DESCRITOR DE DIAGRAMAS DIGITALIZADOS

Alberto Arkader Kopiler

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Edil S. T. Fernandes, Ph.D.

Prof. Eduardo Peixoto Paz, M.Sc.

Prof. Júlio Salek Aude, Ph.D.

Prof. Luis Paulo V. Braga, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

ABRIL DE 1990

KOPILER, ALBERTO ARKADER

Descritor de Diagramas Digitalizados [Rio de Janeiro] 1990

XIII, 161 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1990)

TESE - Universidade Federal do Rio de Janeiro, COPPE

1. Processamento de Imagens I. COPPE/UFRJ II. Título (série)

Aos meus pais e ao meu irmão

Agradeço à direção do Departamento de Eletrônica do Centro de Pesquisas de Energia Elétrica - CEPEL, Engs. Maurício Moszkowicz e Homero G. de Andrade, pela oportunidade de realização deste trabalho.

Aos pesquisadores do Núcleo de Computação Eletrônica (NCE), Eduardo Peixoto Paz e Tarcísio Neves da Cunha, pela valorosa colaboração e orientação, sem as quais este trabalho não teria sido concluído.

Ao professor Edil S. T. Fernandes pela compreensão e orientação deste trabalho.

Aos colegas do Departamento de Eletrônica pelo estímulo e troca de idéias durante toda a realização da pesquisa.

Resumo da TESE apresentada à COPPE/UFRJ como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências (M.Sc.)

DESCRITOR DE DIAGRAMAS DIGITALIZADOS

Alberto Arkader Kopiler

Abril de 1990

Orientador: Prof. Edil Severiano Tavares Fernandes.

Co-orientador: Prof. Eduardo Peixoto Paz.

Programa: Engenharia de Sistemas e Computação.

Um estudo sobre a digitalização de diagramas utilizando técnicas de processamento de imagens, computação gráfica e reconhecimento de padrões foi elaborado.

Um sistema de digitalização de imagens foi montado e foram implementados algoritmos para testar as diversas etapas que envolvem o processo de digitalização, até se chegar a uma descrição primária do diagrama original.

Esta descrição é compacta, e está voltada para o interfaceamento com editores gráficos baseados em sistemas de computação gráfica semelhantes ao GKS.

Abstract of the THESIS presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M.SC.)

DESCRIPTOR OF DIGITIZED DIAGRAMS

Alberto Arkader Kopiler

April, 1990

Thesis Supervisor: Prof. Edil Severiano Tavares Fernandes.

Thesis Co-Supervisor: Prof. Eduardo Peixoto Paz.

Department: Computing and Systems Engineering.

A survey of digitized diagrams using the techniques of image processing, computer graphics and pattern recognition has been developed.

A system for the digitization of diagrams has been assembled and algorithms for testing the several parts that are related to the digitization process have been implemented up to the point of getting a primary description of the original diagram.

This description is compact and is designed to interface to graphics editors based on GKS-like computer graphics systems.

ÍNDICE

PÁGINA

PÁGINA

PÁGINA

FIGURAS

PÁGINA

FIGURAS

PÁGINA

FIGURAS

PÁGINA

FIGURAS

PÁGINA

ALGORITMOS

PÁGINA

ALGORITMOS

PÁGINA

CAPITULO I

I - INTRODUCÃO

O desenvolvimento tecnológico dos últimos dez anos fez com que a computação gráfica e o processamento de imagens por computador se tornassem populares. O reconhecimento de padrões também apresentou um progresso significativo. Por vários anos a computação gráfica, o processamento de imagens e o reconhecimento de padrões foram tratados como áreas separadas. Mais recentemente, a observação de que tanto os equipamentos quanto as técnicas poderiam ser intercambiadas entre estas áreas evidenciou o ganho que poderia ser obtido ao utilizá-las em conjunto [1,2].

Associado ao desenvolvimento tecnológico que propiciou equipamentos mais rápidos, com maior resolução e mais compactos, houve uma redução em seus preços [3,4]. Isto permitiu uma maior popularização da área de processamento de imagens, alargando sua utilização para aplicações em microcomputadores, e não mais somente para computadores de grande porte ou projetos bem subvencionados financeiramente.

Estes fatos, aliados ao interesse nesta área de pesquisa em franca expansão, contribuíram para que fosse feito um estudo sobre a viabilidade da aplicação destes conceitos aos projetos ora em desenvolvimento na área de sistemas de supervisão e controle do Departamento de Eletrônica do

CEPEL (Centro de Pesquisas de Energia Elétrica - ELETROBRÁS). A equipe CDS (Centros de Supervisão) desenvolve uma família de sistemas de supervisão e controle aplicados a sistemas elétricos. Estes sistemas possuem uma interface homem-máquina voltada para telas semigráficas. Estas telas permitem que um operador acompanhe suas variáveis através da visualização de diagramas unifilares que descrevem todo o sistema. Estas telas são editadas "off-line" com auxílio de um editor semigráfico, a partir de DIAGRAMAS desenhados originalmente em papel vegetal. Pensamos, então, em automatizar o procedimento de entradas de dados gráficos "off-line" através do desenvolvimento de um sistema de processamento de imagens. Este sistema deveria digitalizar os diagramas originais, processá-los e inserí-los dentro de um editor gráfico, que utiliza conceitos de computação gráfica, para apresentação e manutenção posteriores. Para isso, se faz necessário um estudo para analisar o problema e os passos necessários para a sua solução, levando em consideração: a relação custo-benefício - sobre quais aspectos é vantajosa esta automatização? , tempo de processamento, quantificação do volume de informação, estado-da-arte da tecnologia, disponibilidade e custo dos equipamentos envolvidos e simplificações para diminuir a complexidade do problema.

O trabalho aqui apresentado se dispõe a exercitar conceitos através da implementação de algoritmos e testes práticos para que se possa chegar a uma conclusão sobre os resultados obtidos, isto é , se é viável ou não o sistema

proposto.

Este estudo se mostra importante, uma vez que notamos uma tendência cada vez maior para a automatização da documentação nos escritórios ("Paperless Office" - Escritório sem Papel e DIP - "Document Image Processing") [5], o crescimento da área de publicação com o auxílio do computador ("Desktop Publishing" - Editoração Eletrônica) [6], o desenvolvimento de sistemas para reconhecimento de caracteres (OCR - "Optical Character Recognition"), figuras ("Shape Analysis") e visão por computador. Muitos destes sistemas utilizam imagens cujas características são semelhantes às dos diagramas analisados, e para os quais o estudo realizado pode ser estendido. Isto é, apesar de se tratar de uma aplicação específica, as técnicas e ferramentas aqui descritas são, em princípio, válidas para outras aplicações mais genéricas.

O presente trabalho pretende contribuir para o conhecimento na área de pesquisa em imagens pelo fato de reunir uma gama de artigos e livros relativos ao problema proposto, comparar técnicas e métodos e decidir por aqueles que mais se adequam para a solução do mesmo. As dificuldades evidenciadas, os algoritmos revistos de forma crítica e o escopo de atuação contribuem para que o profissional ou estudante das áreas de computação gráfica, reconhecimento de padrões e processamento de imagens tenha uma boa referência de erros e acertos, e possa extrair idéias e conceitos, adaptando-os a sua aplicação.

I.1 - Abordagem Empregada

O objetivo deste trabalho é o de analisar a viabilidade da automatização do procedimento de entrada de dados gráficos "off-line", de forma a substituir a entrada manual via um editor semigráfico, aproveitando todos os diagramas já desenhados em papel vegetal. Qualquer alteração posterior para inserção ou modificação seria feita dentro de um editor gráfico, preferencialmente já existente e que suporte entrada de dados via arquivo intermediário [7]. O sistema proposto deve ser encarado como mais uma entrada gráfica para o editor, assim como existem o "mouse" ou o "joystick", ou melhor dizendo, é um arquivo criado para o editor e não pelo editor gráfico.

A abordagem empregada é a de, uma vez definido o problema em questão, analisá-lo, percebendo características que possam simplificá-lo. Antes de partirmos para a estruturação da solução, é preciso fazer uma estimativa da resolução necessária para que possamos determinar os equipamentos mais adequados para a aplicação. Esta estimativa é importante, pois no processo de digitalização de imagens trabalhamos com amostragem em duas dimensões, e queremos ter certeza de que não haverá perdas na imagem digitalizada, podendo esta ser totalmente recuperada. Uma vez feita a estimativa e escolhidos os equipamentos de acordo com a aplicação, utilizamos técnicas e conceitos de processamento de imagens e reconhecimento de padrões para estruturar a solução definindo cada uma de suas etapas. A cada etapa

está associado um algoritmo que processa e compacta a imagem original até que se extraia uma DESCRIÇÃO do diagrama, que serve como entrada para um editor gráfico.

Os algoritmos de cada etapa são testados com padrões simplificados, alguns destes sintetizados pelo editor de imagens LATIM [8] que é utilizado como ferramenta de depuração, atuando decisivamente na validação dos mesmos.

Optamos, na concepção dos algoritmos, por uma abordagem no domínio do espaço ao invés do domínio da freqüência. Isto é, as técnicas utilizadas estão muito mais próximas da computação gráfica [1] do que do processamento de sinais (imagens) [9]. No decorrer da dissertação deste trabalho serão apontados os motivos que justificam a escolha pelo domínio do espaço.

I.2 - Organização do Texto

O restante do texto é organizado da seguinte forma:

O CAPÍTULO II apresenta o desenvolvimento teórico deste trabalho, com as definições, técnicas e conceitos envolvidos. É feita a análise do problema, a estimativa da resolução e a estruturação da solução.

O CAPÍTULO III trata da implementação propriamente dita. Nele são descritos os algoritmos, as linguagens de programação, os equipamentos, os procedimentos experimentais e

as ferramentas de depuração.

O CAPÍTULO IV apresenta os resultados dos testes de implementação para que se faça uma avaliação quantitativa e qualitativa dos algoritmos.

O CAPÍTULO V contem uma discussão onde os resultados são analisados criticamente e são detalhados os objetivos dos testes realizados.

O CAPÍTULO VI é o último capítulo, onde são apresentadas as conclusões de forma sucinta, já tendo as mesmas sido justificadas no decorrer do trabalho. Além disso são feitas recomendações e sugestões resultantes da pesquisa, são enumeradas questões sem resposta e é proposta a continuação do trabalho.

Por fim vem a bibliografia, contendo toda a literatura referenciada neste trabalho e os anexos.

I.3 - Revisão da Literatura

Existem diversas publicações periódicas e livros-texto sobre computação gráfica, processamento de imagens e reconhecimento de padrões. As principais publicações periódicas sobre o assunto são:

a) Computer Graphics and Image Processing; desde 1983 Computer Vision, Graphics and Image Processing (CGIP).

b) IEEE Transactions on Pattern Analysis and Machine Intelligence, a partir de 1979.

c) Proceedings da IEEE.

d) IEEE Transactions on Computers de 1974 a 1979.

e) ACM Transaction on Graphics.

f) Computer Graphics (SIGGRAPH - ACM).

g) IEEE Computer Graphics.

As principais revisões bibliográficas sobre processamento de imagens, reunidas por A. Rosenfeld, são publicadas anualmente na CGIP desde 1973.

As principais revisões sobre computação gráfica, reunidas por G. F. Schrack, são publicadas na CGIP desde 1976.

Existe, ainda, uma boa revisão bibliográfica sobre segmentação de documentos e técnicas de codificação [10] que trata exatamente da aplicação proposta por este trabalho, e da qual foram extraídas importantes referências que auxiliaram o desenvolvimento deste trabalho.

Existem diversos livros-texto sobre o assunto, mas o que mais se aproximou da proposta deste trabalho foi PAVLIDIS [1] nos capítulos 1,2,3,4,7,9 e 12, pela abordagem no do-

mínio do espaço, pela discussão e formalização da teoria, dos problemas e solução relativos a desenhos de apenas dois níveis, muitos deles já transformados em algoritmos e pela própria didática do autor que facilitou de sobremaneira o entendimento do assunto. Podemos citar ainda os livros de GONZALEZ [9] e PRATT [11] que apesar da abordagem no domínio da freqüência, são muito completos e servem como base de comparação com o livro adotado acima.

Na parte de reconhecimento de padrões tomamos principalmente artigos como base, mas não podemos deixar de citar os livros de GONZALEZ [12] e FU [13] que apresentam uma abordagem sintática e são ótima referência: GONZALEZ [12] nas páginas 160 a 175 apresenta um exemplo completo de sistema de reconhecimento de impressões digitais, muito interessante como ponto de partida para este trabalho, e FU [13] apresenta uma introdução muito bem estruturada, mostrando várias técnicas de pré-processamento. Estas técnicas simplificam o problema original, facilitando o posterior reconhecimento, sendo que cada aplicação deve escolher aquela que lhe é mais adequada.

Na parte de computação gráfica não podemos esquecer de citar NEWMAN [7], pois apesar de na época de sua publicação o padrão GKS ainda não estar pronto, e, portanto, não estar nele descrito, o mesmo apresenta princípios de computação gráfica de caráter geral que vão desde o "hardware" até o "software" interativo, e que podem ser estendidos para sistemas baseados no GKS.

Dentre os artigos pesquisados, serão mencionados agora, apenas aqueles mais importantes e que tratam principalmente de sistemas, os outros estarão todos incluídos na bibliografia com a devida referência quando forem citados.

Em [14] foi apresentado um método de codificação para representação vetorial de desenhos de engenharia. Neste artigo foram introduzidas muitas idéias que fazem parte deste trabalho, principalmente a idéia de utilizar um sistema de CAD para manutenção e alteração de diagramas digitalizados, assim como aproveitar a forma vetorial de armazenamento para posterior reconhecimento de textos e outros elementos. Além disso, já foi feita uma estruturação preliminar dos passos do sistema, quais sejam:

- rastreamento e binarização do diagrama.
- remoção de ruído e preenchimento de vazios.
- deteção de contornos.
- seguimento de linhas.
- vetorização.

O sistema foi apresentado, mas não houve a formalização devida, sendo a abordagem mais informativa e superficial.

O artigo [15] discute um sistema para pré-processamento e segmentação de esquemáticos elétricos. O conceito mais importante introduzido foi o da representação da figura através de grafos que descrevem o diagrama em termos de primitivas gráficas. Notamos, portanto, a preocupação do

autor em segmentar o desenho na forma de grafos, descrevendo-o de modo a facilitar o processamento e compactar a figura.

O artigo [16] é o mais antigo, e por isso, o único que trata do pré-processamento de uma forma mais detalhada. Os outros artigos [14,15,17-21] pressupõe que todo o processamento inicial já foi feito, seja no "hardware" seja no "software", isto é, já é fato consumado. Estes artigos também assumem que a imagem a ser tratada se encontra digitalizada em apenas dois níveis, não tratando da conversão de uma imagem de vários níveis para imagens de apenas dois níveis.

Artigos mais recentes [18,19] descrevem sistemas completos para reconhecimento automático de diagramas eletrônicos, mas onde simplificações, como por exemplo símbolos pré-definidos e características conhecidas a priori, são indispensáveis para diminuir a complexidade do problema.

O artigo [17] mostra um sistema para reconhecimento de documentos com vetorizador, isto é, converte imagens varridas ("raster") para a forma de vetores. O sistema extrai características através do uso de grafos com adjacência de linhas.

O artigo [18] descreve um sistema extrator para entendimento de diagramas de circuitos eletrônicos baseados em topologia, mesma abordagem no domínio do espaço adotado

neste trabalho. O autor já pressupõe que tenham sido feitas todas as operações de pré-processamento, inclusive o afinamento, e apresenta como resultado do sistema, três categorias de elementos: símbolos de circuito, linhas de conexão e caracteres. Nele não há restrições quanto à proporção de tamanho entre símbolos e letras, como já é o caso de [20].

O artigo [19] descreve um sistema de leitura automática de diagramas de circuito e reconhecimento de símbolos com estrutura em "loop". Ele possui facilidades de ajuste de símbolos e métodos híbridos, usando heurística e comparação para identificação de símbolos.

O artigo [20] apresenta um algoritmo robusto para separação de imagens compostas por uma mistura de figuras e texto, como são normalmente formados os diagramas com os quais estamos trabalhando. Faz parte de um sistema automático para análise de documentos e se baseia na idéia de separar direto na imagem em dois níveis, texto de figura. O objetivo da separação é poder aplicar algoritmos específicos para reconhecimento de caracteres e símbolos.

O artigo [21] propõe uma técnica híbrida para combinar classificação de padrões estrutural com estatístico, aplicando-a no reconhecimento de caracteres de diferentes fontes e tamanhos.

CAPÍTULO II

II - FUNDAMENTOS TEÓRICOS

Neste capítulo são apresentadas: a relação entre computação gráfica, processamento de imagens e reconhecimento de padrões, a análise do problema, a classificação das imagens, a estruturação da solução e o detalhamento de cada um dos blocos que compõem a solução.

II.1 - RELAÇÃO ENTRE COMPUTAÇÃO GRÁFICA, PROCESSAMENTO DE IMAGENS E RECONHECIMENTO DE PADRÕES

Antes de apresentar o desenvolvimento teórico propriamente dito do trabalho, é importante mostrar como alguns autores definem as áreas de computação gráfica, processamento de imagens e reconhecimento de padrões e a relação entre elas.

Para TOM WILLIAMS [3], a principal diferença entre processamento de imagens e computação gráfica está na origem da imagem. No processamento de imagens, os dados são retirados de uma fonte externa e processados, enquanto que na computação gráfica, a imagem é gerada pelo computador.

Para ROBIN WILLIAMS [2] a computação gráfica geralmente envolve técnicas de síntese, enquanto que o processamento de imagens está mais ligado à ánalise, existindo numerosas técnicas para analisar uma imagem de forma a extrair in-

formação. Na computação gráfica representamos objetos em um computador de forma a facilitar sua visualização e análise. Em alguns casos a própria imagem é o resultado final esperado, como é o caso da confecção das figuras deste trabalho. No processamento de imagens, esta é rastreada (varrida) e armazenada para posterior processamento. Algoritmos para reconhecimento automático, análise usando heurística e estatística estão sendo desenvolvidos. As imagens podem ser também o resultado final do processamento, como por exemplo, na melhoria da apresentação de uma imagem através do aumento do contraste da mesma.

Para PAVLIDIS [1] computação gráfica trata da geração de imagens a partir de informação não pictórica, enquanto que o processamento de imagens lida com problemas em que tanto a entrada quanto a saída são imagens. Para ele é o reconhecimento de padrões que lida com métodos para produzir uma descrição da imagem digitalizada ou associar uma dada imagem a uma classe determinada. Diferentemente dos outros autores, Pavlidis dividiu em reconhecimento de padrões e processamento de imagens, o que eles convencionaram chamar mais geralmente de processamento de imagens. De um certa forma o reconhecimento de padrões é o inverso da computação gráfica. Tudo começa com uma imagem que é transformada pelo reconhecimento de padrões em uma descrição abstrata: números, símbolos ou um grafo; enquanto que a computação gráfica reconstrói a imagem a partir desta descrição.

As definições para estas três áreas neste trabalho são:

- Computação gráfica é a área da computação que cuida da síntese de imagens a partir de sua descrição abstrata.

- Processamento de imagens é a área da computação que cuida da transformação de uma imagem em outra, para que propriedades da mesma sejam alteradas, de acordo com critérios pré-estabelecidos.

- Reconhecimento de imagens é a área da computação que cuida da análise de imagens para que se obtenha uma descrição abstrata da mesma.

A relação entre estas três áreas é mostrada na figura (II.1).

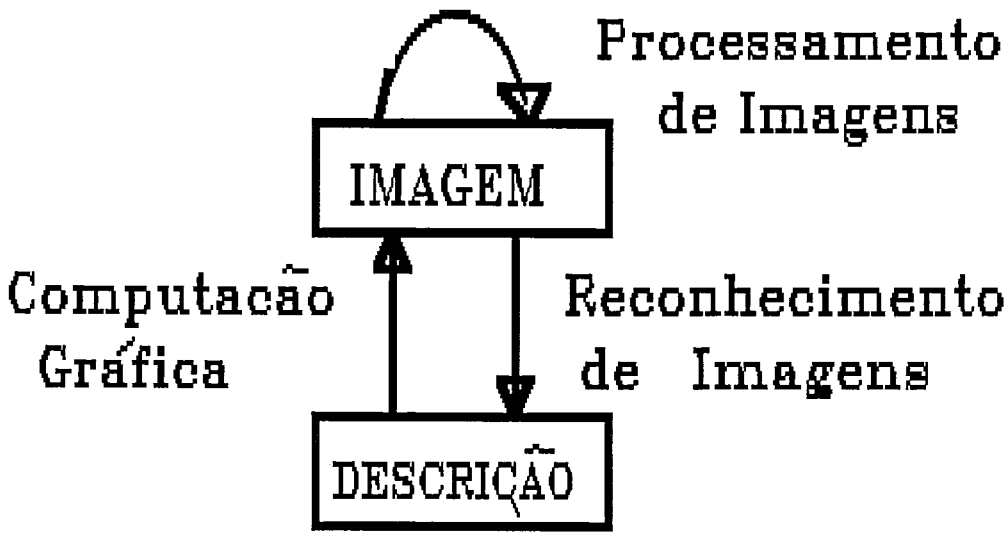

Figura II.1 - Diagrama ilustrando a relação entre computação gráfica, reconhecimento de imagens e processamento de imagens.

Vale a pena ressaltar aqui que as definições adotadas são direcionadas pelo fato de desejarmos fazer interagir estas três áreas, usando as similaridades e complementações que elas fazem entre si. É claro que em cada uma destas áreas existem aplicações que não se enquadram nestas definições. Devemos perceber, também, que passamos a tratar reconhecimento de padrões mais especificamente por reconhecimento de imagens.

Também é importante salientar que os dispositivos de varredura ("raster-scan") utilizados em processamento de imagens, pelo seu baixo preço, vem sendo intensamente utilizados em computação gráfica, apesar de que para esta os dispositivos vetoriais sejam mais adequados. Através de placas gráficas e programas é possível utilizar um dispositivo de varredura para a apresentação de gráficos descritos na forma vetorial. Daí a mistura óbvia de gráficos com imagens.

II.2 - CLASSIFICAÇÃO DAS IMAGENS

As imagens são geralmente representadas por matrizes com elementos inteiros chamados de "pixel" ("picture element"). As imagens coloridas podem ser representadas por três matrizes, uma para cada cor primária (azul, verde e vermelho) ou por uma única matriz onde os bits de cada elemento correspondem a cores diferentes. As imagens podem ser ainda descritas na forma de uma estrutura de dados mais elaborada, como por exemplo em grafos. Na maior parte

das aplicações, esta matriz é quadrada e tem dimensões que variam de 256x256, 512x512 ou 1024x1024. Neste trabalho se manipulará com partes da imagem e não com a imagem inteira, por restrições do equipamento utilizado e pela aplicação que utiliza diagramas de grandes dimensões. Estas partes da imagem são compostas por vários níveis de cinza e são representadas por matrizes. É conveniente pensar nesta imagem como a de um vetor tridimensional (x,y,z), onde a abcissa x representa a coluna de localização do pixel, a ordenada y representa a linha de localização do pixel e a cota z representa um valor inteiro equivalente ao nível de cinza do pixel localizado nas coordenadas (x,y).

As imagens podem ser divididas em quatro classes: 1) imagens coloridas ou com vários níveis de cinza, 2) imagens com apenas dois níveis, 3) curvas e linhas contínuas, e 4) pontos, retas, arcos, círculos e polígonos.

A classe 1 é composta por imagens coloridas ou com vários níveis de cinza, e é a forma usual de imagens de televisão a cores ou preto e branco.

A classe 2 é composta por imagens que possuem apenas dois níveis. A página de um jornal ou a tela de edição de um página de texto são exemplos de imagens de dois níveis ou preto e branco. Estas imagens são normalmente representadas por matrizes com um bit por elemento.

A classe 3 é composta por curvas e linhas contínuas. O

contorno de uma região e formas de onda ou gráficos são exemplos de elementos desta classe. A informação para gerar estas imagens é dada por uma seqüência de pontos que pode ser representada por suas coordenadas (x,y). Uma representação mais eficiente é dada pelo código da cadeia de FREEMAN, aonde é armazenado o ponto inicial com coordenadas (x,y), posicionamento absoluto, sendo que os pontos subseqüentes são armazenados na forma do código da direção (leste=0, nordeste=1, ..., sudeste=7), posicionamento relativo.

A classe 4 é composta por pontos, retas, arcos, círculos ou polígonos, ou seja, primitivas gráficas de um sistema de computação gráfica. Normalmente o ponto é representado por sua coordenada (x,y), a reta é representada pelas coordenadas de seus pontos inicial e final, o arco é dado por três pontos a ele pertencentes, o círculo é definido pelo ponto central e pelo raio, e o polígono pelas coordenadas de seus vértices, podendo ser uma poligonal aberta (ponto inicial # ponto final) ou fechada (ponto inicial = ponto final). Imagens deste tipo são as mais freqüentemente usadas em aplicações gráficas. Apesar do equipamento de apresentação poder ser da classe 2, ou mesmo da classe 1, a representação interna da imagem é da classe 4.

II.3 - Transformação entre Classes

Vários problemas discutidos na literatura podem ser, em última análise, expressos como uma transformação entre

classes, sendo alguns deles representados por transformações dentro de uma mesma classe.

A transformação da classe 1 para a classe 2 é um processo chamado de segmentação. Nele são identificadas as regiões onde a cor ou a intensidade são aproximadamente uniformes. Esta transformação tem como objetivo simplificar o problema original, e/ou extrair informação relativa a alguma propriedade da imagem, ou mesmo melhorar a apresentação de uma imagem de vários níveis em um dispositivo de apenas dois níveis: preto e branco. Como exemplos desta transformação temos:

a) Para a apresentação de imagens de vários níveis em um dispositivo de apenas dois níveis, os artigos [22-26] contêm técnicas e algoritmos para melhorar a apresentação de imagens de vários tons de cinza em dispositivos monocromáticos. Estas técnicas não foram elaboradas para que seja feito um processo de reconhecimento posterior ou codificação, mas algumas idéias podem ser adaptadas para acelerar o processamento de algoritmos mais adequados a nossa aplicação, como por exemplo a média restrita [22].

b) A técnica de deteção de contornos pode ser encontrada nos artigos [27] e [28]. Ela também converte uma imagem de vários níveis em uma de apenas dois níveis, mas, sua finalidade é a de extrair o contorno de uma figura ou separar áreas de nível de cinza constante. O desenho original é formado por linhas finas, sendo que ao aplicar o algoritmo

de deteção de contorno, serão criadas duas linhas marcando o contorno da linha original. Esta característica cria problemas para o processamento posterior de reconhecimento, pois a representação sugerida do código da cadeia pressupõe o acompanhamento da direção, pela linha em si e não pelo seu contorno. A partir do contorno obter a linha original não é trivial e é custoso do ponto de vista do processamento requerido.

c) A técnica de binarização está descrita em [29] e [30] e é a que melhor resultado apresentou para a aplicação proposta.

A transformação da classe 1 para a classe 2, então, é a conversão de uma imagem de vários níveis para apenas dois (quantização). A imagem original pode ter vários níveis (ex: fotografia de uma pessoa) ou apenas dois (ex: diagramas de esquemas eletrônicos). No primeiro caso a transformação tem por objetivo retirar imformações relativas ao contorno, medir a distância entre pontos ou separar áreas uniformes. No segundo caso nós podemos utilizá-la para codificação e reconhecimento de imagens.

A transformação da classe 2 para a classe 3 é feita geralmente por algoritmos de afinamento ou seguimento de contornos. Neste último a região é mapeada em uma curva fechada, enquanto que no primeiro é extraído o esqueleto da figura original.

A transformação da classe 3 para a classe 4, às vezes chamada de segmentação de curvas, é o processo em que se procura os pontos críticos ao longo do contorno. Entre estes pontos, também chamados de característicos, tentamos aproximar uma reta ou uma curva. A que mais se aproximar é definido como ponto, reta ou arco. Analisando de forma mais completa, podemos acescentar ainda círculos, polígonos (linha poligonal aberta ou fechada). Esta transformação está relacionada à área de reconhecimento de imagens.

Podemos observar, em última análise, que a proposta deste trabalho é a transformação da classe 1 para a classe 4. Isto é, a partir de um diagrama do setor elétrico obtemos uma imagem digitalizada, representada por uma matriz com vários níveis de cinza. Esta imagem é convertida para dois níveis, afinada, encontrado seus pontos característicos e segmentada em arcos e retas.

A transformação dentro de uma mesma classe está relacionado ao processamento de imagens, a transformação da classe 1 para a classe 4 ao reconhecimento de imagens e a transformação inversa da classe 4 para a classe 1 à computação ou processamento gráfico.

A seguir é analisado o problema de forma mais específica e é definido o diagrama em blocos da solução.

II.4 - Análise do Problema

Ao iniciarmos a análise, é importante perceber características que simplifiquem o problema original. As características da nossa aplicação como pode ser visto na figura (II.2), são:

a) desenho preto em fundo branco com dimensões próximas ao formato A0.

b) composta de caracteres, linhas e símbolos com traçado fino.

c) a maior parte do desenho corresponde ao seu fundo (80%) e a menor a sua frente (20%).

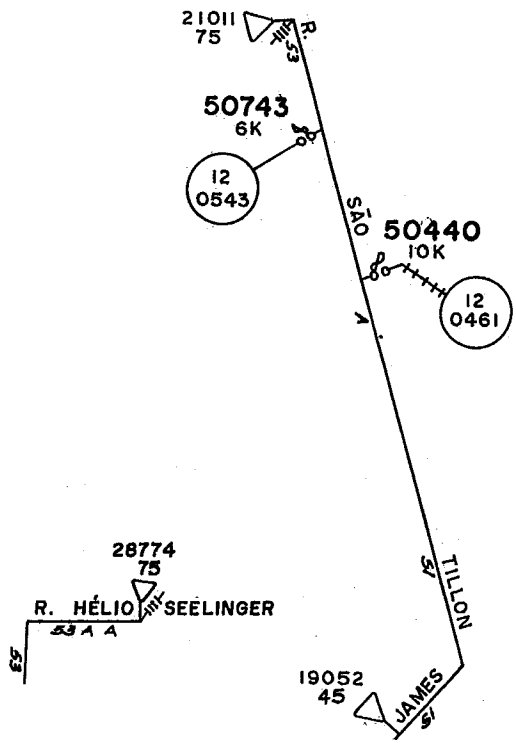

Figura II.2 - Mostra de um pedaço de um diagrama.

Estamos, portanto, trabalhando com figuras da classe 2, ou seja, desenhos e textos que apresentam apenas dois níveis: preto e branco. Ao fazermos a digitalização, apesar da imagem original ser da classe 2, ela adquire níveis de cinza depois de feita a amostragem, passando a ser uma imagem classe 1. Devemos, portanto, aplicar uma transformação da classe 1 para a classe 2, de tal forma a retornar às características originais da imagem de possuir apenas dois níveis e para fins de compactação. O método de transformação a ser adotado deve ser aquele que mais se adequa ao objetivo de reconhecimento e à representação voltada para o código da cadeia. Escolhemos, portanto, a binarização como método de transformação.

Uma vez dispondo de uma imagem da classe 2, percebemos que as linhas, caracteres e símbolos originais do desenho, que são finos, se tornavam grossos. Isto se dá pelo fato de não poder haver perdas na imagem digitalizada, fazendo com que se aumente o número de pontos da matriz de amostragem, de forma a garantir a condição inicial. Devemos, portanto, cogitar da aplicação de uma transformação da classe 2 para a classe 3, para fins de compactação, simplificação e direcionamento para codificação segundo a regra da cadeia. A técnica de afinamento é o processo mais adequado para esta transformação, resultando em um esqueleto de grossura unitária e um único caminho a ser percorrido.

Neste ponto ainda representamos a imagem por uma matriz binária, por razões de simplicidade, facilidade de depura-

ção e visualização dos resultados. Aproveitamos a transformação da classe 3 para a classe 4, na qual é feita a trilha ou seguimento de contorno para busca de pixels característicos, para além da segmentação fazer a codificação segundo o código da cadeia de FREEMAN. Após terem sido descobertos os pixels característicos, devemos percorrer os pixels entre eles, verificando se estes se aproximam mais a uma curva, reta ou conjunto de curvas e/ou retas. Neste último caso mais pixels característicos devem ser incluídos. Este processo faz parte da transformação da classe 3 para a classe 4, e é chamado de reconhecimento primário.

Por fim devemos criar um arquivo compatível com o padrão gráfico do editor gráfico (ex: GKS). Nele são armazenados as primitivas gráficas que descrevem o diagrama original, e a partir das quais poderá ser redesenhado e alterado.

A seguir são apresentadas duas figuras: a figura (II.3) mostra um diagrama em blocos que descreve todas as etapas do sistema de análise de imagens e a figura (II.4) mostra a forma de representação dos dados de cada uma das etapas que devem ser implementadas.

Pela figura (II.3) podemos observar que este processo de análise de imagens é composto pelos seguintes blocos principais:

a) Digitalização

b) Pré-Processamento

c) Pós-Processamento

d) Segmentação

e) Interface com programa de computação gráfica

f) Reconhecimento Primário

g) Classificação

h) Reconhecimento

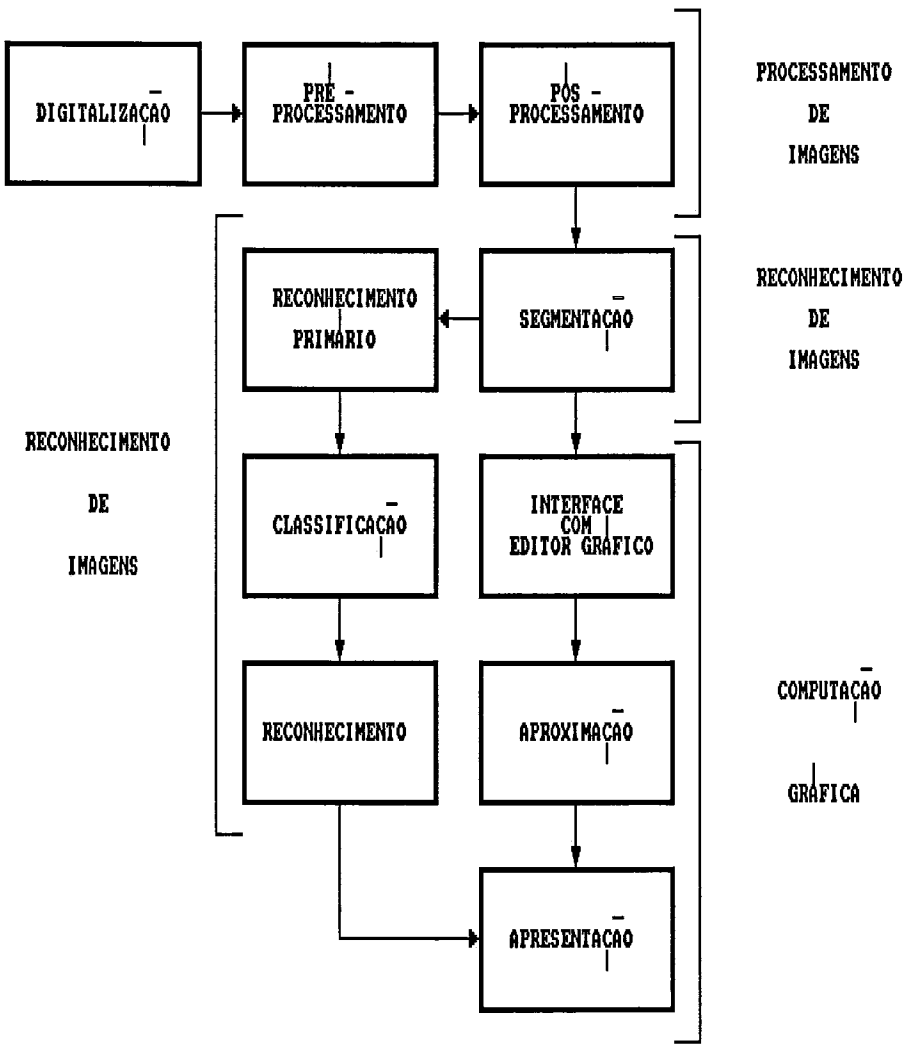

Figura II.3 - Diagrama em Blocos representando o Sistema de Análise de Imagens Proposto.

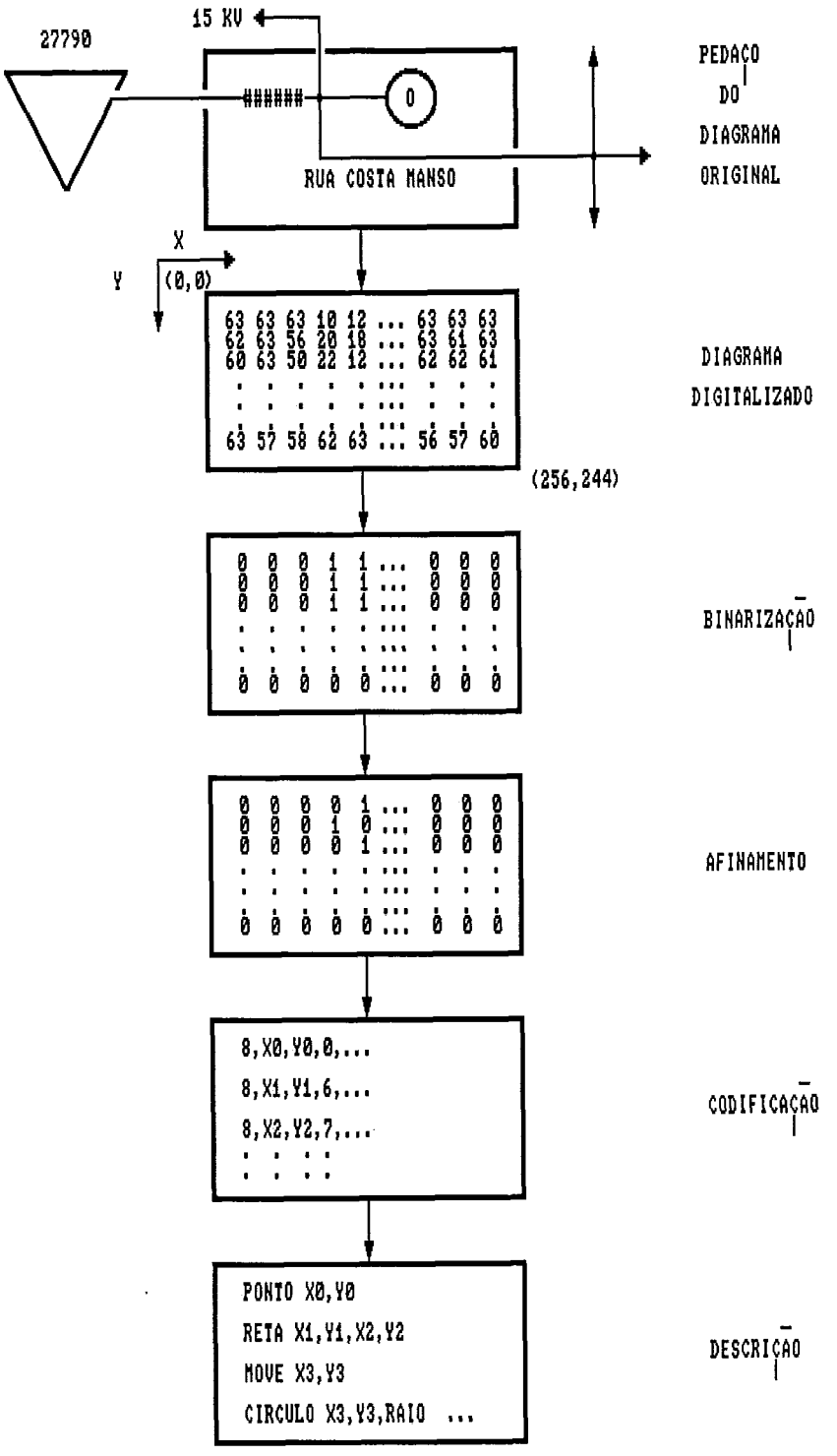

Figura II.4 - Diagrama mostrando o Fluxo de Dados ao longo dos Blocos do Sistema Proposto.

A seguir são descritos cada um dos blocos:

a) Digitalização - A digitalização é o processo pelo qual a partir de um dispositivo de conversão A/D, é feita uma amostragem em duas dimensões de uma imagem, obtendo-se geralmente uma matriz p x q com n níveis de cinza. Normalmente a matriz resultante da amostragem é quadrada (p = q) e assume dimensões que são potência de 2 (p = q = 2**r = 256, 512, 1024) e os valores correspondentes ao número de níveis também (2**s = 64, 128, 256). Neste ponto, ainda não entraremos na escolha do dispositivo de digitalização. A abordagem, agora, será independente de dispositivo, mas é importante ressaltar que devemos fazer a estimativa da resolução dos equipamentos de acordo com a aplicação. É necessário também, calcular o volume de dados a processar e o tempo de processamento para dimensionarmos os dispositivos de armazenamento e processamento. Esta parte será vista no capítulo III, que trata dos materiais e métodos experimentais.

O processo de digitalização consiste, portanto, na digitalização de parte ou de toda a imagem e no seu armazenamento em um arquivo em disco para posterior processamento. O fato de digitalizarmos apenas parte da imagem está relacionado a restrições nos equipamentos disponíveis ou a aplicações em que é suficiente a análise de um pedaço da imagem apenas. No nosso caso, o ideal seria digitalizarmos a imagem de uma só vez, mas restrições quanto ao equipamento não nos permitiu, criando o problema de interligação dos vários pedaços.

b) Pré-Processamento - O pré-processamento consiste em observarmos a imagem digitalizada, percebendo se existem ruídos nela. Casos existam ruídos do tipo gaussiano ou sal e pimenta, e isto pode ser facilmente percebido através da digitalização e visualização de imagens padrão, basta passarmos filtros específicos para eliminar o problema. Podemos perceber, também, se o contraste não está bem ajustado e proceder uma equalização de histograma. Estaremos, portanto, executando transformações dentro da classe 1. Também incluímos dentro da definição de pré-processamento o procedimento de binarização da imagem, pois a imagem originalmente classe 2 ao ser digitalizada passou a se comportar como classe 1. Após o pré-processamento teremos como resultado uma imagem classe 2.

c) Pós-Processamento - O pós-processamento é composto pelos processos de afinamento e codificação. O afinamento compacta a imagem, sendo que as linhas da imagem original devem ter uma grossura homogênea, caso contrário perderemos a informação de grossura relativa às linhas. Para que isto não ocorra é preciso acrescentar mais informação à matriz binária, mas isto não foi necessário no nosso caso.

O processo de afinamento deve ser de tal forma a não alterar a topologia da imagem [31], mantendo a conectividade dos pontos que estão ligados entre si. A partir da matriz binária afinada fazemos uma varredura transformando-a, através da codificação da cadeia, em um vetor de direções, com ponto inicial (x,y) absoluto e direções seguintes re-

lativas.

d) Segmentação - A segmentação consiste em percorrermos o vetor codificado, encontrando os pontos característicos. Os pixels entre cada dois pontos característicos devem ser reconhecidos como arcos, círculos, retas e polígonos. Isto é, obtemos uma descrição primária da imagem original.

e) Interface com o Programa de Computação Gráfica - Neste bloco é feita a interface que compatibiliza a descrição primária obtida com o padrão gráfico do editor já desenvolvido ou a desenvolver. A partir daqui existem duas tendências distintas: a primeira relacionada à computação gráfica onde técnicas de interpolação, suavização e aproximação de curvas (splines e curvas de Bezier [1] e [7]) são usadas para melhorar a apresentação; a segunda relacionado ao reconhecimento de imagens onde tentamos juntar curvas e retas entre si de acordo com critérios pré-estabelecidos para processar o reconhecimento de caracteres e símbolos.

f) Reconhecimento Primário - O reconhecimento primário é a análise das primitivas gráficas de uma imagem e seu agrupamento para posterior classificação.

g) Classificação - A classificação é o processo de analisar os grupos formados no reconhecimento primário segundo um critério, e tentar separar, por exemplo, aqueles que são considerados desenho dos que são mais provavelmente

caracteres. Por exemplo, podemos estabelecer o critério para separação de texto de desenho no qual segmentos interligados formando um grupo com comprimento maior do que um comprimento máximo pré-estabelecido serão considerados desenho, caso contrário serão caracteres. Isto para o caso dos símbolos serem maiores do que as letras. Portanto, o esquema de classificação adotado depende da aplicação. Teremos ao final os grupos separados por classe.

h) Reconhecimento - O reconhecimento é o processo no qual são aplicados algoritmos por classes para identificação dos elementos e sua descrição de uma forma mais compacta. Caso usássemos a classificação do exemplo anterior, poderíamos aplicar um algoritmo específico para reconhecimento de caracteres e outro só para reconhecimento dos símbolos. Caso contrário, teríamos de utilizar um algoritmo mais genérico, que provavelmente seria mais complexo, mais sujeito a confusões entre símbolos e caracteres. Neste ponto percebemos que se tivermos sucesso no reconhecimento de símbolos e caracteres, teremos uma informação mais compacta e poderemos realimentar o editor gráfico não mais somente com retas e arcos, mas em um nível de abstração maior, com caracteres ASCII e código dos símbolos que permitem sua reconstrução a partir de descrições já inseridas dentro do editor.

II.5 - Estruturação da Solução

O escopo deste trabalho só vai até o bloco representando a

interface com o editor gráfico (figura II.3), o que corresponde a uma transformação da classe 1 para a classe 4. A seguir é apresentada a estruturação da solução proposta, sendo que cada um dos blocos é detalhado, colocando as definições e o desenvolvimento teórico necessário.

A solução proposta está estruturada da seguinte forma:

a) Estimativa da Resolução
b) Interligação
c) Pré-Processamento
d) Afinamento
e) Trilha de Caminho
f) Segmentação
g) Interface com o Editor Gráfico

II.5.1 - Estimativa da Resolução

Como já foi levantado anteriormente, procuramos dar uma abordagem independente de equipamentos. De qualquer forma colocamos este passo como o primeiro. Isto se deve ao fato de precisarmos analisar a aplicação para então adquirirmos equipamentos, trabalharmos nos algoritmos, fazermos testes, analisarmos os resultados e concluírmos sobre eles. Portanto, precisamos estimar o volume de dados, a resolução dos dispositivos e o tempo de processamento, antes de qualquer coisa, para o caso de um projeto a ser executado. Pelo caráter acadêmico deste trabalho, achamos por bem fazer uma abordagem mais geral, teórica e didática. No pró-

ximo capítulo detalharemos este item.

II.5.2 - Interligação

A existência desta fase depende do item anterior. A nossa aplicação é composta por diagramas de grandes dimensões (formato A0). Exitem dispositivos rastreadores capazes de digitalizar diagramas de tamanho grande. Neste caso a imagem digitalizada armazenada não estaria dividida, dispensando este processamento. Comercialmente é mais fácil e barato encontrar dispositivos para digitalizar diagramas de formato A3 ou A4, caso em que a interligação é necessária. Esta fase, portanto, depende do dispositivo e só é abordada agora, porque é aqui o lugar ideal para fazermos a junção das partes da imagem. Neste ponto, ainda não escolhemos os dispositivos, só levantamos o problema e sugerimos que o ideal para a nossa aplicação, é o de digitalizar a imagem de uma vez. Se não for possível, teremos que pensar em como interligar as várias partes.

Um dispositivo mecânico de precisão é necessário para posicionar o digitalizador, sendo interessante digitalizar áreas das partes vizinhas para facilitar a interligação. O "hardware" é fundamental neste caso, principalmente se a aplicação envolver mapas geográficos. O processamento necessário, geralmente é o de fazer a equalização dos histogramas entre vizinhos na parte de fronteira, para que não fique visível a junção entre eles. A interligação pode ser feita de forma interativa com o auxílio de um editor, ou

automática, ou mesmo já fazer parte do próprio "hardware", o que seria ideal.

Apesar de ter sido abordada, a interligação não será implementada, pois testes iniciais demonstram a necessidade de um dispositivo mecânico, não disponível. Nos itens posteriores estaremos sempre lidando com partes da imagem, o que não invalida em nada este estudo, uma vez que tudo que for aplicado para uma parte poderá ser estendido para toda a imagem.

A sugestão de fazermos a interligação neste ponto e não posteriormente se deve ao fator da simplicidade e da fidelidade. Se a interligação fosse feita mais tarde estaria mais sujeita a erros de defasagem na superposição e implicaria em uma estrutura de dados mais intrincada.

II.5.3 - Pré-Processamento

Antes de discutir os passos envolvidos no pré-processamento é importante analisar questões do tipo : qual o domínio que deve ser escolhido para descrever a imagem e qual o modelo que melhor se adapta a ela.

II.5.3.1 - Domínio do Espaço X Domínio da Freqüência

É neste momento que aflora a questão se devemos usar o domínio do espaço ou uma transformada no domínio da freqüência (ex: FFT [1,9,11]) , ou ainda alguma outra transforma-

da (exs: Hadamard, Walsh, Hotteling, Cosseno, etc. [9,11]) para processar a imagem. Segundo NETRAVALI [32] bastam dois breves comentários em favor da descrição do processo no domínio do espaço:

1) O efeito da iluminação de um ponto percebido pela visão é um efeito altamente local e depende fortemente da configuração espacial, isto é, de como está disposta a sua vizinhança. Como exemplos podemos citar o efeito "mach band" [9] e [11], onde pixels com a mesma intensidade podem parecer mais claros ou mais escuros dependendo da intensidade dos pixels ao seu redor.

2) O campo visual apresenta uma não homogeneidade muito grande, com uma função de espalhamento do ponto ("point spread function") que muda rapidamente com a localização do campo visual. Como exemplo podemos citar a digitalização por câmera, onde após uma pequena modificação da posição da câmera para uma mesma imagem, geralmente obteremos resultados diferentes.

Estes efeitos são mais facilmente descritos e manipulados no domínio do espaço.

II.5.3.2 - Modelo de Imagem

O termo imagem se refere a função de intensidade de luz em duas dimensões, denominada de f(x,y), onde o valor ou amplitude de f nas coordenadas espaciais (x,y) nos dá a in-

tensidade (brilho) da imagem naquele ponto. Como a luz é uma forma de energia e f(x,y) precisa ser não nulo e finito, ou seja 0 < f(x,y) < infinito. A natureza básica de f(x,y) é caracterizada por duas componentes: a quantidade de luz incidente sobre a cena e a quantidade de luz refletida pela cena. Estas componentes são chamadas de iluminação (i(x,y)) e reflexão (r(x,y)), sendo que

$$f(x,y) = i(x,y) * r(x,y)$$
$$\text{para } 0 < i(x,y) < \text{infinito e}$$
$$0 < r(x,y) < 1$$

A natureza de i(x,y) é determinada pela fonte de iluminação, enquanto que r(x,y) é determinada pelas características do objeto na cena, sendo que r(x,y) = 0 para a absorção total e r(x,y) = 1 para a reflexão total.

A partir de agora, a intensidade de uma imagem monocromática f nas coordenadas (x,y) será chamada de nível de cinza da imagem naquele ponto.

Para que uma imagem fique em uma forma adequada para o processamento de imagem a função f(x,y) precisa ser digitalizada tanto no espaço quanto na amplitude. A digitalização das coordenadas espaciais (x,y) será chamada de amostragem da imagem, enquanto que a digitalização da amplitude será chamada de quantização dos níveis de cinza.

A resolução da imagem, isto é, o grau de detalhe que con-

seguimos perceber, é um compromisso entre o número de níveis de cinza e o número de pontos da matriz de amostragem. Além disso, a qualidade da imagem é uma propriedade subjetiva que depende dos requerimentos da aplicação. Por exemplo: a televisão preto e branco tem 525 linhas, cada linha com 525 elementos e 256 níveis de cinza. Se aumentássemos o número de pontos para 1024 x 1024 e reduzíssemos o número de níveis de cinza para 32, diminuiríamos a "qualidade da imagem" classe 1, mas aumentaríamos a da classe 2. Isto ocorre porque esta última, por apresentar apenas dois níveis de cinza, não é influenciada pela diminuição do número de níveis, mas é beneficiada pelo aumento do número de pontos de amostragem.

II.5.3.3 - Binarização

Como já foi abordado anteriormente, uma imagem originalmente classe 2, ao ser digitalizada passa a se comportar como uma imagem classe 1, por causa da função de espalhamento do dispositivo de digitalização. A imagem passa a apresentar, portanto, uma larga faixa de valores. Podemos tentar retornar a classe 2 através de uma transformação na qual precisaríamos de apenas um bit para representar cada elemento: atribuiríamos o valor 0 para o pixel com valor menor do que um limiar e o valor 1 para o pixel maior ou igual ao mesmo limiar. Este processo é denominado de binarização [29]. O que ocorre é que aos pixels que correspondem a células de amostragem próximas à fronteira de regiões serão atribuídos valores 0 e 1 de uma forma mais ou

menos arbitrária, podendo ser acesos pixels que deveriam estar apagados, e apagados pixels que deveriam estar acesos, criando vazios ou preenchendo espaços de forma equivocada. Por esta razão devemos escolher uma matriz de amostragem suficientemente grande, isto é, com a malha fina, de tal forma que mesmo para a parte mais estreita da região, uma célula da amostragem está inteiramente contida em área de única cor (figura II.5).

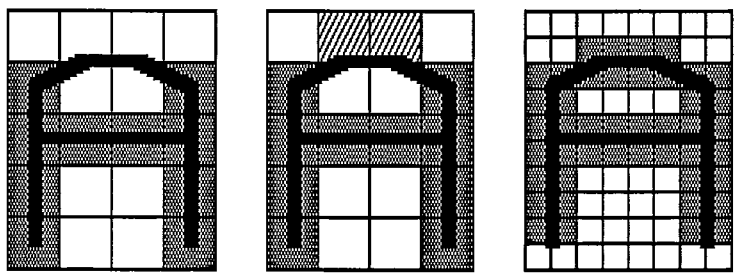

Figura II.5 -

(a) Criação de uma falha na letra devido à resolução insuficiente.

(b) O aumento do número de níveis de cinza permite a detecção da topologia correta do caracter.

(c) O aumento do número de pontos de amostragem da matriz tem o mesmo efeito que (b) sem que seja necessário o aumento do número de níveis de cinza.

A binarização proposta inicialmente é uma seleção global de limiar baseada nos histogramas de nível de cinza. Ela é chamada de global porque é arbitrado um único limiar para toda a imagem. Caso o histograma da nossa aplicação apresentasse uma distribuição ideal bimodal como mostrada na figura II.6, bastaria encontrar o vale entre os dois picos e definir ali o limiar, para que a transformação da classe

1 para a classe 2 tivesse sucesso. Repare o que os dois picos representam: o pico maior representa o fundo do desenho, predominantemente branco, sendo que sua área representa cerca de 80% da área total do desenho; o pico menor representa a frente do desenho correspondendo a 20% da área total. Por isso é que representamos o que está desenhado originalmente em preto pelo pixel aceso e o fundo pelo pixel apagado, para reduzir o tempo de processamento de apresentação. Mas os efeitos citados anteriormente fazem com que a distribuição real se apresente conforme a figura (II.7).

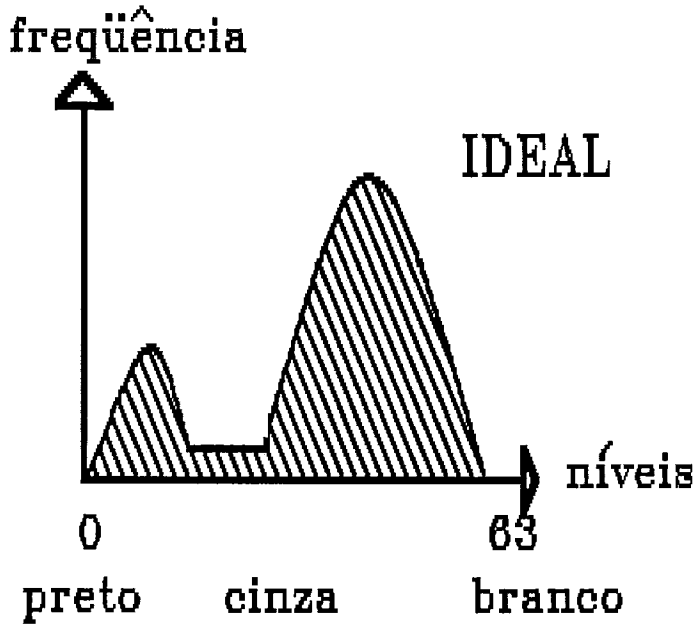

Figura II.6 - Histograma com a Distribuição Ideal Bimodal dos Níveis de Cinza.

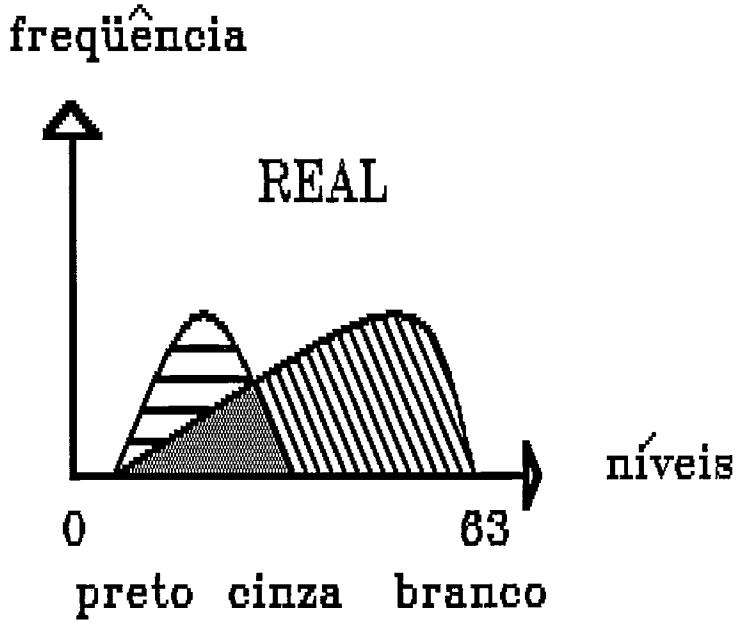

Figura II.7 - Histograma com a Distribuição Real dos Níveis de Cinza.

Pela figura (II.8) podemos visualizar o problema onde é mostrada uma figura antes e depois de digitalizada. Podemos notar pela heterogeneidade da grossura das linhas e curvas da figura original, a tendência de que depois de digitalizada, apesar de todas as linhas terem sido desenhadas com uma única cor, elas apresentem níveis de cinza diferentes entre si. Além disso existe o efeito "mach band" onde uma única linha digitalizada apresenta na sua parte central um nível de cinza mais escuro e conforme se desloca do centro para as bordas vai apresentando tons de cinza mais claros até chegar ao fundo branco do desenho. Com tudo o que foi exposto, chegamos a conclusão que utilizar um único limiar para a binarização de uma imagem não é satisfatório.

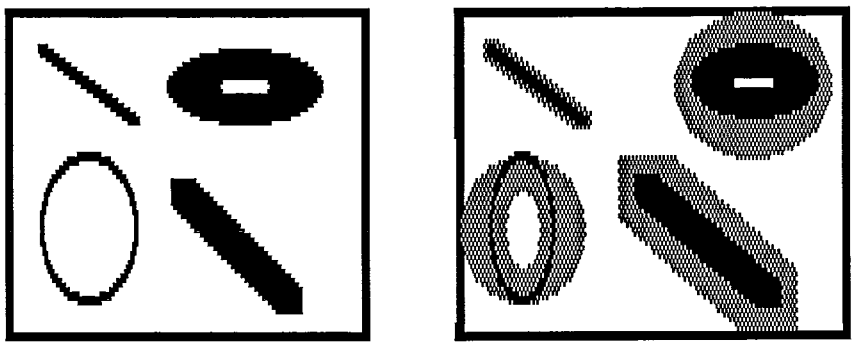

Figura II.8 - Efeito "Mach-Band".

Devemos partir, portanto, para uma binarização local onde são definidas janelas dentro das quais devemos extrair alguma propriedade relevante que sirva de critério de decisão para determinar se o pixel testado ficará aceso ou permanecerá apagado. Estas janelas são definidas, normalmente, como quadrados, por simplicidade e simetria, sendo seu tamanho ideal definido através de testes sobre a imagem, dependendo, portanto, da aplicação. Esta janela não deve ser pequena demais para levar em consideração o efeito dos pixels vizinhos sobre o pixel central e evitar uma sensibilidade grande ao ruído, nem deve ser grande demais para não tendermos a voltar a ter um critério único aplicado a toda a imagem. Além de quadradas estas janelas tem, geralmente, um número ímpar de elementos no lado, por questões de equidistância do elemento central a seus vizinhos como pode ser visto na figura II.9. O valor mais comum destas janelas é de 3x3, 4x4 ou 5x5, sendo que o valor

4x4 por ser uma potência de 2 é mais eficiente computacionalmente falando. Nesta análise desprezamos a diferença de distância entre os vizinhos da diagonal e os vizinhos horizontais e verticais, mas muitos autores costumam atribuir pesos diferenciados para um e outro, sendo o peso menor dado aos vizinhos diagonais.

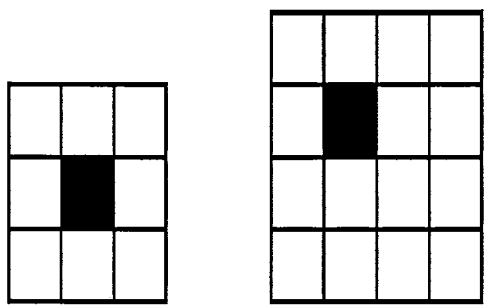

Figura II.9 - janela 3x3 janela 4x4

Além de definir o tamanho ideal da janela, devemos definir o critério para aplicar a binarização. Podemos usar a média dos níveis de cinza dentro da janela, o desvio padrão, a média ponderada onde são dados pesos diferentes de acordo com a distância do pixel central aos vizinhos, ou então tentar perceber alguma característica da aplicação que possa ser utilizada como propriedade para a binarização. É claro que podíamos pensar em tamanho de janelas diferentes e critérios diferentes a serem aplicados para uma mesma imagem heterogênea (ex: texto misturado com fotografias). Este seria o caso da binarização dinâmica. Para nossa

aplicação, como existe uma homogeneidade dos desenhos, podemos usar a binarização local. Após testes, escolhemos a janela 5x5 e a média dos níveis de cinza como critério. Os artigos [27], [29] e [30] apresentam várias técnicas de binarização.

Neste item não foram mencionadas transformações dentro da própria classe 1, pois a necessidade das mesmas vai depender do hardware escolhido e dos requisitos da aplicação. O que podemos adiantar aqui é que se existirem ruídos na imagem que não sejam eliminados pelo próprio hardware devemos aplicar filtros que estão, geralmente, no domínio da transformada, mas que podem também ser do domínio do espaço. Os problemas relativos à não homogeneidade da iluminação podem ser contornados via utilização de filtros homomórficos [9,11].

II.5.4 - Afinamento

Após o pré-processamento obtemos uma imagem representada por uma matriz binária. Devemos notar que esta imagem, se comparada a imagem orginal, apresenta linhas mais grossas. Isto se deve ao efeito "mach band" e ao fato de estarmos escolhendo uma matriz de amostragem com resolução suficiente para evitar perdas em relação à imagem original. Por isso devemos aplicar um processo de afinamento para retornar a imagem para a grossura original unitária. Isto é possível, já que nossa aplicação não está sujeita à restrições no que diz respeito à grossura das linhas, o que

não é verdade, por exemplo para desenhos de arquitetura. Cabe, então, a aplicação de um algoritmo de afinamento para simplificar a imagem binária, facilitando a posterior trilha de contorno do esqueleto e servindo, também, para compactar a imagem original.

É importante chamar a atenção sobre as diferenças entre afinamento, encolhimento e esqueletonização, através de suas definições [11,33].

PRATT [11] apresenta as seguintes definições: Afinamento, intuitivamente, é o processo que transforma um conjunto plano que possui um interior não vazio, em um conjunto que coincide com sua fronteira e, portanto, tem um interior vazio. Sendo que os conjuntos associados a este último tipo são chamados de linhas ou curvas de largura unitária. O processo de esqueletonização significa achar uma aproximação para o eixo mediano, de um conjunto plano. O encolhimento representa a redução de um conjunto para o seu menor equivalente topológico. Já PAVLIDIS [33] apresenta as definições da seguinte forma: o encolhimento e o afinamento são operações irreversíveis que visam a redução de uma região conectada de pixels que estão de acordo com um conjunto de propriedades, para um tamanho menor. O encolhimento reduz a região para um pixel apenas, enquanto que o afinamento reduz a região para uma mínima largura do corte transversal. Esqueletonização é uma operação relacionada às outras que transforma a região em uma representação do seu esqueleto.

As definições de PAVLIDIS estão mais corretas, pois PRATT afirma que as operações são irreversíveis, mas PAVLIDIS mostra um algoritmo para reconstrução da figura original, além do que a definição de PRATT para encolhimento não leva em conta uma figura vasada, isto é, que tem um furo. O importante é entendermos porque o processo por nós adotado chama-se afinamento e não encolhimento. Isto se deve ao fato de querermos reduzir uma figura grossa para outra fina mantendo as propriedades originais, sem com isso degenerar a imagem, o que não é verdade pela definição de encolhimento, e sim, pela definição de afinamento.

Para descrevermos o processo de afinamento é necessária a definição de propriedades, pois é fundamental uma base teórica para garantirmos que ao final do algoritmo obteremos um conjunto que apresenta as mesmas características do conjunto original, só que com uma largura mínima (grossura unitária).

Para a nossa aplicação é fundamental que o processo de afinamento mantenha a conectividade, isto é, se as figuras originais são compostas por linhas conectadas, os esqueletos produzidos tem que continuar conectados.

O processo básico de afinamento é o de percorrer os pixels da borda ou contorno da figura, identificando os que são e os que não são deletáveis. Depois de percorrido o contorno da figura, os pixels marcados como deletáveis são "descascados", deixando apenas aqueles considerados múltiplos ou

esqueletais. Ao final do descascamento sucessivo dos contornos resta um conjunto em que todos os pixels são múltiplos, e este é identificado como sendo o esqueleto da figura.

A seguir são apresentadas definições e propridades importantes para o processo de afinamento e extraídas de [1,18, 31,33,34].

Nesta fase, lidamos com uma matriz retangular binária com M linhas e N colunas. Os elementos da matriz com valores 0 são chamados de elementos-0 (pixels apagados) e os com valores 1 são chamados de elementos-1 (pixels acesos). Façamos X0, X1, X2,...,X7 representarem oito elementos na vizinhança do elemento p como na figura (II.10). É assumido que a primeira linha, a M-ésima linha, a primeira coluna e a N-ésima coluna estão preenchidas com elementos-0. Estas linhas e colunas são chamadas de moldura da figura.

x3	x2	x1
x4	p	x0
x5	x6	x7

Figura II.10 - Elemento **p** e a notação usada para definir a localização relativa de seus vizinhos.

1) Vizinhança - A vizinhança direta do elemento p corresponde ao conjunto dos pixels Xi com i par. A vizinhança indireta corresponde ao conjunto de pixels Xi com i ímpar. A vizinhança do elemento p corresponde ao conjunto dos pixels Xi com 0 <= i <= 7. Dois pixels são ditos vizinhos diretos (vizinhos-d) se as respectivas células compartilharem um lado, e vizinhos indiretos (vizinhos-i) se estas células se tocarem somente em um canto. Vizinhança-4 é o mesmo que vizinhança-d e vizinhança-8 é o mesmo que vizinhança.

2) Conectividade - Dois elementos a1 e a2 com um valor comum são chamados de diretamente conectados ou 4-conectados (8-conectados) se a seqüência de elementos y0(=a1),y1,..., yn(=a2) existir, tal que cada yi está na vizinhança-4 (vizinhança-8) de y i-1 (1 <= i <= n) e todo yi tiver o mesmo valor que a1 e a2.

3) Componente Conectado - Os elementos conectados que obedecem a propriedade de conectividade-4 (conectividade-8) são chamados de componentes conectados-4 (componentes conectados-8). Um componente conectado composto por elementos-0 é chamado de componente-0 e o composto por elementos-1 é chamado de componente-1.

4) Furo - Qualquer componente conectado composto por elementos-0 que não contem a moldura de uma figura é chamado de furo. Um componente conectado composto por elementos-1 sem furo é dito ser simplesmente conectado ou caminho sim-

ples, e com furo é chamado de multiplamente conectado ou caminho fechado.

Para evitar ambigüidades envolvendo a conectividade e contradições topológicas trataremos os elementos-1 pela conectividade-8 e os elementos-0 pela conectividade-4. Assumiremos para a imagem que a frente do desenho, ou seja, o que efetivamente está desenhado será representado por elementos-1 (pixel aceso) e o fundo do desenho será representado por elementos-0 (pixel apagado). Por exemplo, na figura (II.11), consideramos os pixels A e B hachurados como acesos, pertencentes à frente do desenho e efetivamente conectados pela definição de conectividade-8. Já os pixels C e D não hachurados são apagados, pertencentes ao fundo do desenho e não estão conectados pela definição de conectividade-4.

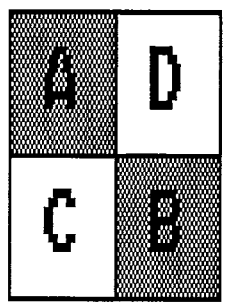

Figura II.11 Definições para evitar contradições na interpretação da conectividade entre elementos-1 e elementos-0.

5) Raiz - A raiz de um componente conectado composto por elementos-1 é definida como "um menos (o número de furos dentro daquele componente)". A raiz de uma figura é definida como " (o número de de componentes-um) menos o número de furos". Esta definição é importante pois nos dá a informação sobre a topologia. Isto quer dizer que, antes e depois da aplicação do afinamento, a raiz tem que permanecer inalterada.

6) O elemento-1 p é chamado de elemento interior se todos os elementos na vizinhança-4 de p forem elementos-1, e é chamado de elemento isolado se todos os elementos da vizinhança-8 de p são elementos-0.

7) Função Cor - C(p)

C(p) = 1, 2, 3 ou 4 => p é um pixel do objeto.

0 => p é um pixel de fundo ou um pixel eliminado do objeto.

obs: a princípio p é binário, mas conforme avançamos no processo de afinamento e processos posteriores, rotulamos os pixels com mais características e eles passam a ser representados por mais bits, e não mais por apenas um. Daí termos definido para C(p) valores maiores do que 1.

8) Função Vizinhança - V(p)

V(p) = Somatório (i=0 a 7) de delta Xi

onde delta Xi=1 para C(Xi) # 0 e C(X i+1) = 0

senão delta Xi = 0

sendo X8=X0

V(p) = 0 => elemento isolado

V(p) = 2 => elemento de passagem ou de conexão

V(p) # 0 e V(p) # 2 => elementos característicos

V(p) = 1 => elemento terminal

V(p) = 3 => elemento de bifurcação

V(p) = 4 => elemento de cruzamento

A função vizinhança fornece a característica do elemento em relação aos seus vizinhos, mostrando se o elemento é isolado ou pertence ao contorno. É útil, também, para trilhar os caminhos, pois, saberemos quantos vizinhos devem ser trilhados.

9) Segmento de figura - É o caminho p0 p1 p2 ... pn para n>=1

onde V(p0), V(pn) = 1, 3 ou 4 e

V(pi) = 2 para i=1, 2, 3,..., n-1,

ou seja, é o caminho onde os elementos inicial e terminal são característicos e os elementos restantes, se existirem, são de passagem.

10) Caminho - É a seqüência de pixels conectados (de p0 a pn).

caminho aberto => p0 # pn

caminho fechado => p0 = pn

11) Largura do caminho - É o número de pixels existentes em um corte transversal de um caminho. Também é chamado de grossura de um caminho, sendo que obter a largura mínima unitária de um caminho é o objetivo ideal para o processo de afinamento. Essa largura mínima corresponde a um único pixel.

12) Comprimento do caminho - É o número de pixels existentes ao longo de um caminho. Quando o caminho tem largura unitária, esta definição não gera problemas. Caso a largura não seja unitária, devemos pensar em termos de um corte longitudinal.

13) Direção de trilha - D (pXi)
D (pXi) = i => Direção do pixel corrente p para os vizinhos Xi, com i = 0,1,2,...,7

14) Pixel ou elemento de contorno - É o pixel que possui ao menos um vizinho direto nulo.

15) Pixel esqueletal - São os pixels que sobram ao final do afinamento, formando o esqueleto da figura. Podem ou não ser iguais ao conjunto de pixels múltiplos dependendo do grau de refinamento do algoritmo de afinamento.

16) Pixel múltiplo - São os pixels considerados indispensáveis para a preservação da topologia e da conectividade da figura, não podendo, portanto, ser eliminados da figura durante o afinamento.

17) Deletabilidade - É a propriedade pela qual se julga se um pixel pode ou não ser deletado da figura. Esta propriedade é menos ou mais refinada dependendo do algoritmo de afinamento. Geralmente a preservação dos pixel múltiplos é uma condição suficiente, mas não necessária para garantir que a topologia e a conectividade fiquem preservadas. Pois, podem existir casos particulares em que só sobram pixels múltiplos e com a eliminação de mais pixels específicos, ainda assim tanto a topologia quanto a conectividade ficam preservadas.

18) Pixel ou elemento característico - É o pixel que não é isolado, interior ou de passagem, ou seja, é um pixel terminal, de bifurcação ou de cruzamento. Os exemplos são apresentados na figura (II.12).

19) Curvatura - É a tendência que um segmento de figura tem de se aproximar mais a uma curva (arco) ou a uma reta.

No processo de afinamento, nos baseamos nas definições de multiplicidade estabelecidas por PAVLIDIS em [1] e [33], onde para distingüir os pixels múltiplos dentro do conjunto dos pixels de contorno são usadas operações locais com janelas 3x3. Um pixel é chamado de múltiplo se ele atender pelo menos um dos seguintes padrões de vizinhança apresentados na figura (II.13).

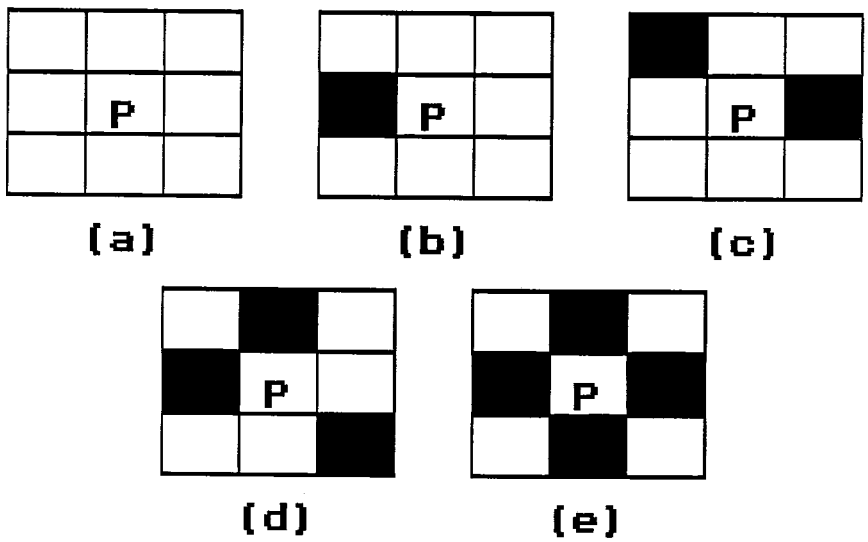

Figura II.12 - Cinco tipos de pixel p : (a) ponto isolado V(p)=0; (b) ponto terminal V(p)=1; (c) ponto de passagem V(p)=2; (d) ponto de bifurcação V(p)=3 e (e) ponto de cruzamento V(p)=4.

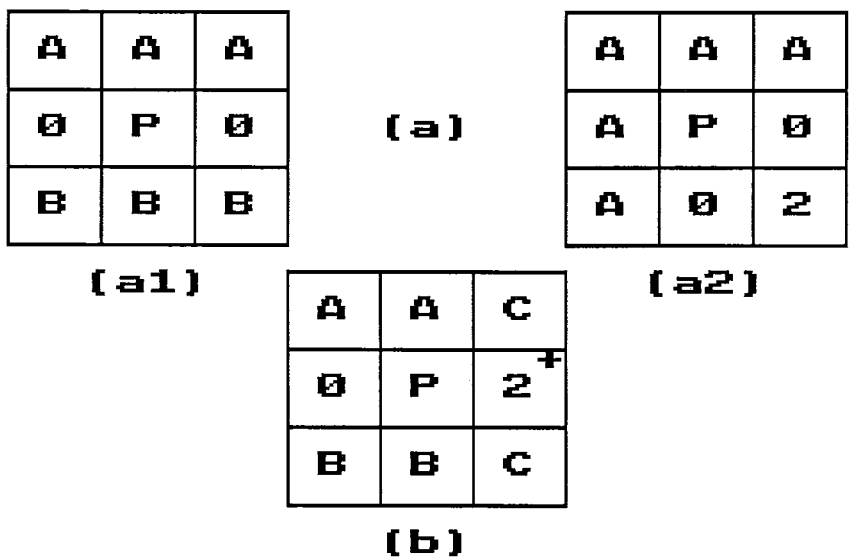

Figura II.13 - Padrões de vizinhança para identificação de pixels múltiplos.

Nos padrões (a) no mínimo um elemento de cada grupo marcado por A ou B precisam ser não nulos. O primeiro padrão de (a) rotacionado uma vez de 90 graus e o segundo padrão rotacionado três vezes de 90 graus também devem ser levados em consideração. Isto corresponde a um total de seis padrões embutidos na definição de (a). Os elementos marcados por 0 significam elementos de fundo ou nulos, os elementos marcados por 1 significam elementos interiores e os elementos marcados por 2 ou mais são elementos do contorno. O sinal (+) identifica que o valor deve ser maior ou igual ao do número junto do sinal. No padrão (b), no mínimo um dos pixels marcados por C devem ser não nulos. Se ambos os pixels rotulados por C são não nulos, então o valor dos pixels rotulados por A e B podem assumir qualquer valor. Caso contrário, no mínimo um dos membros de cada par marcado por A ou B precisam ser não nulos. O padrão (b) e os três padrões produzidos por três rotações sucessivas de 90 graus devem ser levados em consideração.

Existe ainda mais uma condição que se satisfeita identifica um pixel múltiplo independente das condições anteriores. É uma condição estabelecida quando se deseja preservar pixels isolados ou pixels que são terminais. Consiste em se considerar múltiplos os pixels que tiverem no máximo um vizinho não nulo ou não tiverem nenhum vizinho rotulado por um.

É importante ressaltar que nos padrões apresentados os números representam o número de vezes que um algoritmo de

traçado de contorno passou por eles. Um pixel marcado por 0 representa um pixel apagado ou de fundo, isto é, não faz parte do objeto. O pixel marcado por 1, depois que o algoritmo de traçado de contorno tiver atuado, representa pixel interior não pertencente ao contorno. O pixel marcado por 2 ou mais significa pixel de contorno.

Em última análise, o processo de afinamento consiste em se encontrar os pixel do contorno da figura e proceder o teste para identificar os pixels múltiplos e os deletáveis. Os pixels múltiplos são marcados para que não sejam retirados posteriormente, e os pixels deletáveis são descascados da figura. O processo se repete até que só restem pixels múltiplos. Ao final devemos obter uma figura de largura unitária, onde a topologia e a conectividade foram preservadas e só existe um caminho a ser percorrido posteriormente pelo algoritmo de trilha de caminho.

II.5.5 - Trilha do Caminho

A trilha do caminho consiste em percorrermos a matriz binária afinada em busca de pixels característicos, isto é, pixels terminais, de bifurcação ou de cruzamento. Os pontos isolados não serão considerados. Como saída deste procedimento teremos a descrição segundo o código da cadeia (função direção) das figuras que compõem esta imagem. Isto é, ao mesmo tempo em que buscamos os pixels característicos, já vamos codificando a figura. Além do tradicional código da cadeia associado a cada pixel, deverão ser in-

corporadas informações, sobre a característica do pixel que correspondem à função vizinhança (V(P)).

É importante chamar a atenção sobre o caso particular dos caminhos fechados ou ciclos, pois eles não possuem nenhum ponto característico tradicional. Eles só possuem pontos de passagem, precisando, portanto, de um tratamento especial. Outro fator que é bom atentar é o de como percorrer cada caminho apenas uma vez e como não deixar de percorrer nenhum caminho. Devemos, para tanto, marcar os pixels a medida em que eles vão sendo trilhados e colocar em uma pilha pixels de bifurcação e cruzamento, retirando-os quando encontrarmos pixels terminais ou, pixels de cruzamento ou bifurcação em que todos seus vizinhos acesos já tenham sido trilhados. Os caminhos que restarem a ser trilhados, se houver algum, serão caminhos fechados, facilmente trilháveis.

II.5.6 - Segmentação

O processo de segmentação consiste em percorrermos o código da cadeia acrescido das informações sobre a característica de cada pixel, sendo que devemos fazer uma aproximação [1,16,18,36,37,38,39,40] para os pixels presentes entre dois pixels característicos. Esta aproximação pode ficar mais próxima a uma reta, a uma curva ou então a um conjunto só de retas, só de curvas, ou de retas e curvas. O processo consiste em pegar segmentos de figura e aproximá-los a curvas e/ou retas, de forma tal a compactar

ainda mais a figura original e direcionar para uma interface gráfica. Uma vez identificado que um segmento de figura se aproxima a uma reta, basta guardar as coordenadas (x,y) absolutas do pixel inicial e do pixel final ou as coordenadas relativas, ou ainda guardar as coordenadas do pixel inicial, o ângulo e o comprimento para podermos reconstruir a reta. Para um arco bastariam guardar três coordenadas : pixel inicial, pixel final e pixel sobre o arco, coordenadas estas absolutas ou relativas. A representação vai depender de como o editor gráfico formatou suas primitivas gráficas, isto é, quais são os parâmetros necessários para definir uma reta, um arco, uma linha poligonal, etc.

No caso em que um segmento de figura não possa ser aproximado simplesmente por uma única reta ou curva, devemos proceder a identificação de pixels críticos ou dominantes entre os pixels característicos, passando a aproximar segmentos menores. Portanto, o segmento de figura pode ser aproximado a uma ou mais curvas e/ou retas. Neste ponto podemos fazer uma anologia com o processo de afinamento que tem que preservar a topologia e a conectividade. O processo de segmentação tem que preservar a forma ("shape") da imagem e evitar a interseção. A aproximação deve ser feita de modo a não alterar as características de forma da figura original e evitar que aproximação de segmentos de figura localizados a uma pequena distância um do outro acabem por se interceptar, quando na figura original não estavam interceptados, e vice-versa, segmentos inter-

ceptados passem a ficar não interceptados.

II.5.7 - Interface com o Editor Gráfico

A medida em que fazemos a segmentação, podemos ir armazenando em arquivo em disco a descrição da figura. O primeiro segmento de figura encontrado se for aproximado a uma reta poderá ser "traduzido" como um movimento absoluto do cursor seguido de um traçado relativo de uma reta,

MOVE_ABSOLUTO X0,Y0

RETA_RELATIVO DELTA_X1, DELTA_Y1

ou então, um movimento absoluto apenas:

RETA_ABSOLUTO X0,Y0 X1,Y1

Apesar de parecer mais atraente a segunda opção, é preferível utilizar movimentos relativos, pois, estes têm , normalmente, uma execução mais rápida e no cômputo geral acabam por ser mais compactos se comparados aos movimentos absolutos.

As primitivas gráficas são geralmente:

MOVE_RELATIVO DELTA_X1, DELTA_Y1

MOVE_ABSOLUTO X1,Y1

RETA_RELATIVO DELTA_X1,DELTA_Y1

RETA_ABSOLUTO X1,Y1 X2,Y2

ARCO_RELATIVO DELTA_X1,DELTA_Y1 DELTA_X2,DELTA_Y2

ARCO_ABSOLUTO X1,Y1 X2,Y2 X3,Y3

POLIGONO_RELATIVO DELTA_X1,DELTA_Y1 ... DELTA_XN,DELTA_YN

POLIGONO_ABSOLUTO X1,Y1 ... XN,YN

PONTO_RELATIVO

PONTO_ABSOLUTO X1,Y1

Podem existir ainda primitivas mais específicas como, CÍRCULO, QUADRADO, TRIÂNGULO, etc...

Existem atualmente padrões gráficos como o GKS, o PHIGS, o CORE voltados para a computação gráfica com formatos de arquivo e primitivas gráficas próprias. Mais recentemente surgiram padronizações para arquivos de editoração eletrônica (TIFF, GEM, WINDOWS) e programas pintores do tipo "PAINTBRUSH", com formato de arquivo com extensão .PCX, ou "AUTOCAD" com extensão .DWG. A CCITT tem formalizado a codificação de imagens para padronização de transmissão via FAX, de imagens de dois ou vários níveis [35, capítulo 5, página 211]. Para nossa aplicação o importante é achar um editor gráfico que aceite a entrada via arquivo externo, não criado dentro dele, mas sintetizado "artificialmente". Uma vez encontrado, devemos nos informar quais são as primitivas de desenho disponíveis e o formato do arquivo. Daí basta gerar um arquivo com formato e primitivas compatíveis para podermos usar todo o poder deste editor nas modificações do diagrama descrito.

CAPÍTULO III

III - TÉCNICAS, EQUIPAMENTOS E IMPLEMENTAÇÃO

Neste capítulo descrevemos os materiais e métodos utilizados, ou seja, os equipamentos utilizados, as ferramentas de software, o método experimental e a implementação propriamente dita.

III.1 - IMPLEMENTAÇÃO

Na figura (II.3) do capítulo anterior, foi feito um diagrama em blocos da solução proposta. Agora apresentamos na figura (III.1) um novo diagrama em blocos numerado, cuja nomenclatura é mais específica e está mais próxima da implementação.

A seguir cada um dos blocos é analisado de uma forma suscinta, mostrando: a entrada, a saída , o processamento e os problemas encontrados.

Primeiramente devemos observar a aplicação, percebendo características que possam simplificar o processamento e outras que são indispensáveis para que não haja uma degeneração do diagrama original. As características principais da nossa aplicação são: a) desenho monocromático preto em fundo branco com dimensões próximas do formato A0; b) desenho composto por caracteres, linhas e símbolos com tra-

çado fino e predominantemente homogêneo; c) cerca de 20% do diagrama pertencem ao desenho, sendo que os outros 80% correspondem ao fundo.

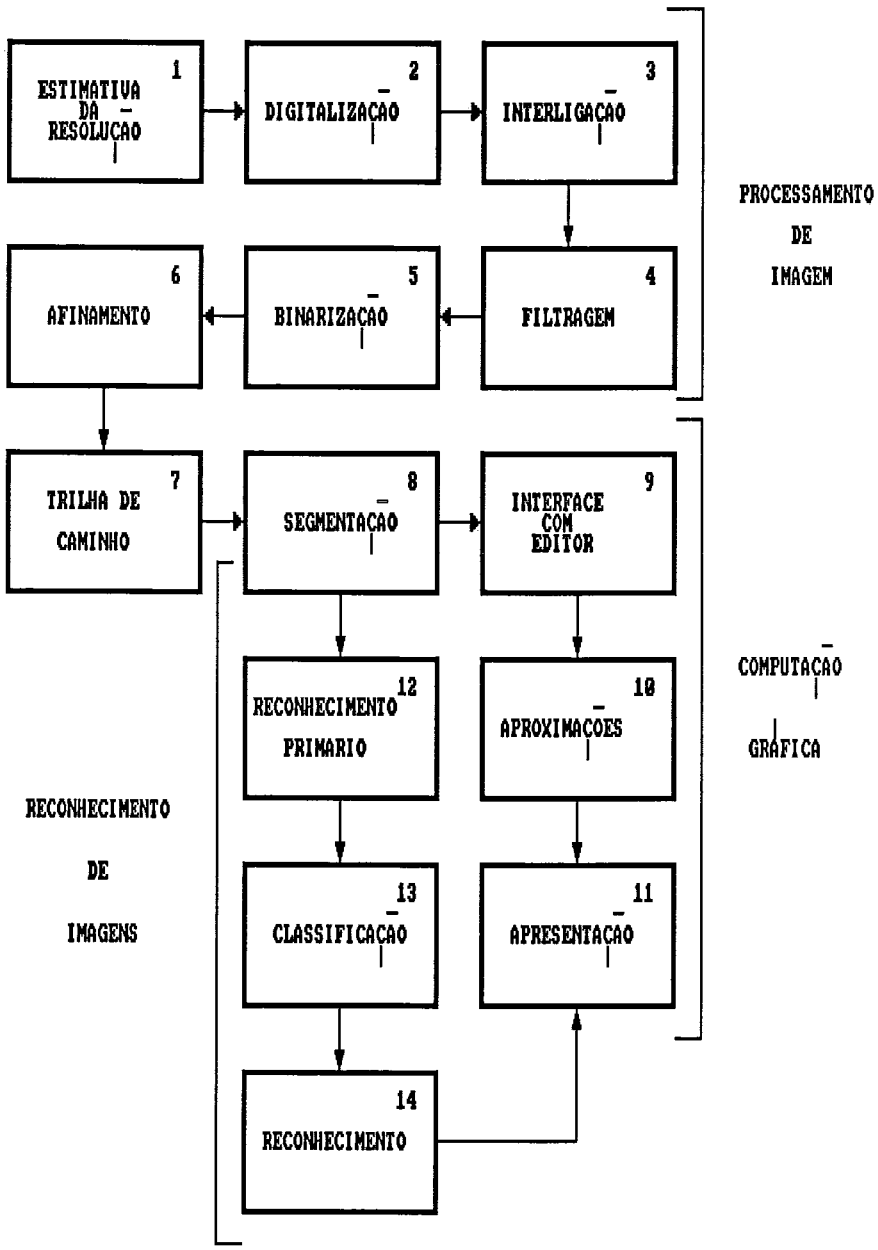

Figura III.1 - Diagrama em Blocos da Implementação.

1) ESTIMATIVA DA RESOLUÇÃO -

ENTRADA: dimensões do diagrama, dimensões do menor elemento do diagrama, características da imagem (simplificações e restrições).

SAÍDA: número de pixels na horizontal x número de pixels na vertical (resolução do dispositivo).

PROCEDIMENTO: identificar o menor elemento do diagrama (ex:caracter), associar a este elemento uma matriz capaz de representá-lo, calcular o número de elementos que cabem dentro da área total do desenho, multiplicar o número de elementos que cabem na vertical e na horizontal pelo número de pixels da matriz de representação obtendo a resolução estimada em pixels, multiplicar a resolução pelo número de bits correspondentes ao número de níveis de cinza obtendo o tamanho do arquivo da imagem.

2) DIGITALIZAÇÃO -

ENTRADA: imagem contínua.

SAÍDA: parte da imagem discreta, matriz com vários níveis de cinza.

PROCESSAMENTO: amostragem da imagem no espaço e na amplitude (quantização dos níveis de cinza).

EQUIPAMENTO: scanner, digitalizador por câmera ou outro dispositivo apropriado.

PROBLEMAS: escolher dispositivo adequado à aplicação, resolução do dispositivo insuficiente, iluminação não homogênea ou insuficiente, tamanho da imagem maior do que a capacidade do dispositivo.

3) INTERLIGAÇÃO -

ENTRADA: várias partes da imagem com superposição entre vizinhos e informação sobre a localização de cada pedaço na imagem original.

SAÍDA: imagem representada por uma única matriz com vários níveis de cinza.

PROCESSAMENTO: superposição entre os pedaços e equalização de histograma nas fronteiras entre vizinhos para não deixar as junções visíveis.

PROBLEMAS: distorções na imagem digitalizada, deslocamento ou defasagem devido à descalibração do dispositivo de digitalização.

SUGESTÃO: dispositivo de tamanho igual ou maior do que a imagem para que esta fase não seja necessária, ou interligação a partir de dispositivo ótico e mecânico de precisão.

4) FILTRAGEM -

ENTRADA: matriz com vários níveis de cinza e ruído.

SAÍDA: matriz com vários níveis de cinza.

PROCESSAMENTO: filtros específicos para eliminação de ruídos do tipo sal e pimenta, filtro homomórfico e subtração pela imagem de uma folha em branco para compensar iluminação não homogênea.

PROBLEMAS: identificar o(s) tipo(s) de ruído (gaussiano, sal e pimenta, iluminação não homogênea, etc...).

5) BINARIZAÇÃO -

ENTRADA: matriz com vários níveis de cinza.

SAÍDA: matriz com apenas dois níveis.

PROCESSAMENTO: conversão no espaço de vários níveis para dois níveis, utilizar binarização local com janela (ex: 4x4) e escolher propriedade para testar se o pixel ficará aceso ou apagado (ex: média dos níveis de cinza).

PROBLEMAS: efeito "mach band" e contraste simultâneo, digitalização com resolução insuficiente, histograma com indefinição quanto à separação das áreas brancas e pretas.

6) AFINAMENTO -

ENTRADA: matriz com dois níveis e linhas grossas.

SAÍDA: matriz com dois níveis e linhas finas com largura unitária (esqueleto).

PROCESSAMENTO: traçar o contorno da figura, identificando entre os pixels de borda os que são deletáveis e os que são múltiplos, descascar os pixels deletáveis, mantendo os múltiplos, repetir o procedimento até que só reste o esqueleto da figura.

PROBLEMAS: manter a topologia e a conectividade da imagem, garantir que só haverá um caminho a ser percorrido na trilha posterior do esqueleto.

7) TRILHA DE CAMINHO -

ENTRADA: matriz com esqueleto.

SAÍDA: representação da imagem pelo código da cadeia de FREEMAN e identificação dos pixels característicos.

PROCESSAMENTO: varrer a imagem, identificando os pixels característicos (terminal, de bifurcação ou de cruzamento), armazenar em um vetor as coordenadas absolutas (x,y) do pixel inicial e a direção relativa dos pixels interligados pelo código da cadeia, também devemos guardar a característica do pixel dado pela função vizinhança.
PROBLEMAS: existência de figuras que não têm pixels característicos (ciclos), garantir que um caminho só será trilhado uma vez.
JUSTIFICATIVA: 20% é desenho e 80% é fundo; representação pelo código da cadeia é compacta e facilita a trilha posterior.

8) SEGMENTAÇÃO -
ENTRADA: vetor com esqueleto codificado pelo código da cadeia e informação sobre os pixels característicos.
SAÍDA: retas e arcos.
PROCESSAMENTO: analisar a tendência entre pixels característicos e aproximar este segmento de figura a uma reta ou um arco.
PROBLEMAS: existência de linhas poligonais fechadas sem pixels característicos (ciclos), existência de segmentos de figura com pontos críticos além dos característicos que fazem com que um único segmento seja aproximado a um conjunto de retas e/ou arcos, definição de critério para análise da curvatura.

9) INTERFACE C/ EDITOR -

ENTRADA: vetor com retas e arcos.

SAÍDA: arquivo com primitivas gráficas.

PROCESSAMENTO: converter as retas e arcos aproximados para primitivas gráficas com posicionamento absoluto ou relativo. As primitivas podem ser por exemplo: MOVE X0,Y0 RETA X0,Y0,X1,Y1 ARCO X0,Y0,X1,Y1,X2,Y2 LINHA X0,Y0,X1,Y1,...,Xn,Yn (ABERTA) PONTO X0,Y0 LINHA X0,Y0,X1,Y1,...,X0,Y0 (FECHADA) MOVE_RELATIVO DELTAX0,DELTAY0.

PROBLEMAS: editores sem entrada para arquivo gerado externamente ou com formatação de arquivos desconhecida ou complicada.

10) APROXIMAÇÕES -

ENTRADA: retas, arcos, linhas poligonais.

SAÍDA: linhas poligonais suavizadas.

PROCESSAMENTO: splines e curvas de Bezier aplicadas para suavizar mudanças bruscas na direção das linhas (dentes de serra).

11) APRESENTAÇÃO -

ENTRADA: descrição do diagrama.

SAÍDA: imagem do diagrama na tela, na impressora ou noutro dispositivo gráfico.

PROCESSAMENTO: a partir da descrição do diagrama em primitivas gráficas é feita sua reconstrução na tela do vídeo ou da impressora com a ajuda de um editor gráfico que utiliza conceitos de computação gráfica (janelas, zoom, rota-

ção, escala, etc).

12) RECONHECIMENTO PRIMÁRIO -

ENTRADA: retas, arcos, linhas.

SAÍDA: grupos de retas, arcos e linhas.

PROCESSAMENTO: a partir de critérios pré-estabelecidos, retas, arcos e linhas são agrupados de forma a se aproximarem às classes definidas.

PROBLEMAS: estabelecer critérios para unir as primitivas em grupos.

SUGESTÃO: representar as retas, arcos e linhas e a relação entre elas por uma estrutura sintática.

13) CLASSIFICAÇÃO -

ENTRADA: grupos de retas, arcos e linhas.

SAÍDA: classes de símbolos, caracteres e linhas.

PROCESSAMENTO: a partir de critérios pré-estabelecidos (ex: comprimento de um segmento de figura) é feita a separação de símbolos, caracteres e linhas.

PROBLEMAS: definir propriedades e critérios baseados na aplicação e específicos para simplificar e facilitar a classificação.

SUGESTÃO: análise sintática.

14) RECONHECIMENTO -

ENTRADA: classes de símbolos e caracteres.

SAÍDA: símbolos e caracteres reconhecidos.

PROCESSAMENTO: são feitos algoritmos específicos para reconhecer símbolos e caracteres separadamente, os símbolos

e caracteres válidos são definidos a priori para facilitar o reconhecimento, o código dos caracteres (ex:ASCII) e dos símbolos são mais compactos e podem ser realimentados para apresentação.
PROBLEMAS: reconhecimento de caracteres de fontes diferentes, confusão entre símbolos e caracteres, caracteres escritos a mão e unidos dificultam reconhecimento, restrições quanto a direção dos caracteres para evitar confusão (ex: letras M e W), restrições quanto à proporção do tamanho entre caracteres e símbolos.

A seguir são detalhados os algoritmos e técnicas correspondentes aos blocos de 1,2,4,5,6,7,8 e 9 (figura III.1), que foram efetivamente implementados.

III.1.1 - ESTIMATIVA DA RESOLUÇÃO

Antes de escolher o hardware mais adequado para a nossa aplicação, é necessário fazer uma estimativa da resolução para o dispositivo de digitalização. Isto é importante para evitar perdas de informação já no primeiro estágio do processamento da imagem, devido a uma amostrageminsuficiente. Ao fazermos a estimativa da resolução, podemos também calcular o tamanho, expresso em número de bytes, do arquivo para armazenamento da imagem. Isto nos permitirá dimensionar a capacidade do dispositivo de armazenamento de arquivos em disco.

Para que possamos fazer uma estimativa da resolução é necessário conhecermos a aplicação. No nosso caso, estamos digitalizando mapas do setor elétrico que são compostos por desenhos e textos. Devemos estabelecer o menor elemento representativo a ser reconhecido sem perdas. Foi escolhido o menor caracter alfanumérico encontrado. Este caracter tem dimensões de 1,5 mm (vertical) por 1,0 mm (horizontal) (figura III.2), e o desenho tem dimensões de 540 mm (vertical) por 780 mm (horizontal).

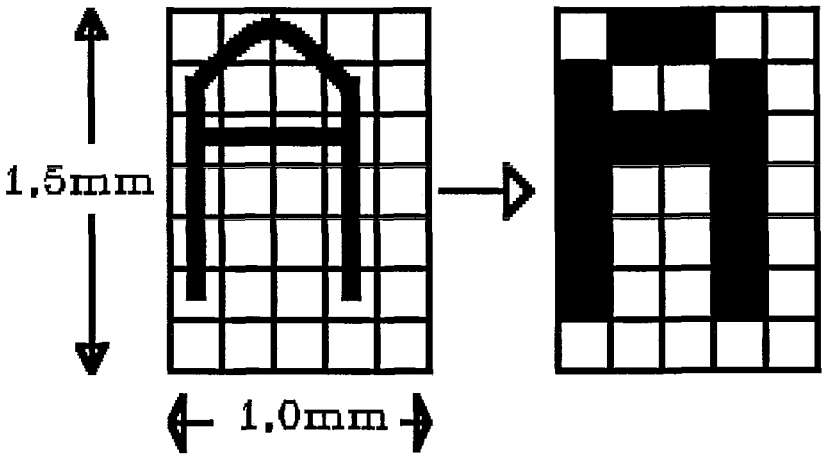

Figura III.2 - Dimensões do menor elemento representativo do diagrama.

Dividindo o tamanho total pelo tamanho de cada elemento, teremos o número máximo de elementos que cabem dentro de uma folha.

Nv = Vt / Ve = 540 / 1,5 ~ 366
Nh = Ht / He = 780 / 1,0 ~ 780 } 366 x 780 elementos

Onde,

Nv = número de elementos na vertical

Nh = número de elementos na horizontal

Vt = altura total do desenho (vertical)

Ht = largura total do desenho (horizontal)

Ve = altura do elemento (vertical)

He = largura do elemento (horizontal)

Sabendo que cada elemento é um caracter, podemos representá-lo de uma forma ideal através de uma matriz 5x7 (horizontal x vertical), como se faz habitualmente em impressoras e monitores. Multiplicando estes fatores pelos elementos na horizontal e na vertical, obtemos:

Pv = Nv * 7 = 2562

Ph = Nh * 5 = 3900

Pt = Pv * Ph = 2562 * 3900 = 9.911.800 pontos

Onde,

Pv = número de pontos na vertical

Ph = número de pontos na horizontal

Pt = número total de pontos

Isto é, quase dez milhões de pontos (bits) ou mais de um milhão de bytes (caracteres). Daí existir aquele velho ditado que diz: "Uma imagem vale mais do que mil palavras" (ou mais ...).

Habitualmente, só conseguimos uma resolução com esta ordem de grandeza por meio de digitalizadores rastreadores

("scanners" [4]). Eles percorrem o papel iluminando linha por linha, coluna por coluna, sendo, portanto, lentos. Existem "scanners" comerciais com a resolução de impressoras laser, ou seja, 300 dpi ("dots per inch" - pontos por polegada). Fazendo algumas contas, temos que:

1 polegada = 2,54 cm = 300 pontos

. .
. . 2,54 mm = 30 pontos

. .
. . 1,0 mm ~ 12 pontos

Logo,

300 dpi ~ 12 pontos / mm

Isto significa, assumindo que as resoluções horizontal e vertical são iguais, uma matriz de 12x12 pontos / mm2, ou uma matriz de 12x18 para englobar o menor elemento (1,0 x 1,5 mm2) de nossa aplicação. Esta matriz é maior do que a matriz 5x7 definida a priori. Esta maior resolução, também vai significar mais pontos a serem armazenados, isto é, mais memória de massa livre será necessária. O problema destes rastreadores, além da sua lentidão, está nas suas dimensões, pois geralmente, só estão disponíveis comercialmente os que digitalizam folhas de tamanho A4 e A3. Isto quer dizer que para figuras maiores teremos que particionar o desenho e recompô-lo após a digitalização. Outro problema está associado à quantidade de informação. Na nossa previsão obtivemos como resultado que um diagrama para ser totalmente recuperado, sem perdas, deveria ser amostrado com um "scanner" de 2562x3900 pontos, ou seja, quase dez milhões de pontos. Se a saída do digitalizador

fôr direta em dois níveis (apenas preto e branco), necessitaríamos de 1 bit para representar cada pixel, ou seja, 10 M bits ou 1,25 M bytes. Observando que apenas 20% da área total da figura é ocupada por informação de desenho ou texto (os outros 80 % são informação de fundo) podemos aplicar técnicas de compactação [1,35] que reduzirão o tamanho do arquivo a ser armazenado.

Sabendo que cerca de 20% da área é útil, teremos uma redução de 10 M bits para 2 M bits ou 250 K bytes aproximadamente. Para cada ponto representado por um bit, precisamos associar mais informação, caso contrário não conseguiremos recuperar a figura original. Esta informação a ser adicionada pode ser por exemplo: coordenada (x,y) do ponto inicial da cadeia de pontos e a direção relativa ao próximo ponto. Esta nova representação auxilia no traçado do contorno da figura, além de compactar a figura original.

Associada a essa compactação poderíamos utilizar uma codificação "Huffman","Run Length Encode") que atribui pesos às direções mais freqüentes ("Huffman"), ou que se vale da repetição da direção dos pontos adjacentes para reduzir o número total de bits para uma média menor do que seria sua representação direta ("Run Length Encode").

A seguir comparamos alguns tipos de codificação para que tenhâmos uma idéia da ordem de grandeza da compactação.

a) Caso todos os pontos estejam conectados dentro de uma vizinhança

código da cadeia: 8 direções

representação :

3 bits + 4 bytes { x - 2 bytes 0 <= x <= 65535
* { y - 2 bytes 0 <= y <= 65535
* apenas para pontos no início da cadeia

n = número de pontos

média = (3*n + 32) / n ~ 3 bits

taxa de compactação = 10 M / 6 M (bits / bits) = 1,66

b) Caso existam pontos afastados da cadeia principal, isolados ou formando outras cadeias secundárias.

código da cadeia : 8 direções + 1 indicando ponto inicial (código = 8)

representação : 4 bits + 4 bytes (para todos os pontos iniciais)

ni = número de pontos iniciais

n = número de pontos não iniciais

média de bits = (4*n + 32*ni) / (n+ni)

para n >> ni => média ~ 4 bits

taxa de compactação = 10 M / 8 M (bits / bits) = 1,25

c) Associando código de Huffman ao código da cadeia.

direção	código	p	
0	0	12%	
1	01	12%	
2	011	12%	
3	0111	12%	
4	01111	12%	
5	011111	12%	
6	0111111	12%	
7	01111111	12%	
8	011111111	4%	(pontos iniciais)

média de bits = (0,12 (1+2+3+4+5+6+7+8) + 0,04*9) = 4,68 bits

taxa de compactação = 10 M / 9,36 M (bits / bits) = 1,06

Como a probabilidade de cada direção foi estipulada como igual, este método não se mostra satisfatório.

d) Código da cadeia diferencial

Este método se mostra satisfatório, caso os pontos não estejam muito afastados um dos outros em relação à direção. Ele associa um número (0,1,2,3,4,-1,-2,-3) à diferença entre a direção do ponto corrente e a direção do próximo ponto. Associaremos ainda o valor 5, para indicar o início de uma nova cadeia de pontos. Ele associa um código com menor número de bits para variações suaves do traçado de contorno e maior número de bits para variações bruscas. Segundo PAVLIDIS [1] temos em média 2 bits por pixel.

direção	código	p
0	0	40%
+1	01	20%
-1	011	20%
+2	0111	7,5%
-2	01111	7,5%
+3	011111	1,25%
-3	0111111	1,25%
+4	01111111	1,25%
5	011111111	1,25%

média de bits = (0,4 + 0,2(2+3) + 0,075(4+5) + 0,0125(6+7+8+9) = 2,45 bits

taxa de compactação : (10 M / 4,9 M) bits/bits = 2,05

O exemplo d), acima, é o mais adequado, uma vez que apresenta a melhor taxa de compactação da figura original, além de facilitar o traçado de contorno pela representação através da direção. Realmente é o melhor dentro dos exemplos aqui expostos, mas deve ser considerado apenas um passo no sentido de compactar a figura original ainda mais. A escolha de uma codificação mais elaborada com compactação mais eficiente vai depender da aplicação e dos recursos disponíveis. Se dispusermos de muito espaço de armazenamento e pouca rapidez de processamento devemos utilizar uma codificação mais simples, enquanto que se dispusermos de pouco espaço de armazenamento e rapidez no processamento devemos optar por uma codificação mais elaborada. Isto é, existe um compromisso entre codificação e processamento, pois geralmente quanto mais compacto é um código, mais processamento será necessário para a decodi-

ficação.

Até agora nesta estimativa, tratamos do caso ideal do problema sem considerar o ruído, os erros de quantização e amostragem e, consideramos a saída do digitalizador em apenas dois níveis.

No caso não ideal, a figura apresenta ruídos e manchas que tem que ser eliminadas através de filtragens no domínio do espaço ou da freqüência [1,9,11]. A imagem original, apesar de apresentar dois níveis bem marcantes, após ser digitalizada aparecerá na saída com um número maior de níveis, principalmente se dispusermos apenas de digitalizador por câmera de vídeo [41], como é o nosso caso. Isto faz com que a imagem digitalizada aparente um resultado bom para os olhos, sem os dentes de serra que normalmente encontramos nas telas dos computadores com modo gráfico de resolução insuficiente.

Na verdade o que ocorre é que a baixa resolução dos dispositivos é compensada por um número maior de níveis de cinza (ou cores). Existe uma relação de custo-benefício entre o número de pontos de um dispositivo e o número de níveis de cinza. Estipular quantos pontos na vertical e na horizontal, e quantos níveis deve ter um digitalizador (entrada) e um monitor (saída) vai depender da aplicação e das ferramentas disponíveis. Na figura (II.5) apresentamos um exemplo da digitalização de uma letra, onde a resolução do digitalizador com saída em apenas dois níveis não foi su-

ficiente para recuperar o caracter, surgindo como conseqüência falhas. Caso dispuséssemos de um digitalizador com três níveis de saída, conseguiríamos recuperar totalmente o caracter. O ideal para o nosso caso seria o de conseguir um digitalizador de maior resolução, de tal forma que a região mais estreita do caracter estivesse totalmente contida dentro de uma célula de amostragem e a cor dentro dela fosse única (figura II.5c).

O número de níveis de cinza ou cores de um digitalizador está relacionado ao processo de quantização, enquanto que a resolução, ou número de pontos na horizontal x número de pontos na vertical, está relacionada ao processo de amostragem. A amostragem é feita em duas dimensões (horizontal e vertical) e tem que ser suficiente para evitar perdas como ficou estabelecido no início deste tópico : o menor elemento é um caracter de dimensões 1,5 mm x 1,0 mm; e "a parte mais estreita tem que estar totalmente contida dentro de uma célula de amostragem e a cor dentro desta célula é única". Se a amostragem não for suficiente teremos um erro de amostragem. No caso ideal, ao definirmos uma matriz 5x7, garantimos que a parte mais estreita está dentro do que foi definido. Teremos de garantir ainda que cada ponto desta matriz 5x7 só exista um nível de cinza (ou cor) e que as cores dos pontos desta matriz 5x7 estejam acima de um limiar, de tal modo que na hora da quantização em apenas dois níveis, se acenda o ponto da matriz correspondente. Pontos amostrados que são acesos quando deveriam permanecer apagados e vice-versa constituem em erros de

quantização. O nível de quantização não é crítico para nossa aplicação que está restrita a dois níveis. O mesmo não pode ser dito de uma aplicação que possuísse muitas variações de tonalidade, como é o caso da maioria das imagens reais (flores, rosto, arco íris, etc...).

No caso real dispusemos apenas de um digitalizador por câmera com resolução de 256 (H) x 244 (V) e 64 níveis de cinza para exercitar as técnicas e algoritmos. Com isso os problemas de iluminação e resolução ficam críticos. Na prática isto significa que estaremos lidando com imagens com sombras em que, apesar dos desenhos e textos serem escuros em fundo claro, poderá haver confusão entre o fundo e a frente do desenho, caso não tomemos as providências alertadas no parágrafo anterior.

Como a resolução de nosso digitalizador é pequena, fica evidente que não poderemos digitalizar a figura do mapa de uma só vez, pois perderíamos todos os detalhes. Precisaríamos, portanto, aproximar a câmera do objeto, e até utilizar uma lente macro para garantir a resolução exigida. Com isso podemos calcular o número de partes que seriam necessárias para dividir a figura original sem perdas:

NPv = 2562 / 244 ~ 10

NPh = 3900 / 256 ~ 16

NPt = NPv * NPh = 160 pedaços

Onde,

NPv = número de partes na vertical

NPh = número de partes na horizontal

NPt = número total de partes

Portanto, teríamos que executar 160 vezes o processo de digitalização, ou seja, 160 arquivos em disco de 256 x 244 pontos (= 62464 pixels). Sabendo que cada pixel corresponde a 6 bits, para representar até 64 níveis, teremos um total de 374.784 bits ou 46848 bytes. No total geral teremos : 46848 x 160 = 7.495.680 bytes, ou seja, aproximadamente 60.000.000 de bits (59.965.440). Isto é, 6 vezes maior do que para o cálculo do "scanner", o que era esperado uma vez que estipulamos um "scanner" que só apresenta dois níveis de saída, enquanto que nosso digitalizador por câmera apresenta 64 níveis (6 bits).

Podemos levantar a seguinte questão: por que não, aumentar o número de pontos amostrados pelo digitalizador, de tal forma a diminuir o número de pedaços em que seriam divididas a figura original? Primeiramente, o nosso digitalizador não é programável como, por exemplo, o modo gráfico do IBM PC, onde podemos passar do modo 320 x 200 para 640 x 200 com a diminuição do número de cores disponíveis de 4 para 2 em prol de uma maior resolução. Em segundo lugar nossa câmera possui uma resolução de 270 x 300 (H x V) e transforma o sinal ótico em sinal elétrico, sinal de vídeo composto compatível com monitores, videocassetes e TV's. Como a resolução da câmera está muito próxima da do nosso

digitalizador, de nada adiantá aumentar o número de pontos amostrados, se a própria resolução da câmera não é suficiente.

Ressalvo aqui que o digitalizador por câmera não é o dispositivo ideal para executar o tipo de aplicação a que estamos nos propondo. O ideal seria um "scanner" de tamanho A0 com 300 dpi. Mas como só dispomos deste digitalizador, testaremos os algoritmos, filtros e o processamento de diagramas com ele, o que não será ideal, mas sim didático e satisfatório, desde que tomemos os cuidados necessários no que diz respeito à resolução e à iluminação, ou seja, o processo de digitalização.

Outro fator que pode "assustar" a princípio, é o tamanho dos arquivos digitalizados que só podem ser salvos em discos rígidos, ou em diskettes com capacidade maior do que 500k bytes. Em defesa entra o fato de já dispormos de equipamentos mais rápidos, com maior capacidade de processsamento e armazenamento. Como exemplo podemos citar o PC AT com 80386, clock de 25 Mhz e memória RAM expansível até 16 M bytes e disco rígido de até 640 M bytes. Além disso, estão surgindo novos dispositivos de armazenamento ótico, como por exemplo o CD-ROM ("Compact Disk Read Only Memory"), WORM ("Write Once Read Many") com capacidade de armazenamento da ordem de giga-bytes e tempo de acesso de milisegundos (memória "CACHE"). Os monitores estão cada vez com maior resolução: ao padrão CGA ("Color Graphics Adapter") do IBM PC (640 x 200 - 2 cores) e ao HÉRCULES

(640 x 350 - 2 cores) seguiu o EGA ("Enhanced Graphics Adapter") (640 x 200 - 16 cores), e agora o VGA ("Video Graphics Adapter") (640 x 480 - 256 cores).

A tendência é de surgirem monitores com mais pontos e tecnologias de cristal líquido colorido de alta resolução para permitir a apresentação de uma folha tamanho A0 de uma só vez na tela. É claro que existem técnicas que permitem apresentar um desenho grande através de janelas em um monitor, mas determinados desenhos precisam de uma apresentação como um todo para termos uma melhor visão do que ele representa. É claro que também existem técnicas para apresentar o "jeitão" do desenho em um monitor de menores dimensões, mas a tendência sempre existe de eliminarmos as limitações através do "hardware". As próprias memórias RAM dinâmicas estão com mais capacidade e menor tempo de acesso : chips de 1 M bits e 80 ns.

TAXA DE COMPACTAÇÃO DE UMA FIGURA

figura original digitalizada : 60.000.000 bits
(2562 x 3900 x 6 bits)
processamento: 64 níveis => 2 níveis / 6
(2562 x 3900 x 1 bit) : 10.000.000 bits
processamento: 20% do desenho / 5
(((2562 x 3900) / 5) x 1bit) : 2.000.000 bits
processamento: direção ("differencial chain code") x 2
(((2562 x 3900) / 5) x 2 bits) : 4.000.000 bits
cálculo em BYTES / 8
Tamanho do arquivo final => 500 K bytes

Podíamos continuar a compactação, mas isso implicaria em poupar memória ao custo de mais processamento. Precisa existir uma relação de custo-benefício ótima. Achamos razoável, partirmos do valor de 500 K bytes e se houver necessidade poderemos compactá-lo ainda mais (por exemplo "Run Length Encode" da direção).

Resumindo, os passos para a estimativa da resolução são:

- PROBLEMA: Escolher o hardware adequado.
- Digitalização : Qual dispositivo usar ? (ex: scanner, câmera, etc...)
- Fazer uma estimativa da resolução necessária:
- Escolher o menor elemento representativo: caracter.
- Estipular no mínimo uma matriz m x n para amostrar este elemento: matriz 7x5.
- Encontrar a relação entre o tamanho total do desenho e o tamanho do elemento: - Calcular o número de elementos que cabem nas dimensões horizontal e vertical do desenho.
- Multiplicar o número de elementos encontrados pelo número de pixels estipulado na matriz de amostragem m x n.
- Comparar com a resolução dos dispositivos de digitalização.
- Escolher o dispositivo que atende os requisitos da aplicação.

CONCLUSÃO : Devemos utilizar um SCANNER tamanho A0 para a digitalização. Pelo volume de informações prevemos um processamento intensivo para execução dos algoritmos, sendo o

ideal utilizar um computador com grande capacidade de processamento e preferencialmente com processador de ponto flutuante. Os dispositivos de armazenamento em disco devem ser rápidos e com grande capacidade. A utilização de microcomputadores de última geração e de discos óticos é sugerida.

III.1.2 - DIGITALIZAÇÃO

Como dispositivo de digitalização utilizamos uma placa digitalizadora [41-44] associada a uma câmera de vídeo. A câmera focaliza uma parte do diagrama e converte a imagem ótica para um sinal de vídeo que é amostrado pela placa digitalizadora e armazenado na memória da própria placa. A placa armazena uma imagem formatada com 244 linhas e 256 colunas, sendo que cada pixel é representado por um entre 64 níveis de cinza possíveis. São portanto 6 bits, sendo o pixel preto representado pelo nível 0, e o pixel branco representado pelo nível 63.

A placa digitalizadora é ligada a um microcomputador do tipo PC-XT através da interface serial (COM1). A imagem guardada em mémoria é então transmitida serialmente pela placa para o computador, onde é armazenada em um arquivo em disco com tamanho de 62464 bytes, uma vez que cada pixel está associado a um byte de armazenamento.

Esta placa vem acompanhada de outra chamada de placa apresentadora, que permite que vejamos a imagem digitalizada

em um monitor do tipo televisão preto e branco, com vários níveis de cinza. Existem duas formas de visualizar os resultados: ligando as placas diretamente e visualizando a imagem digitalizada "on line", ou colocando o computador como intermediário ora ligando a placa digitalizadora para receber a imagem, ora ligando a placa apresentadora para mostrar o resultado com vários níveis de cinza.

A visualização dos vários níveis de cinza foi importante para a investigação do efeito "mach-band", da iluminação não homogênea e do compromisso entre número de níveis de cinza e a resolução da imagem.

Como nossa câmera é de tecnologia VIDICOM e não CCD ("Charged Coupled Device") como as mais modernas, é fundamental uma boa iluminação. Na figura (III.3) é mostrado o esquema da digitalização, sendo que as lâmpadas são de 100 Watts.

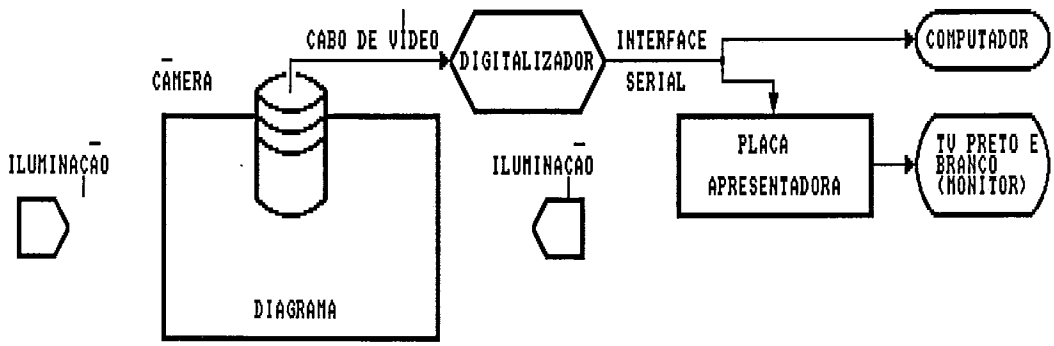

figura III.3 - Esquema da digitalização.

III.1.3 - FILTRAGEM

Para visualizar os problemas da iluminação basta digitalizar uma folha em branco. Ao examinar a imagem digitalizada e analisar seu histograma, percebemos manchas escuras na imagem e um histograma distribuído pelos níveis de cinza, ao invés de uma concentração em torno do nível 63 (figura III.4).

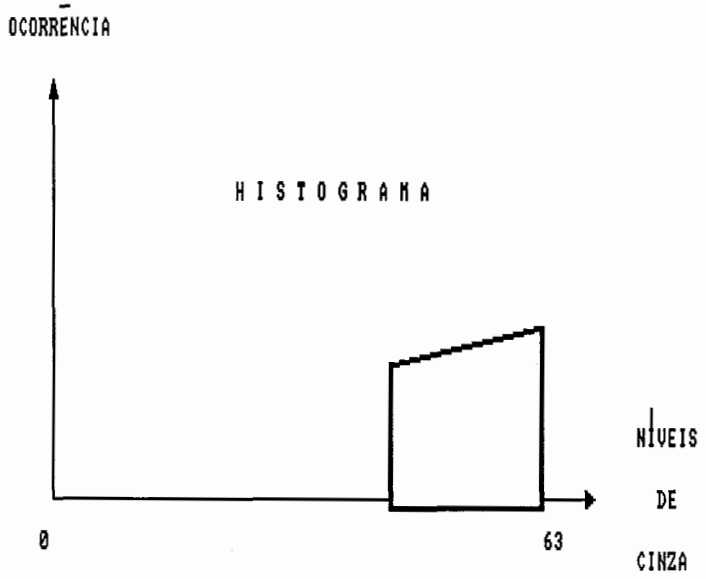

Figura III.4 - Histograma de uma folha em branco digitalizada.

Para minimizar o problema da iluminação não homogênea podemos utilizar o seguinte procedimento de filtragem:

1) Digitalizar a imagem desejada.
2) Digitalizar uma folha em branco nas mesmas condições de iluminação.

3) Inverter a imagem da folha em branco.

Isto é, o nível 63 passa a ser 0.

o nível 0 passa a ser 63.

o nível 27 passa a ser 36.

nível de cinza novo = 63 - nível de cinza antigo.

4) Somar a imagem desejada com a imagem invertida da folha em branco.

Observação: Para eliminar os efeitos da iluminação não homogênea não subtraímos direto a imagem da folha em branco porque a definição de pixel preto está associado ao nível 0 e a de pixel branco está associado ao nível 63. Na folha em branco, os níveis de ruído são aqueles que se afastam do nível 63, sendo o pior caso o da mancha negra de nível 0. Se subtraíssemos a imagem original digitalizada direto da folha em branco digitalizada, estaríamos subtraindo de 0 (mancha negra), daí a necessidade de inverter a imagem da folha em branco e somá-la à imagem original.

O efeito "mach-band" pode ser melhor visualizado colocando a placa apresentadora em um modo de resolução menor (64x61). Observamos claramente a variação dos níveis de cinza de um tom mais forte até um mais fraco até chegar ao fundo branco do desenho. Este efeito faz com que nosso olho não perceba mudanças bruscas na direção das linhas, que seriam nítidas como dentes de serra em um monitor de apenas dois níveis. Por isso, quando digitalizamos pela primeira vez um pedaço do diagrama, apesar dele apresentar caracteres de tamanho reduzido (1,0 mm x 1,5 mm), conse-

guíamos ler tudo o que estava escrito e reconhecer todos os desenhos. Só percebemos que os níveis de cinza tinham um papel importante para a melhoria da qualidade da imagem, quando, depois de aplicarmos o processo de binarização, obtivemos resultados ruins com apenas dois níveis. Isto significa dizer que para melhorar os resultados da binarização temos que aumentar o número de pontos de amostragem, isto é, temos que aproximar a câmera do papel (" zoom"), pegando uma área menor do diagrama. Com o mesmo número de pontos de amostragem, mas diminuindo a área digitalizada do diagrama, teremos como resultado o aumento da resolução. Em contrapartida, a imagem será fragmentada em um número maior ainda de pedaços.

Já foi abordado anteriormente, mas é importante enfatizar a necessidade de dispormos de um dispositivo mecânico acoplado à câmera, caso desejemos unir todos os fragmentos do diagrama.

Outro fato importante que deve ser levado em consideração é o da influência do fundo branco sobre o desenho. Na digitalização por câmera a maior área de fundo branco tende a saturar sobre o desenho, fazendo com que as linhas do desenho fiquem interrompidas. Neste caso é indicado o uso do filtro homomórfico, pois a simples subtração por uma imagem branca não é suficiente para compensar todos os efeitos da iluminação não homogênea.

O procedimento de compensação da iluminação foi implementado através de primitivas do processamento de imagens, software que acompanha a placa digitalizadora:

```
INVERT BRANCA.PIC
ADD IMAGEM.PIC BRANCA.PIC RESULTADO.PIC
```

A seguir são listados algumas destas primitivas de processamento de imagens, escritas em TURBO PASCAL:

```
ADD.PAS pic1 pic2 <pic3>
função: pic3 = pic1 + pic2.

COMPARE.PAS pic1 n <pic2>
função: pic2 = pic1 se pic1 >= n
        senão = 0.

INVERT.PAS pic1 <pic2>
função: pic2 = 63 - pic1.

MASK.PAS pic1 pic2 <pic3>
função: pic3 = pic1 se pic2 > 0
        senão = 0.

MULTIPLY.PAS pic1 n <pic2>
função: pic2 = pic1*n.

SHOW.PAS pic1
função: envia a imagem pic1 para a placa apresentadora.
```

GRAB.PAS pic1/n/c

função: armazena a imagem digitalizada expandida transmitida pela placa digitalizadora em arquivo em disco com nome pic1. A opção /c armazena a imagem comprimida e a opção /n evita que a imagem seja enviada para a placa apresentadora.

COMPRESS.PAS pic1 <pic2>

função: pic2 é a versão de pic1 codificada pela técnica de "run-length".

EXPAND.PAS pic1 <pic2>

função: pic2 é a versão não "run-length-encoded" de pic1. Os outros programas assumem os arquivos como expandidos.

EDGE.PAS pic1 <pic2>

função: pic2 contem a informação do contorno de pic1.

HISTO.PAS pic1

função: apresenta um histograma da intensidade dos pixels de pic1.

TRESH.PAS pic1 n <pic2>

função: pic2 = 1 se pic1 >= n
senão = 0.

FILTER.PAS pic1 <pic2>

função: pic2 é uma versão de pic1 após a passagem de um filtro passa-baixa.

Os arquivos incluídos importantes são:

DECLARES.P - declarações de constantes, tipos e variáveis globais.
HEXUTIL.P - rotinas para conversão hexadecimal.
SERIAL.P - código da interface serial.
PICTURES.P - rotinas para manipulação dos arquivos de imagem.
IMAGES.P - rotinas para processamento de imagens.

Não foi implementado o filtro homomórfico, mas sua ausência será compensada na binarização, como será visto logo a seguir.

III.1.4 - BINARIZAÇÃO

A binarização é o processo de conversão de 63 níveis de cinza para apenas dois níveis. Conforme já foi discutido no capítulo anterior, a binarização local é a mais adequada, uma vez que a utilização de um único limiar global não apresenta resultados satisfatórios. Como critério para determinar se o pixel ficará aceso ou permanecerá apagado, foi utilizado a média de uma vizinhança limitada por uma janela p x q. Foram feitos testes com janelas de tamanhos diferentes e acabamos optando por uma janela quadrada onde p=q=5.

O algoritmo de binarização é o seguinte: Como entrada temos uma matriz Y x X, onde o par ordenado (Y,X) corres-

ponde à posição do pixel na imagem e o conteúdo da matriz naquele ponto ao nível de cinza do pixel. Percorreremos a matriz da direita para a esquerda e de cima para baixo. A cada pixel visitado é feita a soma deste e dos vizinhos na janela p x q. O resultado final da soma é dividido pelo número de pixels da janela e é comparado com o pixel visitado, considerado como central. Se a média for menor do que o valor do pixel central, ele permanecerá apagado, e caso a média seja maior ou igual ao valor do pixel, o pixel será aceso. A lógica é a de acender o pixel com menor nível de cinza, isto é, aquele que tende a ser preto, pois este é mais provável pertencer ao desenho e não ao fundo. Acender os pixels pretos é mais eficiente do que acender os brancos, uma vez que os primeiros correspondem a apenas 20% do total de pixels do diagrama. Para otimizar o algoritmo, podemos utilizar a técnica da média restrita [22] : se o nível de cinza do pixel central for menor do que um limiar mínimo estipulado, este será considerado preto; se for maior do que um limiar máximo estipulado, este será considerado branco; caso contrário é feito o cálculo da média na vizinhança que é comparada ao valor do pixel central (figura III.5). Assim, o cálculo custoso da média só será feito nos pixels com nível de cinza intermediário.

O fato da binarização ser local e de utilizarmos como propriedade a média, compensa em parte o efeito "mach-band" e a ausência do filtro homomórfico. Ao analisarmos uma área, verificamos se ela é homogênea ou de alto contraste. Se ela é homogênea, é mais provável pertencer ao fundo do

diagrama. Se é uma área de contraste, é mais provável pertencer à frente do diagrama. Isto permite que áreas de nível de cinza diferentes sejam consideradas como pertencentes ao desenho, isto é, estamos analisando diferenças relevantes entre níveis de cinza em uma janela do diagrama, e não o nível absoluto dos pixels. Esta análise é importante para a definição do tamanho da janela de vizinhança, uma vez que uma janela muito pequena, em que coubesse integralmente uma parte do desenho, seria considerada uma área homogênea e, portanto, de fundo, a menos que o nível de cinza estivesse abaixo do limiar mínimo. Já uma janela muito grande poderia ser tendenciosa, fazendo o fundo do desenho predominar sobre a frente ou vice-versa. Daí a importância de testes para definir o melhor tamanho de janela para a nossa aplicação.

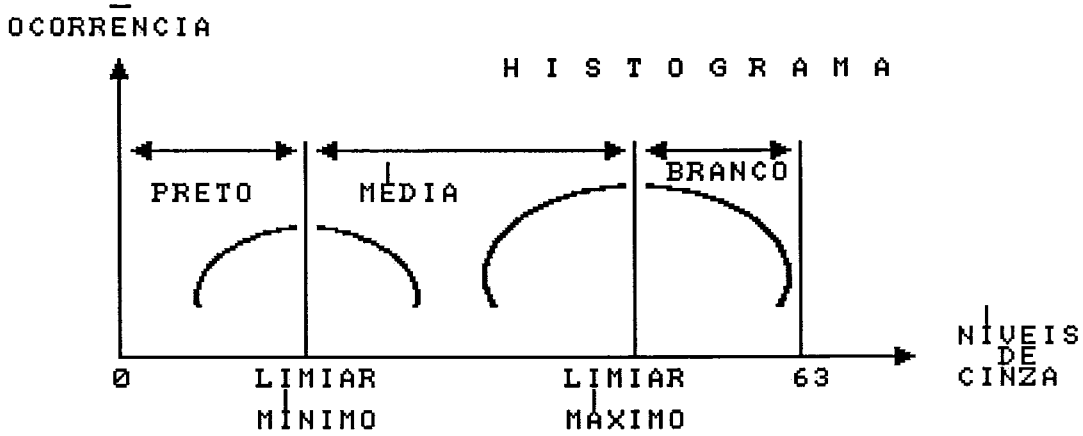

Figura III.5 - Ilustração da técnica da média restrita.

Os passos do algoritmo proposto, escritos em português estruturado são:

ROTINA BINARIZAÇÃO

ENTRADA: matriz nível(y,x), limiar_mínimo, limiar_máximo, janela p x q, delta_nível

SAÍDA: matriz binária B(y,x)

FUNÇÃO: Converte imagem de 63 níveis em outra de apenas 2 níveis.

```
PARA linha = (p+1)/2 ATÉ (m-p/2) FAÇA
  PARA coluna = (q+1)/2 ATÉ (n-q/2) FAÇA
    INÍCIO
      B(linha,coluna):=0;
      SE nível(linha,coluna) < limiar_mínimo ENTÃO
        INÍCIO
          ACENDE_PIXEL(linha,coluna);
          B(linha,coluna):=1;
        FIM
      SENÃO
        SE nível(linha,coluna) <= limiar_máximo ENTÃO
          INÍCIO
            média:=CALCULA_MÉDIA(linha,coluna,p,q);
            SE média > nível (linha,coluna) + delta_nível
            ENTÃO
              INÍCIO
                ACENDE_PIXEL(linha,coluna);
                B(linha,coluna):=1;
              FIM
          FIM
    FIM
```

algoritmo I - Rotina de Binarização.

FUNÇÃO CALCULA_MÉDIA

ENTRADA: linha, coluna, p, q

FUNÇÃO: Calcula a média dos níveis de cinza dentro de uma área p x q.

```
  média:=0;
  PARA i:=(1-p)/2 ATÉ p/2 FAÇA
    PARA j:=(1-q)/2 ATÉ q/2 FAÇA
      média:=média+nível(i+linha,j+coluna);
  média:=média/(p*q);
  RETORNA média;
FIM
```

Onde, nível(y,x) é a matriz correspondente à intensidade dos pixels do pedaço do diagrama digitalizado; limiar_mínimo é o nível limite abaixo do qual um pixel é considerado preto; limiar_máximo é o limite acima do qual um pixel é considerado branco; janela p x q é a área de processamento para o cálculo da média sendo sua medida horizontal q unidades e a sua medida vertical p unidades; delta_nível é a medida da diferença que pode ser controlada pelo algoritmo para que uma região seja considerada homogênea ou de contraste; B(y,x) é a matriz binária que guarda o resultado do processamento de binarização.

Quanto aos limites, temos que:
nível(y,x) é uma matriz com 1 <= linha <= 244,
1 <= coluna <= 256 e 0 <= nível(y,x) <= 63;
0 <= limiar_mínimo <= 63 ;
0 <= limiar_máximo <= 63;
0 <= delta_nível <= 63;
0 <= B(y,x) <=1;
e a janela pxq tem 1 <= p <= 244 e 1 <= q <=256.

É importante atentar para os limites do rastreamento das linhas e das colunas. Por simplificação, consideramos uma moldura, que para o caso da janela 4x4, é composta pela primeira linha, pela primeira coluna, pelas duas últimas linhas e pelas duas últimas colunas (figura III.6).

Uma vez feita a binarização, podemos visualizar a imagem diretamente em um monitor monocromático do PC no modo grá-

fico de 640 por 200 com auxílio do TURBO GRAPHIX TOOLBOX [46], ou então usar o LATIM [8] para visualizar a imagem.

A complexidade do algoritmo de binarização é de ordem: O (m x n x p x q) para o pior caso, e O (m x n) para o melhor caso.

No melhor caso o limiar_máximo tenderia a ser igual ao limiar_mínimo, e teríamos, então, uma binarização simples. No pior caso sempre seria calculada a média da vizinhança.

Resumindo:

- Estipular o tamanho da janela. ex: janela 4x4.
- Obter propriedades dentro desta área. ex: média dos níveis de cinza, desvio padrão, derivada, laplaceano, etc.
- Estabelecer critério para acender ou apagar ponto. ex: se p(y,x) <= média, então acende o ponto.

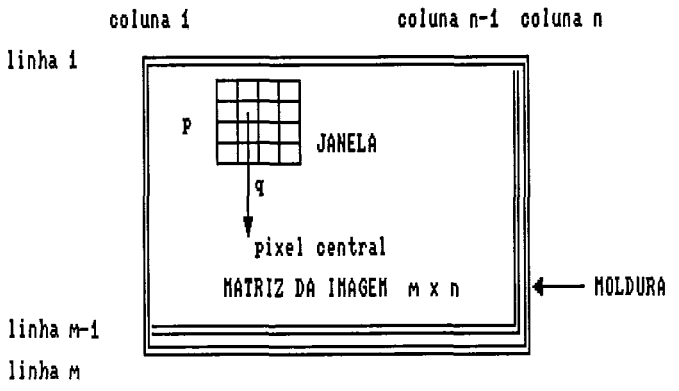

Figura III.6 - figura mostrando a matriz da imagem m x n, sua moldura e a janela de processamento p x q.

III.1.5 - AFINAMENTO

Já foi explicado no item II.5.4 do capítulo II, as razões da aplicação de algoritmos de afinamento: as linhas grossas da imagem já binarizada são convertidas para linhas finas (como as originais), por questões de compactação e simplificação da trilha posterior de caminho. Um ponto importante é o de garantir que a topologia e a conectividade da imagem permaneçam inalteradas após a aplicação do algoritmo, e que só haja um caminho a ser percorrido na posterior trilha de caminho.

Os capítulos VII e IX de PAVLIDIS [1] apresentam a teoria relativa ao afinamento, formalizando e definindo todos os conceitos necessários. Os aspectos e definições principais foram ressaltados no capítulo II deste trabalho. Agora apresentaremos o desenvolvimento do algoritmo e sua descrição em português estruturado.

O algoritmo de afinamento é aplicado sobre uma matriz binária, isto é, só contem elementos com valor 0 ou 1. Esta matriz foi resultante da aplicação do algoritmo de binarização. Devemos percorrê-la da esquerda para a direita e de cima para baixo. Todo ponto aceso ("1") deve ser analisado para identificar aqueles que são de contorno. Pela definição 14 do item II.5.4, temos que um pixel é de contorno quando possui pelo menos um vizinho direto nulo, sendo considerado o contorno externo ou interno (furo). Uma vez encontrado um pixel de contorno, devemos proceder a análi-

se de sua multiplicidade (figura II.13, definição 16), isto é identificar se ele é múltiplo ou não. Se ele for múltiplo, é feita uma marca no pixel para evitar que ele seja eliminado posteriormente. Caso o pixel não seja múltiplo ele é eliminado, uma vez que não é importante para a conectividade. O algoritmo é repetido até que só restem pixels múltiplos na imagem binária. Portanto, é feito um descascamento dos pixels em excesso no contorno, só restando ao final o esqueleto da figura.

Os pontos importantes do algoritmo são:

a) A rotulação do pixel como múltiplo é feita para evitar que em uma passagem posterior do algoritmo este mesmo pixel seja eliminado.

b) A visita aos pixels de contorno não deve ser sequencial, mas sim paralela, para evitar o encolhimento da imagem original. Portanto, devemos percorrer inicialmente todos os pixels de contorno que tenham o vizinho direto X0 (figura II.10) nulo, marcar aqueles que são múltiplos, e só ao final da identificação do último pixel de contorno nestas condições é que procedemos a eliminação dos pixels já percorridos não múltiplos. Devemos repetir o procedimento mais três vezes para os pixels de contorno que tenham os vizinhos X2, X4 e X6 nulos. Ao percorrermos o contorno de uma forma mista entre sequencial e paralela, evitamos a degradação da imagem.

ROTINA AFINAMENTO

ENTRADA: matriz B(y,x).

SAÍDA: matriz B(y,x) afinada.

FUNÇÃO: Converte uma imagem de dois níveis com linhas grossas para outra com linhas finas, mantendo a conectividade e a topologia.

```
fica:=VERDADEIRO;
ENQUANTO fica FAÇA
  INÍCIO
    fica:=FALSO;
    PARA vizinho:=0,2,4 e 6 FAÇA
      INÍCIO
        PARA linha:=2 ATÉ n-1 FAÇA
          PARA coluna:=2 ATÉ m-1 FAÇA
            INÍCIO
              SE B(linha,coluna)=1 ENTÃO
                INÍCIO
                  SE CONTORNO(linha,coluna,vizinho) ENTÃO
                    INÍCIO
                      SE ESQUELETO(linha,coluna) ENTÃO
                        INÍCIO
                          B(linha,coluna):=2;
                          ACENDE(linha,coluna);
                        FIM
                      SENÃO
                        INÍCIO
                          B(linha,coluna):=3;
                          fica:=VERDADEIRO;
                        FIM
                    FIM
                FIM
            FIM
            PARA j:=1 ATÉ n FAÇA
              PARA i:=1 ATÉ m FAÇA
                SE B(i,j)=3 ENTÃO
                  B(i,j):=0;
      FIM
  FIM
```

Algoritmo II - Rotina de Afinamento Clássico de PAVLIDIS.

FUNÇÃO CONTORNO

ENTRADA: linha, coluna, vizinho

FUNÇÃO: Identifica se o pixel pertence ou não ao contorno.

```
SE vizinho de B(linha,coluna)=0 ENTÃO
  RETORNA VERDADEIRO
SENÃO
  RETORNA FALSO;
```

FUNÇÃO ESQUELETO

ENTRADA: linha,coluna

FUNÇÃO: Comparar a vizinhança do pixel com os padrões de multiplicidade (figura II.13a) retornando VERDADEIRO ou FALSO.

```
PARA k:=0 ATÉ 7 FAÇA
  X[k]:=vizinho Xk de B(linha,coluna);

skeleto:=FALSO; cont:=0;
ENQUANTO (skeleto=FALSO) E (cont<=5) FAÇA
  INÍCIO
    CASO cont
      INÍCIO
        0: SE (X[4] OU X[0])=0 ENTÃO
             SE  ( (X[1] OU X[2] OU X[3] <> 0) E
                   (X[5] OU X[6] OU X[7] <> 0) ) ENTÃO
               skeleto:=VERDADEIRO;
        1: SE (X[2] OU X[6])=0 ENTÃO
             SE  ( (X[3] OU X[4] OU X[5] <> 0) E
                   (X[7] OU X[0] OU X[1] <> 0) ) ENTÃO
               skeleto:=VERDADEIRO;
        2: SE  (X[0] OU X[2])=0 ) E (X[1] <> 0)  ENTÃO
          { SE (X[3] OU X[4] OU X[5] OU X[6] OU X[7]) <> 0
               ENTÃO } skeleto:=VERDADEIRO;
        3: SE  (X[2] OU X[4])=0 ) E (X[3] <> 0)  ENTÃO
          { SE (X[5] OU X[6] OU X[7] OU X[0] OU X[1]) <> 0
               ENTÃO } skeleto:=VERDADEIRO;
        4: SE  (X[4] OU X[6])=0 ) E (X[5] <> 0)  ENTÃO
          { SE (X[7] OU X[0] OU X[1] OU X[2] OU X[3]) <> 0
               ENTÃO } skeleto:=VERDADEIRO;
        5: SE  (X[6] OU X[0])=0 ) E (X[7] <> 0)  ENTÃO
          { SE (X[1] OU X[2] OU X[3] OU X[4] OU X[5]) <> 0
               ENTÃO } skeleto:=VERDADEIRO;
      FIM
     cont:=cont+1;
  FIM
  RETORNA skeleto;
```

A função ESQUELETO foi concebida para testar se um pixel é esqueletal ou não, ou seja, se deve ou não ser preservado.

A variável cont serve para controlar cada critério a ser testado, sendo que caso o teste resulte em verdadeiro, não será feito mais nenhum teste.

O caso 0 serve para comparar a vizinhança 3x3 do pixel analisado com o padrão A1 de multiplicidade apresentado na figura (II.13a).

O caso 1 é idêntico ao 0, a menos do padrão, que é o A1 da figura (II.13) rotacionado de 90 graus.

O caso 2 compara a vizinhança 3x3 do pixel analisado com o padrão A2 da figura (II.13).

Os casos 3, 4 e 5 correspondem à comparação do padrão A2 da figura (II.13) rotacionado de 90, 180 e 270 graus, respectivamente.

O padrão da figura (II.13b) não foi utilizado, pois optamos pela implementação do algoritmo de afinamento mais simples (afinamento clássico). No algoritmo de afinamento com trilha de contorno é que é necessário o teste deste padrão e das rotacões de 90, 180 e 270 graus.

A parte do programa relativo à função ESQUELETO que aparece entre colchetes ({}) serve para chamar a atenção para o trecho responsável pela preservação de pixels terminais. Se retirarmos o trecho entre colchetes, estaremos preservando os pontos terminais.

A função contorno identifica se um pixel pertence ou não ao contorno, a partir da análise dos seus vizinhos diretos nulos.

Existem vários algoritmos para afinamento [1,33,34], sendo que os principais são o clássico (descrito acima no algoritmo II), o assíncrono e o baseado na trilha de contorno. Optamos pelo algoritmo clássico, uma vez que o algoritmo assíncrono pressupõe uma arquitetura paralela para sua implementação e o algoritmo de trilha de contorno necessita de processamento posterior (edição) para eliminação de pixels preservados em exagero.

Nos testes realizados com o algoritmo clássico transformado por PAVLIDIS [1, capítulo 9] observamos resultados satisfatórios. Os testes foram feitos com o auxílio do LATIM [8]. Este programa permite que editemos todos os padrões da figura (II.13a), tendo a facilidade de fazer as rotações de 90 graus, podendo salvar cada um dos padrões gerados em arquivos em disco.

O procedimento de teste (depuração) do algoritmo foi, portanto, o de sintetizar todas as vizinhanças determinadas pela definição dos padrões da figura (II.13a). Feito isso, aplicamos o algoritmo a cada um dos padrões e salvamos o resultado em arquivos em disco. Depois, subtraímos o padrão antes e depois do afinamento, tomando o cuidado para que os pixels acesos antes e depois do afinamento tenham níveis diferentes (ex: aceso antes=0FFH, aceso depois=02H) e configurando o LATIM para que os pixels que sobraram do afinamento fiquem com um nível de cinza ou cor diferente na tela. Com isso pudemos observar que nem todos os padrões de multiplicidade são preservados (figura III.8), por um determinismo existente no algoritmo. Observamos também que mais pixels são preservados do que era esperado.

A seguir são apresentados os resultados do afinamento dos padrões (figuras III.7 a III.12). O quadrado cheio indica pixel preservado e o "x" indica pixel eliminado. Foram editados e afinados todos os padrões possíveis para uma matriz 3x3, conforme a figura (II.13a), colocando, por exemplo o grupo "AAA" em "000" e gerando todos os "BBB" possíveis de "000" a "111", depois colocamos o grupo "AAA" em "001" e geramos todos os "BBB" possíveis e assim por diante. O importante é que o pixel central de um padrão múltiplo seja preservado.

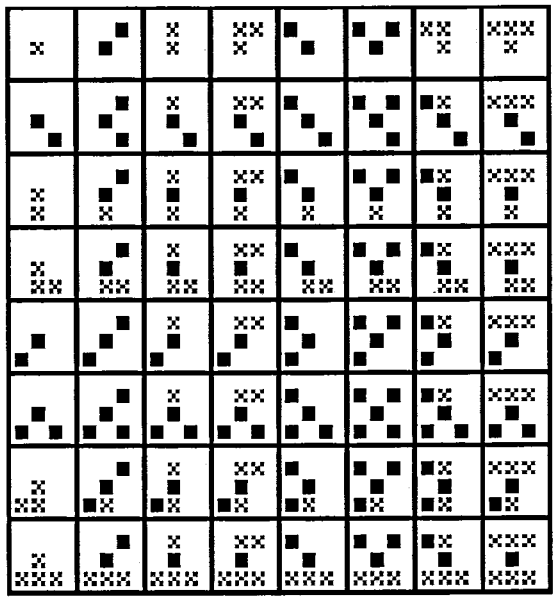

Figura III.7 - padrão A1 da figura II.13.

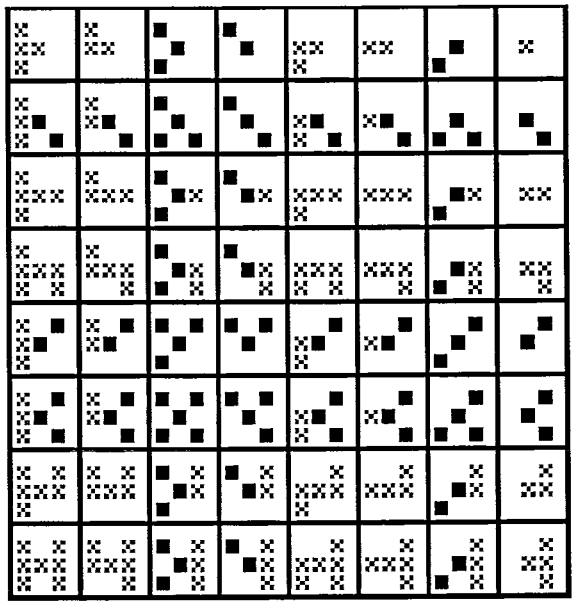

Figura III.8 - padrão A1 da figura II.13 com rotação de 90º.

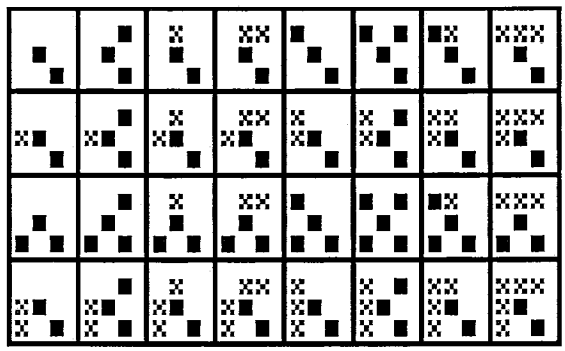

Figura III.9 - padrão A2 da figura II.13.

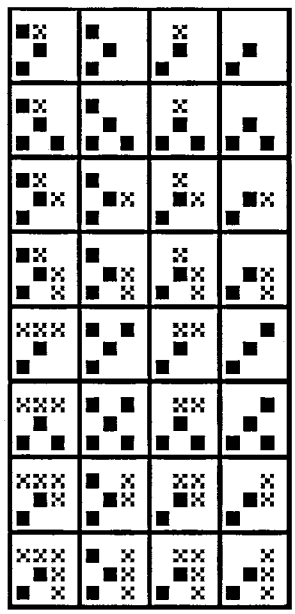

Figura III.10 - padrão A2 da figura II.13 com rotação de 90º.

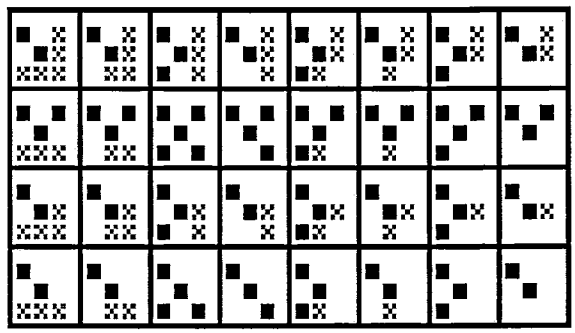

Figura III.11 - padrão A2 da figura II.13 com rotação de 180º.

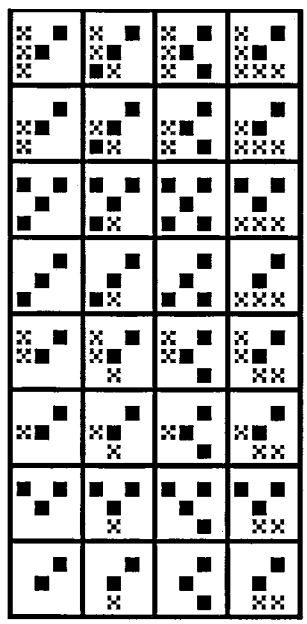

Figura III.12 - padrão A2 da figura II.13 com rotação de 270º.

A partir da observação dos resultados e da análise do algoritmo descobrimos que o determinismo ocorre em função da ordem escolhida para procura dos pixels de contorno. No início procuramos todos os pixels de contorno que não possuam vizinho direto X0, sendo que no final são eliminados todos aqueles que não são considerados múltiplos. Isto faz com que em passos posteriores do algoritmo, pixels originalmente múltiplos não sejam assim considerados, pois a eliminação de pixels em passos anteriores acaba por transformá-los em pixels deletáveis. O número de padrões não respeitados é de 16 em 72 possíveis (22%), mas o fundamental é ver se a topologia e a conectividade são preservadas apesar do que foi observado. Sintetizamos uma série de figuras nas quais os padrões que não foram preservados são colocados em pontos críticos de linhas: no meio e nos extremos (figura III.13). Observamos a partir dos resultados que a conectividade era mantida. Chegamos à conclusão que a conectividade é sempre mantida apesar de tudo. O máximo que ocorre é o descascamento por vezes exagerado dos extremos da figura. Esta é uma limitação do algoritmo, o que faz com que aumentem os requisitos da digitalização, para que obtenhamos figuras com mais pixels e mais longas para compensar a "fome" do algoritmo. A seguir são apresentados resultados de testes de conectividade e topologia (figura III.14). Observe como o mínimo desenho possível de um quadrado formado por quatro pixels é preservado, ou seja, o algoritmo preserva a topologia.

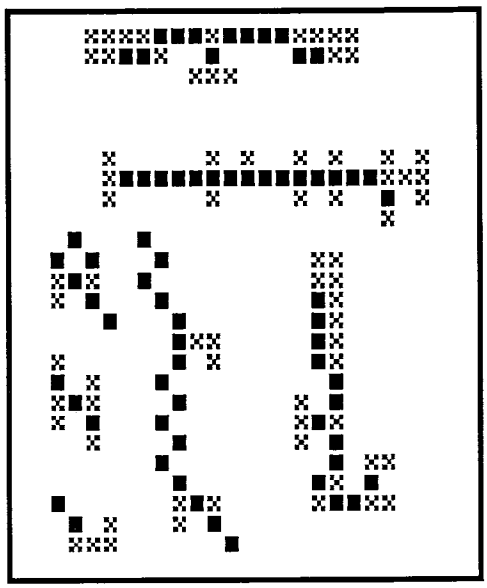

Figura III.13 - Resultado dos testes (1) para prova das propriedades de conectividade e topologia.

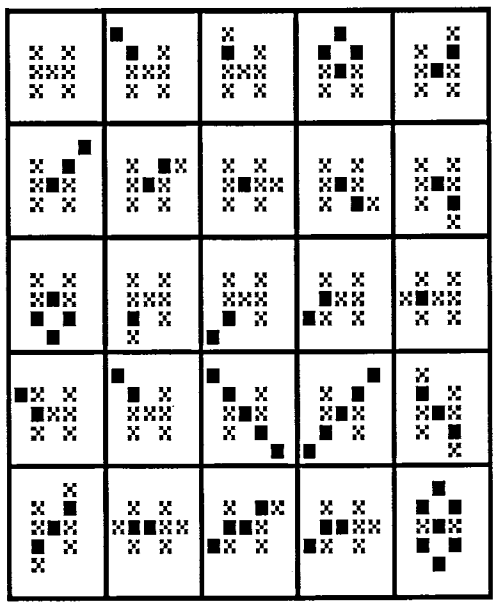

Figura III.14 - Resultado dos testes (2) para prova das propriedades de conectividade e topologia.

III.1.6 - TRILHA DE CAMINHO

Uma vez que já afinamos a imagem através da aplicação do algoritmo II, podemos partir para a trilha de caminho. A preservação apenas dos pixels múltiplos e o algoritmo de afinamento garantem que ao final só restarão pixels esqueletais. Estes pixels são mantidos porque são indispensáveis para garantir a conectividade e a topologia da imagem. Sabendo, então, que só restam pixels indispensáveis, é de supor que a simplificação das vizinhanças dos pixels (eliminação dos pixels não múltiplos) favoreça a implementação de um algoritmo de trilha de caminho que garanta que só existirá um caminho a ser trilhado. Com um algoritmo de trilha que estabeleça critérios de avanço bem definidos, baseados na vizinhança, e que marque caminhos já visitados é possível garantir que só percorreremos cada caminho uma única vez.

No curso da trilha do segmento da figura, cada ponto característico pode ser identificado pelo valor de sua cor (definição 7 do item II.5.4) ou computando a função vizinhança (definição 8 item II.5.4) do pixel corrente.

No início todos os pixels da figura afinada estão em "1" e o fundo em "0", sendo que a matriz binária B(y,x) é formada por bytes, apesar de só precisar até aqui de um bit para a representação da imagem binária afinada. Conforme o algoritmo de trilha vai sendo executado, os pixels recém visitados vão sendo marcados como já percorridos para evi-

tar a trilha repetida de um segmento de figura. Além da marcação de já percorridos, os pixels vão sendo cadastrados pela sua característica (terminal, bifurcação ou cruzamento) ou pela ausência de característica (pixel de passagem). No caso dos pixels de passagem, podíamos ter optado pela sua eliminação ao invés da marcação como já percorrido, mas para fins de depuração do algoritmo optamos pela marcação.

Se faz um rastreamento sobre a matriz B(y,x) da direita para a esquerda e de cima para baixo, até encontrarmos o primeiro ponto característico. Para tanto buscamos o primeiro ponto ligado (em "1") e calculamos sua função vizinhança : se o ponto for isolado (V=0) ou de passagem (V=2) continuamos a busca até encontrar um ponto com V=1, 3 ou 4, ou seja , um ponto terminal, de bifurcação ou cruzamento. Uma vez encontrado, entramos na rotina TRACER. Esta rotina percorre os pixels acesos, através de um algoritmo de traçado de contorno tradicional [1, algoritmo 7.1], até que um novo pixel característico seja encontrado. A rotina TRACER foi feita a partir do algoritmo de trilha de segmento descrito em [18].

Na rotina TRACER, cadastramos o pixel inicial de entrada como já percorrido, e com a informação sobre sua característica. Além disso já preenchemos um byte do vetor de saída C(z) com este ponto marcado como inicial (Direção=8) e mais dois bytes com a posição (y,x), isto é, a linha e a coluna de localização deste pixel. Acendemos

este pixel no monitor e o colocamos em uma PILHA. Então enquanto houver elemento na pilha, procederemos a trilha do caminho desta figura conectada, isto é, todo pixel que estiver conectado ao inicial e aos seus vizinhos será trilhado e assim por diante. Para encontrar outras figuras que não estejam ligadas a essa devemos continuar a execução do algoritmo III em busca de outro pixel característico que ainda não tenha sido trilhado. Ao final, caso existam, só restarão os "loops" ou ciclos não trilhados, isto é, figuras compostas apenas por pixels de passagem. Neste caso usamos um algoritmo de traçado de contorno igual ao TRACER só que não há necessidade de usar a pilha. Chamamos esta rotina de TRACICLO.

Todo pixel característico de bifurcação e cruzamento recém trilhado é colocado na pilha. Os critérios para retirada de pixels da pilha são os seguintes: a) ao trilharmos um pixel característico terminal; b) ao visitarmos um pixel característico de bifurcação ou cruzamento já trilhado; c) ao trilharmos um pixel característico de bifurcação ou cruzamento que já tenha todos os seus vizinhos trilhados.

Portanto, a partir do pixel inicial, procuramos seu vizinho segundo o critério definido pela rotina TRAÇADO. Testamos se o mesmo é característico : se for encerramos, pois já obtivemos o segmento de figura; caso contrário ele só pode ser pixel de passagem, e só existirá um único caminho a seguir. Seguimos na direção definida pelo pixel e seu único vizinho ainda não visitado e repetimos o proce-

dimento até encontrarmos outro pixel característico.

Ao partir de um pixel característico com mais de um vizinho (3 ou 4), os pixels de passagem vão sendo marcados como já percorridos na imagem binária para evitar a repetição de caminhos já trilhados. Ao final do algoritmo de trilha, o novo pixel característico é cadastrado juntamente com seus vizinhos, caso ainda não tenha sido visitado, para que os possíveis segmentos de figura pendentes sejam trilhados posteriormente.

Se for encontrado um pixel terminal, devemos retornar ao pixel característico anterior para retomar o processo da trilha. Se não houver caminho de trilha, voltamos para o pixel característico anterior, e trilhamos novos caminhos até encontrarmos um pixel terminal. O processo de ida e volta prossegue até retornarmos ao pixel inicial.

Devemos atentar aos "loops", isto é , a volta a um pixel característico já visitado. Devemos perceber que um pixel já foi visitado e não recadastrá-lo, além de tratá-lo como se fosse um pixel terminal.

Na volta ao pixel inicial não devemos encerrar o procedimento imediatamente, devemos nos certificar de que não há mais trilhas partindo do pixel inicial e seguí-las caso seja verdade, ou encerrar em caso contrário.

Depois disso, ainda devemos continuar rastreando para ver

se ainda existe um pixel característico não visitado, pois existem muitos segmentos de figura isolados.

Para o pixel inicial e cada vez que um pixel é retirado da pilha, na saída do vetor C(z) é preenchido um byte com código de ponto inicial (Direção=8) e mais dois bytes para a linha e coluna correspondentes. A cada pixel de passagem está associado o código de FREEMAN com a direção (D=0,1,...,7) que é colocada no vetor de saída C(z). Ao final teremos um arquivo composto por inúmeros segmentos com código de início 8, pontos de passagem com código de 0 a 7, sendo que cada um deles tem também informação sobre sua característica. Nas figuras III.15 e III.16 é apresentada a codificação dos bits dentro do byte que representa cada pixel no vetor de saída C(z) e na matriz de entrada B(y,x).

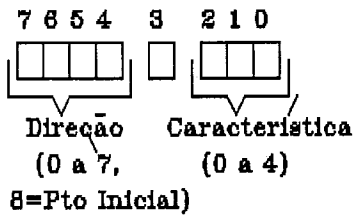

Figura III.15 - Representação da codificação dos bits do byte associado a cada pixel do vetor de saída C(z).

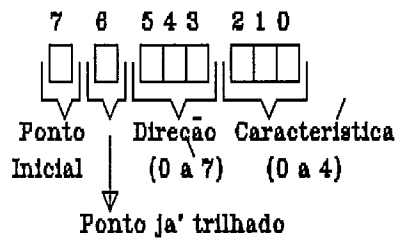

Figura III.16 - Representação da codificação dos bits do byte associado a cada pixel da matriz de entrada B(y,x).

ROTINA TRILHA DE CAMINHO

ENTRADA: matriz B(y,x) afinada

SAÍDA: vetor C(z) com código da cadeia de FREEMAN, informação sobre pixels característicos e informação sobre colocação ou retirada da PILHA.

```
PARA linha:=2 ATÉ m-1 FAÇA;
  PARA coluna:=2 ATÉ n-1 FAÇA;
    SE B(linha,coluna) <> 0 ENTÃO
      SE ( V(linha,coluna) <> 0 ) E
         ( V(linha,coluna) <> 2 ) ENTÃO
        SE NOT T(linha,coluna) ENTÃO
          TRACER (linha,coluna);
```

```
PARA linha:=2 ATÉ m-1 FAÇA;
  PARA coluna:=2 ATÉ n-1 FAÇA;
    SE B(linha,coluna) <> 0 ENTÃO;
      SE ( V(linha,coluna) = 2 ) ENTÃO
        SE NOT T(linha,coluna) ENTÃO
          TRACICLO(linha,coluna);
```

algoritmo III - Trilha de Caminho

A seguir descrevemos a rotina TRACER de uma forma mais global, e em português estruturado de modo a facilitar a compreensão do algoritmo. A rotina TRACICLO, a função T e todas as rotinas e funções chamadas dentro das rotinas principais serão descritas em português ou português estruturado.

ROTINA TRACER

ENTRADA: linha, coluna do pixel característico inicial

SAÍDA: vetor C(z) preenchido.

DESCRIÇÃO:

1. Coloca elemento inicial na pilha.
2. Salva elemento e posição (linha,coluna) no vetor de saída C(z).
3. Cadastra e acende pixel.
4. ENQUANTO a pilha não está vazia FAÇA (4-28)
5. Colocar início como VERDADEIRO.
6. ENQUANTO pixel é de passagem OU início FAÇA (6-16)
7. Colocar achou FALSO, cont igual a 0 e início igual a FALSO.
8. ENQUANTO não achou E cont <= 4 FAÇA (8-16)
9. Procura vizinho aceso segundo a rotina TRAÇADO.
10. SE achou ENTÃO (10-16)
11. SE pixel é de passagem ENTÃO (11-15)
12. Cadastra pixel em B(linha,coluna) como trilhado e característica.
13. Avança na direção do vizinho e determina direção.
14. Atualiza linha e coluna.
15. Salva direção em C(z).
16. Acende pixel.
17. SE achou ENTÃO (17-25)
18. SE pixel ainda não foi cadastrado ENTÃO (18-20)
19. Cadastra em B(linha,coluna) como trilhado e característica.
20. Salva característica em C(z).
21. SE pixel é terminal ENTÃO (21-23).
22. Pega elemento da pilha.
23. Salva em C(z) característica, código de ponto inicial, linha e coluna.
24. SE pixel é de bifurcação ou cruzamento ENTÃO. (24-25)
25. Coloca elemento na pilha.
26. SENÃO SE todos os vizinhos já foram trilhados ENTÃO. (26-28)
27. Pega elemento da pilha.
28. Salva em C(z) como inicial, linha e coluna.

```
Status=COLOCA_NA_PILHA (linha, coluna);
C(z):=V(linha,coluna) OR 80h;
C(z+1):=linha; C(z+2):=coluna;
B(linha,coluna):= V(linha,coluna) OR 0C0h;
ACENDE_PIXEL(linha,coluna);
ENQUANTO existe elemento na pilha FAÇA
  INÍCIO
    início:=VERDADEIRO;
    ENQUANTO ( V(linha,coluna)=2 ) OU início FAÇA
      INÍCIO
        achou:=FALSO; cont:=0; início:=FALSO;
        ENQUANTO ( NOT achou ) OU ( cont <= 4 ) FAÇA
          INÍCIO
            achou:=TRAÇADO (linha,coluna,vizinho);
            SE achou ENTÃO
              INÍCIO
                direção:=AVANÇA(linha,coluna,vizinho);
                SE (V(linha,coluna)=2) ENTÃO
                  INÍCIO
                    C(z+i):=(direção) OR 2;
                    B(linha,coluna):= 42h;
                  FIM
                ACENDE_PIXEL(linha,coluna);
              FIM
          FIM
      FIM
    SE achou ENTÃO
      INÍCIO
        SE B(linha,coluna) = 1 ENTÃO
          INÍCIO
            B(linha,coluna):= V(linha,coluna) OR 40h;
            C(z+j):= V(linha,coluna);
            SE V(linha,coluna) = 1 ENTÃO
              INÍCIO
                status:=RETIRA_DA_PILHA(linha,coluna);
                SE status ENTÃO
                  INÍCIO
                    B(linha,coluna):=
                          B(linha,coluna) OR 80h;
                    C(z+j):=C(z+j) OR 80h;
                    C(z+j+1):= linha; C(z+j+2):= coluna;
                  FIM
              FIM
            SENÃO
              Status:=COLOCA_NA_PILHA(linha,coluna);
          FIM
      FIM
    SENÃO
      INÍCIO
        status:=RETIRA_DA_PILHA(linha,coluna);
        SE status ENTÃO
          INÍCIO
            C(z+k):=C(z+k) OR 80h;
            C(z+k+1):= linha; C(z+k+2):= coluna;
          FIM
      FIM
  FIM
```

ROTINA TRACICLO

ENTRADA: linha, coluna

SAÍDA: vetor C(z) preenchido.

```
Xinicial:=coluna; Yinicial:=linha;
C(z):=82h;
C(z+1):=linha; C(z+2):=coluna;
B(linha,coluna):= 0C2h;
ACENDE_PIXEL(linha,coluna);
início:=VERDADEIRO;
ENQUANTO (Xinicial <> coluna) OR (Yinicial <> linha) OR
inicio FAÇA;
  INÍCIO
    achou:=FALSO; cont:=0; início:=FALSO;
    ENQUANTO ( NOT achou ) OU ( cont < 4 ) FAÇA
      INÍCIO
        achou:=TRAÇADO (linha,coluna,vizinho);
        SE achou ENTÃO
          INÍCIO
            direção:=AVANÇA(linha,coluna,vizinho);
            C(z+i):=direção OR 2;
            B(linha,coluna):= 42h;
            ACENDE_PIXEL(linha,coluna);
          FIM
        SENÃO
          cont:=cont+1;
      FIM
  FIM
```

FUNÇÃO AVANÇA

ENTRADA: linha, coluna, vizinho.

SAÍDA: Retorna a direção definida entre a posição (linha,coluna) e o vizinho para o qual avançamos na trilha do caminho (D=0,1,..., ou 7), e atualiza a posição (linha, coluna).

FUNÇÃO V

ENTRADA: linha, coluna

FUNÇÃO: Retorna 0 para pixel isolado, 1 para pixel terminal, 2 para pixel de passagem, 3 para pixel de bifurcação e 4 para pixel de cruzamento.

FUNÇÃO T

ENTRADA: linha, coluna.

FUNÇÃO: Retorna VERDADEIRO se pixel já foi trilhado, caso contrário retorna FALSO.

FUNÇÃO TRAÇADO

ENTRADA: linha, coluna

SAÍDA: vizinho X[k] não trilhado.

FUNÇÃO: Retorna VERDADEIRO caso encontre o vizinho ou FALSO em caso contrário.

```
dir:=6;
SE ( X[k:=dir] de B(linha,coluna) <> 0 ) E
   ( X[k] não trilhado ) ENTÃO
  INÍCIO
    vizinho:=X[k];
    achou:=VERDADEIRO;
  FIM
SENÃO
  INÍCIO
    SE ( X[k:=dir-1] de B(linha,coluna) <> 0 ) E
       ( X[k] não trilhado ) ENTÃO
      INÍCIO
        vizinho:=X[k];
        achou:=VERDADEIRO;
        dir:=dir-2;
      FIM
    SENÃO
      INÍCIO
        SE ( X[k:=dir+1] de B(linha,coluna) <> 0 ) E
           ( X[k] não trilhado ) ENTÃO
          INÍCIO
            vizinho:=X[k];
            achou:=VERDADEIRO;
          FIM
        SENÃO
          dir:=dir+2;
      FIM
  FIM
RETORNA achou;
```

FUNÇÃO COLOCA_NA_PILHA

ENTRADA: linha, coluna.

FUNÇÃO: Insere elemento na pilha com informação sobre a localização do pixel, isto é, o par (linha,coluna). Retorna VERDADEIRO para inserções com sucesso e FALSO para pilha cheia.

FUNÇÃO RETIRA_DA_PILHA

SAíDA: linha, coluna.

FUNÇÃO: Retira elemento da pilha colocando à disposição a informação de localização do pixel, isto é, o par (linha, coluna). Retorna VERDADEIRO para retiradas com sucesso e FALSO para pilha vazia.

A seguir são mostradas três configurações consideradas críticas para o término correto do algoritmo III: a) dois pixels característicos ligados duas vezes entre si, b) dois pixels característicos ligados três vezes entre si, e c) dois pixels carcterísticos ligados quatro vezes entre si (figura III.17). As ligações são compostas por pixels de passagem.

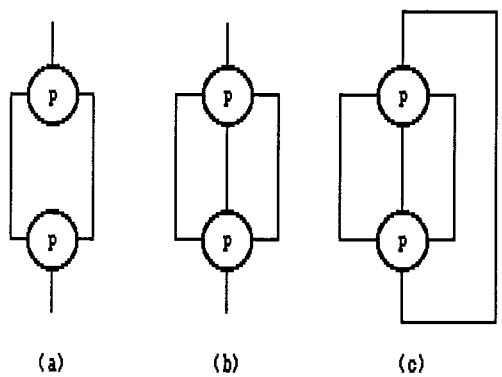

Figura III.17 - Configurações críticas de término do algoritmo de Trilha de Caminho.

III.1.7 - SEGMENTAÇÃO

Foi feita uma pesquisa bibliográfica sobre a segmentação [1,16,18,36,37,38,39,40], só que não chegamos a implementá-la. A sugestão que damos para a implementação é a de encontrar a melhor definição, aplicável a um algoritmo, da propriedade da curvatura. Uma linha importante a ser pesquisada é a da computação gráfica, onde existem vários algoritmos de traçado de retas e curvas em um gradeado discreto [45]. Podemos pensar em adaptar estes algoritmos para que eles percorram os pixels de cada segmento, analisando se este se aproxima mais a uma curva ou a uma reta. A análise do erro, isto é, quanto um pixel pode estar afastado de uma vizinhança e ainda assim ser considerado como pertencente ao contorno de uma reta ou uma curva, é fundamental. É importante perceber que os segmentos de figura podem não ser compostos apenas por uma única reta ou uma única curva, e sim por um conjunto de retas e curvas como pode ser visto na figura III.18.

Neste ponto é interessante analisar a aplicação para percebermos se existe alguma característica que possa ser utilizada para a simplificação do problema. Percebemos que nosso desenho é formado basicamente por segmentos de reta, mas os caracteres são formados por curvas e retas. Portanto, como estamos tratando indiferentemente tanto figuras como caracteres não podemos simplificar a nossa análise apenas para retas. O que pode ser feito é controlar o refinamento do algoritmo de tal forma que um conjunto de re-

tas possa ser aproximado a uma única curva, ou uma curva com pequena inclinação possa ser aproximada a uma reta. Estas simplificações podem ser utilizadas desde que não alterem o significado do desenho. Devemos, por exemplo, evitar que segmentos que antes não se interceptavam passem a se interceptar em função das simplificações utilizadas.

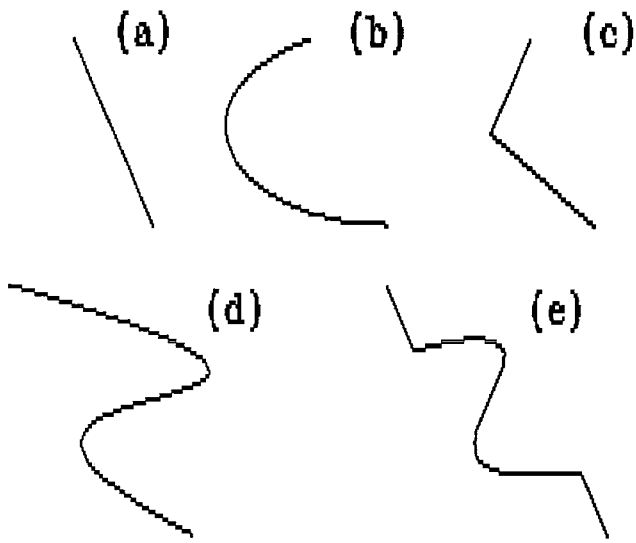

Figura III.18 - Segmentos de figura compostos por uma única reta (a), por uma única curva (b), por duas retas (c), por duas curvas (d) e por curvas e retas (e).

O algoritmo sugerido deve percorrer o vetor C(z) que contem a imagem codificada segundo o código da cadeia. Todo byte do vetor que está marcado como inicial é seguido de dois bytes representando a linha e a coluna, respectivamente. Com isso temos o posicionamento absoluto da localização do primeiro pixel a ser aceso, sendo que os seguintes são dados por deslocamentos relativos associados à direção. Ao percorrer o vetor devemos testar a característi-

ca de cada byte para identificar até onde vai o segmento de figura corrente. Não há necessidade mais de usar uma pilha, uma vez que sempre que há um desvio da posição corrente, este estará associado a um pixel característico marcado como inicial, e que é seguido da posição absoluta dada pela linha e a coluna. Se encontrarmos um pixel característico que não está marcado como inicial, devemos partir dele para encontrar outro segmento, não havendo necessidade de desvio. De cada segmento de figura percorrido é extraída a informação de comprimento e inclinação. É feita, então, uma aproximação a uma reta, uma curva ou um conjunto de retas e curvas. A saída do algoritmo pode ser direto um arquivo com chamadas às primitivas gráficas. Por exemplo, se analisamos um determinado segmento de figura, que é inicial e pode ser aproximado a uma reta, teremos como saída:

```
MOVE_ABSOLUTO Xinicial, Yinicial;
RETA_RELATIVO Delta_X, Delta_Y;
```

A partir daí todos os outros movimentos podem ser considerados relativos, mesmo que eles sejam originalmente absolutos, bastando para isso conhecermos a posição corrente e calcularmos o deslocamento necessário para chegarmos à nova posição.

Podemos acender os pixels de cada segmento percorrido de forma a acompanhar o funcionamento do algoritmo.

Obtemos, portanto, neste ponto, uma descrição compacta e direcionada para o interfaceamento com programas editores elaborados segundo as técnicas e definições de computação gráfica.

III.1.8 - INTERFACE COM O EDITOR GRÁFICO

A interface com o editor gráfico pode ser implementada já embutida no algoritmo sugerido anteriormente. Este item foi criado porque podem existir nuances do editor gráfico que precisam ser analisadas. Informações para inicialização, como número de pixels na horizontal e na vertical, tamanho do arquivo, cor de frente, cor de fundo, atributos de linha, título, etc, podem ser necessários e muitas vezes precisam estar explícitos. Para isso teremos que verificar através da análise da documentação do editor o que falta ser acrescentado ao arquivo gerado. Além disso, precisamos descobrir quais são as primitivas disponíveis e quais são as limitações do editor, por exemplo, quanto ao tamanho dos arquivos. É interessante escolher um editor que possua documentação técnica e facilidade para interfaceamento via arquivo intermediário externo. Para facilitar a depuração, o ideal é que o arquivo gerado esteja no formato ASCII, pois, caso contrário a depuração será dificultada.

Geralmente as primitivas encontradas de nosso interesse são: acender ponto, desenhar reta relativa ou absoluta, desenhar polígono absoluto ou relativo e aberto ou fecha-

do, desenhar arco, desenhar círculo, desenhar retângulo e posicionar cursor relativo ou absoluto.

III.2 - EQUIPAMENTOS, FERRAMENTAS E LINGUAGENS

Os algoritmos correspondentes aos blocos 1,2,4,5,6,7 de Processamento de Imagens (figura III.1) foram implementados em TURBO PASCAL 3.0 e testados com padrões simplificados, alguns destes sintetizados pelo editor de imagens LATIM [8] que foi, também, utilizado como ferramenta de depuração. Os equipamentos utilizados foram uma câmera JVC-VIDICOM com lente macro, uma placa digitalizadora [41], e um micro PC-XT.

CAPÍTULO IV

IV - RESULTADOS

A seguir mostramos os resultados obtidos pelo processamento dos algoritmos de Binarização (I), de Afinamento (II) e de Trilha de Caminho (III). Os algoritmos foram aplicados a três imagens básicas digitalizadas com 256 colunas, 244 linhas e 64 níveis de cinza. A imagem A (LIDIA.pic, figura B.1) corresponde a um rosto e possui originalmente diversos tons de cinza. A imagem B (MAPA.pic, figura B.2) representa uma parte de um diagrama que possui apenas dois níveis. A imagem C (COLUMBIA.pic, figura B.3) corresponde a um desenho de apenas dois níveis impresso por uma impressora matricial. Também digitalizamos uma folha em branco (FUNDO.pic, imagem D) para analisarmos o seu histograma.

IV.1 - BINARIZAÇÃO

Apresentamos nas figuras (IV.1a) a (IV.1h) os histogramas das imagens de A a D, ou seja, as distribuições do número de pixels pelos níveis de cinza existentes. Cada imagem é acompanhada de dois histogramas: o primeiro é proporcional e o segundo tem a escala das ordenadas não linear para ressaltar baixas ocorrências de níveis de cinza.

A análise destes histogramas é importante para a determinação dos parâmetros a serem definidos para o algoritmo 1

de binarização. O limite máximo e o limite mínimo para a imagem B podem ser por exemplo 35 e 60, para a imagem C, 51 e 62, e para a imagem A, 11 e 46.

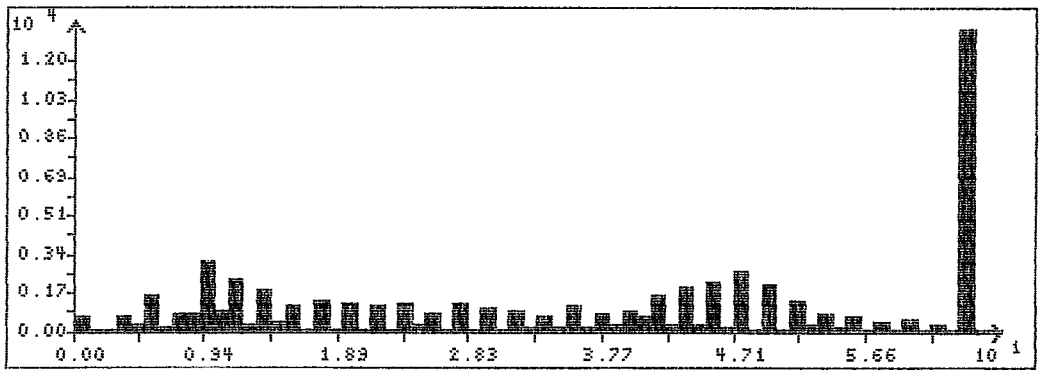

Figura IV.1a - Histograma da Imagem A.

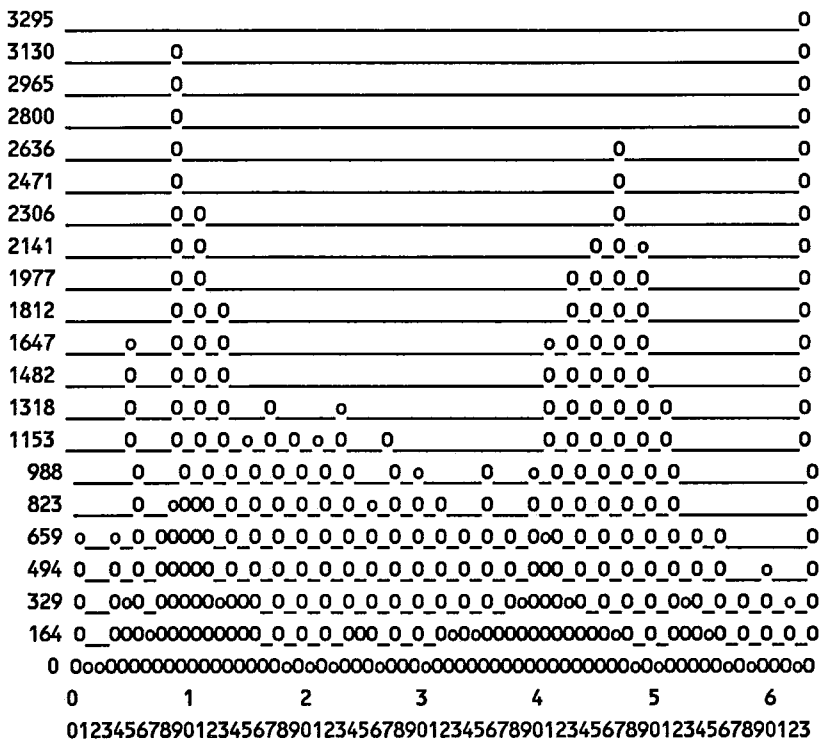

Figura IV.1b - Ampliação do Histograma da Imagem A.

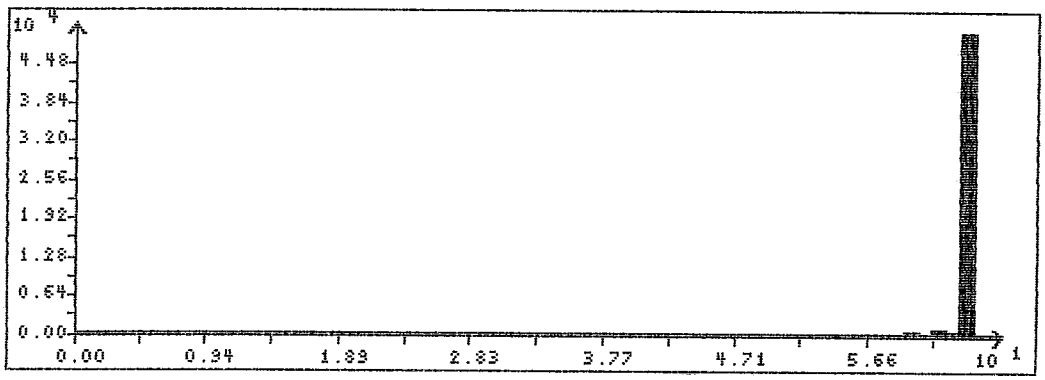

Figura IV.1c - Histograma da Imagem B.

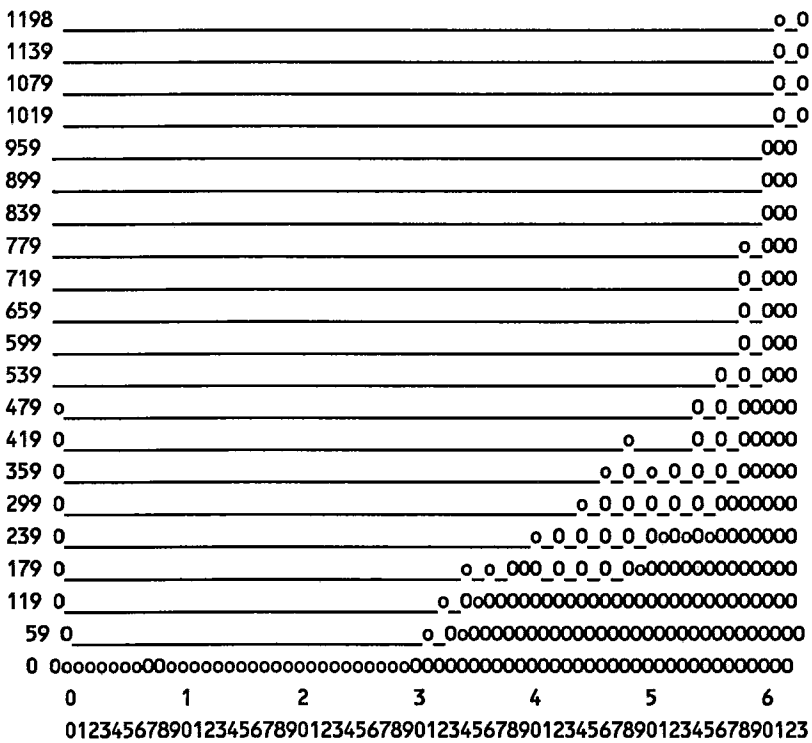

Figura IV.1d - Ampliação do Histograma da Imagem B.

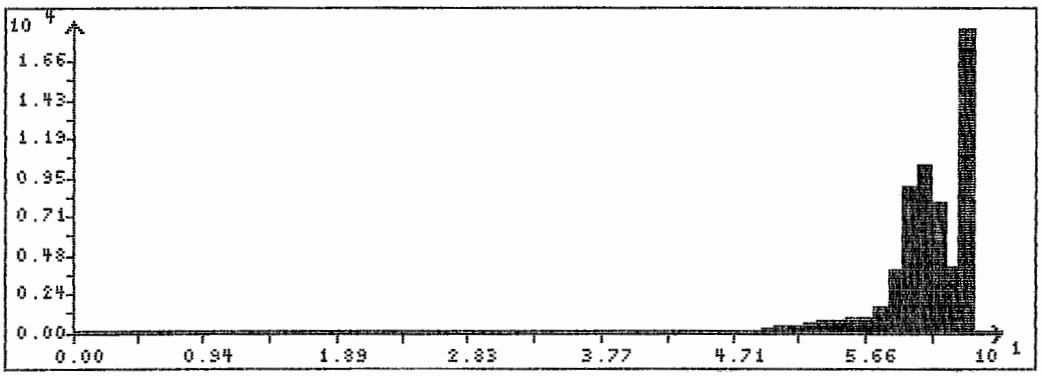

Figura IV.1e - Histograma da Imagem C.

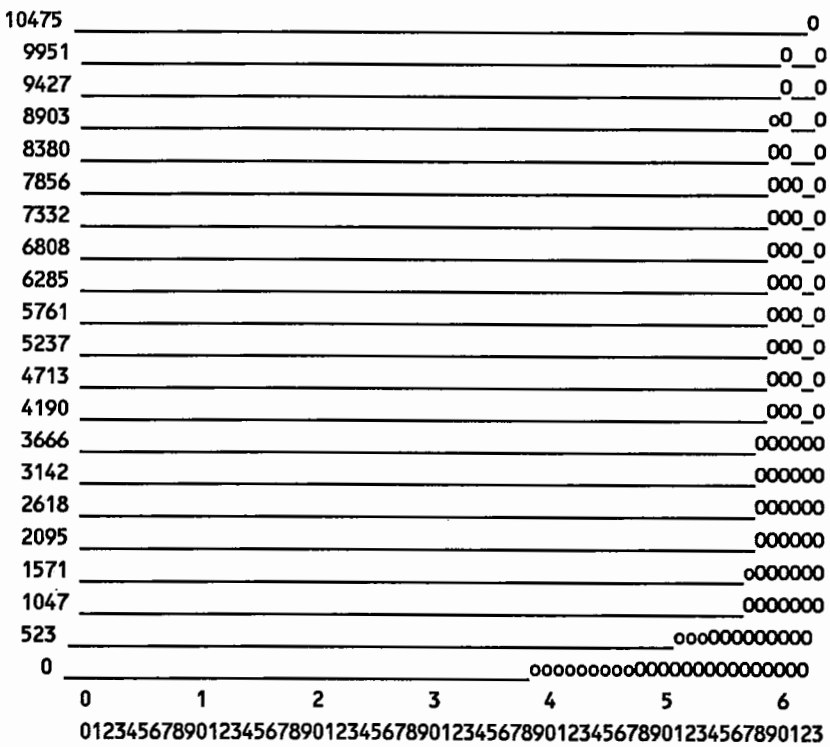

Figura IV.1f - Ampliação do Histograma da Imagem C.

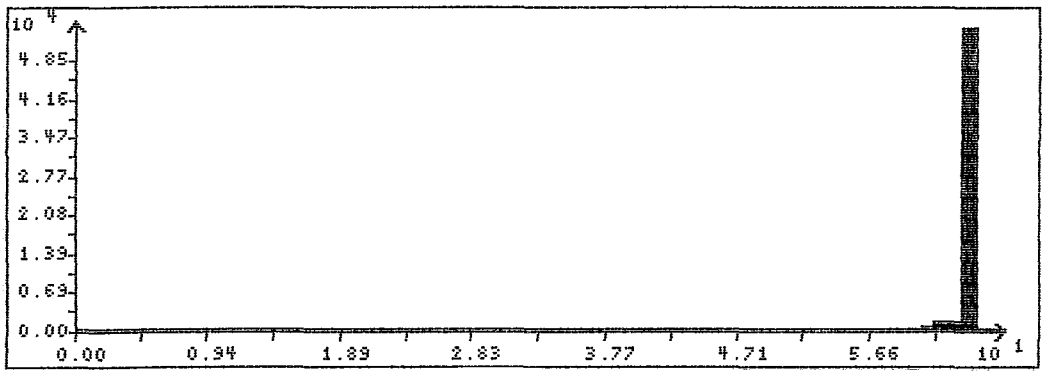

Figura IV.1g - Histograma da Imagem D.

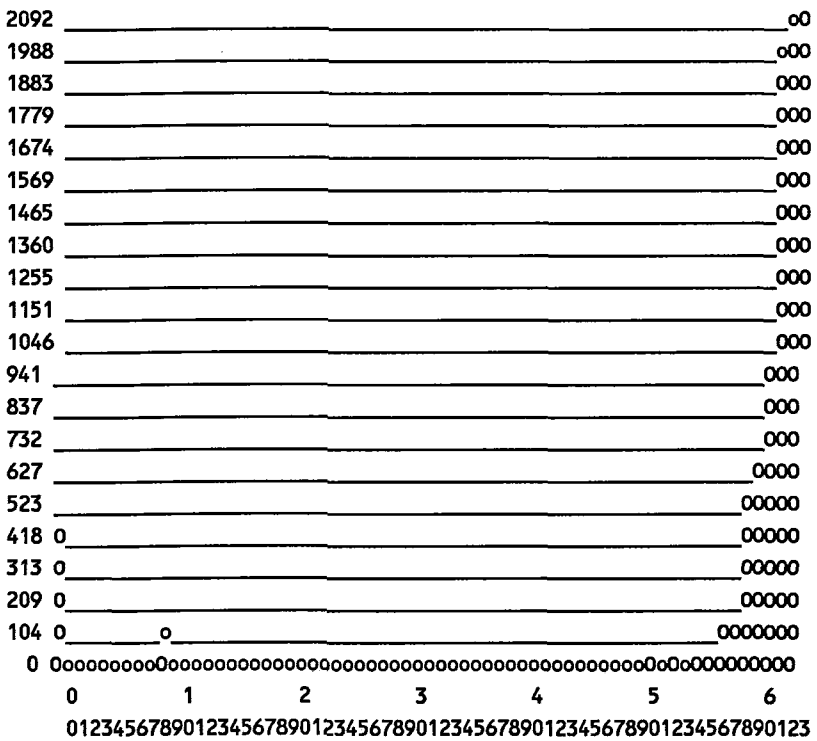

Figura IV.1h - Ampliação do Histograma da Imagem D.

A seguir são apresentados os resultados da aplicação do algoritmo 1 para diversos tamanhos de janela de processamento.

Figura IV.2a - Binarização de LIDIA.pic com janela 5x5, limite mínimo = 9, limite máximo = 48, delta nível = 0.

Figura IV.2b - Binarização de LIDIA.pic com janela 1x1, limite mínimo = 32, limite máximo = 32, delta nível = 0.

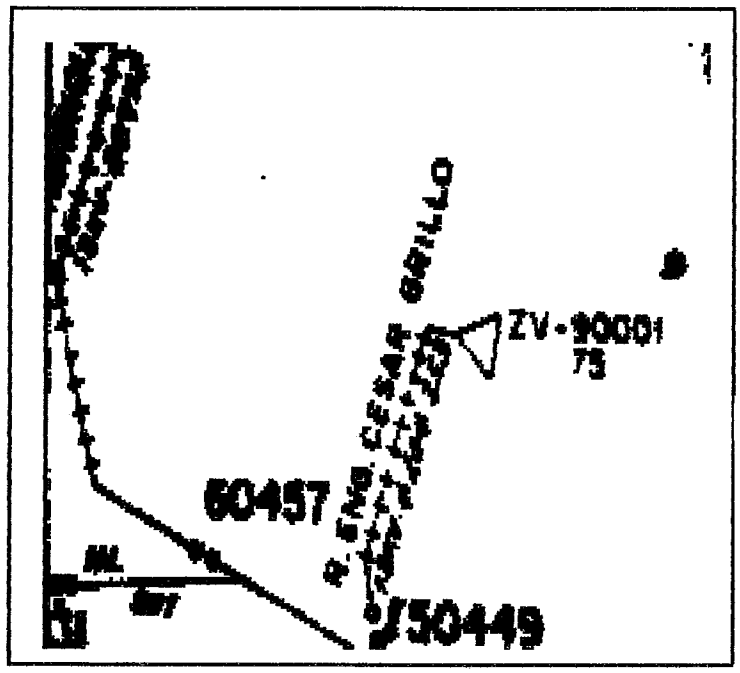

Figura IV.2c - Binarização de MAPA.pic com janela 1x1, limite mínimo = 55, limite máximo = 55, delta nível = 0.

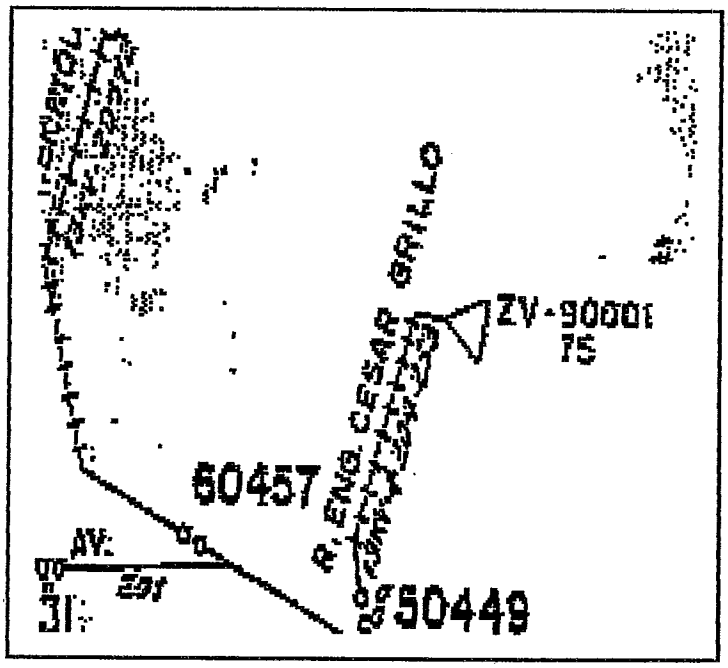

Figura IV.2d - Binarização de MAPA.pic com janela 3x3, limite mínimo = 30, limite máximo = 62, delta nível = 0.

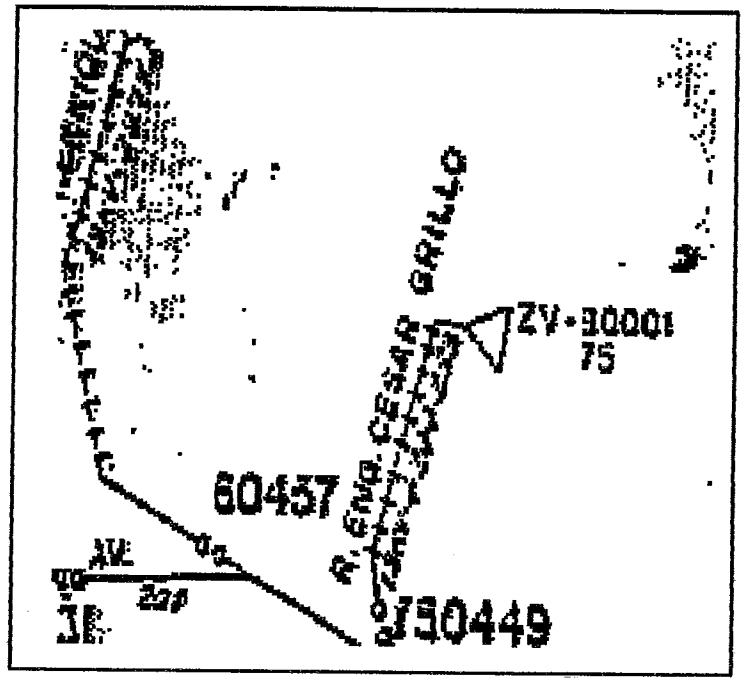

Figura IV.2e - Binarização de MAPA.pic com janela 4x4, limite mínimo = 30, limite máximo = 62, delta nível = 0.

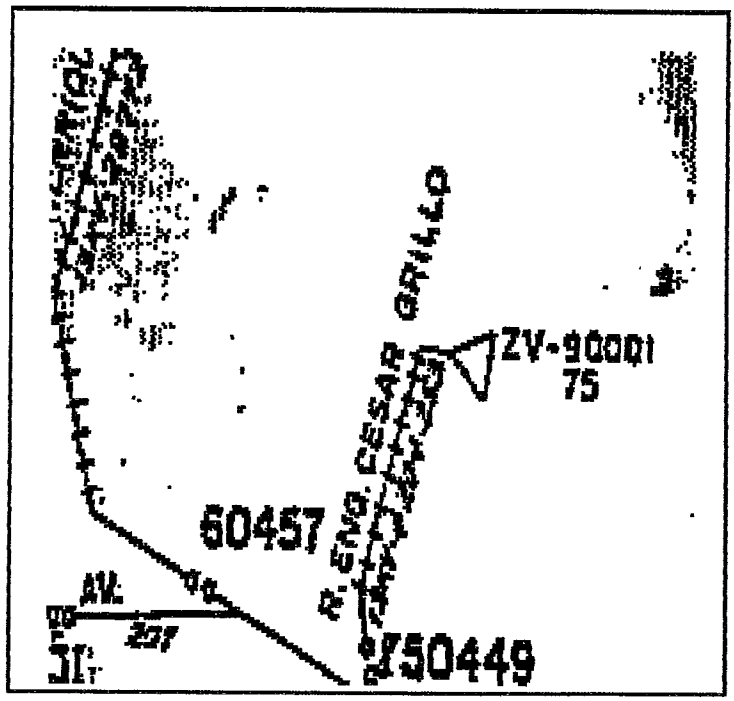

Figura IV.2f - Binarização de MAPA.pic com janela 5x5, limite mínimo = 30, limite máximo = 62, delta nível = 1.

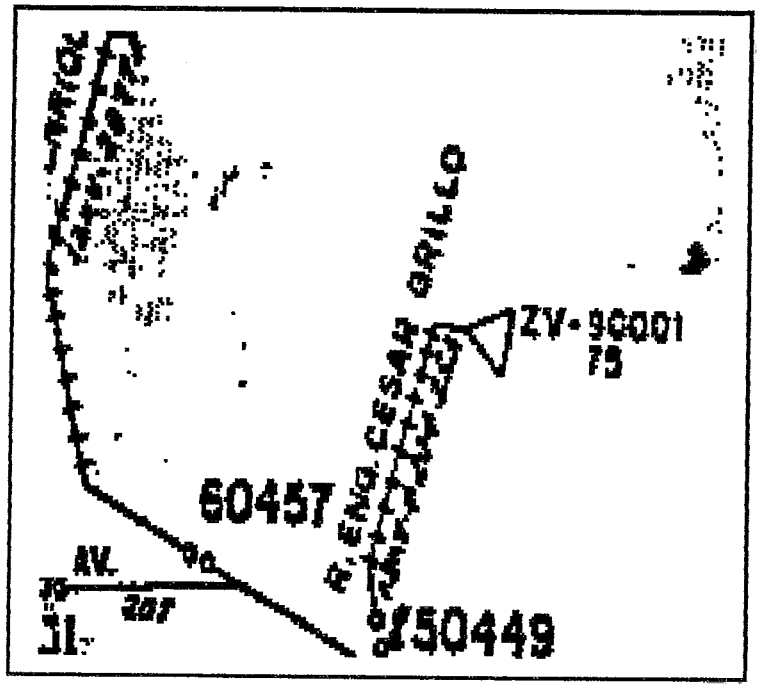

Figura IV.2g - Binarização de MAPA.pic com janela 8x8, limite mínimo = 30, limite máximo = 62, delta nível = 0.

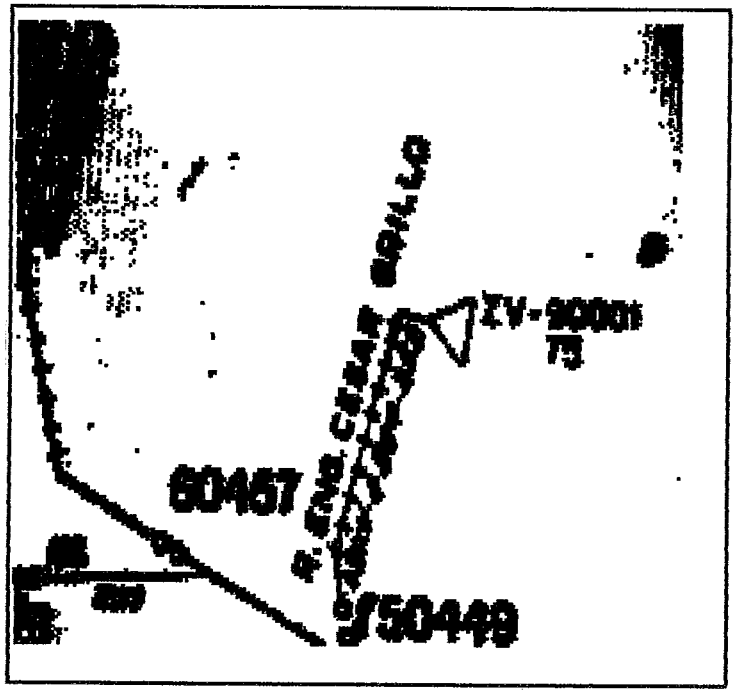

Figura IV.2h - Binarização de MAPA.pic com janela 5x5, limite mínimo = 30, limite máximo = 62, delta nível = 10.

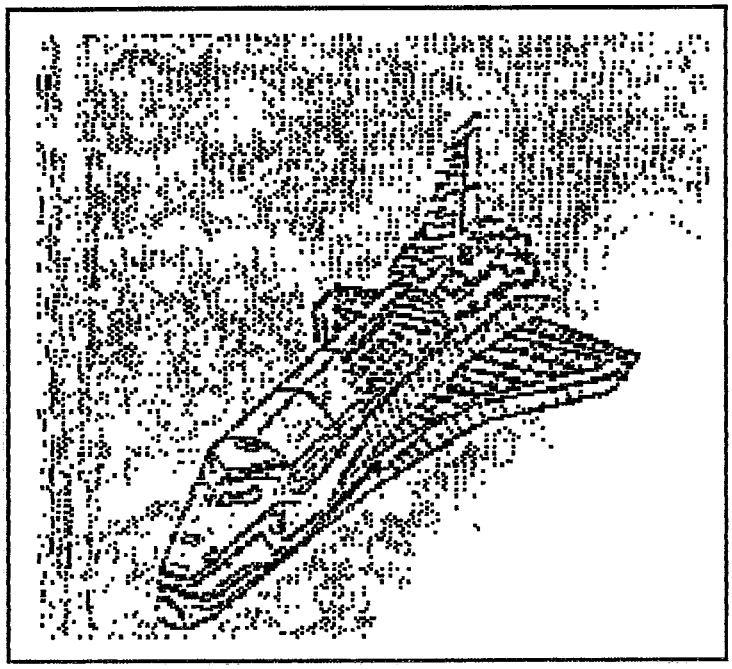

Figura IV.2i - Binarização de COLUMBIA.pic com janela 5x5, limite mínimo = 39, limite máximo = 62, delta nível = 0.

IV.2 - AFINAMENTO

A seguir são apresentados resultados da aplicação do algoritmo II. Nas figuras (IV.3a) a (IV.3c) o afinamento é aplicado a imagens resultantes da binarização. As demais figuras são resultado do afinamento de imagens sintetizadas no LATIM (figuras IV.4a a IV.4g).

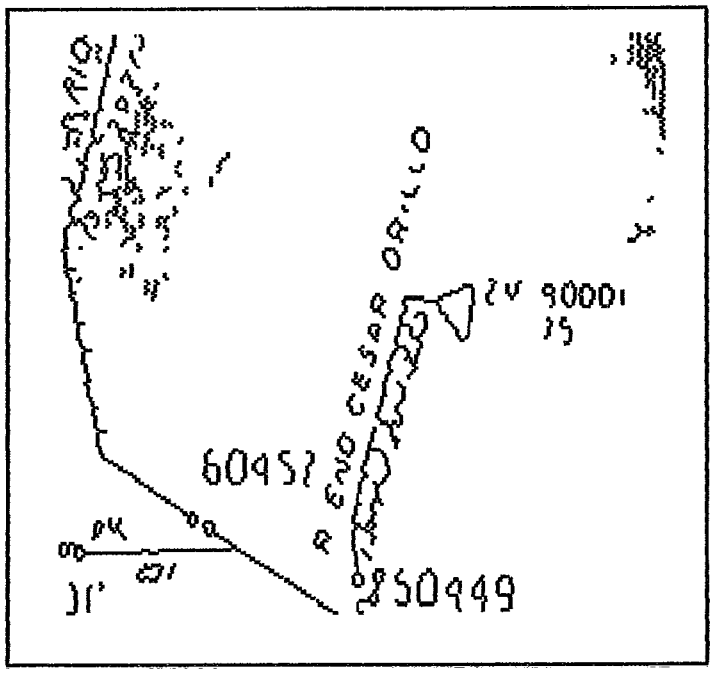

Figura IV.3a - Afinamento de MAPA.BIN (figura IV.2f).

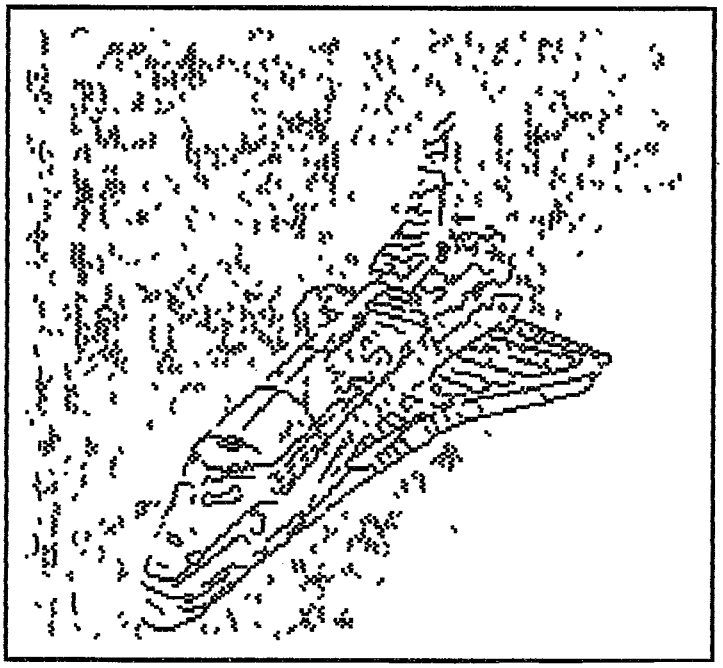

Figura IV.3b - Afinamento de COLUMBIA.BIN (figura IV.2i).

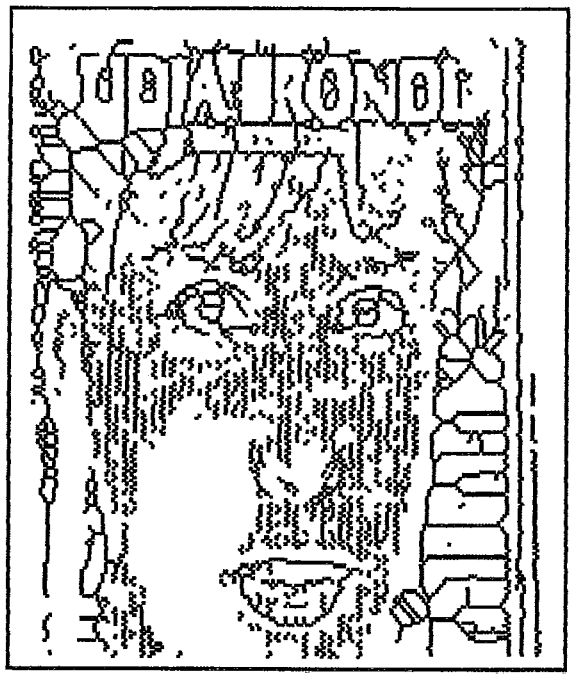

Figura IV.3c - Afinamento de LIDIA.BIN (figura IV.2a).

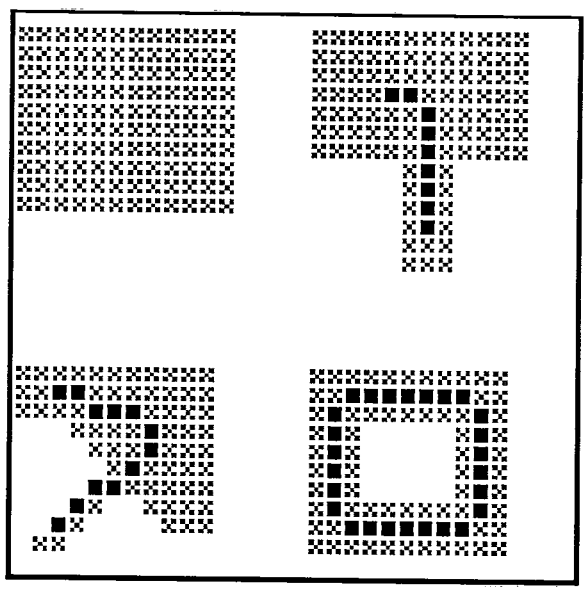

Figura IV.4a - Afinamento de PADRÕES.LAT.

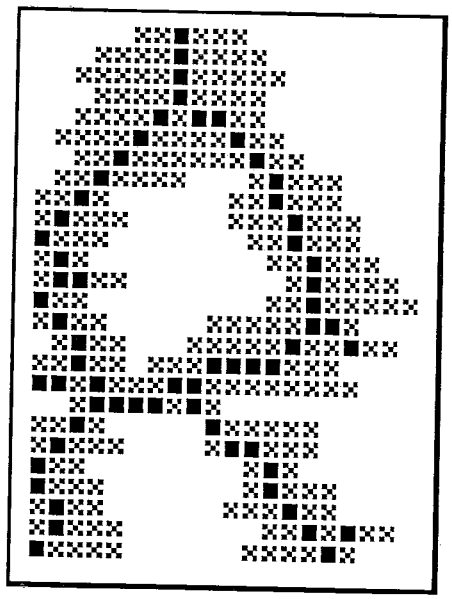

Figura IV.4b - Afinamento da letra A (A.LAT).

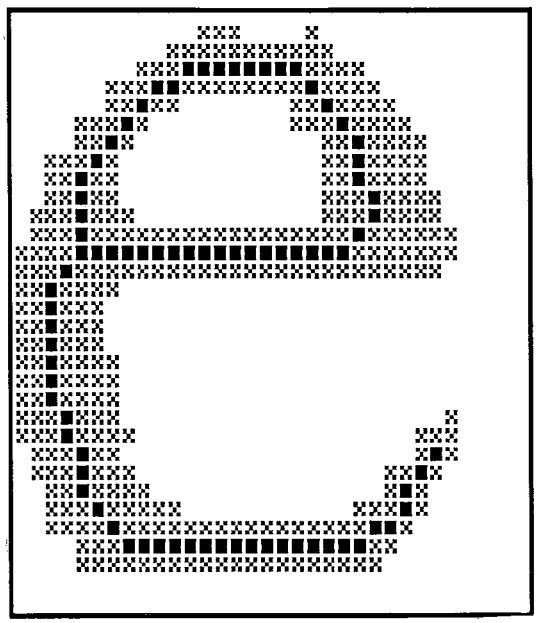

Figura IV.4c - Afinamento da letra E (E.LAT).

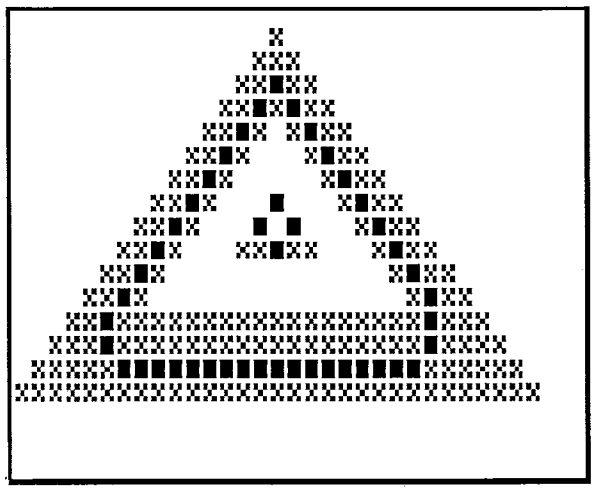

Figura IV.4d - Afinamento do TRIÂNGULO (TRI.LAT).

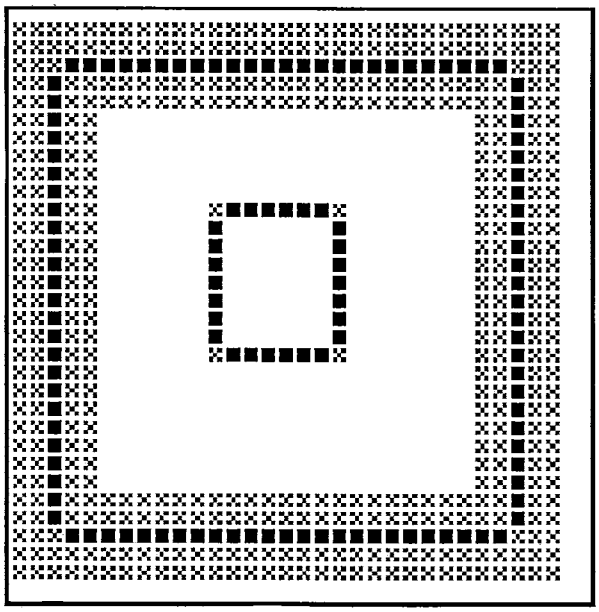

Figura IV.4e - Afinamento dos QUADRADOS (QUAD.LAT).

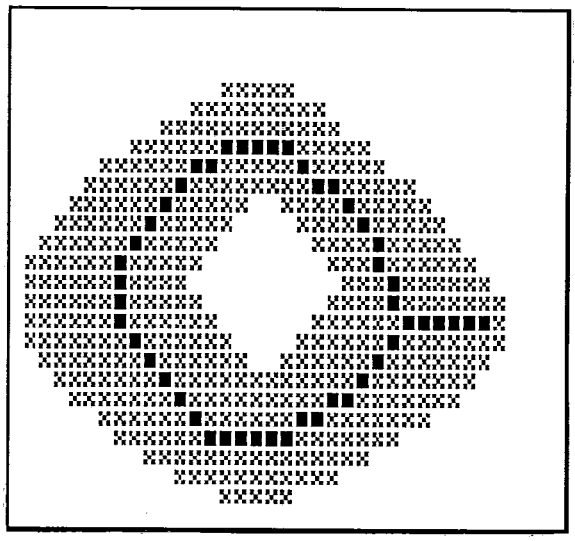

Figura IV.4f - Afinamento do CÍRCULO (CIR.LAT).

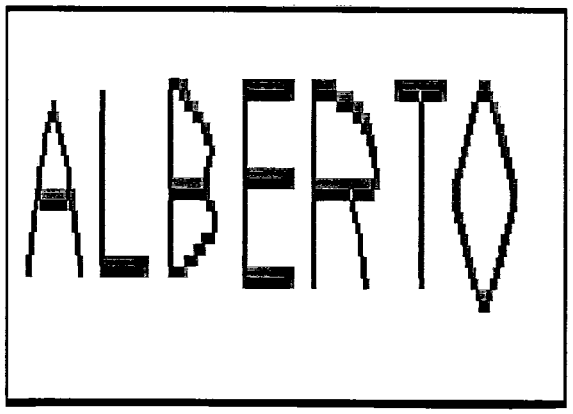

Figura IV.4g - Afinamento do "ALBERTO" (BETO.LAT).

IV.3 - TRILHA DE CAMINHO

A trilha de caminho não possue imagens para mostrar os resultados, já que nesta fase temos como saída um arquivo com a codificação segundo o código da cadeia acrescido da informação da característica dos pixels. Para acompanhar se o algoritmo III funciona corretamente devemos examinar a imagem afinada de entrada e o arquivo de saída, e também acompanhar na tela se todos os caminhos a serem trilhados foram percorridos. Se cada pixel trilhado for aceso na tela, poderemos verificar ao final se todos os caminhos foram trilhados.

A seguir apresentamos o resultado da aplicação do algoritmo de Trilha de Caminho à letra "A" apresentada na figura (IV.4b), ou seja, a seqüência de bytes com o código da cadeia, a característica dos pixels e a marcação de pixel inicial conforme a codificação representada na figura (III.15).

```
E1  08  01
62  62  53  52  52  52  52  52  52  72  62  53  72  72  62  53  42
81  03  11
70  62  53  52  52  62  72  62  52
81  04  13
00  02  02  12  02  13  02  02  02  12  12  03  72
81  0F  0F
20  22  22  32  22  32  22  32  32  32  42  32
83  09  12
60  72  72  02  72  62  72  72  72  12
81  02  0D  00
81  08  04
81  08  01
81  08  01
```

CAPÍTULO V

V - DISCUSSÃO

Os resultados apresentados no capítulo IV serviram para analisarmos o desempenho dos algoritmos de binarização, afinamento e trilha de caminho. Além disso ficaram claras suas limitações e sucetibilidade a ruído. As imagens apresentadas serviram para auxiliar na depuração dos algoritmos, sendo que seu processamento para diversos parâmetros permitiu, através da comparação, que os ajustássemos da melhor forma possível.

A qualidade dos resultados, principalmente os da binarização, foi analisado de forma comparativa e subjetiva, pois a "qualidade" de um mesmo resultado visual pode ser melhor ou pior dependendo da pessoa [32].

A seguir são discutidos todos os resultados apresentados no capítulo anterior.

V.1- BINARIZAÇÃO

Nas figuras (IV.2.a) e (IV.2.b), aplicamos o algoritmo de binarização a uma imagem que apresenta originalmente vários tons de cinza (LÍDIA, figura B.1), sendo considerada uma imagem classe 1. Podemos observar que nos histogramas correspondentes a essa imagem (figuras IV.1.a e IV.1.b) existe uma boa distribuição dos pixels pelos níveis de

cinza. Podemos, também, observar que um grande número de pixels estão associados ao nível 63, nível branco, ou seja, ao fundo da imagem.

Na figura (IV.2.b) fizemos uma binarização com janela 1x1, ou seja, foi analisado pixel a pixel sem levar em consideração sua vizinhança. Escolhemos o valor 32 tanto para o nível superior quanto para o nível inferior e a variação de nível (delta nível) igual a zero. Com isso nosso algoritmo de binarização se comporta da mesma forma que o algoritmo de binarização simples, pois estamos analisando pixel a pixel (janela 1x1) e só temos um único nível de binarização, uma vez que o nível inferior é igual ao superior. O resultado deste processamento, apresentado na figura (IV.2.b), mostra que os detalhes da imagem original (figura B.1) não foram preservados, sendo que foram formados blocos de áreas brancas e pretas. Isto era esperado uma vez que existia informação da imagem associado aos níveis de cinza, e uma conversão simples de vários níveis para apenas dois faz com que infomações sejam perdidas.

Já na figura (IV.2.a) foi aplicada a mesma imagem o algoritmo de binarização, mas com janela 5x5, limite inferior igual a 9, limite superior igual a 48 e delta nível nulo. Percebemos agora mais detalhes do que na figura anterior, mas em compensação o ruído (rosto da Lídia) ficou em evidência. Quanto maior for o tamanho da janela, mais insensível ao ruído ficará o resultado, mas menos detalhes serão identificados, havendo uma tendência do resultado da

figura (IV.2.a) ficar igual ao da figura (IV.2.b).

Nas figuras (IV.2.c) a (IV.2.g) foi aplicado o algoritmo de binarização a um pedaço da imagem da figura (B.2). Esta imagem apresenta apenas dois níveis sendo classificada como imagem da classe 2. O histograma desta imagem está representado nas figuras (IV.1.c) e (IV.1.d). Podemos observar neste histograma uma grande concentração de pixels à direita, isto é, não existe uma boa distribuição de pixels pelos níveis de cinza. Esta concentração na região mais clara do histograma, ou seja, na área que apresenta níveis de cinza de valor mais elevado, mostra a influência do fundo branco sobre a frente escura da imagem do diagrama. Mostra, também, que não existe uma distinção clara entre as regiões preta e branca do diagrama, ficando visível a dificuldade de obtermos bons resultados se aplicarmos à imagem um algoritmo de binarização simples.

Na figura (IV.2.c) escolhemos os parâmetros de forma a que obtivéssemos um resultado correspondente a uma binarização simples, ou seja, janela 1x1, limite inferior igual ao limite máximo com valor 55 e delta nível nulo. Podemos notar que com esses parâmetros o resultado obtido não foi satisfatório, um vez que houve perda de conectividade em diversos pontos (linhas ligadas na imagem original passaram a ficar descontínuas), e houve modificação na topologia (números e letras com um furo passaram a ficar sem furo algum).

Nas figuras (IV.2.d) a (IV.2.g) testamos vários tamanhos de janela diferentes: 3x3, 4x4, 5x5 e 8x8, mas com o mesmo limite superior e inferior. A conectividade e a topologia não foram mantidas em nenhum destes resultados, mas foi a figura (IV.2.f) que apresentou o melhor resultado comparativo. O fato da conectividade e topologia não terem sido mantidas já neste ponto compromete os resultados dos processamentos posteriores que passarão a ser analisados a partir da conectividade e topologia apresentadas após o processamento da binarização. Percebemos, também, a existência de ruído, menos evidente na figura (IV.2.c) e mais evidente na figura (IV.2.h).

A figura (IV.2.h) tem todos os parâmentros iguais aos da figura (IV.2.f), a menos do delta nível que na primeira é igual a 10 e na última é igual a 1. Este parâmetro ajusta a sensibilidade ao contraste: quanto maior for este parâmetro, mais sensível a pequenas variações de nível de cinza ficará o algoritmo de binarização, isto é, acabará acentuando pequenas variações de nível de cinza. Por isso, a figura (IV.2.h) parece "borrada" se comparada à figura (IV.2.f).

A figura (IV.2.i) apresenta a binarização com janela 5x5 da imagem B.3 (COLÚMBIA). Esta imagem também possuiu originalmente apenas dois níveis. O histograma está representado nas figuras (IV.1.e) e (IV.1.f). O histograma também apresenta concentração de pixels à direita, com pico no nível 63, isto é, um grande número de pixels brancos de

fundo. O ruído devido à iluminação não homogênea está espalhada por toda a imagem.

As figuras (IV.1.g) e (IV.1.h) representam o histograma de uma folha em branco digitalizada nas mesmas condições de iluminação das outras imagens. Como não poderia deixar de ser, existe uma grande concentração de pixels à direita do histograma. Nas imagens anteriores não foi feita a filtragem com compensação da iluminação nào homogênea. O histograma foi apresentado a título de comparação com os outros histogramas das imagens. Podemos perceber que no caso das imagens originalmente monocromáticas a concentração de pixels à direita do histograma mostra o problema da iluminação, uma vez que mesmo uma folha totalmente branca, ao ser digitalizada adquire níveis de cinza de valor menor do que 63.

Concluímos a partir destes resultados que para melhorar a qualidade do processo de binarização devemos melhorar a qualidade do processo de digitalização. Devemos utilizar um "scanner" e filtros específicos para obtermos uma imagem de melhor qualidade, sem ter que modificar e "complicar" o algoritmo de binarização. Ja' abordamos na estimativa da resolução (item III.1) a necessidade de um dispositivo de digitalização adequado para nossa aplicação. De qualquer modo, os resultados obtidos são resultantes de imagens reais e servem para que os problemas existentes venham à tona.

A seguir para testar os algoritmos de afinamento e trilha de caminho utlizaremos as imagens reais binarizadas de melhor qualidade e sintetizaremos outras para testes mais específicos.

V.2 - AFINAMENTO

Nas figuras (IV.3.a) a (IV.3.c) foram apresentados os resultados da aplicação do algoritmo de afinamento a imagens reais. A figura (IV.3.a) é o resultado do afinamento da figura (IV.2.f). Observando cuidadosamente as duas figuras podemos notar que a conectividade e a topologia são mantidas. Podemos observar, também, que o algoritmo é guloso e "come" um pedaço das extremidades dos elementos do diagrama. Com isso o ruído também acaba diminuindo.

A figura (IV.3.b) apresenta o afinamento da figura (IV.2.i). A figura (IV.3.c) apresenta o afinamento da figura (IV.2.a). O que foi discutido para a figura anterior, vale também para essas duas.

As figuras (IV.4.a) a (IV.4.g) apresentam o resultado da aplicação do algoritmo de afinamento a imagens sintetizadas. Foi feita uma superposição entre a imagem original e a resultante do afinamento para facilitar a visualização dos resultados.

Na figura (IV.4.a) notamos que o algoritmo de afinamento não funciona adequadamente quando o padrão não é muito

alongado, não tendo tendência para nenhuma direção, como é o caso do desenho superior esquerdo, onde foi feito um encolhimento total sem preservação de pixels.

No desenho superior direito, a tendência para baixo favoreceu a preservação de alguns pixels. No desenho inferior esquerdo só houve "crescimento" em duas direções, uma vez que a outra direção não se encontrava suficientemente alongada. O desenho inferior direito teve sua topologia e conectividade inteiramente preservadas, pois o algoritmo de afinamento, apesar das limitações observadas nesta figura, sempre preserva os furos não importando o número de pixels deste elemento.

As figuras (IV.4.b) e (IV.4.c) apresentam o afinamento das letras "A" e "E" respectivamente. A letra "A" está mais ruidosa, isto é, apresenta mais pixels acesos e apagados do que o devido e que deformam a letra original. Notamos, então, que o algoritmo de afinamento é sensível ao ruído, havendo "crescimento" de pixels formando indevidamente continuações de linhas. Já a letra "E", por ser menos ruidosa, apresentou um resultado muito melhor. É interessante notar a preservação da topologia, da conectividade e a simplificação do caminho a ser percorrido posteriormente pelo algoritmo de trilha de caminho.

As figuras (IV.4d) a (IV.4f) apresentam algumas figuras geométricas: triângulos, quadrados e círculo, respectivamente. Na figura (IV.4.d) colocamos um triângulo vasado

dentro de outro para testar se o algoritmo de afinamento conseguia "enxergar" figuras dentro de outras figuras. Na figura (IV.4.e), idem, mas só que agora colocamos um quadrado dentro do outro.

Na figura (IV.4.f), notamos no círculo novamente o ruído fazendo o algoritmo "crescer" linhas incorretamente. Verificamos, então, a importância de que os estágios anteriores entreguem para o bloco de afinamento figuras com pouco ruído para simplificar o processamento de afinamento, ou seja, para que o algoritmo apresentado já seja suficiente.

Na figura (IV.4.g), afinamos a palavra "ALBERTO" para testar o algoritmo aplicado a vários elementos isolados entre si. É interessante observar que o algoritmo apresenta resultados satisfatórios para letras com pouco ruído.

V.3 - TRILHA DE CAMINHO

O único resultado apresentado foi o do código da cadeia aplicado à figura (IV.4.b).

CAPÍTULO VI

VI - CONCLUSÃO

O presente trabalho apresentou um descritor para diagramas digitalizados. Foi utilizado o domínio do espaço na confecção dos algoritmos. Cada pedaço da imagem é binarizado, afinado e trilhado, devendo ser segmentado para que seja extraída uma descrição compatível com a de um editor gráfico.

A digitalização por câmera para aplicações como a deste trabalho não é recomendada. A baixa resolução, o ruído, as distorções, a necessidade de interligação e a iluminação não homogênea são problemas que tornam desaconselhável a utilização deste dispositivo na digitalização de diagramas. Mas do ponto de vista didático e acadêmico, a utilização de um dispositivo barato permitiu que fizéssemos uma análise do problema e implementássemos diversos algoritmos para processamento da imagem.

A teoria e as técnicas abordadas neste trabalho são de caráter geral e podem ser estendidas para outras aplicações.

Com base na experiência decorrente deste trabalho, podemos reiterar a afirmação de que em processamento de imagens não existe uma solução geral para todos os problemas, por mais que eles pareçam semelhantes. Cada caso é um caso di-

ferente em que a solução vai depender da aplicação. O conhecimento antecipado de características da imagem são fundamentais para a simplificação do problema. Muitas vezes, somente a partir de tentativas e experiências, é que conseguimos obter as propriedades e parâmetros adequados para determinada aplicação.

Uma recomendação importante é a de utilizarmos uma ferramenta de depuração de imagens. Assim como existem depuradores simbólicos que atuam durante ou após a execução do programa, devemos utilizar depuradores e editores de imagens para analisarmos os resultados e validarmos ou não os algoritmos.

Não podemos deixar de discutir a viabilidade de um projeto de automatização baseado no "Descritor de Diagramas Digitalizados". Pelo que foi visto até aqui concluímos que existem técnicas e equipamentos que tornam viáveis este projeto. Mas, como são necessários equipamentos com grande resolução (digitalizador), capacidade de armazenamento e rapidez de processamento para atender os requisitos da aplicação, é necessário um levantamento do custo-benefício do projeto. Do ponto de vista de custo, o desenvolvimento e instalação de um projeto deste porte são caros, pois envolvem equipamentos e programas de ponta. Mas por outro lado, devemos levar em consideração o benefício que teríamos ao automatizar uma quantidade grande de diagramas, pela organização, facilidade de acesso, manutenção, alteração e apresentação, liberando um espaço físico enorme ocu-

pado pela documentação. Além disso, a tendência é a de que os custos baixem e se torne comum a automatização da documentação.

Vale a pena ressaltar que o sistema proposto por este trabalho não se restringe a automatizarmos apenas a documentação. A finalidade é a de criarmos um sistema de informação que permita que um operador acompanhe suas variações diretamente em telas gráficas, através da visualização dos diagramas digitalizados e já devidamente processados e com seus elementos principais reconhecidos. O objetivo final, é, portanto, o de gerarmos o banco de dados do sistema a partir do reconhecimento de seus elementos a partir dos diagramas originais. É claro que esse objetivo parece, a princípio, ser pretencioso, mas, na verdade requer apenas mais estudos e pesquisas para que seja concretizado.

A proposta deste trabalho foi a de mostrar todos os blocos necessários para implementação de um sistema de informação gerado a partir da digitalização de diagramas. Destes blocos foi escolhido um subconjunto que permitiu pelo menos chegarmos à automatização da documentação. A escolha de apenas um subconjunto é devida ao tamanho e complexidade do sistema que indica uma equipe, e não um esforço isolado, para implementação do sistema completo.

Atualmente temos prontos os algoritmos de filtragem, binarização, afinamento e trilha de caminho, e um sistema de digitalização por câmera. Fica faltando, portanto, o algo-

ritmo de segmentação e a interface com um editor gráfico para completar o subconjunto de automatização da documentação.

Para continuação deste trabalho propomos uma avaliação da propriedade da curvatura e a implementação de um algoritmo de segmentação, que permita efetivamente a interface com um editor gráfico existente. Seria interessante, também, fazer testes de digitalização com um "scanner", computar tempos de processamento dos algoritmos e quantidade de informação envolvida, e fazer uma análise estatística das propriedades da imagem.

É sugerida a extensão deste trabalho para o reconhecimento de imagens, onde faríamos o reconhecimento dos componentes do diagrama (linhas de conexão, caracteres e símbolos) através de uma análise sintática. Outra sugestão é a de implementar os algoritmos de binarização, afinamento, trilha de segmento e segmentação em hardware, para acelerar o processamento.

BIBLIOGRAFIA

[1] PAVLIDIS, T., Algorithms for Graphics and Image Processing, Computer Science Press, Rockville, 1982.

[2] WILLIAMS, R., "Image Processing and Computer Graphics", Computer Graphics and Image Processing, vol. 10, pp. 183-193, 1979.

[3] WILLIAMS, T., "Board Level Solutions Open New Territory for Image Processing", Computer Design, pp. 53-66, May, 1, 1987.

[4] STANTON, T., BURNS, D. e VENIT, S., "Page-to-disk Technology: Nine State-of-the-art Scanners", PC Magazine, pp. 128-177, vol. 5, Number 16, September, 30, 1986.

[5] WILLIAMS, T., "Document Processing Moves to The Desktop", Computer Design, pp. 63-70, vol. 28, number 13, july 1, 1989.

[6] BAESELER, F. e HECK, B. DESKTOP PUBLISHING - Editoração Eletrônica, McGraw Hill, 1988.

[7] NEWMAN, W. F., e SPROULL, R. F., Principles of Interactive Computer Graphics, McGraw-Hill, New York, Second Edition, 1979.

[8] PAZ, E. P., e CUNHA, T. N., "LATIM: Laboratório de Tratamento de Imagens", VIII Semicro, Seminário de Microcomputadores, Rio de Janeiro, 1989.

[9] GONZALEZ, R. C. e WINTZ, P., Digital Image Processing, Springer-Verlag, 1977.

[10] NADLER, M., "Document Segmentation and Coding Techniques", Computer Vision, Graphics and Image Processing, pp. 240-262, vol. 28, 1984.,

[11] PRATT, W. K., Digital Image Processing, John Wiley & Sons, 1978.

[12] GONZALEZ, R. C. e THOMASON, M. G., Syntactic Pattern Recognition, Springer-Verlag, 1974.

[13] FU, K. S., Syntactic Pattern Recognition and Applications, Prentice-Hall, Inc., Englewood, New Jersey, 1982.

[14] RAMACHANDRAN, K., "Coding Method for Vector Representation of Engineering Drawings", Proceedings IEEE, pp. 813-817, vol. 28, Number 7, July, 1980.

[15] BLEY, H., "Segmentation and Preprocessing of Electrical Schematics", Computer Vision, Graphics and Image Processing, pp. 271-288, vol. 28, 1984.

[16] SAKAI, T., NAGAO, M. e MATSUSHIMA, H., "Extraction of Invariant Picture Sub-structures by Computer", Computer Graphics and Image Processing, pp. 81-96, vol. 1, 1972.

[17] PAVLIDIS, T., "A Vectorizer and Feature Extractor for Document Recognition", Computer Vision, Graphics and Image Processing, pp. 111-127, vol. 35, 1986.

[18] FAHN, C., WANG, J. e LEE, J., "A Topology-Based Component Extractor for Understanding Electronic Circuit Diagrams", Computer Vision, Graphics and Image Processing, vol. 44, pp. 119-138, 1988.

[19] OKAZAKI, A., KONDO, T., MORI, K., TSUNEKAWA, S. e KAWAMOTO, E., "An Automatic Circuit Diagram Reader with Loop-Structure-Based Symbol Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 331-341, vol. 10, number 3, May, 1988.

[20] FLETCHER, L. A. e KASTURI, R., "A Robust Algorithm for Text String Separation from Mixed Text/ Graphics Images", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 910-918, vol. 10, number 6, November, 1988.

[21] BAIARD, H. S., "Feature Identification for Hybrid Structural/ Statistical Pattern Classification", Computer Vision , Graphics and Image Processing, pp. 318-333, vol. 42, 1988.

[22] JARVIS, J. F., JUDICE, C. N., e NINKE, W. H.,"A Survey of Techniques for the Display of Continous Tone Pictures on Bilevel Displays", Computer Graphics and Image Processing, pp. 13-40, vol. 5, 1976.

[23] JARVIS, J. F., JUDICE, C. N., e NINKE, W. H.,"Using Ordered Dither to Display Continous Tone Pictures on AC Plasma Panel", Proceeding of S.I.D., Vol. 15/4, Fourth Quarter, 1974.

[24] JARVIS, J. F., e ROBERTS, C. S., "A New Technique for Displaying Continous Tone Images on a Bilevel Display", IEEE Transaction on Communications, pp. 891-897, august, 1976.

[25] KNOWLTON, K., e HARMON , L., "Computer Produced Gray Scales", Computer Graphics and Image Processing, pp. 1-20, vol. 1, 1972.

[26] MORRIN, T. H., "A Black-White representation of a Gray-Scale Picture", IEEE Transaction on Computers, pp. 184-186, february, 1974.

[27] DAVIS, L. S., "A Survey of Edge Detection Techniques", Computer Graphics and Image Processing, pp. 248-270, vol. 4, 1975.

[28] SMITH, M. W., e, DAVIS, W. A., "A New Algorithm for Edge Detection", Computer Graphics and Image Processing, pp. 55-62, vol. 4, 1975.

[29] WESZKA, J. S., "A Survey of Threshold Selection Techniques", Computer Graphics and Image Processing, pp. 259-265, vol. 7, 1978.

[30] TING, D., e, PRASADA, B., "Digital Processing Techniques for Encoding of Graphics", Proceedings of the IEEE, pp. 757-769, vol. 68, number 7, july, 1980.

[31] YOKOI, S., TORIWAKI, J., e, FUKUMURA, T., "An Analysis of Topological Properties of Digitized Binary Pictures Using Local Features", Computer Graphics and Image Processing, pp. 63-73 , vol. 4, 1975.

[32] NETRAVALI., A. N., e, LIMB, J. O., "Picture Coding: a Review", Proceedings of the IEEE, pp. 366-406, vol. 68, number 3, march, 1980.

[33] PAVLIDIS, T., "A Thinning Algorithm for Discrete Binary Images", Computer Graphics and Image Processing, pp. 142-157, vol. 13, 1980.

[34] ARCELLI, C., e, DI BAJA, G. S., "A One-Pass Two-Operation Process to Detect the Skeletal Pixels on the 4-distance Transform", IEEE Transaction on Pattern Analysis and Machine Intelligence, pp. 411-414, vol. 11, number 4, april, 1989.

[35] EKSTROM, M. P., Digital Image Processing Techniques, Academic Press Inc, London, 1984.

[36] TEH, C. H., e, CHIN, R. T., "On the Detection of Dominant Points on Digital Curves", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 859-872, vol. 11, number 8, august, 1989.

[37] PARENT, P., e, ZUCKER, S. W., "Trace Inference, Curvature Consistency, and, Curve Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 823-839, vol. 11, number 8, august, 1989.

[38] FAHN, C. S., WANG, J. F., e, LEE, J. Y., "An Adaptive Reduction Procedure for the Piecewise Linear Approximation of Digitized Curves", pp. 967-973, vol. 11, number 9, september, 1989.

[39] DAVIS, L. S., "Understanding Shape: Angles and Sides", IEEE Transactions on Computers, pp. 236-242, vol. C-26, number 3, march, 1977.

[40] FREEMAN, H., e, DAVIS, L. S., "A Corner-Finding Algorithm for Chain-Coded Curves", IEEE Transactions on Computers, pp. 297-303, vol. C-26, number 3, march, 1977.

[41] CIARCIA, S., "Build a Gray-Scale Video Digitizer. Part 2: Digitizer/ Transmitter", BYTE, pp. 129-138, june, 1987.

[42] CIARCIA, S., "Build a Gray-Scale Video Digitizer. Part 1: Display/Receiver", BYTE, pp. 95-106, may, 1987.

[43] CIARCIA., S., "Using The Imagewise Video Digitizer. Part 1: Image Processing", BYTE, pp. 113-119, july, 1987.

[44] CIARCIA, S., "Using The Imagewise Video Digitizer. Part 2: Colorization", BYTE, pp. 117-121, august, 1987.

[45] SPROULL, R. F., "Using Program Transformations to Derive Line-Drawing Algorithms", ACM Transactions on Graphics, pp. 259-273, vol. 1, number 4, october, 1982.

[46] TURBO GRAPHIX TOOLBOX, Owner's Handbook, Borland International Inc., version 1, 1985.

APÊNDICE A

"Dando Nome aos Bois"

A seguir são colocadas várias palavras em inglês técnico que não possuem uma tradução em português que seja de consenso geral.

Thresholding = Binarização

Thinning = Afinamento

Skeletonizing = Esqueletonização

Medial Axis = Eixo Mediano

Shape Analysis = Análise ou Reconhecimento de Forma

Feature Points = Pontos Característicos

Shrinking = Encolhimento

Contour Tracing = Trilha ou Seguimento de Contorno

Papeless Office = Escritório Sem Papel

Desktop Publishing = Editoração Eletrônica

Raster-scan = Varredura

Point Spread Function = Função de Espalhamento do Ponto

OCR (Optical Character Recognition) = Reconhecimento de Caracteres Digitalizados.

CCD (Charge Coupled Device) = Dispositivo Sensor Ótico de alta resolução e grande sensibilidade a luz utilizado nas câmeras modernas.

APÊNDICE B

IMAGENS A, B e C

Nas páginas seguintes são apresentadas as figuras B.1, B.2 e B.3.

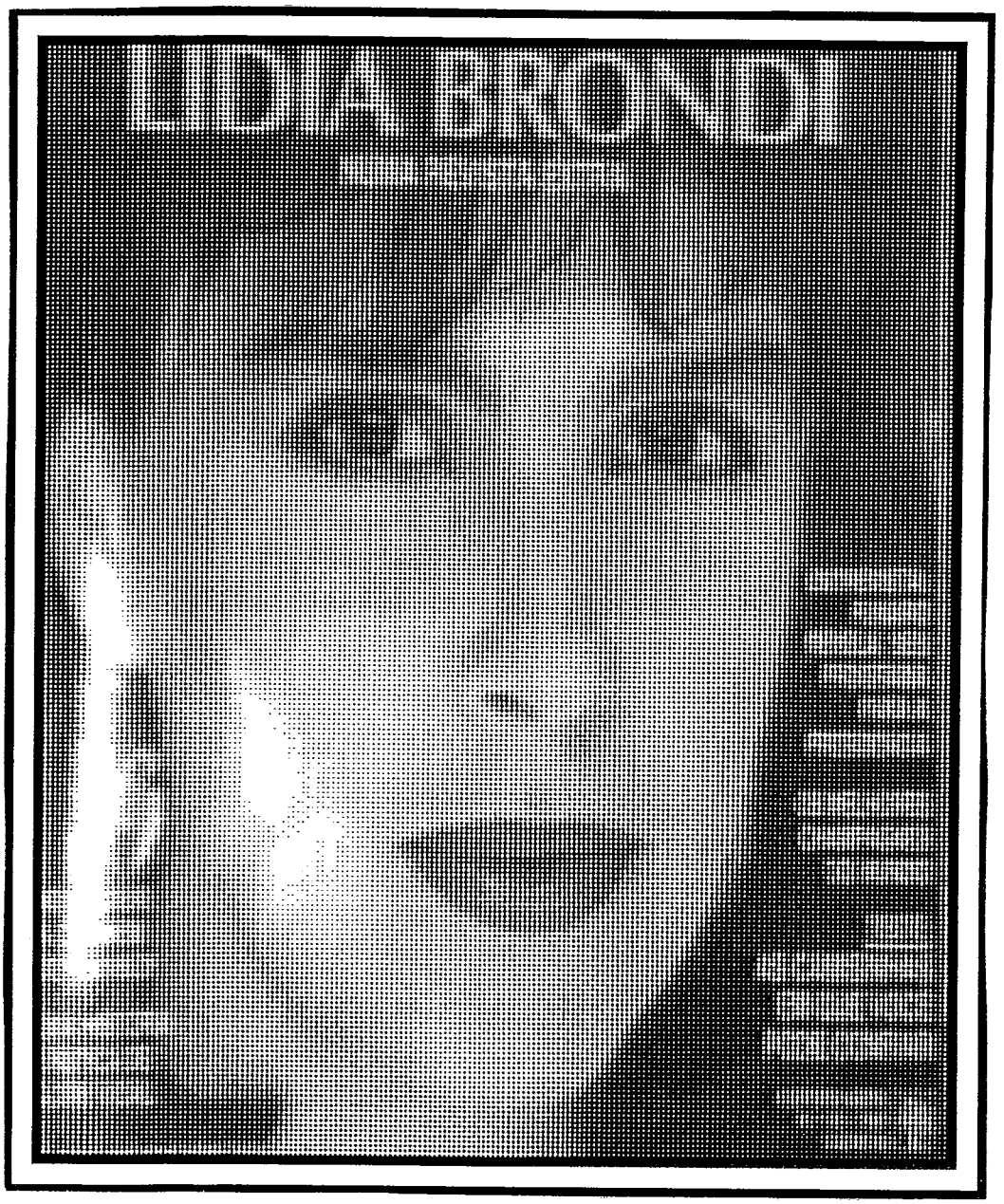

figura B.1 - LÍDIA.PIC

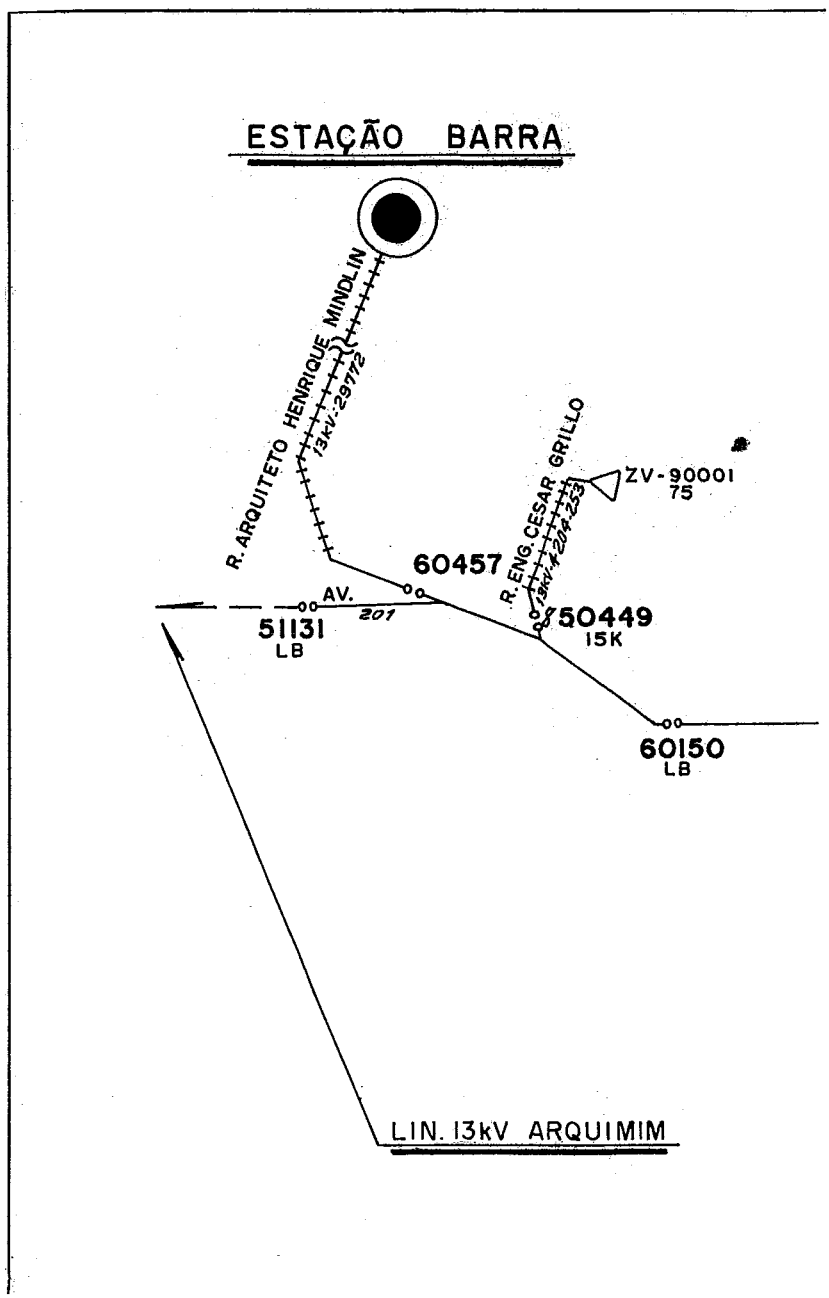

figura B.2 - MAPA.PIC

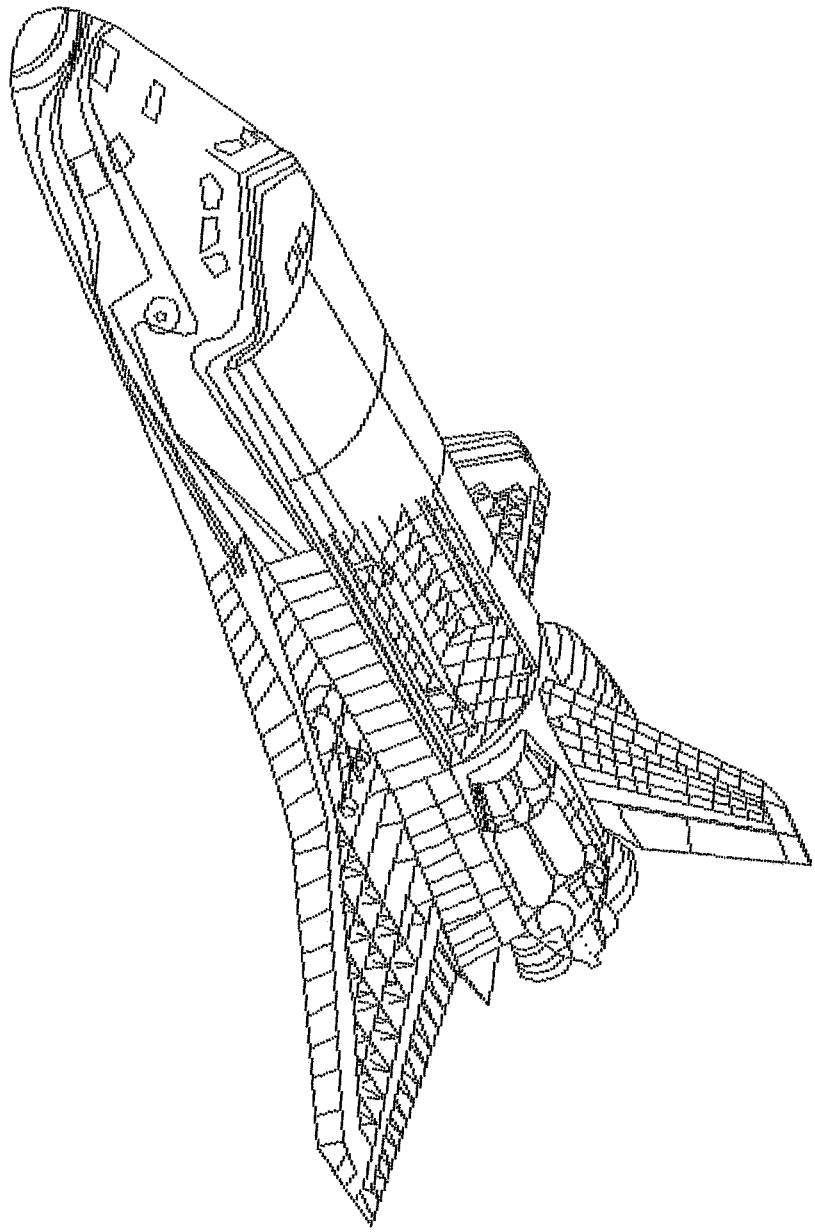

figura B.3 - COLÚMBIA.PIC