pontos;   // Define um fator de qualidade do ajustamento
                            do segmento i
  se Q > limiar
    Quebre_o_segmento(i);                                // Quebra o segmento em dois (i e novo)
  }
repita para os segmentos que foram quebrados();
```

Figura 78 – Algoritmo utilizado para medir a qualidade do ajustamento e decidir as situações onde a quebra (*splitting*) é necessária.

O valor de *Q* utilizado no algoritmo foi obtido empiricamente, sendo igual a 1.0 (um), podendo ainda ser modificado, de acordo com os resultados obtidos. O valor de *N* (número de pixels no segmento) utilizado no denominador é justificado por ser mais interessante manter os segmentos com um grande número de pixels (segmentos compridos). Assim, um mesmo resíduo resultará em um valor de Q maior que aquele que seria obtido se o segmento fosse maior.

CAPITULO 4

RESULTADOS EXPERIMENTAIS

4.1 GERAÇÃO DE UMA IMAGEM PADRÃO

Uma imagem padrão é gerada para testes, a partir de uma imagem criada em tempo de execução pelo próprio sistema (imagem sintética - figura 79). Este procedimento foi adotado pela vantagem de serem conhecidos os valores corretos dos vértices dos polígonos, e consequentemente os parâmetros *a* e *b* de cada um dos segmentos que delimitam os polígonos formados (os números dentro dos círculos são os números dos segmentos - linhas de borda).

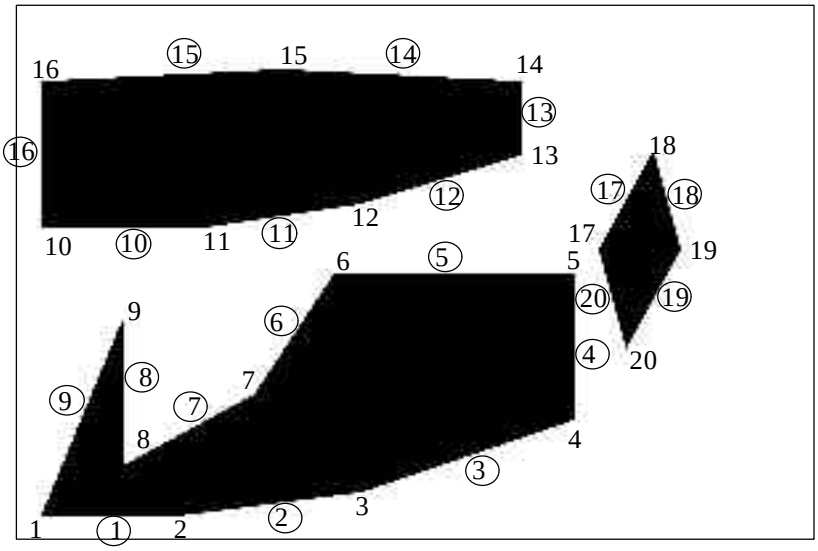


Figura 79 –Imagem gerada sinteticamente.

O seu formato foi escolhido de modo a conter bordas com várias direções e cantos com ângulos retos, agudos e obtusos, de forma a verificar o comportamento dos vários algoritmos implementados em diversas situações. A tabela 3 mostra as coordenadas e os parâmetros dos vinte segmentos que formam os três polígonos que aparecem na imagem padrão. Tais valores foram utilizados como referência, para a análise dos resultados obtidos nos experimentos.

Tabela 2 – Coordenadas e parâmetros dos segmentos que formam os três polígonos da imagem padrão.

Segmento	Vértice 1		Vértice 2		Tipo	a	b
	x	y	x	Y	(parametrização)		
1	10	210	60	210	Horiz ($y=ax+b$)	0,000000	210,000000
2	60	210	130	200	Horiz ($y=ax+b$)	-0,142857	218,571428
3	130	200	210	170	Horiz ($y=ax+b$)	-0,375000	248,750000
4	210	170	210	110	Vert($x=ay+b$)	0,000000	210,000000
5	210	110	120	110	Horiz ($y=ax+b$)	0,000000	110,000000
6	120	110	90	160	Vert($x=ay+b$)	-0,600000	186,000000
7	90	160	40	190	Horiz ($y=ax+b$)	-0,600000	214,000000
8	40	190	40	130	Vert($x=ay+b$)	0,000000	40,000000
9	40	130	10	210	Vert($x=ay+b$)	-0,375000	88,750000
10	10	190	70	90	Horiz ($y=ax+b$)	0,000000	90,000000
11	70	90	130	80	Horiz ($y=ax+b$)	-0,166666	101,666666
12	130	80	190	60	Horiz ($y=ax+b$)	-0,333333	123,333333
13	190	60	190	30	Vert($x=ay+b$)	0,000000	190,000000
14	190	30	100	25	Horiz ($y=ax+b$)	0,055555	19,444444
15	100	25	10	30	Horiz ($y=ax+b$)	-0,055555	30,555555
16	10	30	10	90	Vert($x=ay+b$)	0,000000	10,000000
17	210	110	240	60	Vert($x=ay+b$)	-0,500000	270,000000
18	240	60	250	100	Vert($x=ay+b$)	0,250000	225,000000
19	250	100	230	140	Vert($x=ay+b$)	-0,50000	300,000000
20	230	140	210	110	Vert($x=ay+b$)	0,250000	195,000000

Antes de ser utilizada nos experimentos, a imagem sintética que é vista na figura 79 passou por um processo de degradação, sendo utilizado o modelo apresentado em Gonzalez (1993), que pode ser visto na figura 80. Neste modelo, a imagem de entrada $f(x,y)$ (figura 79) sofre um espalhamento (borramento), definido na função H, e em seguida é acrescentado 10% de ruído aleatório $h(x,y)$.

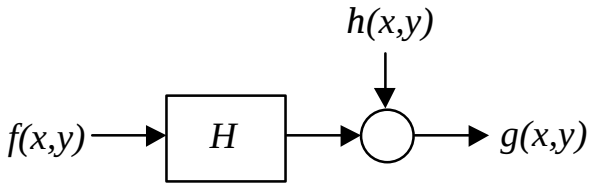


Figura 80 – Modelo de degradação utilizado para a obtenção da imagem padrão.

Após a aplicação deste processo de degradação sobre a imagem apresentada na figura 79, obtém-se a imagem apresentada na figura 81.

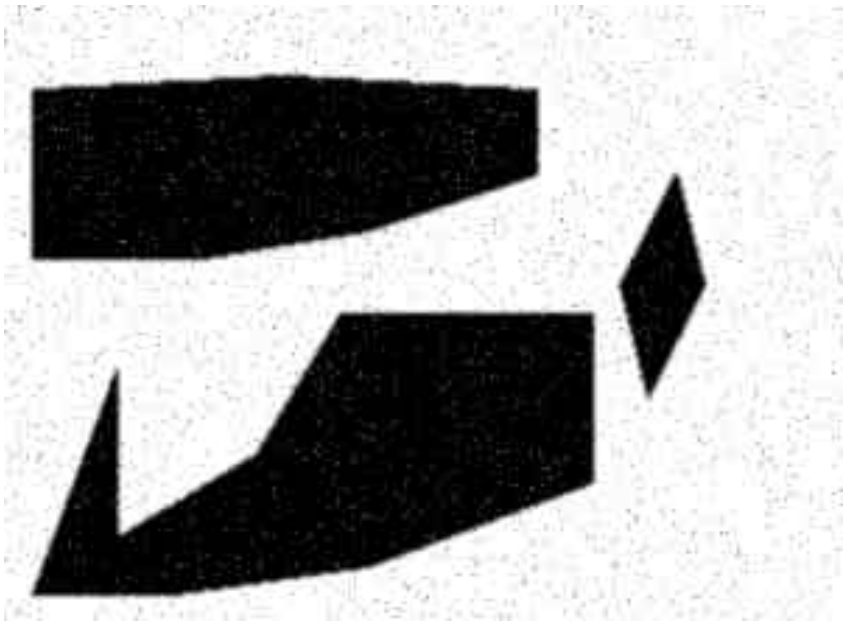


Figura 81 – Imagem utilizada na parte experimental deste trabalho, obtida pelo processo de degradação.

4.2 EXPERIMENTOS COM A IMAGEM PADRÃO

Após apresentados os resultados obtidos com várias combinações de algoritmos implementados, usando a imagem padrão, são também apresentados os valores numéricos dos parâmetros *a* e *b* (ou *a** e *b**) e ainda a imagem resultante (vetores obtidos). Os valores dos parâmetros podem ser comparados com os valores verdadeiro, conhecidos a priori.

As figuras 82 a 91 resumem os experimentos realizados com a imagem padrão. Nestas figuras, (a) apresentam as seqüências de algoritmos utilizadas nas etapas do

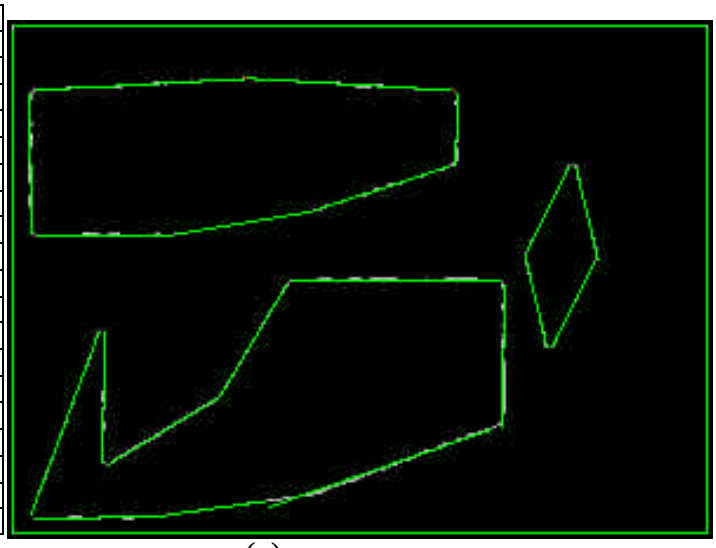
processo; (b) os resultados numéricos obtidos para os parâmetros *a* e *b* (ou *a** e *b**); (c) mostram as imagens dos vetores obtidos.

Suavização	Detecção	Limiarização	Afinamento	Conexão	Ajustamento
Média 3x3	Nevatia e Babu	Otsu	Supressão não Máxima	Varredura e Rotulação	M.M.Q.

(a)

	a	b	$erro_a$	$erro_b$
1	-0,01834	211,20662	-0,01834	1,20662
2	-0,13872	218,70570	0,00413	0,13428
3	-0,34902	244,71919	0,02598	-4,03081
4	-0,01535	212,62424	-0,01535	2,62424
5	-0,00097	109,74039	-0,00097	-0,25961
6	-0,60128	185,62877	-0,00128	-0,37123
7	-0,59947	213,30617	0,00053	-0,69383
8	-0,02034	43,82946	-0,02034	3,82946
9	-0,37162	87,55453	0,00338	-1,19547
10	-0,00070	90,79973	-0,00070	0,79973
11	-0,16569	102,14553	0,00097	0,47886
12	-0,32457	122,49409	0,00876	-0,83925
13	-0,02831	191,92311	-0,02831	1,92311
14	0,05877	18,32142	0,00321	-1,12303
15	-0,05369	29,92427	0,00186	-0,63129
16	0,00960	8,91947	0,00960	-1,08053
17	-0,49585	268,86459	0,00415	-1,13541
18	0,23874	226,53229	-0,01126	1,53229
19	-0,49988	300,25427	0,00012	0,25427
20	0,23485	196,39070	-0,01515	1,39070

(b)



(c)

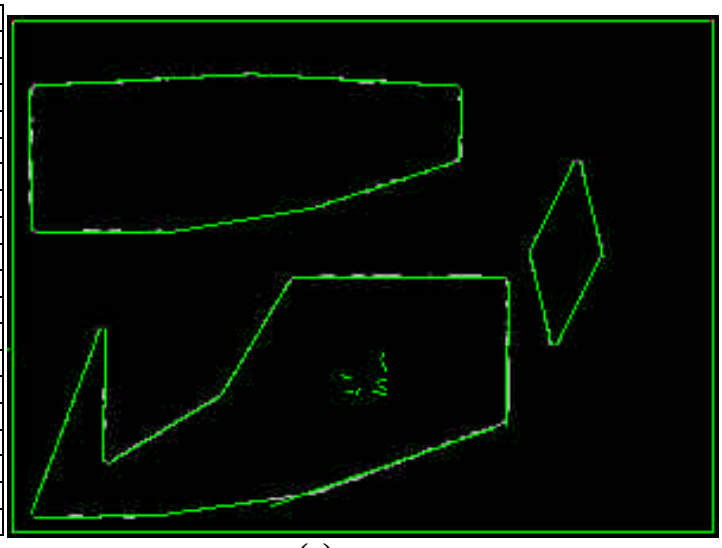
Figura 82 – Experimento 1.

Suavização	Detecção	Limiarização	Afinamento	Conexão	Ajustamento
Média 3x3	Nevatia e Babu	Otsu Local	Supressão não Máxima	Varredura e Rotulação	M.M.Q.

(a)

	a	b	$erro_a$	$erro_b$
1	-0,01834	211,20662	-0,01834	1,20662
2	-0,13872	218,70570	0,00413	0,13428
3	-0,34902	244,71919	0,02598	-4,03081
4	-0,01535	212,62424	-0,01535	2,62424
5	-0,00097	109,74039	-0,00097	-0,25961
6	-0,60128	185,62877	-0,00128	-0,37123
7	-0,59947	213,30617	0,00053	-0,69383
8	-0,02034	43,82946	-0,02034	3,82946
9	-0,37162	87,55453	0,00338	-1,19547
10	-0,00070	90,79973	-0,00070	0,79973
11	-0,16569	102,14553	0,00097	0,47886
12	-0,32457	122,49409	0,00876	-0,83925
13	-0,02831	191,92311	-0,02831	1,92311
14	0,05877	18,32142	0,00321	-1,12303
15	-0,05369	29,92427	0,00186	-0,63129
16	0,00960	8,91947	0,00960	-1,08053
17	-0,49585	268,86459	0,00415	-1,13541
18	0,23874	226,53229	-0,01126	1,53229
19	-0,49988	300,25427	0,00012	0,25427
20	0,23485	196,39070	-0,01515	1,39070

(b)



(c)

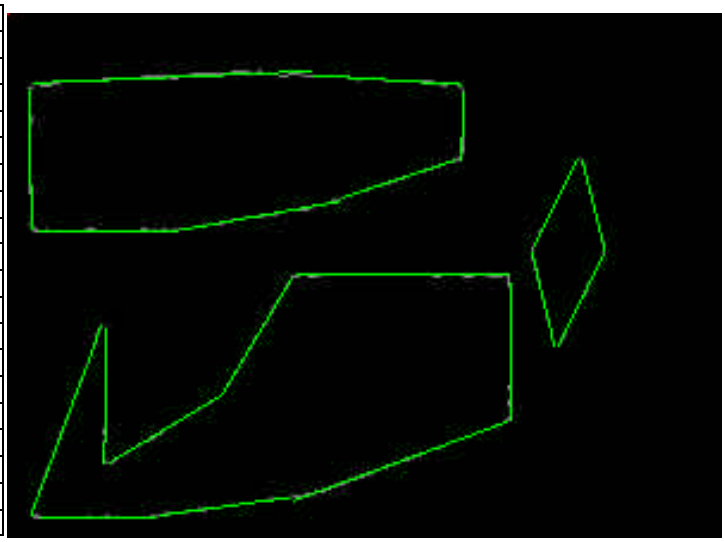
Figura 83 – Experimento 2.

Suavização	Detecção	Limiarização	Afinamento	Conexão	Ajustamento
Mediana	Sobel	Otsu	Supressão não Máxima	Varredura e Rotulação	M.M.Q.

(a)

	a	b	$erro_a$	$erro_b$
1	0,00000	211,00000	0,00000	1,00000
2	-0,13575	218,61301	0,00711	0,04158
3	-0,36820	247,96042	0,00680	-0,78958
4	-0,00516	211,54706	-0,00516	1,54706
5	0,00459	108,52841	0,00459	-1,47159
6	-0,59839	185,23131	0,00161	-0,76869
7	-0,60329	213,60500	-0,00329	-0,39500
8	-0,01485	43,17199	-0,01485	3,17199
9	-0,37174	87,67943	0,00326	-1,07057
10	0,00020	90,86079	0,00020	0,86079
11	-0,17921	103,39611	-0,01255	1,72944
12	-0,32879	122,96791	0,00454	-0,36542
13	-0,04588	192,88545	-0,04588	2,88545
14	0,06395	17,39021	0,00840	-2,05424
15	-0,04201	29,28130	0,01355	-1,27426
16	0,01679	8,29120	0,01679	-1,70880
17	-0,50722	269,83530	-0,00722	-0,16470
18	0,24981	225,72777	-0,00019	0,72777
19	-0,49340	299,52139	0,00660	-0,47861
20	0,23972	195,86342	-0,01028	0,86342

(b)



(c)

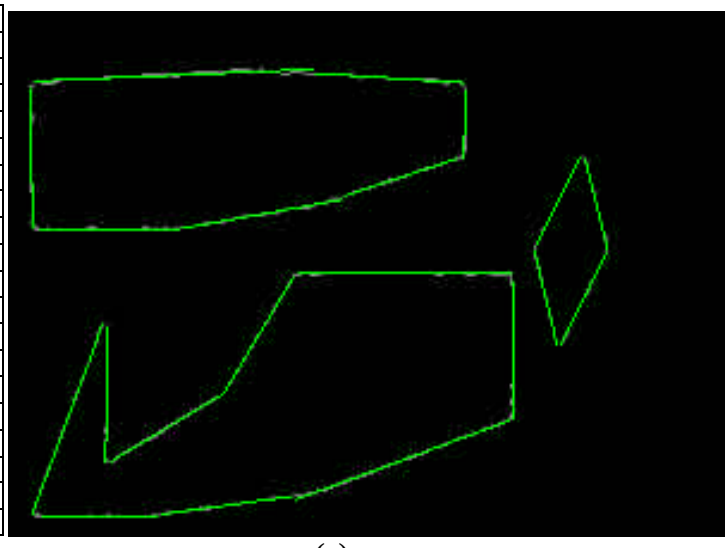
Figura 84 – Experimento 3.

Suavização	Detecção	Limiarização	Afinamento	Conexão	Ajustamento
Mediana	Sobel	Triângulo	Supressão não Máxima	Varredura e Rotulação	M.M.Q.

(a)

	a	b	$erro_a$	$erro_b$
1	0,00000	211,00000	0,00000	1,00000
2	-0,13575	218,61301	0,00711	0,04158
3	-0,36820	247,96042	0,00680	-0,78958
4	-0,00516	211,54706	-0,00516	1,54706
5	0,00459	108,52841	0,00459	-1,47159
6	-0,59839	185,23131	0,00161	-0,76869
7	-0,60329	213,60500	-0,00329	-0,39500
8	-0,01485	43,17199	-0,01485	3,17199
9	-0,37174	87,67943	0,00326	-1,07057
10	0,00020	90,86079	0,00020	0,86079
11	-0,17921	103,39611	-0,01255	1,72944
12	-0,32879	122,96791	0,00454	-0,36542
13	-0,04588	192,88545	-0,04588	2,88545
14	0,06395	17,39021	0,00840	-2,05424
15	-0,04201	29,28130	0,01355	-1,27426
16	0,01679	8,29120	0,01679	-1,70880
17	-0,50722	269,83530	-0,00722	-0,16470
18	0,24981	225,72777	-0,00019	0,72777
19	-0,49340	299,52139	0,00660	-0,47861
20	0,23972	195,86342	-0,01028	0,86342

(b)



(c)

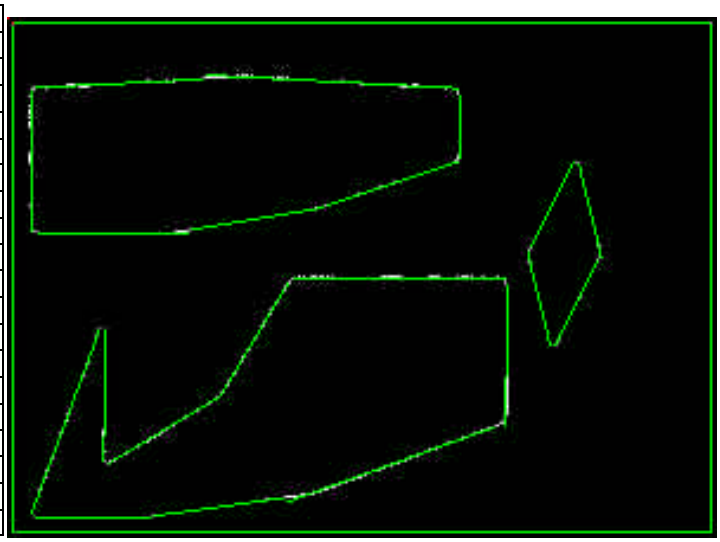
Figura 85 – Experimento 4.

Suavização	Detecção	Limiarização	Afinamento	Conexão	Ajustamento
Mediana	Nevatia e Babu	Otsu	Supressão não Máxima	Varredura e Rotulação	M.M.Q.

(a)

	a	b	$erro_a$	$erro_b$
1	0,00235	210,89647	0,00235	0,89647
2	-0,14060	218,90752	0,00226	0,33609
3	-0,36170	246,94905	0,01330	-1,80095
4	-0,02321	214,01521	-0,02321	4,01521
5	0,00292	109,16531	0,00292	-0,83469
6	-0,60168	185,67220	-0,00168	-0,32780
7	-0,60036	213,37772	-0,00036	-0,62228
8	-0,01614	43,35565	-0,01614	3,35565
9	-0,37125	87,50348	0,00375	-1,24652
10	-0,00634	91,17115	-0,00634	1,17115
11	-0,16788	102,34634	-0,00121	0,67967
12	-0,33308	123,88848	0,00025	0,55515
13	-0,00684	191,20416	-0,00684	1,20416
14	0,05269	19,60544	-0,00286	0,16099
15	-0,04932	29,92170	0,00623	-0,63386
16	0,00771	9,33119	0,00771	-0,66881
17	-0,50481	269,66690	-0,00481	-0,33310
18	0,24094	226,33746	-0,00906	1,33746
19	-0,49977	300,25836	0,00023	0,25836
20	0,24094	195,71533	-0,00906	0,71533

(b)



(c)

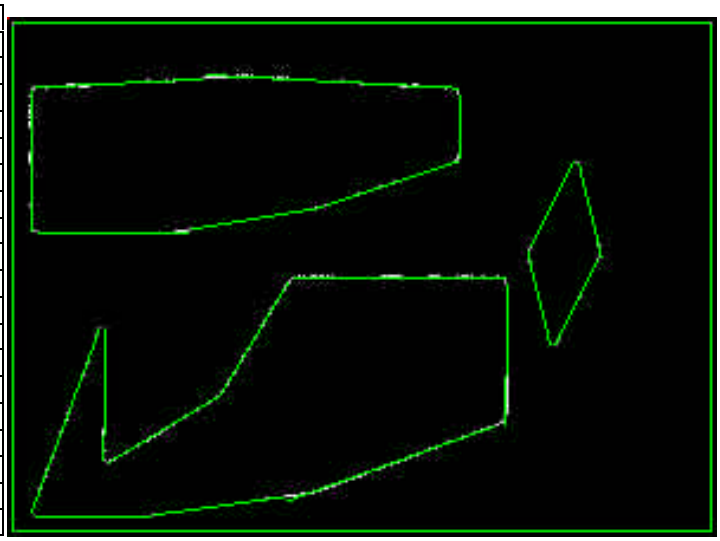
Figura 86 – Experimento 5.

Suavização	Detecção	Limiarização	Afinamento	Conexão	Ajustamento
Mediana	Nevatia e Babu	Otsu Local	Supressão não Máxima	Varredura e Rotulação	M.M.Q.

(a)

	a	b	$erro_a$	$erro_b$
1	0,00235	210,89647	0,00235	0,89647
2	-0,14060	218,90752	0,00226	0,33609
3	-0,36170	246,94905	0,01330	-1,80095
4	-0,02321	214,01521	-0,02321	4,01521
5	0,00292	109,16531	0,00292	-0,83469
6	-0,60168	185,67220	-0,00168	-0,32780
7	-0,60036	213,37772	-0,00036	-0,62228
8	-0,01614	43,35565	-0,01614	3,35565
9	-0,37125	87,50348	0,00375	-1,24652
10	-0,00634	91,17115	-0,00634	1,17115
11	-0,16788	102,34634	-0,00121	0,67967
12	-0,33308	123,88848	0,00025	0,55515
13	-0,00684	191,20416	-0,00684	1,20416
14	0,05269	19,60544	-0,00286	0,16099
15	-0,04932	29,92170	0,00623	-0,63386
16	0,00771	9,33119	0,00771	-0,66881
17	-0,50481	269,66690	-0,00481	-0,33310
18	0,24094	226,33746	-0,00906	1,33746
19	-0,49977	300,25836	0,00023	0,25836
20	0,24094	195,71533	-0,00906	0,71533

(b)



(c)

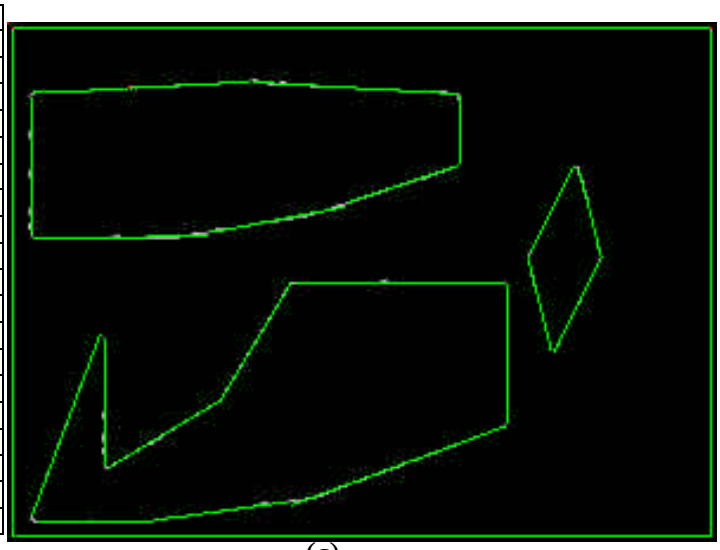
Figura 87 – Experimento 6.

Suavização	Detecção	Limiarização	Afinamento	Conexão	Ajustamento
Mediana Análise Variância	Sobel	Otsu	Supressão não Máxima	Varredura e Rotulação	M.M.Q.

(a)

	a	b	$erro_a$	$erro_b$
1	0,00000	211,00000	0,00000	1,00000
2	-0,13841	218,88910	0,00445	0,31767
3	-0,36893	248,10291	0,00607	-0,64709
4	0,00164	210,75464	0,00164	0,75464
5	0,00031	109,91573	0,00031	-0,08427
6	-0,59951	185,44385	0,00049	-0,55615
7	-0,60057	213,41879	-0,00057	-0,58121
8	-0,00838	42,13173	-0,00838	2,13173
9	-0,37526	88,32533	-0,00026	-0,42467
10	-0,01657	91,46490	-0,01657	1,46490
11	-0,19004	104,82251	-0,02337	3,15584
12	-0,33359	123,84723	-0,00026	0,51390
13	0,01802	190,09811	0,01802	0,09811
14	0,05685	19,10602	0,00130	-0,33843
15	-0,05400	30,31693	0,00155	-0,23863
16	-0,00430	10,02535	-0,00430	0,02535
17	-0,50000	269,00000	0,00000	-1,00000
18	0,25867	225,10355	0,00867	0,10355
19	-0,49434	299,65140	0,00566	-0,34860
20	0,25096	194,74394	0,00096	-0,25606

(b)



(c)

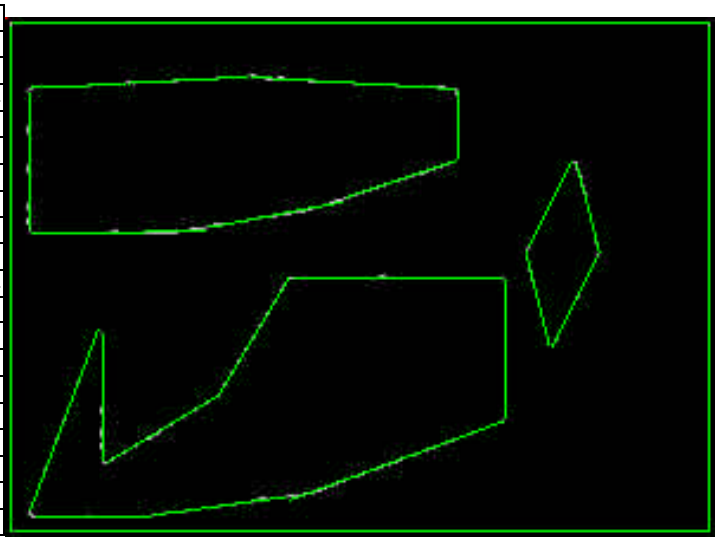
Figura 88 – Experimento 7.

Suavização	Detecção	Limiarização	Afinamento	Conexão	Ajustamento
Mediana Análise Variância	Sobel	Pun	Supressão não Máxima	Varredura e Rotulação	M.M.Q.

(a)

	a	b	$erro_a$	$erro_b$
1	0,00000	211,00000	0,00000	1,00000
2	-0,13841	218,88910	0,00445	0,31767
3	-0,36893	248,10291	0,00607	-0,64709
4	0,00164	210,75464	0,00164	0,75464
5	0,00031	109,91573	0,00031	-0,08427
6	-0,59951	185,44385	0,00049	-0,55615
7	-0,60057	213,41879	-0,00057	-0,58121
8	-0,00838	42,13173	-0,00838	2,13173
9	-0,37526	88,32533	-0,00026	-0,42467
10	-0,01657	91,46490	-0,01657	1,46490
11	-0,19004	104,82251	-0,02337	3,15584
12	-0,33359	123,84723	-0,00026	0,51390
13	0,01802	190,09811	0,01802	0,09811
14	0,05685	19,10602	0,00130	-0,33843
15	-0,05400	30,31693	0,00155	-0,23863
16	-0,00430	10,02535	-0,00430	0,02535
17	-0,50000	269,25674	0,00000	-0,74326
18	0,25867	225,10355	0,00867	0,10355
19	-0,49434	299,65140	0,00566	-0,34860
20	0,25096	194,74394	0,00096	-0,25606

(b)



(c)

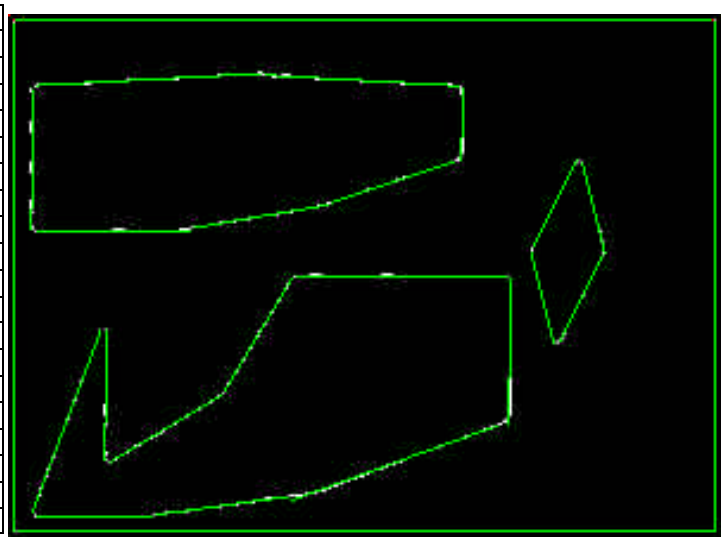
Figura 89 – Experimento 8.

Suavização	Detecção	Limiarização	Afinamento	Conexão	Ajustamento
Mediana Análise Variância	Nevatia e Babu	Otsu	Supressão não Máxima	Varredura e Rotulação	M.M.Q.

(a)

	a	b	$erro_a$	$erro_b$
1	0,00000	211,00000	0,00000	1,00000
2	-0,13890	218,80099	0,00395	0,22957
3	-0,36254	247,11816	0,01246	-1,63184
4	-0,02081	213,67412	-0,02081	3,67412
5	0,00337	109,33372	0,00337	-0,66628
6	-0,59862	185,29108	0,00138	-0,70892
7	-0,60088	213,39442	-0,00088	-0,60558
8	-0,01704	43,42290	-0,01704	3,42290
9	-0,37521	88,28199	-0,00021	-0,46801
10	-0,00689	91,14814	-0,00689	1,14814
11	-0,17014	102,65894	-0,00347	0,99228
12	-0,33251	123,80042	0,00082	0,46709
13	-0,02254	191,65291	-0,02254	1,65291
14	0,05585	19,21552	0,00030	-0,22892
15	-0,05263	30,17027	0,00292	-0,38529
16	-0,00650	10,02669	-0,00650	0,02669
17	-0,50756	269,89417	-0,00756	-0,10583
18	0,25202	225,48347	0,00202	0,48347
19	-0,50344	300,67172	-0,00344	0,67172
20	0,24681	195,02795	-0,00319	0,02795

(b)



(c)

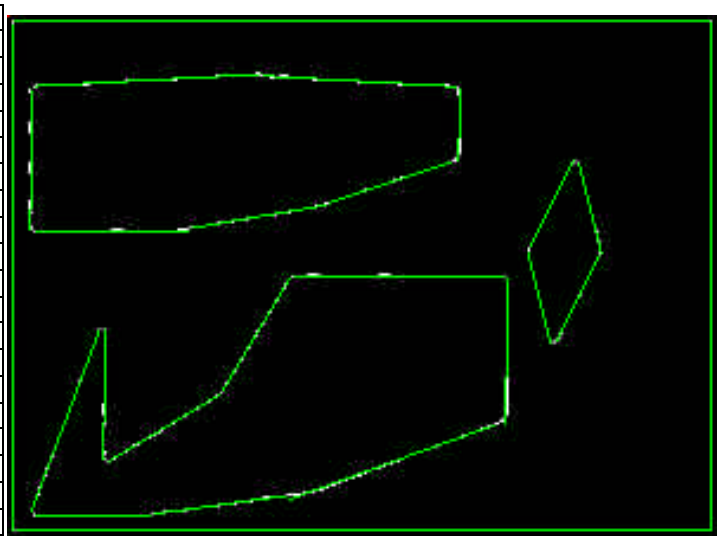
Figura 90 – Experimento 9.

Suavização	Detecção	Limiarização	Afinamento	Conexão	Ajustamento
Mediana Análise Variância	Nevatia e Babu	Pun	Supressão não Máxima	Varredura e Rotulação	M.M.Q.

(a)

	a	b	$erro_a$	$erro_b$
1	0,00000	211,00000	0,00000	1,00000
2	-0,13890	218,80099	0,00395	0,22957
3	-0,36254	247,11816	0,01246	-1,63184
4	-0,02081	213,67412	-0,02081	3,67412
5	0,00337	109,33372	0,00337	-0,66628
6	-0,59862	185,29108	0,00138	-0,70892
7	-0,60088	213,39442	-0,00088	-0,60558
8	-0,01704	43,42290	-0,01704	3,42290
9	-0,37592	88,41216	-0,00092	-0,33784
10	-0,00689	91,14814	-0,00689	1,14814
11	-0,17014	102,65894	-0,00347	0,99228
12	-0,33251	123,80042	0,00082	0,46709
13	-0,02254	191,65291	-0,02254	1,65291
14	0,05585	19,21552	0,00030	-0,22892
15	-0,05263	30,17027	0,00292	-0,38529
16	-0,00650	10,02669	-0,00650	0,02669
17	-0,50756	269,89417	-0,00756	-0,10583
18	0,25202	225,48347	0,00202	0,48347
19	-0,50344	300,67172	-0,00344	0,67172
20	0,24681	195,02795	-0,00319	0,02795

(b)



(c)

Figura 91 – Experimento 10.

A tabela 4 apresenta o E.M.Q. (erro médio quadrático) e o E.M. (erro médio) dos erros verdadeiros, obtidos pela diferença entre os valores estimados e valores conhecidos na geração de imagens.

O EMQ é calculado pela expressão (85) e o erro médio pela (86).

$$EMQ = \sqrt{\frac{\sum e_v^2}{n}} \tag{85}$$

$$EM = \frac{\sum |e_v|}{n} \tag{86}$$

onde *n* é o número de retas.

Tabela 3 – Tabela de erros obtidos nos resultados dos experimentos.

	E.M.Q.		E.M.	
experimento	a	b	a	b
1	0,012352	1,660168	0,008721	1,276701
2	0,012352	1,660168	0,008721	1,276701
3	0,013057	1,425218	0,008643	1,168448
4	0,013057	1,425218	0,008643	1,168448
5	0,008350	1,435836	0,006028	1,057686
6	0,008350	1,435836	0,006028	1,057686
7	0,008387	1,029973	0,005141	0,702040
8	0,008387	1,019051	0,005141	0,689203
9	0,008973	1,353050	0,005987	0,929876
10	0,008976	1,351110	0,006023	0,923367

Pode se verificar que os melhores resultados estão relacionados com a utilização do detector de Nevatia e Babu na etapa de detecção de bordas.

4.3 EXPERIMENTOS COM IMAGENS REAIS

As figuras 92 a 101 apresentam os resultados obtidos com a aplicação das mesmas seqüências de etapas utilizadas nos experimentos com a imagem padrão, quando aplicadas sobre um detalhe de uma imagem aérea real.

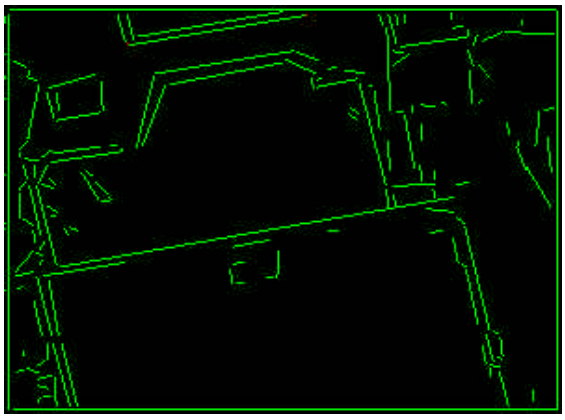
Imagem original

Suavização – Filtro da Média $_{3x3}$

Detecção de Bordas – Nevatia e Babu

Limiarização – Otsu

Afinamento – Supressão não Máxima



Conexão – Varredura e Rotulação

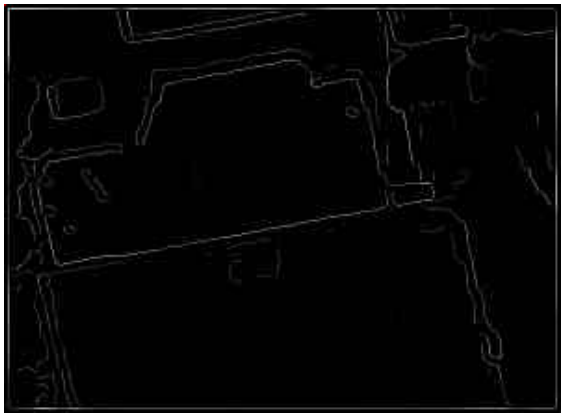
Figura 92 – Experimento 1 (imagem aérea).

Imagem original

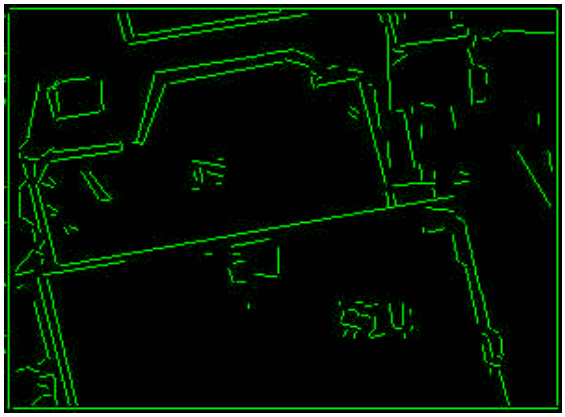
Suavização – Filtro da Média $_{3x3}$

Detecção de Bordas – Nevatia e Babu

Limiarização – Otsu Local



Afinamento – Supressão não Máxima



Conexão – Varredura e Rotulação

Figura 93 – Experimento 2 (imagem aérea).


Imagem original


Suavização – Filtro da Mediana


Detecção de Bordas – Operador de Sobel


Limiarização – Otsu


Afinamento – Supressão não Máxima

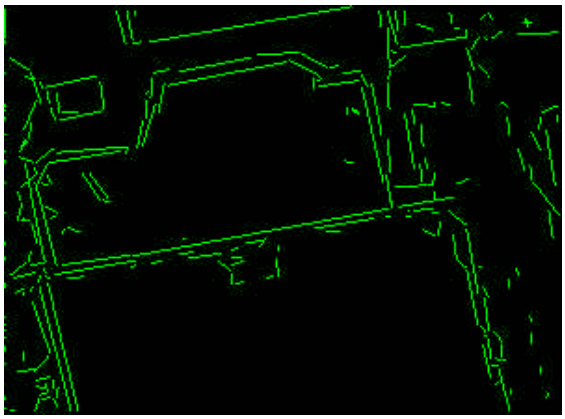

Conexão – Varredura e Rotulação

Figura 94 – Experimento 3 (imagem aérea).

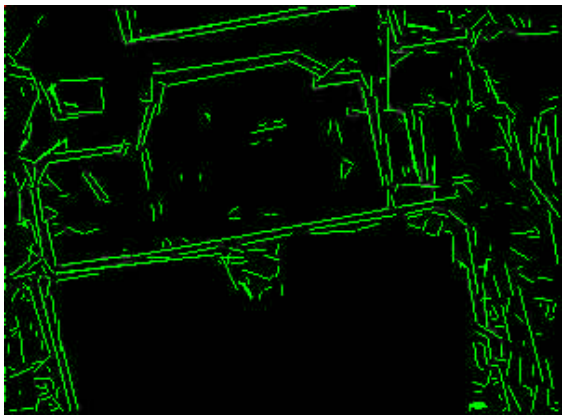
Imagem original

Suavização – Filtro da Mediana

Detecção de Bordas – Operador de Sobel

Limiarização –Triângulo

Afinamento – Supressão não Máxima

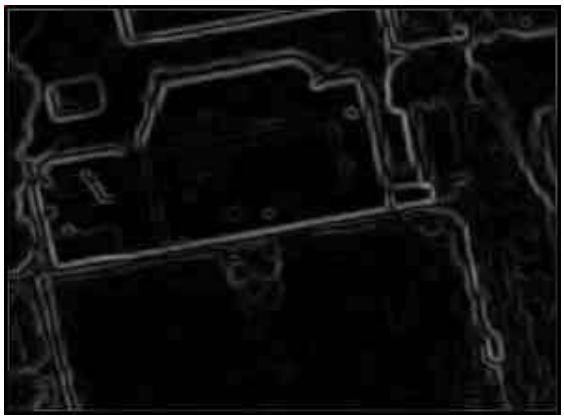


Conexão – Varredura e Rotulação

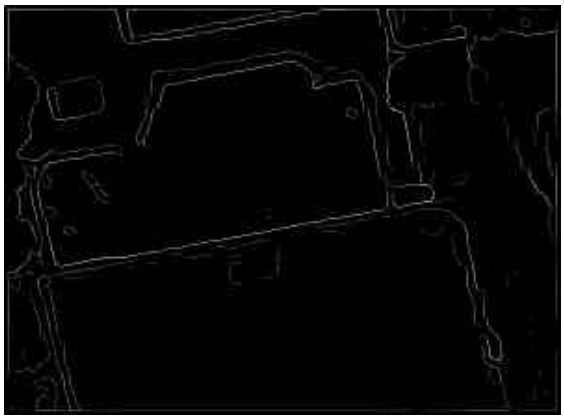
Figura 95 – Experimento 4 (imagem aérea).


Imagem original


Suavização – Filtro da Mediana


Detecção de Bordas – Nevatia e Babu


Limiarização – Otsu


Afinamento – Supressão não Máxima

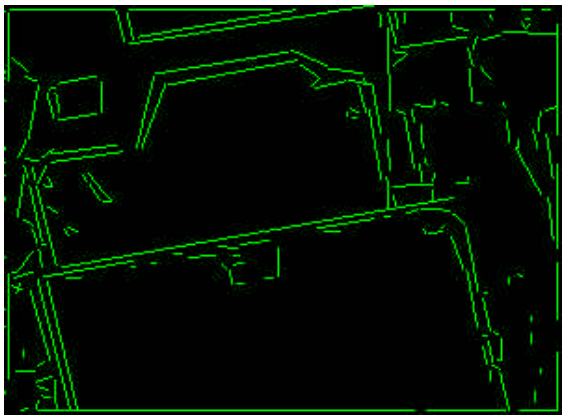
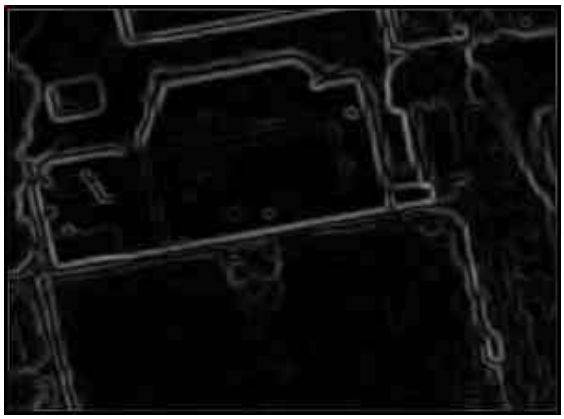

Conexão – Varredura e Rotulação

Figura 96 – Experimento 5 (imagem aérea).

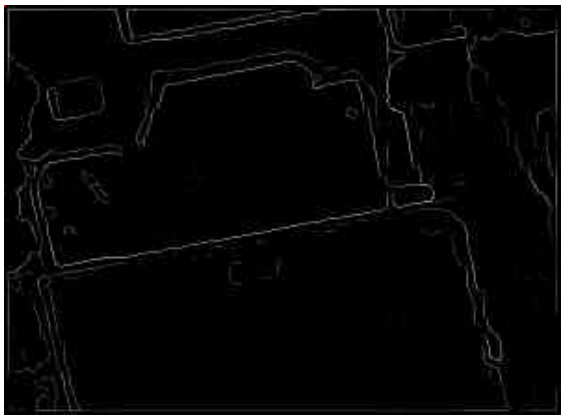
Imagem original

Suavização – Filtro da Mediana

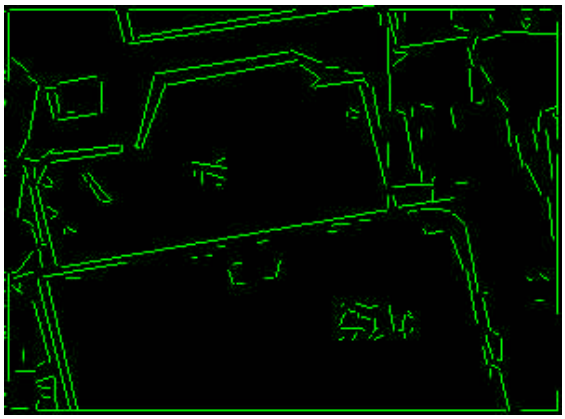


Detecção de Bordas – Nevatia e Babu

Limiarização – Otsu Local



Afinamento – Supressão não Máxima

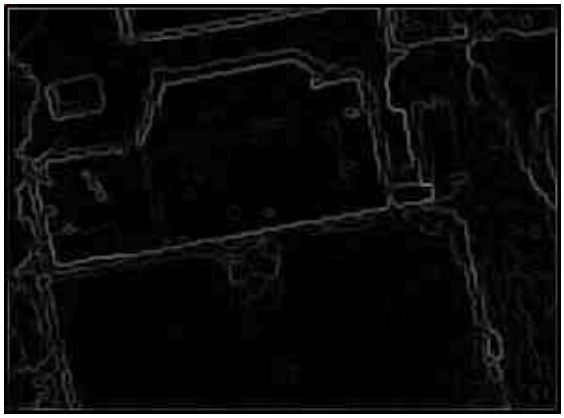


Conexão – Varredura e Rotulação

Figura 97 – Experimento 6 (imagem aérea).

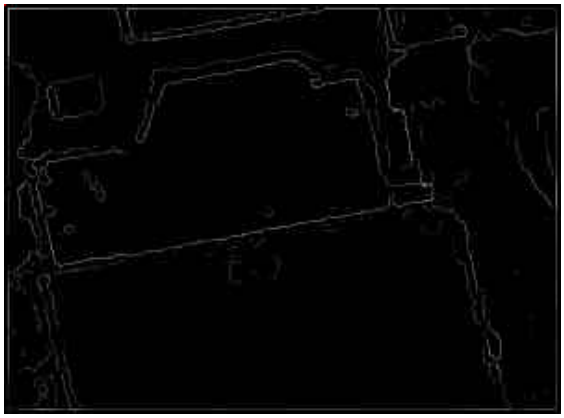
Imagem original

Suavização – Filtro da Mediana Variância Mínima

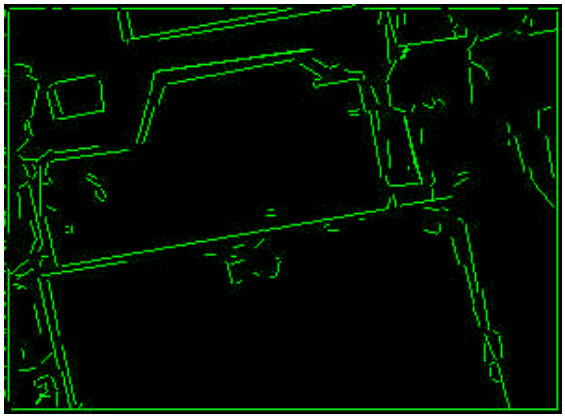


Detecção de Bordas – Operador de Sobel

Limiarização – Otsu



Afinamento – Supressão não Máxima

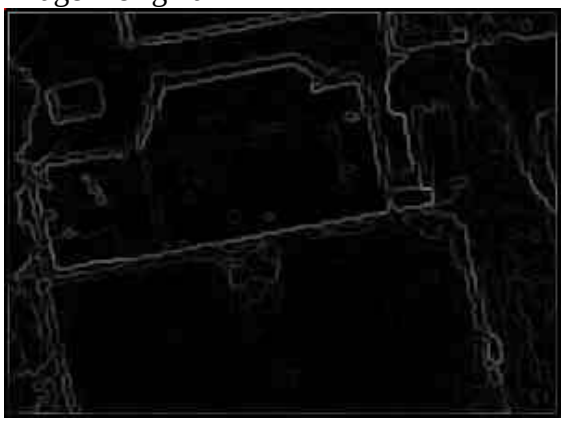


Conexão – Varredura e Rotulação

Figura 98 – Experimento 7 (imagem aérea).

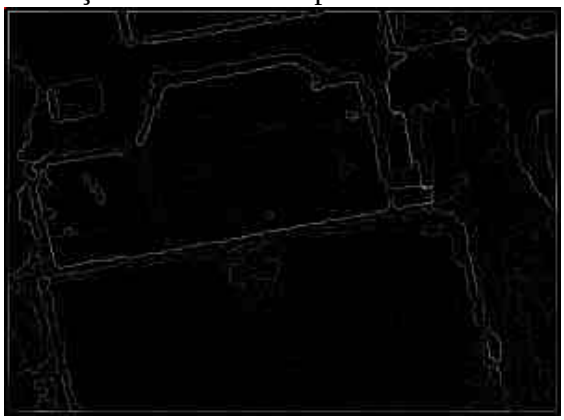
Imagem original

Suavização –Mediana Variância Mínima

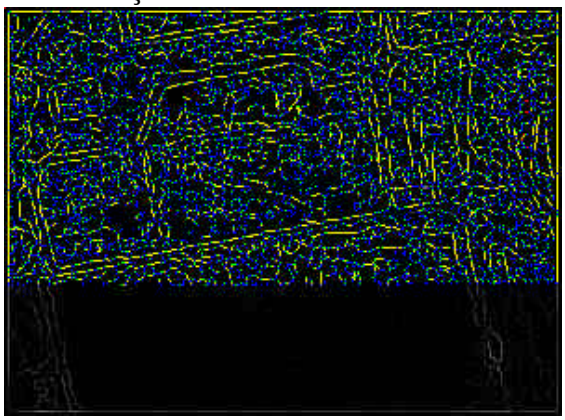


Detecção de Bordas – Operador de Sobel

Limiarização – Pun



Afinamento – Supressão não Máxima



Conexão – Varredura e Rotulação

Após a conexão dos segmentos colineares

Não foi possível fazer o ajustamento.

Figura 99 – Experimento 8 (imagem aérea).

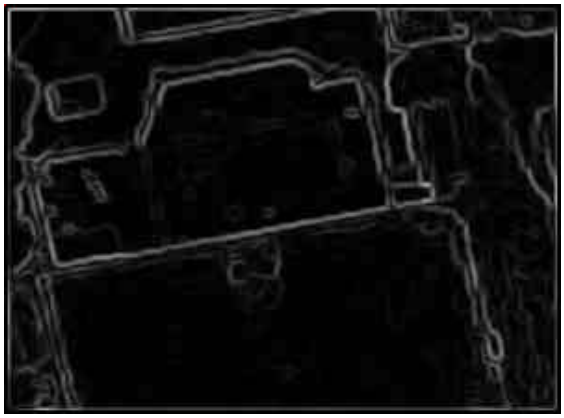
O problema verificado no experimento 8 ocorreu porque o detector de Sobel encontrou muitos detalhes na imagem, e a etapa de limiarização (Método de Pun) não conseguiu eliminar o excesso de informação. Isto gerou um número muito elevado de vetores na imagem (mais de 6000 vetores). A etapa de conexão dos segmentos colineares diminuiu este valor, mas a etapa de ajustamento não consegue chegar ao fim (problemas com recursividade).

O mesmo problema não ocorreu no experimento número 7, que também utilizou o operador de Sobel na etapa de detecção de bordas, porque a etapa de limiarização (Método de Otsu) foi capaz de eliminar o excesso de informação.

O Método de Pun não se mostrou muito eficiente para o trabalho com imagens de bordas de imagens aéreas (imagens reais), porém, foi utilizado neste experimento com uma imagem real, para repetir na íntegra o mesmo experimento realizado com a imagem padrão. Naquele experimento o Método de Otsu Local não pode ser utilizado, devido a ocorrência de regiões totalmente preenchidas com um único tom de cinza (ocasião em que os métodos de limiarização não funcionam), o que dificilmente ocorre com imagens reais. Por isto, os experimentos 8 e 10 foram repetidos, sendo aplicado o Método de Otsu Modificado (local) na etapa de limiarização.

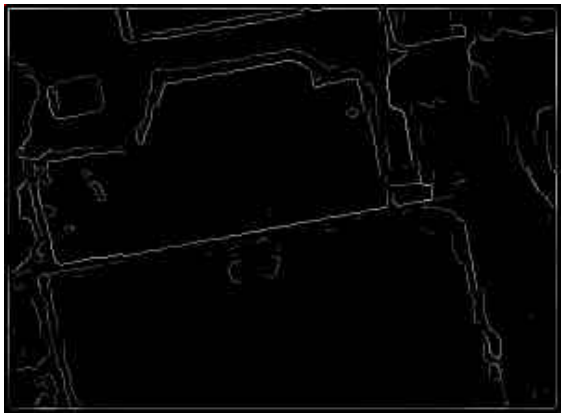
Imagem original

Suavização – Filtro da Mediana Variância Mínima

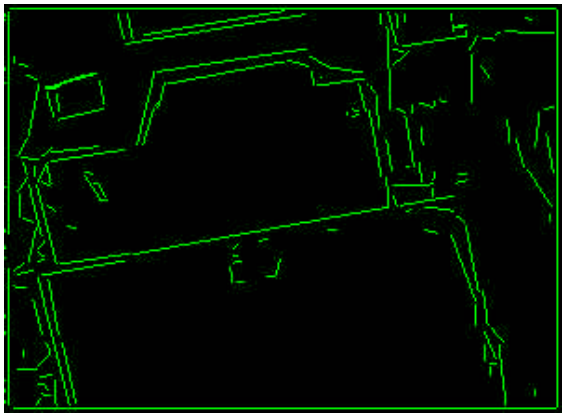


Detecção de Bordas – Nevatia e Babu

Limiarização – Otsu



Afinamento – Supressão não Máxima

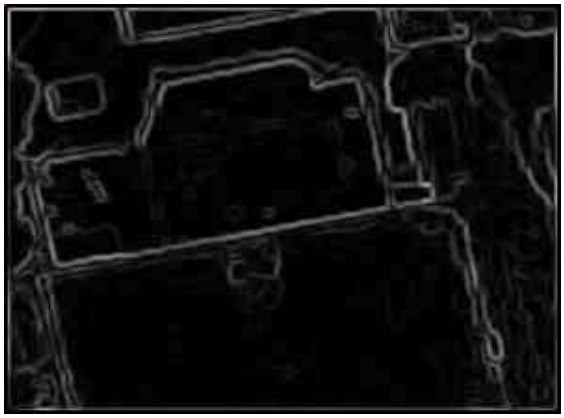


Conexão – Varredura e Rotulação

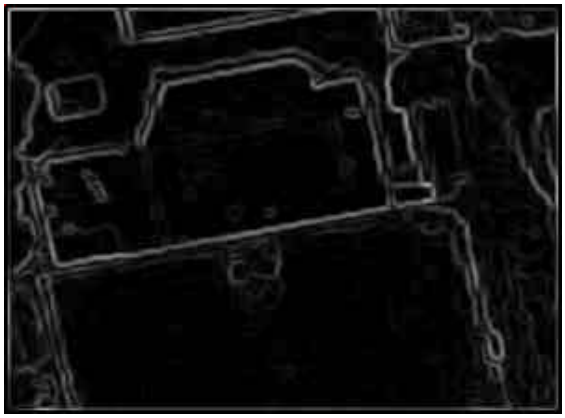
Figura 100 – Experimento 9 (imagem aérea).

Imagem original

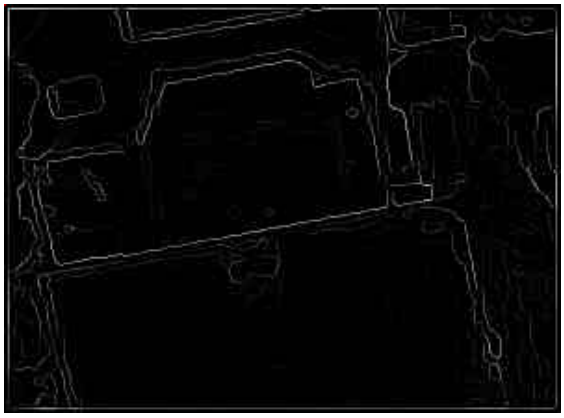
Suavização – Filtro da Mediana Variância Mínima



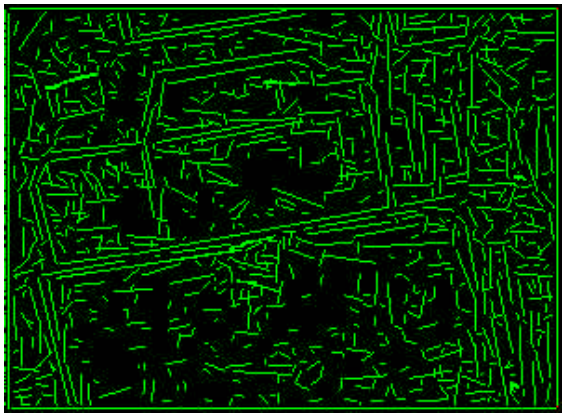
Detecção de Bordas – Nevatia e Babu



Limiarização – Pun



Afinamento – Supressão não Máxima

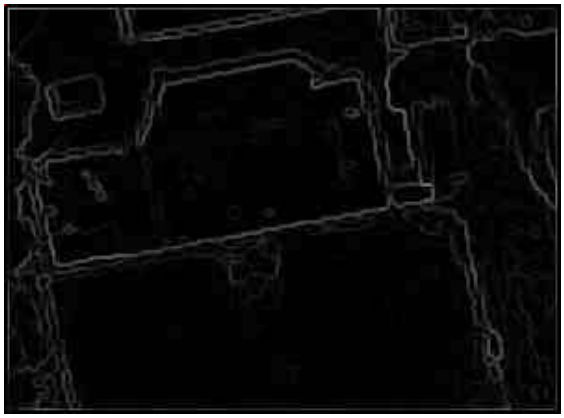


Conexão – Varredura e Rotulação

Figura 101 – Experimento 10 (imagem aérea).

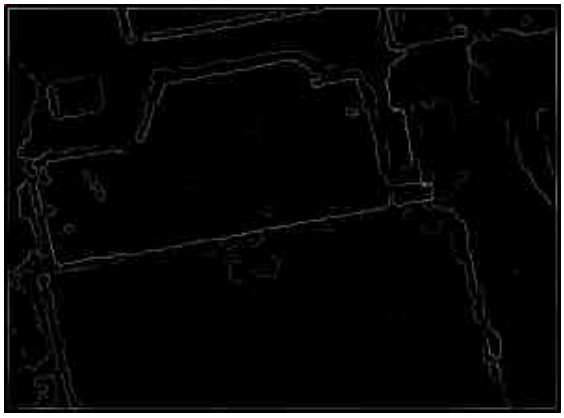
Imagem original

Suavização – Filtro da Mediana Variância Mínima

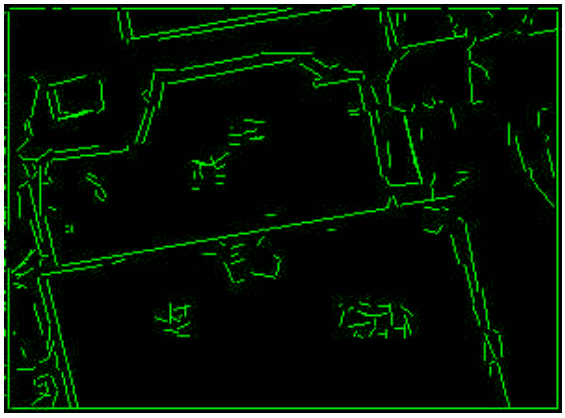


Detecção de Bordas – Operador de Sobel

Limiarização – Otsu Local



Afinamento – Supressão não Máxima

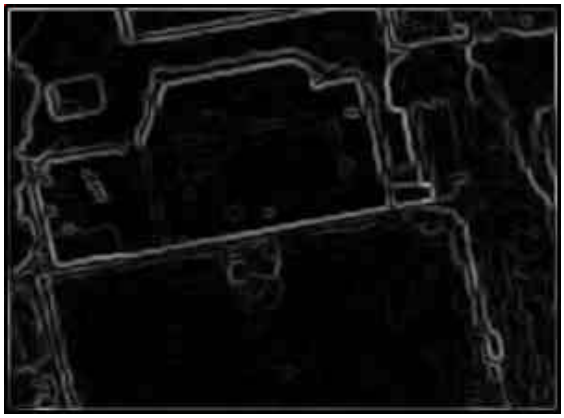


Conexão – Varredura e Rotulação

Figura 102 – Repetição do experimento 8 (imagem aérea), com o Método de Otsu Modificado (Local) na etapa de limiarização.

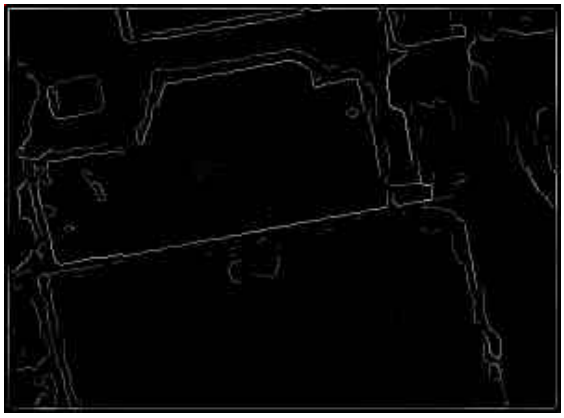
Imagem original

Suavização – Filtro da Mediana Variância Mínima

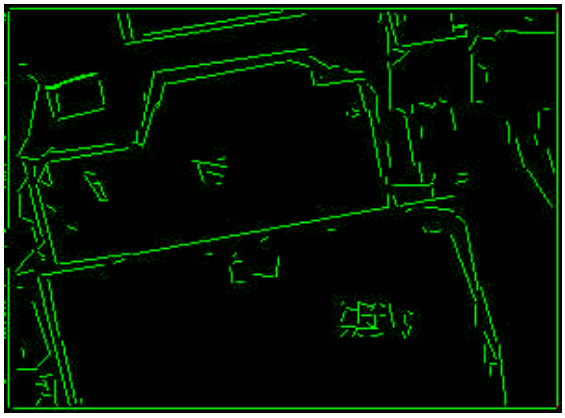


Detecção de Bordas – Nevatia e Babu

Limiarização – Otsu Local



Afinamento – Supressão não Máxima



Conexão – Varredura e Rotulação

Figura 103 – Repetição do experimento 10 (imagem aérea), com o Método de Otsu Modificado (Local) na etapa de limiarização.

Uma análise visual das imagens obtidas nos experimentos deve considerar que bons resultados são alcançados, quando são definidos segmentos grandes, sem muitas quebras, e ainda se verifica a ausência de pequenos segmentos.

Para facilitar a visualização dos resultados, na presença de um número muito grande de segmentos muito pequenos, o sistema desenvolvido possui uma barra de rolagem, onde o usuário pode definir o tamanho mínimo (número de pixels) que um vetor precisa ter para ser visualizado na imagem de saída.

Os experimentos realizados mostraram que na etapa de suavização, o filtro da mediana da região de menor variância é o método que apresenta os melhores resultados. Embora o filtro da média apresente uma boa suavização, não elimina os ruídos, apenas faz um espalhamento dos mesmos. O filtro da mediana apresenta um resultado intermediário entre estes dois.

Na etapa de detecção de bordas, o operador de Nevatia e Babu apresentou um resultado melhor que o operador de Sobel, pois, quando o operador de Sobel é aplicado após uma etapa de suavização onde não foram obtidos bons resultados, verifica-se que o mesmo detecta muitos detalhes, e isto causa a geração de um número excessivo de vetores na etapa seguinte. Este problema pode ser resolvido na etapa de limiarização, caso seja utilizada um método que seja capaz de eliminar este excesso de informação.

Quanto aos métodos de limiarização implementados verificou-se que o Método de Otsu apresenta resultados satisfatórios quando aplicado em imagens de borda, porém, a modificação sugerida no trabalho, denominada Método de Otsu Modificado, que trabalha de forma local, apresentou melhores resultados. Este método apresentou como desvantagem o fato de que dentro das imagens de borda podem ocorrer regiões onde todos os pixels apresentam o mesmo tom de cinza (geralmente áreas de fundo), e neste caso, o Método de Otsu Modificado apresenta problemas. Por outro lado, quando se utiliza o método na imagem toda (Método de Otsu), esta situação dificilmente ocorre. Uma pequena correção que pode ser feita e que deverá resolver este problema é fazer uma verificação dos níveis de cinza presentes em cada região que o método define, e para os casos de regiões com apenas um valor de brilho, assumir que tal região já está devidamente limiarizada, interrompendo o processo.

O método de Pun e do Triângulo apresentaram resultados menos significativos que os resultados obtidos pelos Método de Otsu e Método de Otsu Modificado (proposto neste trabalho).

Na etapa de afinamento de linhas, foi implementado o Método da Supressão-não-Máxima, que foi utilizado nos experimentos realizados no capítulo 4. Além deste método foi implementado também um método que consiste em uma modificação deste método (Supressão-não-Máxima generalizada), que apresenta como vantagem poder operar com direções de bordas com qualquer valor inteiro na faixa de 0 a 360 graus. No caso de utilizar o método tradicional de supressão-não-máxima, isto não é possível, sendo necessário realizar uma discretização destes valores de direções em apenas oito ou doze.

Na etapa de conexão, foi implementado o Método de Varredura e Rotulação (*Scan & Label*), além de um método (proposto) que também faz uma rotulação dos pixels, porém, não utiliza as máscaras de busca apresentadas no método de varredura e Rotulação. Este método faz a rotulação a partir de um pixel inicial, propagando-se para os demais pixels por um processo de inundação (*seed-fill*). Este método, embora não tenha sido apresentado no capítulo de resultados, foi utilizado em vários testes, apresentando o comportamento esperado. Uma desvantagem observada é que o mesmo rotula apenas os pixels que estão em uma vizinhança-8 do pixel de início (o que pode ser modificado, com o aumento da vizinhança na rotina de inundação, porém, restará definir a forma como serão definidos os pixels ponte, a serem inseridos para não deixar quebras nos segmentos) . Outro problema apresentado é que este método não retorna no final da execução os pixels início e fim do segmento rotulado.

O Método de Varredura e Rotulação consegue fazer a rotulação de pixels em uma vizinhança maior (criando pixels ponte no momento da rotulação), obtendo assim um menor número de segmentos, e ainda de maior tamanho, o que é desejável nesta etapa.

O primeiro problema apresentado pelo Método da Inundação (não retornar os pixels início e fim do segmento) pode ser resolvido na etapa seguinte do processamento, com a execução da rotina que faz a determinação dos pixels extremos de todos os segmentos. Quanto ao segundo problema (geração de muitos segmentos pequenos), também pode ser solucionado na etapa de conexão de segmentos colineares (também implementada), que une pequenos segmentos dando origem a segmentos maiores.

A Transformada de Hough, apesar de ser uma técnica robusta e bem conhecida, apresenta grandes dificuldades de implementação (necessidade de uma estrutura muito grande para apresentar resultados adequados), e também não consegue definir o início e

o fim das retas determinadas (o que pode ser realizado em uma etapa posterior a identificação das linhas retas). Por outro lado, os problemas de oclusão que ocorrem em grande abundância nas imagens aéreas podem ser resolvidos utilizando esta técnica. A forma de implementação utilizando o agrupamento q -r representa uma excelente melhora no processo, dispensando a necessidade de uma estrutura muito grande, permitindo uma boa otimização com relação a implementação em um computador.

É possível que uma combinação das técnicas apresentadas seja uma boa opção para utilizar na etapa de conexão (*linking*).

A última etapa implementada no sistema foi o ajustamento dos pixels que possuíam um mesmo rótulo, sendo então obtidos os vetores procurados. Nesta etapa foi utilizado Método dos Mínimos Quadrados – MMQ.

Uma possível continuação deste trabalho deverá ser feita no sentido de obter objetos a partir dos vetores definidos nas rotina desenvolvidas neste sistema. Esta análise deverá considerar o fechamento dos polígonos.

A geração de arquivos de vetores com formatos padronizados, tais como os conhecidos arquivos do tipo DXF ou DWG são também de grande importância em um sistema de conversão de imagens digitais para elementos vetoriais, pois somente desta forma será possível a manipulação dos vetores obtidos, nos mais diversos sistemas de CAD comerciais. A geração destes arquivos é uma tarefa bem trivial, sendo que tais arquivos são caracterizados por possuírem um cabeçalho, onde são colocadas as informações dos elementos (linhas - vetores) que possui, seguidas das coordenadas que definem os vetores.

Finalmente, as rotinas desenvolvidas no sistema deverão ser disponibilizadas sob a forma de uma biblioteca, para que possam ser utilizadas no desenvolvimento de sistemas mais específicos (restituição automática, orientação automática etc.).

CAPITULO 5

CONCLUSÕES E RECOMENDAÇÕES

Neste trabalho foram implementados vários algoritmos para a extração automática de feições (linhas retas). Alguns dos algoritmos existentes precisaram sofrer algumas adaptações, para que pudessem resolver vários problemas observados.

Foram também implementadas algumas etapas complementares, não descritas na literatura, e que se mostraram importantes para a solução de problemas não mencionados na bibliografia.

A parte experimental apresentou algumas seqüências de processamento, e os resultados obtidos (parâmetros *a* e *b* ou *a** e *b** de algumas linhas de borda) foram comparados com os valores verdadeiros (conhecidos na imagem padrão), permitindo uma análise das seqüências realizadas.

Os resultados do capítulo de experimentos mostraram que:

- Na etapa de suavização, o Método da Mediana da região de menor variância (também conhecido por método de suavização com preservação de bordas e cantos) apresentou os melhores resultados.

- Na etapa de Detecção de bordas, o operador de Sobel resultou em um número excessivo de informações. Este problema é explicado pela utilização de uma máscara de tamanho 3x3. A utilização de uma máscara 5x5 deve apresentar uma diminuição neste número de informações. Quando a etapa de suavização remove o excesso de detalhes (ruídos e bordas insignificantes) o operador de Sobel se mostra eficiente.
- O Método de Otsu Local apresentou os melhores resultados nas imagens utilizadas, tendo como único inconveniente, a sua incapacidade de trabalhar em regiões com um único tom de cinza (isto ocorre muito por causa da divisão da imagem em regiões pequenas). Este pequeno inconveniente pode ser resolvido com algumas modificações no algoritmo, tal como sugerido no capítulo 4.
- Na etapa de conexão, um problema ainda a ser corrigido é o aproveitamento dos rótulos que são desativados no processo de conexão de segmentos colineares (quando dois segmentos são unidos, o rótulo do segundo é desativado). Esta correção permitirá uma melhor utilização da matriz de rótulos. Uma solução definitiva será a utilização de uma estrutura de dados dinâmica (ponteiros), o que possibilitará a eliminação completa (da memória) dos rótulos desativados, além de uma criação de novos rótulos mais flexível.
- Após a realização dos experimentos, o sistema desenvolvido sofreu uma reestruturação, passando a utilizar estruturas dinâmicas (ponteiros para matrizes – uma estrutura possível de ser utilizada nas linguagens de última geração), tornando possível a aplicação das rotinas desenvolvidas em imagens de qualquer tamanho (e não apenas em imagens de tamanho fixo - 300 x 200 pixels). Observa-se finalmente que o ambiente de programação utilizado (C^{++} *Builder*) consiste em uma ferramenta de programação com muitos recursos e deverá atender possíveis exigências de acréscimos no sistema desenvolvido.

REFERÊNCIAS BIBLIOGRÁFICAS

ARTERO, A. O.; TOMMASELLI, A. M. G., Técnicas para a Extração de Feições Lineares em Direções Pré-especificadas, In: **Anais do XIX Congresso Brasileiro de Cartografia**, Recife, 1999.

BORATTO, F., **Basic para Engenheiros e Cientistas**, LTC-Livros Técnicos e científicos, Rio de janeiro, 1987.

CAROLI, A., **Matrizes, Vetores, Geometria Análitica: teoria e exercícios**, Editora Nobel, São Paulo, 1984.

CASTLEMAN, K. R., **Digital Image Processing**, Prentice Hall, New Jersey, 1996.

DUDANI, S. A.; LUK, A. L., **Locating Straight-Line Edge Segments on Outdoor Scenes**, Pattern Recognition, Vol. 10, pp.145-157, 19787

EKESTRON, M. P., **Digital Image Processing Techniques**, Academic Press, London, 1983.

GEMAEL, C., **Introdução ao Ajustamento de Observações: Aplicações Geodésicas**, Editora UFPR, Curitiba, 1994.

GONZALEZ, R. C.; WOODS, R. E., **Digital Image Processing**, Addison Wesley, New York, 1993.

HUMES, A. F. P. C.; MELO, I. S. H.; YOSHIDA, L. K.; MARTINS, W. T., **Noções de Cálculo Numérico**, Mc Graw-Hill, São Paulo, 1984.

HUSSAIN, Z., **Digital Image Processing, Pratical Applications of Parallel Processing Techniques**, Ellis Horwood Limited, West Sussex, 1991.

JAIN, R.; KASTURI, R.; SCHUNCK, B. G., **Machine Vision**, Mc Graw-Hill International Editions, Singapore, 1995.

LOW, A., **Computer Vision and Image Processing**, Mc Graw-Hill International Editions, Singapore, 1991.

NEWTON, W., An Approach to the Identification of Forest in thematic mapper Imagery Within the Context of Change Detection System, **M.Sc. Tesis, University College London**, 1993.

PAINE, S. H.; LODWICK, G. D., Edge Detection and Processing of Remotely Sensed Digital Images, **Photogrammetria**, vol. 43, n. 6, pp. 323-336, 1989.

PARKER, J. R. **Algorithms For Image Processing and Computer Vision**, John Wiley & Sons, New York, 1996.

PRATT, W. K., **Digital Image Processing**, John Wiley & Sons, New York, 1991.

PRESS, W. H.; TEUKOLSKY, S. A.; VETTERLING, W.T.; FLANERRY, B.P., **Numerical Recipies in C**, Cambridge University Press, New York, 1992.

SAHOO, P. K.; SOLTANI, S.; WONG, A. K. C., An Survey of Thresholding Techniques, **Computer Vision, Graphics and Image Processing**, n. 41, pp. 233 – 260, 1988.

TOMMASELLI, A. M. G.; TOZZI, C. L., Extração de Linhas Retas em Imagens Digitais, In: **Anais do XVI Congresso Brasileiro de Cartografia**, Rio de Janeiro, 1993.

TOMMASELLI, A. M. G.; TOZZI, C. L. A Recursive Aproach to Space Resection using Straigh Lines, **Photogrammetric Engineering and Remote Sensing**, vol. 62, n. 1, pp. 57 – 66, 1996.

TOMMASELLI A. M. G., Extração Automática de Feições Lineares em Imagens Digitais para a Aplicações Cartográficas, **Relatório de Bolsa de Produtividade em pesquisa, CNPq**, 1999.

VENKATESWAR, V.; CHELLAPA, R., Extraction of Straight Lines in Aerial Images, **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 14, nº11, 1992.

ZHOU, Y. T.; VENKATESWAR, V.; CHELLAPA, R., Edge Detection and Linear Feature Extracting using a 2-d Random Field Model, **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 11, nº1, 1989.