Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e Computação

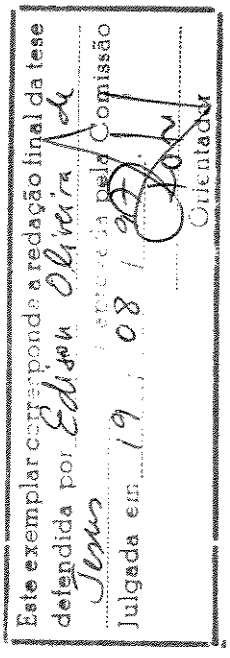
Departamento de Computação e Automação

SRIDE

SISTEMA RECONHECEDOR DE IMAGENS DE DIAGRAMAS ELÉTRICOS

uma tese em

Engenharia Elétrica



por

Edison Oliveira de Jesus

Submetida para requisição do

Grau de Doutor

19 de Agôsto de 1997

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

J499s

Jesus, Edison Oliveira de
SRIDE sistema reconhecedor de imagens de diagramas elétricos / Edison Oliveira de Jesus.--Campinas, SP: [s.n.], 1997.

Orientador: Roberto de Alencar Lotufo
Tese(doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Reconhecimento de padrões. 2. Processamento de imagens. 3. Desenho (Engenharia). 4. Percepção visual. I. Lotufo, Roberto de Alencar. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Membros da Banca Examinadora:

Dr. Roberto de Alencar Lotufo - Orientador
DCA/FEEC/UNICAMP-SP

Dr. Clésio Luiz Tozzi
DCA/FEEC/UNICAMP-SP

Dr. Márcio Luiz de Andrade Netto
DCA/FEEC/UNICAMP-SP

Dr. Sdnei de Brito Alves
ICI/DMC/EFEI/ITAJUBÁ-MG

Dr. Gerald Jean Francis Banon
DPI/INPE-SP

DCA/FEEC/UNICAMP : Departarmento de Controle e Automação da Faculdade de Engenharia Elétrica e Computação da Universidade Estadual de Campinas

ICI/DMC/EFEI/ITAJUBÁ: Departamento de Matemática e Computação do Instituto de Ciências da Escola Federal de Engenharia de Itajubá

DPI/INPE: Departamento de Processamento de Imagens do Instituto Nacional de Pesquisas Espaciais

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

RESUMO

Este trabalho refere-se ao desenvolvimento de um novo método de vetorização de desenhos de engenharia aplicado à diagramas de circuitos elétricos, sem a utilização de afinamento clássico ou obtenção de esqueleto, como normalmente ocorre com os métodos tradicionais. Posteriormente a esta vetorização desenvolveu-se um sistema de reconhecimento dos símbolos que compõem o diagrama de circuito bem como suas interconexões.

A metodologia desenvolvida neste projeto baseia-se na obtenção dos **pontos característicos** da imagem (cruzamentos de linhas, cantos e finais de linhas), os quais podem ser origem dos vetores representativos da imagem. Os resultados obtidos mostram que diferentemente dos processos de afinamento e de extração de esqueleto, esta nova metodologia apresenta as seguintes características: a) o processamento é mais rápido; b) é independente da largura das linhas; c) a obtenção dos pontos de cruzamentos, cantos e fins de retas é facilitada; d) é mais resistente à ruídos tais como a presença de pontos espúrios ou pequenas interrupções de linhas; e) trata a imagem original sem nenhuma perda de informação, proporcionando maior facilidade no processamento para reconhecimento de padrões e interpretação do desenho.

A classificação dos símbolos representativos dos elementos que constituem um circuito elétrico (como por exemplo resistor, capacitor, terra, etc.) é realizada em duas formas

dependendo do tipo do símbolo. Para os símbolos abertos utiliza-se classificação sintática, ou seja, regras específicas que caracterizam os símbolos esperados num diagrama de circuito. Na classificação dos símbolos fechados utiliza-se um método estatístico, levando-se em consideração um vetor de características próprio para os símbolos fechados pertencentes a um diagrama de circuito elétrico.

Apresentamos também uma pesquisa em interpretação de desenhos, reconhecemos linhas pertencentes aos símbolos do desenho, linhas de interconexão e também a grande maioria dos símbolos do desenho.

Por último, o sistema computacional **SRIDE** desenvolvido para realizar o reconhecimento de imagens de diagramas elétricos, está integrado à pacotes comerciais que visam dar maior eficiência aos resultados obtidos: **SPICE** é usado para auto conferência do layout do diagrama obtido no reconhecimento; **CAD** é usado para permitir ao usuário visualizar e modificar o desenho obtido; **OCR** é usado para o reconhecimento dos caracteres extraídos automaticamente do desenho.

Palavras Chaves: Desenhos de engenharia, Reconhecimento de padrões, Vetorização, Interpretação de Desenhos, Estudo de Imagens.

ABSTRACT

This work is concerned to the developing of a new process to vectorize low-level engineering drawings applied to electrical circuits diagrams, without using thinning neither skeleton tracing, as is common in the traditional methods. After this vectorization, was developed a recognizer to the symbols which constitute the diagram as such their interconections.

This project presents a methodology based in finding the feature points of the image (corners, ends of lines and crosses) which can be the origin of the vectors that represent the image. The results show that this method is a new line tracing method, possessing all the functions of the traditional thinning process, and it has some significant superiorities: a) it has faster processing speed; b) it is independent from the line width value; c) it is easy to extract cross points, corners and end lines; d) it has hight ability of resisting noise, such as spurs on lines and small line break; e) it treats the original image with no loss of information, while thinning loses it in the traditional method, it is more suitable for further detail processing, entity recognition and drawing interpretation.

In this project we also do a research on drawing interpretation, we recognize lines which belong to the entities, interconection lines and also some entities of the drawing.

Two forms of classifying symbols which represent the elements of a circuit diagram (like for instance resistor, capacitor, ground, etc.) are used, depending of the symbol type. To the open symbols we use symtatic classification, i. e., specific rules which characterizes this kind of symbol in a circuit diagram. Loop symbols are classified by using stocastic classification, where a feature vector is created to represent each closed symbol of the electrical circuit diagram.

Finally, the computational system **SRIDE** developed to recognize electric diagrams images, has been integrated to some comercial packages, in order to ensure the efficiency of the results: **SPICE** is used to guarantee the layout of the electrical circuit which has been recognized by the system; **CAD** allows the user to view and to modify the resultant drawing; an **OCR** is used to automatically recognize the characters extracted from the drawing.

Keywords: Engineering Drawings, Pattern Recognition, Drawing Vectorization, Drawing Interpretation, Image Understanding.

AGRADECIMENTOS

Aproveito esta oportunidade para expressar meus sinceros agradecimentos ao meu orientador Prof. Roberto de Alencar Lotufo da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, SP, não somente por iniciar e guiar-me nesta pesquisa, mas também pelo grande incentivo e demais ensinamentos a mim dirigidos durante este período de convivência, sem os quais a realização deste trabalho teria sido muito mais difícil.

Gostaria também de expressar meus agradecimentos ao Prof. George Nagy do Electrical and Computer Systems Engineering at the Ressenlaer Polytechinc Institute, NY, USA, o qual me acolheu e orientou com muita dedicação durante os 2 anos de estadia naquele instituto, para realização de grande parte do trabalho.

Sou muito grato também a todos os meus colegas professores do Departamento de Matemática e Computação da Escola Federal de Engenharia de Itajubá, no qual sou lotado, e principalmente ao Grupo de Sistemas de Computação, pelo seu apoio e assistência durante o período de desenvolvimento deste trabalho.

Finalmente, agradeço aos professores Clésio Luiz Tozzi, Márcio Luiz de Andrade Netto, Gerald Jean Francis Banon e Sidnei de Brito Alves pelas sugestões apresentadas, as quais contribuiram de modo significante no aperfeiçoamento deste trabalho.

DEDICATÓRIA

Gostaria de expressar meu mais profundo respeito e gratidão aos meus familiares, principalmente à minha esposa Mariana e aos meus filhos Plínio, Cynthia e Flávio, os quais acreditaram no meu trabalho, colaboraram com muito esforço nos momentos difíceis, incentivaram-me e apoiaram-me em todas as fases de seu desenvolvimento. A eles dedico este trabalho.

CAPÍTULO 1

1. INTRODUÇÃO

1.1 MOTIVAÇÃO

O rápido progresso na indústria de computadores visto nos últimos tempos, fez com que o acesso a essas máquinas por mais e mais pessoas tenha aumentado em grande escala. CPUs estão cada vez mais rápidas e mais baratas, possibilitando que pesquisas de uma forma geral sejam beneficiadas, devido a maior disponibilidade de equipamentos. Entre as aplicações beneficiadas com este progresso estão aquelas voltadas à visão computacional e processamento de imagens. É fácil de perceber que atualmente em nossos escritórios, secretárias estão fazendo uso não mais da velha máquina de escrever, e sim de modernos computadores pessoais, mais fáceis e mais ágeis no desenvolvimento das tarefas diárias. De uma forma mais avançada, sistemas de reconhecimento de caracteres (OCR) e digitalizadores (*scanners*) estão sendo utilizados na tarefa de análise de documentos: um texto não precisa ser mais redigitado ou datilografado, pois agora ele pode ser digitalizado e ter os caracteres reconhecidos e convertidos para uma forma eletrônica de fácil modificação, rápida transmissão e eficiente armazenamento. Também o processamento de desenhos está sendo rapidamente

automatizado através dos pacotes comercialmente acessíveis de CAD. No entanto, a análise de desenhos ainda não avançou tanto quanto as pesquisas dedicadas ao OCR. O reconhecimento automático de desenhos está ainda no estágio de desenvolvimento.

Dois problemas de grande interesse na análise de documentos são: a) conversão automática de um desenho em papel para uma forma eletrônica tal como o CAD; b) armazenamento eficiente desses documentos. Os sistemas computacionais em desenvolvimento hoje em dia têm como objetivo o tratamento de imagens relacionadas com desenhos manuais ou não, podendo envolver mapas, fluxogramas, diagramas lógicos, plantas arquitetônicas, desenhos mecânicos e elétricos [Fahn e outros 1988, Nagasamy 1990, Okazaki 1988 e Vaxiviére 1992].

Hoje a necessidade de conversão de desenhos do papel para um formato eletrônico é muito grande, principalmente quando se trata de projetos de engenharia os quais envolvem um volume muito grande de informações. Em se tratando de desenhos de circuitos elétricos, muitos foram feitos em papel, são antigos mas ainda estão sendo usados e precisam de alterações periodicamente. A conversão automática não somente economiza tempo na entrada de dados como também na revisão do desenho.

Uma vez que o desenho tenha sido convertido para um formato eletrônico, ele pode ser usado de muitas maneiras: inventários podem ser produzidos automaticamente; verificação do desenvolvimento lógico pode ser realizada através da associação dos símbolos do diagrama com funções lógicas; criação e manutenção de uma biblioteca de desenhos, etc.

Esta pesquisa visa o desenvolvimento de um sistema computacional para processar desenhos de diagramas de circuitos elétricos ou eletrônicos, os quais têm como característica principal o fato de pertencerem a uma classe de desenhos que são constituídos de símbolos e linhas de interconexão.

O sistema computacional desenvolvido, o **SRIDE** (Sistema Reconhecedor de Imagens de Diagramas Elétricos), procura incorporar as soluções para os problemas de interesse na área de análise de documentos aplicados à engenharia, ou seja, a conversão automática do desenho e seu eficiente armazenamento e ainda procede o reconhecimento da maioria dos elementos constituintes do desenho. Estamos propondo uma técnica inovadora para a segmentação de símbolos no diagrama de circuitos, o que assegura aumento de confiabilidade, velocidade e produtividade na automação da conversão eletrônica do desenho. O armazenamento é gerenciado por funções de um banco de dados de desenhos, as quais garantem muita flexibilidade na recuperação e manuseio das informações mantidas neste banco.

A Figura 1.1.1 a seguir, mostra a estrutura do sistema computacional desenvolvido. A entrada de dados é constituída de um desenho, no caso um diagrama de circuito elétrico ou eletrônico, a qual é capturada via *scanner* e armazenada num arquivo de dados do tipo **TIFF**. O próximo passo é a binarização da imagem através da escolha de um limiar (*threshold*). Em seguida inicia-se as fases de extração: extração de caracteres, extração de linhas e extração de características da imagem em estudo. Neste ponto, existe a possibilidade do usuário proceder a visualização do desenho e também do reconhecimento dos caracteres extraídos através da utilização de um OCR (reconhecedor de caracteres) disponível no mercado.

Uma vez obtidas todas estas informações, o sistema inicia o processo de reconhecimento dos símbolos que compõem o diagrama elétrico ou eletrônico. Os elementos até este momento não reconhecidos, são os candidatos da próxima fase: o reconhecimento das linhas de interconexão dos símbolos do diagrama.

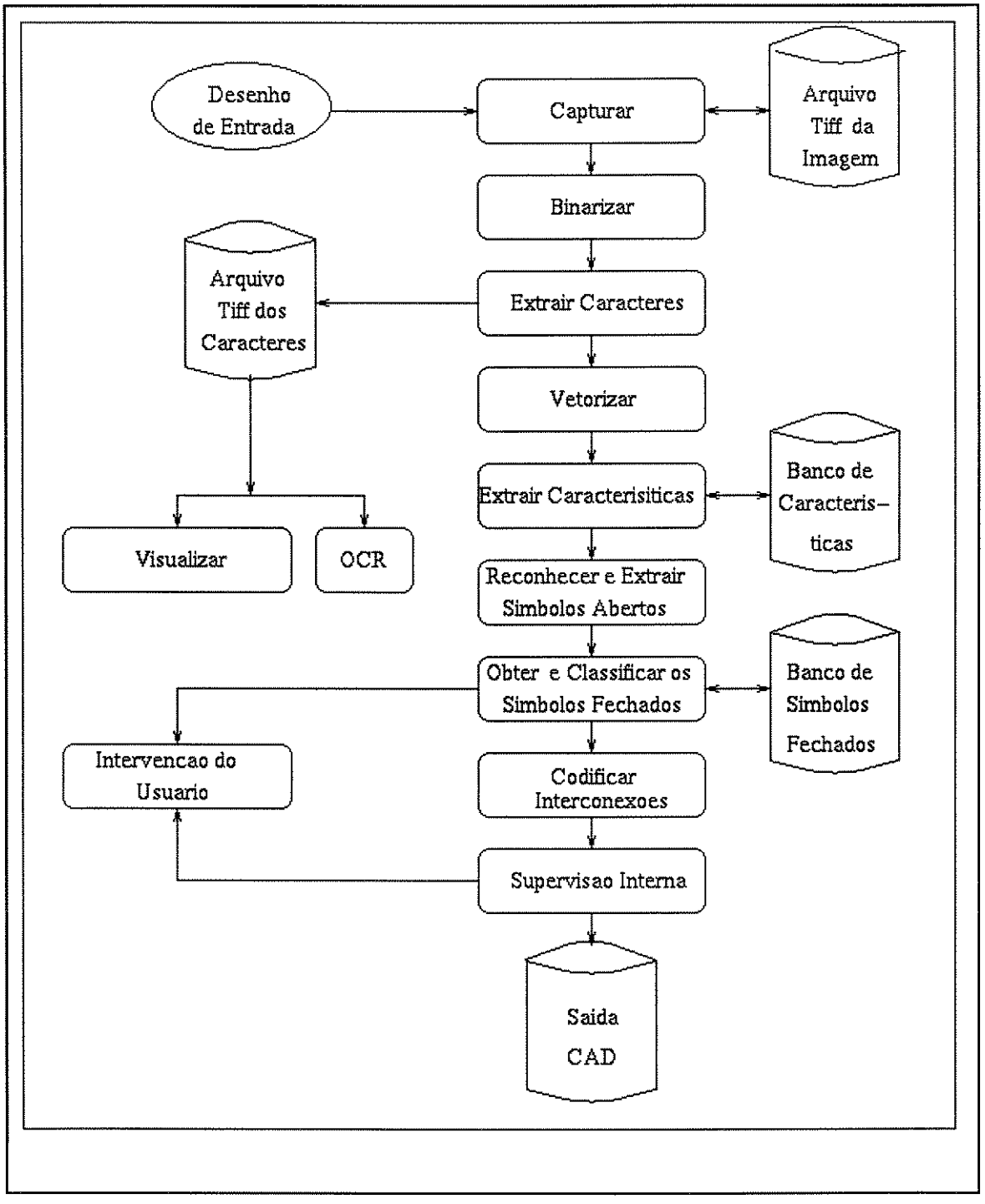


Figura 1.1.1 Arquitetura do sistema SRIDE

Estabelecidos todos os elementos componentes do diagrama, estes são enviados a um supervisor interno para verificação da topologia do circuito que o desenho representa. Nesta pesquisa, o SPICE é o analisador topológico integrado ao sistema para este fim. O usuário interage com o sistema:

a) quando novos símbolos são reconhecidos pelo sistema e este necessita de suas identificações para proceder a inclusão destes símbolos no banco de símbolos;

b) ou sempre que correções se façam necessárias no desenho reconhecido; para isto a saída do desenho pode ser sempre visualizada através de um CAD comercial.

Nesta pesquisa, o sistema SRIDE está integrado aos pacotes comerciais: OmniPage para OCR, TurboCAD para CAD e Spice para análise elétrica do circuito.

1.2 Organização do Trabalho

O trabalho está organizado da seguinte forma: neste capítulo é apresentada a motivação da realização da pesquisa; no capítulo 2 é mostrada a evolução histórica das pesquisas anteriores realizadas na área de análise de desenhos de engenharia; o capítulo 3 descreve os métodos de pré-processamento, vetorização, levantamento de características e classificação de símbolos utilizados nesta pesquisa; o sistema computacional e os resultados são sumarizados e discutidos na capítulo 4; finalmente, conclusões e direções para futuros trabalhos são tratados no capítulo 5.

CAPÍTULO 2

2. REVISÃO BIBLIOGRÁFICA

2.1 INTRODUÇÃO

Tradicionalmente, informações são armazenadas e transmitidas através de documentos e estes geralmente são compostos de textos ou figuras escritos (a mão ou através de um equipamento qualquer) em um pedaço de uma folha de papel. Com o passar do tempo porém, este papel sofre uma degradação natural provocando perda total ou de parte da informação ali armazenada. Este problema fica bem evidenciado no caso das bibliotecas onde grande parte do material fica guardado por longo período de tempo, e além disto é utilizado de diferentes formas por diferentes pessoas, o que introduz maior desgaste ao papel manuseado, acelerando assim seu processo de degradação.

Outro problema relacionado a documentos refere-se ao fator utilização: pode-se ter necessidade de uma simples consulta ao documento ou uma tarefa mais específica, tal como a alteração da informação ali armazenada. Isto leva-nos a considerar também o fator volume de documentos, o que implica numa tarefa extra: a localização de um determinado documento entre os muitos documentos guardados.

Com a utilização do computador, todos estes problemas podem ser solucionados, pois muitas pesquisas têm sido desenvolvidas com objetivo de tratamento de textos, editoração, computação gráfica e desenhos auxiliados por computador. Assim, tarefas como localização, manuseio e manutenção de documentos são realizados por softwares dotados de várias opções, as quais possibilitam ao usuário uma variedades muito grande de tarefas realizáveis com documentos. No início dos anos 80 foi lançada a idéia do *paperless office* , o que agora é uma realidade em muitos ambientes que lidam com documentos. Evidentemente, documentos ainda são impressos em papel, tendo em vista à necessidade de disseminação e mesmo leitura (nem todos os usuários tem acesso ao computador, ou nem sempre o prazer da leitura é o mesmo usando-se o computador). O problema agora é outro: manusear o fluxo de documentos eletrônicos e em papel de forma integrada e eficiente. Uma solução seria tratar um documento eletrônico tal com o escrito no papel, ou seja, possibilitar a leitura e alteração de forma bem confortável ao usuário, podendo-se adicionar a isto tarefas específicas de tal forma a aumentar a produtividade do serviço. Exemplo disto são os editores e processadores de texto, onde o usuário pode buscar e alterar a ocorrência de várias cadeias (*strings*) de caracteres; pode colunizar automaticamente um texto e assim por diante.

Resumindo, o papel continuaria o meio mais confortável para a leitura e modificação, no entanto estas e outras tarefas podem ser compatíveis com a utilização do computador. Desta forma pode-se concluir que o objetivo da análise de imagens de documentos é a interpretação automática da informação visual contida numa imagem em 2D, compreendendo qualquer mistura de textos, gráficos e figuras.

O processamento de textos e o processamento gráfico são duas categorias diferentes tratadas pela análise de imagens de documentos. No processamento de textos o objetivo é o tratamento do *layout* das páginas e o reconhecimento dos caracteres.

Este tipo de análise inclui:

- determinação da inclinação do texto quando da sua captura;
- obtenção das colunas, parágrafos, linhas de textos e palavras;
- reconhecimento dos caracteres.

No processo gráfico faz-se o tratamento de linhas, símbolos e regiões. Para isto as seguintes tarefas devem ser realizadas:

- afinamento de linhas (*thinning*) ;
- detecção de linhas (*fitting*) ;
- detecção de curvas e cantos ;
- reconhecimento de símbolos.

A análise destes componentes geralmente cai dentro do domínio do **processamento de imagens** e **reconhecimento de padrões**.

2.2 PROCESSAMENTO DE IMAGENS

Em essência, o processamento de imagens é parte do processamento de sinais e refere-se aos sinais bidimensionais, onde algumas técnicas utilizadas são comparáveis àquelas utilizadas na revelação de filmes fotográficos. A principal idéia por trás do processamento de imagens é a extração de informações pertinentes.

Existem várias aplicações de grande interesse e vulto hoje em dia para o processamento de imagens: sensoriamento remoto por satélites, processamento de imagens médicas, métodos de compressão de imagens visando sua transmissão e armazenamento, processamento de imagens de documentos, entre outras.

Pode-se extrair informações de tais imagens através do uso de algumas operações básicas:

- operações pontuais:

  Manipulam a intensidade dos pixels na imagem;

  Exemplos:

  * contraste - torna os valores mais escuros para preto e os mais claros para branco, e ajusta linearmente os valores intermediários.
  * densidade - mostra somente os pixels cuja intensidade está dentro de uma faixa específica. Esta operação é usada quando se deseja salientar ou classificar um objeto na imagem.

- operações espaciais:

  São distribuídas em várias categorias;

  * procedimentos de registros - usados para sobrepor imagens e com isto detectar por exemplo, distorções;
  * filtros - procedimentos matemáticos que permitem destacar partes da imagem. Exemplo: ao isolar-se os componentes de alta frequência numa cena, pode-se detectar as bordas dos elementos da imagem.
  * textura - A variação no brilho dos pixels numa região, pode ser importante para o reconhecimento da imagem. A textura é geralmente calculada como sendo o desvio padrão dos vizinhos mais próximos (*nearest neighbours*) ao redor do pixel, e este desvio padrão de cada pixel leva a uma nova imagem.
  * extração de características e classificação - São ferramentas poderosas na análise de imagens. Por exemplo, se certas características da imagem são únicas, o sistema pode "aprender" como achar estas características na

imagem. O **reconhecimento de padrões**, uma ciência que usa estatísticas, geometria e inteligência artificial, pode ser usada na tarefa de extração de características e classificação de objetos.

* compressão de dados - torna-se importante quando a imagem precisa ser transmitida ou quando grande massa de dados deve ser armazenada. A idéia é descrever a imagem usando técnicas estatísticas tais como análise dos principais componentes ou reduzir o tamanho do conjunto de dados, usando técnicas de redução como por exemplo tamanho da corrida.

2.3 DESENHOS DE ENGENHARIA

Os desenhos de uma forma geral podem ser classificados em 3 categorias: mapas, mecânicos e simbólicos [Kong e Rosenfeld 1989]. Mapas e desenhos mecânicos descrevem regiões geográficas ou objetos mecânicos, e por isto eles são desenhados numa escala determinada. Desenhos simbólicos usam símbolos para representar objetos ou conceitos e não usam nenhuma escala.

Geralmente mapas são multi-coloridos e muito complicados pois contêm muitas linhas e regiões em diferentes cores ou textura. Leitores automáticos de mapas já estão sendo pesquisados [Ejiri 1984] com o objetivo de sua conversão do papel para o computador.

Desenhos mecânicos também são complicados, pois contêm projeções 2-D de objetos 3-D, conjunto de dimensões, superposição de objetos, hachureas, etc. A análise de objetos é realizada através do estudo de linhas e regiões na imagem, guiados por informações de bibliotecas ou por regras específicas [Dori e outros 1989, Vaxiviere e outros 1989].

Desenhos simbólicos incluem diagramas de circuitos lógicos, fluxogramas, diagramas de subestações elétricas, diagramas de encanamentos ou instrumentação, diagramas de fiação em geral, PERTS, etc. Dentre estes, os diagramas de circuitos são os mais estudados [Fahn e outros 1988, Fukada 1982 e 1984, Okazaki 1988]. Diagramas de encanamentos e instrumentação [Furuta 1984, Lin 1985] e fluxogramas [Abe 1985] também têm sido bastante estudados.

Embora tenham diferentes objetivos, estes desenhos têm em comum o fato de serem constituídos por símbolos, caracteres e linhas de interconexão. Todos os artigos pesquisados nesta revisão bibliográfica têm em comum as ações de baixo nível, tais como vetorização, segmentação de caracteres ou extração de características. No entanto, as ações de alto nível, os métodos de interpretação, dependem em muito dos tipos de desenhos envolvidos no estudo. Não existe ainda um sistema capaz de manusear qualquer tipo de desenho.

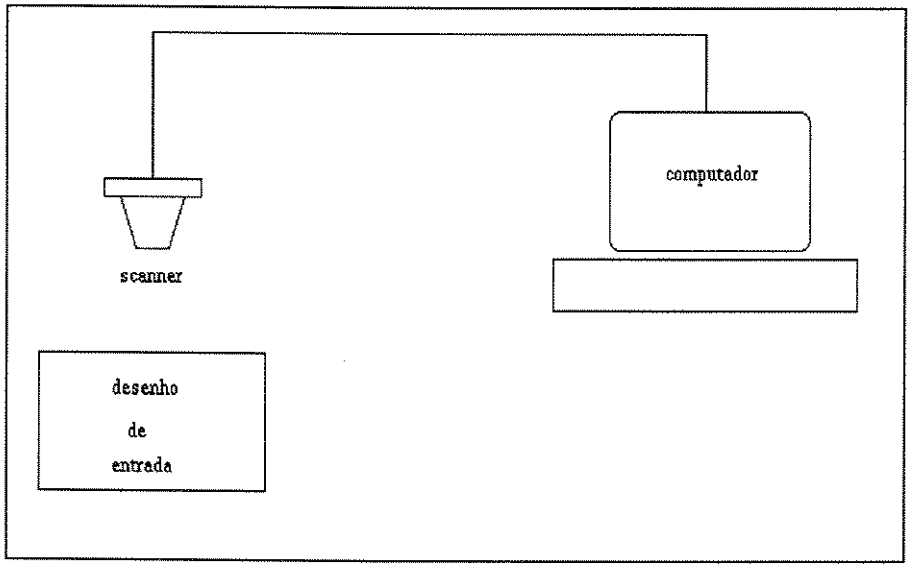


Figura 2.3.1 - Um típico ambiente computacional usado para a entrada de desenhos de engenharia

Nesta pesquisa a entrada de um sistema tal como o mostrado na Figura 2.3.1, pode ser constituída de qualquer espécie de documento. Entretanto esta pesquisa é voltada à

diagramas de circuitos e portanto, para sistemas deste tipo, pode-se ter como entrada de dados documentos tais como desenhos elétricos/eletrônicos os quais podem também possuir textos;

Um **desenho de engenharia** é um produto gráfico. Sua interpretação é baseada em vários conhecimentos do domínio específico, processados por um usuário humano especialista (*expert*) naquele domínio de conhecimento. Em geral, desenhos de engenharia consistem de diversas vistas geométricas de um produto, anotações (valores e identificações) tais como num desenho de diagrama de um circuito elétrico ou eletrônico (veja exemplo de um desenho eletrônico na Figura 2.3.2 a seguir).

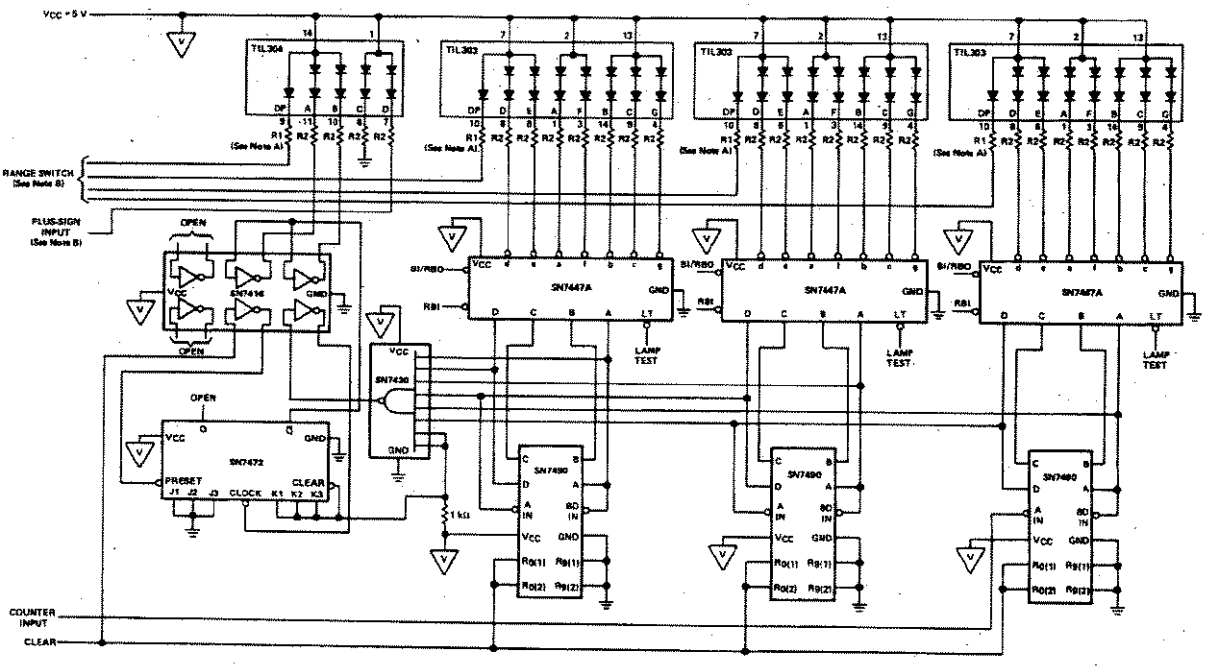


Figura 2.3.2 - Exemplo de um diagrama típico de um circuito eletrônico

Um sistema para interpretação de desenhos de engenharia pode ter várias aplicações práticas além de ser também de grande interesse teórico. Um sistema automático de interpretação poderia ser usado por exemplo para converter um ambiente baseado em papel para um sistema CAD - CAM (Computer Aided Design e Computer Aided Manufacture) ou para eficientemente transmitir ou receber informações de um desenho em papel.

Outra aplicação em potencial é a interface homem - máquina para um sistema CAD - CAM, propiciando um método alternativo para o usuário descrever as formas de um objeto.

Assim como no processamento de textos, o processamento de desenhos de engenharia, mesmos convertidos para uma forma eletrônica pode tornar-se bastante complicado. Desenhos são armazenados eletronicamente num computador, em diferentes formatos e em diferentes tamanhos e ainda em diferentes sistemas. Daí a necessidade de um sistema eletrônico que resolva o problema contornando todas estas limitações.

2.4 VANTAGENS DE APLICAÇÕES EM CAD

A utilização do CAD traz as seguintes vantagens:

- CAD é um sistema interativo para o manuseio de desenhos, cuja utilização já é bem conhecida comercialmente;
- modificações de desenhos já existentes, se freqüentes ou complexos, é tremendamente melhorada usando-se CAD;
- torna-se muito fácil processar grande variedade de desenhos e automaticamente criar bancos de dados CAD;
- é fácil a atualização de dados técnicos;
- é fácil a recuperação de documentos para reprodução;
- a manutenção dos desenhos em geral também é facilitada.

2.5 APLICAÇÕES GERAIS

Existem muitas aplicações em sistemas de análise de documentos: [Lai e Kasturi 1994]:

- processamento de cartas e pacotes no correio com o fim de automatizar o manuseio e ordenação dos mesmos;
- processamento de cheques bancários e outros documentos financeiros;
- leitura de pautas musicais;
- reconhecimento e verificação de assinaturas;
- manuseio de livros em bibliotecas;
- leitura de artigos em jornais e revistas.

O processamento de imagens pode ser aplicado em muitas áreas, dependendo apenas da especialidade do sistema construído, e por isso pode-se gerar diferentes sistemas para diferentes domínios. A maior aplicação para desenhos de engenharia é a conversão de desenhos para o formato de arquivos CAD (projeto auxiliado por computador), cujo principal objetivo é converter arquivos de desenhos já existentes numa forma eletrônica padrão de acesso a estas informações.

2.6 AMBIENTE DE CAPTURA DE DESENHOS

Em geral, o ambiente usado para obtenção de desenhos capturados de folhas de papel, tem a forma apresentada na Figura 2.3.1. O digitalizador possui rotinas internas para a

binarização da imagem as quais podem ser usadas para melhorar-se a obtenção de desenhos limpos sem a introdução de buracos ou sujeiras espúrias. Entretanto, para desenhos de qualidade pobre pode ser necessária a captura da imagem em nível de cinzas e depois determinar um limiar (*threshold*) para sua binarização.

Existem muitos formatos eletrônicos disponíveis para arquivos de imagens: TIFF, FIGS, HPGL, PBM e GIF. Cada um tem vantagens e desvantagens. Um dos mais usados é o formato TIFF, porque permite uma grande variedade de aplicações e também é independente de arquiteturas de computadores, sistemas operacionais e periféricos gráficos. Ele manuseia imagens em branco e preto, níveis de cinza e coloridas, permitindo ao usuário ajustar os periféricos tais como digitalizador, monitor e impressora, à características únicas. Além disto, TIFF suporta muitos tipos de compressão de arquivos.

2.7 REVISÃO BIBLIOGRÁFICA

2.7.1 DIAGRAMAS ESQUEMÁTICOS

As aplicações de reconhecimento de padrões em diagramas esquemáticos, são apresentadas cronologicamente nos seguintes trabalhos: [Masui e outros 1980], [Bunke 1982], [Fukada 1982], [Shimizu e outros 1982], [Takagi e outros 1982], [Sato e Tojo 1982], [Pavlidis 1982], [Bley 1984], [Fukada 1984], [Furuta e outros 1984], [Groen e Munster 1984], [Okazaki e outros 1988], [Fahn e outros 1988], que serão comentados a seguir:

- [Masui e outros - 1980] descrevem um algoritmo e uma máquina usados para o reconhecimento de padrões em circuitos escritos a mão. Os padrões são desenhados numa seção de papel usando-se certas regras de desenvolvimento. Baseados em vários processamentos de imagens uma máquina foi desenvolvida para processar imagens usando-se operações paralelas.

  Em cada circuito desenhado, os símbolos representam os elementos do circuito lógico e as linhas representam as interconexões entre os símbolos. O algoritmo desenvolvido usa o *hardware* de uma máquina a qual pode reconhecer em alta velocidade, todos os objetos padrões de um circuito. O processo consiste de 3 tipos de processamento de imagem: reconhecimento de símbolos, medida da espessura de linhas e representação da linha. Cada processamento é executado como uma unidade de uma pequena área retangular da imagem de entrada.

  O circuito é desenhado sob algumas regras que definem o desenvolvimento da máquina, tais como: direções das linhas, marcas para símbolos, tipos de papel onde o circuito é desenhado e espessura das linhas.

  O reconhecimento dos símbolos é feito em 4 passos: determinação da região de processamento; classificação da região; discriminação do símbolo e conexão entre regiões. A representação das linhas é feita em 3 passos: determinação da região de processamento, representação das linhas na região local e representação das linhas na região global.

- [Bunke - 1982 (1)] introduz uma abordagem para reconhecimento sintático de padrões onde uma gramática é usada para gerar uma ferramenta. Duas classes

de diagramas são estudadas: diagramas de circuitos e fluxogramas. A tarefa em cada caso é a extração de uma descrição da imagem de entrada fornecida ao sistema.

Neste trabalho o diagrama de circuitos é fornecido ao sistema de interpretação na forma de vetores, ou seja, o pré-processamento e a extração de linhas não são executados. Os segmentos representativos do diagrama de circuito são transformados em grafos, assim como o resultado da interpretação da entrada.

A novidade apresentada neste trabalho diz respeito a forma de se processar a imagem de entrada. Nos métodos convencionais um *parser* de uma dada gramática avalia a representação dos padrões apresentados, enquanto que o autor propõe a obtenção dos mesmos resultados aplicando-se aos padrões apresentados, uma seqüência programada de produções de acordo com fluxogramas, admitindo neste caso as mesmas características de trabalho de um diagrama de circuito.

- [Bunke -1982 (2)] compara três diferentes algoritmos que manuseiam somente símbolos e também descreve um sistema que automaticamente extrai uma discriminação de um diagrama de circuito usando uma árvore de decisão contendo o conhecimento *a priori* sobre os símbolos.

  O problema é atacado em 3 diferentes maneiras: relaxação probabilística, árvore de decisão e gramática com grafo programado. O primeiro método é baseado em princípios numéricos enquanto que as outras duas abordagens pertencem ao campo do reconhecimento sintático de padrões.

- [Bunke - 1983 (3)] apresenta um sistema para interpretação automática de diagramas de circuitos usando árvores de decisão, as quais contêm o conhecimento *a priori* sobre a forma dos símbolos constituintes do circuito. O controle da interpretação das linhas é realizado pela árvore de decisão enquanto que a interpretação de textos é controlada por ambos, os resultados da interpretação das linhas e um conjunto de autômatos finitos de estado que definem as denotações.

  A interpretação de linhas é realizada em 2 etapas: busca de conexões e reconhecimento de símbolos. Ambas as rotinas são alternadas até que todas as linhas sejam interpretadas. A interpretação de textos também é realizada em 2 passos: concatenação de simples caracteres em cadeias; e interpretação destas cadeias de caracteres. Os resultados obtidos são mostrados graficamente através da visualização do diagrama interpretado.

- [Fukada - 1982] descreve um sistema que detecta linhas, cruzamentos de linhas, junções L e símbolos lógicos. Entretanto, os diagramas de circuitos lógicos precisam ser desenhados sob rígidas condições a fim de se permitir a aplicação das regras do sistema.

  As linhas são detectadas através de um sensor de linhas. Quando este sensor de linhas detecta uma mudança particular no sinal da linha, estas mudanças são analisadas através do esqueleto da imagem. A conectividade das linhas é estudada via afinamento das áreas onde as mudanças ocorreram. Finais de linhas são examinadas contra os pinos de entrada dos elementos a fim de reconhece-los como portas.

Os diagramas devem ser desenhados sob algumas regras: usar somente linhas retas; símbolos lógicos devem ser simétricos às linhas de entrada ou saída; linhas têm espessuras definidas; símbolos devem ter limites de tamanhos, etc.

O algoritmo primário é composto de 4 rotinas: busca de linhas (*tracking*); processamento das áreas significativas; processamento de símbolos lógicos e detecção do ponto inicial para a busca de linhas.

- [Shimizu - 1982 e outros] descrevem um sistema no qual um diagrama lógico de circuito escrito a mão é automaticamente lido e os dados são descritos usando-se vetores de tal forma que um sistema computacional possa processá-los. O processo por eles apresentados é baseado em extração de características, descrição de formas e interpretação das características.

  Embora sejam escritos a mão, os diagramas de entrada devem obedecer algumas regras de construção tais como: a necessidade de uma grade referência para os elementos do diagrama; a representação dos 36 tipos de símbolos alfanuméricos permitidos é descrita pelas suas posições e códigos num diagrama de circuito impresso. Eles usam *chain code* para representarem linhas e arcos e também usam estruturas de dados hierárquicas, dinâmicas e relacionadas em operações de processamento interativo. O processamento das linhas dos desenhos é realizado com os dados originais em 8 bits de níveis de cinza, permitindo maior confiabilidade na detecção das linhas apagadas ou pobres, tarefa de difícil realização se o afinamento fosse o método escolhido.

  Segmentos de figuras são definidos através dos respectivos *chain codes* e através do conhecimento das relações entre os segmentos conectados,

transforma-se os *chain codes* em segmentos de linhas e arcos.

- [Sato e Tojo - 1982] descrevem uma abordagem para reconhecimento e compreensão computacional de diagramas escritos a mão. Tal sistema gera a descrição de um dado diagrama através de um conjunto de segmentos de linhas.

  O processo de interpretação de diagramas segue o seguinte esquema: adquirir a imagem do diagrama escrito a mão através de um digitalizador, detectar as primitivas do desenho tais como segmentos de linha, arcos, círculos, etc.; gerar descrições das relações geométricas entre as primitivas, usando regras de ligação para uma dada classe de diagrama; identificar símbolos usando seus modelos; gerar um desenho do resultado interpretado.

  As regras de ligações entre as primitivas constituem o conhecimento *a priori* para uma dada classe de figuras. Por exemplo, uma porta AND pode ser detectada através da verificação da presença das primitivas arco, corda de arco e segmento de reta. Desta forma, mudando-se as regras de ligação entre as primitivas pode-se mudar o domínio do conhecimento do sistema, tornando-o apto ao reconhecimento de outros objetos.

- Um sistema experimental para codificar vetores e arcos através de afinamento, mapeamento em grafos, ajustes, eliminação de ruídos, e reconhecimento de símbolos e caracteres é descrito por [Pavlidis - 1982]. Ele descreve uma abordagem para separar textos de não textos assim como um pós processamento para remover serifs.

  O processamento de textos e informações gráficas é realizado por um

sistema que converte as imagens a serem processadas em grafos cujos elementos são vetores ou arcos. Este sistema é constituído de 5 partes: afinamento da imagem binária de tal forma que na imagem resultante cada pixel escuro tenha poucos vizinhos escuros; mapeamento da imagem afinada em um grafo de tal forma que este grafo não tenha nenhum nó com grau 2 (aquele com exatamente 2 arcos conectando-se ao nó); os elementos do grafo são aproximados para segmentos de reta ou arcos; o grafo é analisado e características tais como laços (*loops*) - veja definição no ítem 3.4.1 - ou linhas retas, etc. são identificadas; subgrafos são classificados na busca de textos ou símbolos.

- [Bley - 1984] apresenta um algoritmo de um passo para segmentação de uma imagem binária em componentes primários (grupos de pixels pretos conectados).

  A segmentação da imagem resulta num grafo representativo da imagem binária. Cada nó do grafo representa um componente primário. Cadeias de caracteres no desenho são localizadas computando-se os componentes conectados, isto é, subgrafos conectados e depois fazendo-se o agrupamento (*cluster*) dos pequenos componentes conectados. Linhas são computadas em dois diferentes métodos. Primeiro, os componentes primários são juntados a um conjunto de linhas classificadas o qual descreve o conjunto de linhas dominantes do desenho. Depois, um sistema de produções é utilizado para analisar os detalhes no conjunto de linhas dominantes.

  Os subgrafos são armazenados numa memória de trabalho do sistema de

produção. A cada estágio, durante a execução do programa, as condições das regras são avaliadas e desta forma os laços são detectados.

- [Furuta e outros - 1984] descrevem um sistema que reconhece linhas representativas de encanamentos em diagramas instrumentais. Eles desenvolveram métodos de extração de informações sobre as interconexões contidas nestes diagramas para a construção do respectivo desenho.

  Os desenhos de entrada são manuais e os seguintes módulos foram desenvolvidos para o sistema de reconhecimento: extração de caracteres; desenho de linhas e segmentação de pequenos símbolos; reconhecimento de pequenos símbolos; reconhecimento de caracteres; reconhecimentos de símbolos grandes e confecção do desenho.

  A imagem de entrada é armazenada usando **representação codificada** (*code representation*), ou seja, códigos de corridas dos pixels pretos (ponto inicial da corrida e tamanho) .

  As linhas são detectadas, primeiro através do seu ponto inicial de busca, seguido da busca do ponto final da linha. A detecção destes pontos é feita mediante a observância de regras definidas *a priori*.

  A segmentação de símbolos também é feita considerando-se corridas de pixels dentro de uma área limitada. O reconhecimento de símbolos é feito baseado na extração de características geométricas dos pontos pertencentes aos símbolos, tais como número de pontos iniciais, número de pontos finais, número de interseções, número de linhas, etc.

- [Groen e Munster - 1984] descrevem um método para análise de diagramas esquemáticos de tal forma que eles possam ser usados diretamente em sistemas de bancos de dados com o formato de CAD - CAM. Na metodologia apresentada por eles, é usado uma abordagem *top down* , ou seja , investiga-se primeiro a topologia global da imagem usando operações binárias de vizinhanças local e depois disto os diferentes objetos achados são classificados. Eles usam um *hardware* especial para realizar as operações binárias tais como erosão, dilatação e esqueletização. A extração de linhas e a separação de textos da imagem são realizadas utilizando-se componentes conectados, de tal forma que o maior componente na imagem assume-se como parte das linhas do desenho e o menor componente assume-se como texto. Desta forma, o tamanho do componente é o parâmetro de decisão.

  Afinamento de linhas, obtenção de laços e classificação probabilística são as chaves para a detecção dos objetos na imagem. O único parâmetro utilizado no trabalho para a detecção dos objetos na imagem é o tamanho do objeto. As interconexões são obtidas através da deleção na imagem dos objetos já classificados.

- [Okazaki e outros - 1988] descrevem um método para reconhecimento de símbolos que possuam laços num diagrama de circuito. O método consiste em dois processos: segmentação de símbolos e identificação de símbolos. Eles usam heurísticas a fim de mediar a escolha entre casamento (*template matching*) e extração de características e também uma estratégia de controle de decisão é levada em consideração a fim de completar a análise.

Considerando a grande incidência de laços que constituem os símbolos num diagrama de circuito, os autores desenvolveram um método para o reconhecimento dos mesmos, baseados em que símbolos podem ser compostos por um ou mais laços primitivos, os tamanhos, direções e posições dos símbolos são fixos e símbolos são interconectados através de linhas.

Num primeiro estágio do processo, as regiões mínimas para análise são isoladas nas regiões onde um dado laço é candidato a ser um símbolo. Os símbolos são verificados contra as características pré definidas para os símbolos possíveis num diagrama de circuitos.

O objetivo do próximo estágio do processo é calcular o tipo exato do símbolo conhecido na sua região mínima para análise. Esta tarefa é executada usando-se o casamento de padrões. Entretanto em alguns casos quando a diferença entre as características nos padrões é muito pequena, este método não é muito conveniente e daí a necessidade de utilização de outro método para classificação. Neste caso foram usadas várias características da imagem para cada região mínima: número de linhas conectadas, número de agrupamentos, características locais, etc. A mediação é levada em conta por heurísticas aplicadas hierarquicamente às imagens.

Para que o leitor automático funcione, foi necessário definir regras para o desenho dos diagramas: necessidade de réguas para traçar linhas e padrões (*templates*) para traçar símbolos; limites nos tamanhos dos caracteres nas direções possíveis; limites nos tamanhos mínimos para segmentos de retas e na distancia mínima entre elementos. Quando toda a imagem é reconhecida, os dados são transmitidos a um sistema CAD para análise pelo usuário.

- [Fahn e outros - 1988] descrevem um sistema automático de interpretação para diagramas de circuitos eletrônicos baseados em contexto topológico. Um algoritmo usando aproximações lineares de heurísticas foi usado para um primeiro reconhecimento de segmentos de retas na imagem. Também agrupamentos são usados para a extração de componentes.

  O sistema de interpretação de circuitos eletrônicos varre a imagem do circuito segundo o esqueleto dos objetos. Cada objeto no diagrama de circuitos é composto de um conjunto de segmentos os quais são delimitados por pontos característicos que são detectados durante a varredura inicial. Durante a varredura do diagrama de circuito, os elementos que constituem cada símbolo do circuito ou caracter são combinados num agrupamento (*cluster*) a fim de se analisar o contexto dos segmentos da figura dentro de certas limitações. Baseados em modelos para os componentes essenciais de um diagrama, os agrupamentos relacionados são identificados como símbolos ou caracteres. Se os segmentos da figura são excluídos dos agrupamentos, à eles são atribuídas as características de linhas de interconexão.

  Os elementos de figura são obtidos da imagem binária afinada, usando-se funções de vizinhança, as quais definem se um dado pixel constitui um ponto final, uma linha ou um cruzamento. O processo de reconhecimento constitui-se da classificação simples dos elementos de figura baseado em suas formas.

  Um método de melhor pesquisa relacional, o qual emprega técnica de pesquisa de primeira profundidade sobre regras especificadas, é utilizado para extrair os componentes essenciais da imagem binária (informações descritivas da imagem tais como arcos ou laços).

Grafos e uma gramática sintática são usados para gerar as sentenças que representam estes componentes essenciais, os quais uma vez extraídos são grupados em 3 categorias em termos de símbolos, caracteres ou linhas de interconexões a fim de serem interpretados por funções subsequentes no processamento do sistema.

2.7.2 DESENHOS DE ENGENHARIA

Alguns sistemas foram desenvolvidos para trabalharem com desenhos de engenharia e são mostrados cronologicamente nos seguintes trabalhos: [Clement 1981], [Bunke e Allerman 1982], [Haralick e Queeney 1982], [Ejiri e outros 1984], [Karima e outros 1985], [Dori 1989], [Vaxiviere e Tombre 1989], [Kasturi e outros 1990], [Nagasamy e Langrana 1990], [Filipski e Flandrena 1992], [Yu e outros 1994].

- [Clement - 1981] descreve um experimento para a representação em vetores de segmentos de linhas em desenhos. O desenho é primeiramente capturado com uma determinada resolução e depois usa-se uma resolução mais alta para a inspeção detalhada de linhas finas que se encontram muito juntas umas as outras. Um esquema de processamento seqüencial é usado, no qual a limiarização (*thresholding*) e a largura de linha são levados em conta de medida a medida. Áreas não interpretadas são marcadas no banco de dados. O principal problema ocorre nas conexões de linhas e em alguns pequenos agrupamentos ou bolhas. Tal estratégia é similar àquela produzida por sistemas CAD.

A estratégia utilizada pelo autor assume que a posição do desenho dentro da área do digitalizador é conhecida, e uma descrição das dimensões das margens do desenho é dada. Desta forma toda a análise da imagem é concentrada nesta área de trabalho. Esta análise consiste no reconhecimento de caracteres aplicado ao título, número do desenho e outras anotações dentro do desenho. O método utilizado faz uma classificação inicial do desenho em áreas contendo nenhuma linha e áreas contendo algumas linhas que serão futuramente analisadas. O resultado é um conjunto de pontos contendo medidas de linhas retas ou arcos.

- [Bunke e Allerman - 1982] apresentam um sistema para reconhecer símbolos e conexões num diagrama de circuito usando relaxação probabilística. A idéia principal é designar um vetor com a probabilidade das possíveis interpretações para cada vértice de um diagrama de circuito. Um vértice é o local onde segmentos de linha encontram outros segmentos através de seus extremos. As probabilidades são sucessivamente mudadas a fim de se obter um único e consistente valor que as representam.

  Neste tipo de abordagem, conhecimento baseado em relaxação, o conhecimento *a priori* é expresso através de coeficientes que definem a compatibilidade de interpretação da imagem nas posições de vizinhança. Neste caso não é feito nenhum casamento imediato entre o conhecimento *a priori* e as partes da imagem de entrada. Ao contrário, uma interpretação inicial da imagem é estimada através de passos prévios e depois é refinada sob a influencia dos coeficientes de compatibilidade. O processo de relaxação aplicado a diagramas de circuitos é baseado na configuração dos segmentos de linhas que se

conectam através de um vértice. Como resultado obtém-se um vetor contendo as probabilidades de interpretação correta para cada vértice que compõe os segmentos de linhas na imagem.

- [Haralick e Queeney - 1982] desenvolveram um algoritmo que possibilita o computador descrever objetos em 3D, em desenhos de engenharia decompostos de 2 a 6 vistas 2D ortogonais. Eles mostram que a descrição do desenho de engenharia consiste em resolver o seguinte problema: obter a equação de superfície para cada face, baseados nos segmentos de linhas que constituem o desenho de engenharia. Desta forma, a análise de um objeto consiste em considerar cada objeto do desenho como sendo constituído de 3 conjuntos de partes componentes: um conjunto de pontos, um conjunto de linhas e um conjunto de faces.

  Regras estabelecem o relacionamento entre os conjuntos. O processo de compreensão do desenho consiste em identificar cada face do objeto e as coordenadas de cada vértice de cada face. Uma árvore de pesquisa dividida em 4 partes é utilizada para a realização desta tarefa.

- [Ejiri e outros - 1984] descrevem métodos de análise estrutural para reconhecimento de desenhos de engenharia e mapas. Estes métodos de análise estrutural também têm sido utilizados para reconhecer caracteres e símbolos em diagramas de circuitos lógicos, diagramas de partes mecânicas e mapas topográficos. Alguns algoritmos importantes foram implementados em *hardware* (exemplo: digitalizadores automáticos). Basicamente o método

utilizado busca por pixels de mesma cor a fim de se determinar o vetor mais longo que contém os pixels desta determinada cor. Após todos os pixels serem investigados, obtém-se as coordenadas dos extremos de todas as retas da imagem.

Baseados nestas retas, o reconhecimento de linhas e figuras é realizado. Aplicações em desenhos de circuitos , diagramas lógicos e mapas são apresentados. Os resultados são obtidos através de uma lista de vetores prontos para uso de sistemas CAD.

- [Karima e outros - 1985] apresentam um estudo que explora a captura automática de desenhos de engenharia. Eles obtiveram os resultados através de problemas práticos e baseados na revisão da literatura, eles concluíram que embora avanços substanciais tenham sido feitos em direção à leitura automática de desenhos, este assunto ainda continua a ser um desafio para a tecnologia atual.

- [Dori - 1989] apresenta um algoritmo para reconhecimento de dimensões em desenhos de engenharia o qual emprega abordagem de conhecimentos de geometria e reconhecimento sintático junto com um autômato específico determinístico finito (DFA). Primeiro o problema de distinguir objetos dentre linhas é atacado. Depois, o conjunto de dimensões, seus componentes, tipos e variedades são definidos e ilustrados. A metodologia empregada consiste em reconhecer dimensões em desenhos de máquinas que integram um autômato finito, uma tabela contendo a geometria dos elementos do diagrama e

descritores de linhas (comprimento, largura, tipo inteira ou tracejada, etc.). Primeiro, é realizada uma classificação automática de linhas e objetos da imagem com ênfase na complexidade introduzida pelas linhas de dimensões, as quais descrevem as dimensões das projeções das linhas. Depois os conjuntos-dimensões (entidade gráfica num desenho de máquina que estabelece a medida entre 2 projeções de contorno de um objeto) e seus elementos, são obtidas baseadas numa classificação funcional. Os descritores de linhas esperados são comparados com aqueles obtidos e então o reconhecimento é verificado. Havendo correspondência, o reconhecimento foi bem sucedido.

- [Kasturi e outros - 1990] descrevem um sistema automático para interpretação de linhas em desenhos. Este sistema foi projetado para gerar uma descrição do conteúdo de um desenho de linhas em papel, em termos de várias primitivas gráficas e suas relações espaciais.

  O algoritmo utilizado para desenvolvimento do sistema descrito neste trabalho, emprega transformada de Hough aos centróides dos componentes conectados da imagem. Após um afinamento na imagem, linhas são segmentadas em linhas retas e linhas curvas. Os segmentos de linhas e suas interconexões são analisados a fim de se localizar os laços redundantes mínimos os quais são adequados para gerar uma sucinta descrição dos gráficos. Tal descrição inclui a localização e atributos das formas poligonais simples, círculos e linhas de interconexão e também uma descrição da relação espacial entre eles.

  O sistema gera uma saída contendo a descrição dos objetos que constitui o desenho.

- [Nagasany e Langrana - 1990] apresentam um método para pré-processamento e vetorização de imagens capturadas e digitalizadas de desenhos de engenharia para transferência dos resultados para bancos de dados CAD - CAM. Os passos de pré-processamento incluem remoção de ruídos, enchimento ordenado, segmentação da imagem, afinamento de linhas e extração dos contornos. São apresentados algoritmos para conversão de *raster* para vetores capazes de reconhecerem linhas retas, círculos, arcos e seções cônicas.

  A metodologia é compreendia nos seguintes passos:

  ⇒ pré-processamento da imagem de desenhos obtidas da digitalização do *bit map* a fim de eliminar ruídos e outras características indesejáveis;

  ⇒ segmentação da imagem, separando linhas finas das linhas grossas ou áreas cheias (tais como ocorre nas setas);

  ⇒ extração das linhas da imagem afinada;

  ⇒ extração dos contornos das regiões cheias para reconhecimento de símbolos tais como as setas, importantes para separar os conjuntos de dimensões dos objetos reais na imagem;

  ⇒ vetorização das linhas.

- [Filipski e Flandrena - 1992] apresentam uma discussão sobre os diferentes aspectos e necessidades relativos ao problema da conversão CAD, e descreve a arquitetura geral e algoritmo para um sistema CAD. Eles também descrevem alguns subsistemas, tais como: digitalização, vetorização, reconhecimento de textos, processamento de contexto e edição para limpeza para um particular sistema de conversão, o GTX5000.

- [Vaxiviere e Tombre - 1989] desenvolveram um sistema chamado *CELESSTIN* o qual integra vários módulos em um sistema de interpretação baseado em quadro negro (*blackboard*). Este sistema vetoriza as partes gráficas e monta as linhas resultantes em blocos. Os resultados são transferidos para um sistema CAD. Neste sistema, vetores, blocos e entidades são unidades estruturais para as quais são definidas uma sintaxe para combinar estruturas de baixo nível em alto nível. No nível de semântica são introduzidos conhecimentos de engenharia mecânica. Desta forma o sistema apresentado demonstra capacidade de usar os conhecimentos de engenharia mecânica para ler e desenhar documentos técnicos.

- [Yu e outros - 1994] apresentam um projeto para converter *bitmaps* de uma classe de desenhos de engenharia para uma representação de alto nível. Usando um pacote comercial de software, as imagens binárias capturadas são manualmente pré processadas a fim de se remover caracteres e cadeias de caracteres, depois afinadas e então vetorizadas. Depois toda a segmentação é realizada baseada em algumas propriedades genéricas de desenho de fluxogramas e diagramas.

2.8 PRINCIPAIS ARTIGOS DE REFERÊNCIA

O desenvolvimento deste trabalho de tese foi embasado principalmente em informações colhidas nos seguintes trabalhos:

- [Okazaki - 1988] onde a idéia de segmentação de laços pode ser largamente utilizada no desenvolvimento do sistema computacional objeto desta pesquisa, uma vez que eles são encontrados em grande quantidade em diagramas de circuitos eletrônicos.

- [Kasturi - 1980] tem excelentes informações para o processamento de documentos, principalmente quando se deseja segmentá-los em imagens de textos e imagens gráficas.

- [Bunke - 1982] seu trabalho é importante pois passa a idéia de como usar o conhecimento *a priori* sobre os elementos constituintes de um diagrama de circuito.

2.9 SISTEMAS COMERCIAIS DISPONÍVEIS

Nos EUA podemos encontrar softwares comerciais relacionados com o propósito desta pesquisa porém nenhum deles têm o mesmo objetivo de nossa pesquisa, ou seja, reconhecer os elementos envolvidos num diagrama de circuito. Dentre estes softwares, tivemos a oportunidade de experimentar os seguintes:

1) **PROTEL FOR WINDOWS**: desenvolvido pela Protel Technology Inc., San Jose, CA.

   É um software para desenvolvimento de circuitos impressos (PCB - Printed Circuit Board) baseado em ambiente MS Windows. Possui edição de múltiplas

folhas e biblioteca de símbolos; produz como saída o desenho na tela do circuito impresso desejado.

2) **IsSPICE 3 SIMULATOR**: distribuído pela Intusoft, San Pedro, CA.

É um simulador de circuitos. Ele avalia o comportamento do circuito antes de sua construção. O pacote na realidade é constituído de 3 softwares: o ICAPS que faz a simulação propriamente dita; o SPICENET é um programa para entrada de esquemáticos que gera uma lista (*netlist*) de entrada para o SPICE; o PRESPICE contém uma biblioteca com modelos de elementos para o SPICE.

3) **RASTATION R2V** e **CAD OVERLAY**: desenvolvidos pela Image System

R2V é um vetorizador que automaticamente converte toda ou parte de imagens raster digitalizadas, para o formato CAD. CAD OVERLAY permite que a imagem *raster* e a vetorizada sejam vistas simultaneamente, preservando-se escalas e correlação entre os elementos. Possuem versões para DOS, Windows e Unix.

4) **LEHDAR PCB-CAD**: desenvolvido pela Lehdar Systems Corp., Blaine, WA.

É um sistema integrado utilizado para capturar e extrair informações de PCBs. Verifica conexões e cria uma lista (*netlist*) do circuito lido. As informações são oriundas de uma biblioteca de símbolos integrada ao sistema. Estão disponíveis bibliotecas para esquemáticos em geral, circuitos eletrônicos, circuitos elétricos e redes em geral.

5) **CAPFAST**: desenvolvido pela Phase Three Logic, Inc., Beaverton, OR.

É um editor de circuitos elétricos ou eletrônicos, baseado numa larga biblioteca de símbolos integrada à ele. Possui interfaces para o simulador SPICE e também para a criação de PCBs. Está disponível também para DOS e Unix. Nesta pesquisa, Cap Fast teve como objetivo apenas, aferir a média do tempo necessário à captação manual de diagramas de circuitos. Ele não está integrado ao sistema SRIDE.

6) **THE DESIGN CENTER**: desenvolvido pela MicroSim Corp. Irvine, CA.

É um software para editar, simular e analisar circuitos analógicos ou digitais. Também é baseado numa grande biblioteca de símbolos e pode ser rodado no Unix. Possui um analisador gráfico para formas de ondas produzidas pelo PSPICE.

Capítulo 3

3. MÓDULO RECONHECEDOR

3.1 INTRODUÇÃO

O principal objetivo desta pesquisa é desenvolver um *software* que reconheça diagramas de circuitos elétricos desenhados em papel (2D). Basicamente este tipo de desenho contém símbolos, linhas que interconectam estes símbolos e caracteres que formam a documentação do desenho.

Nossa metodologia é baseada em propriedades simples que são genéricas aos símbolos e linhas de conexão pertencentes a um diagrama de circuito. Por exemplo, uma linha de conexão não tem laços. Uma linha de conexão (exceto se for linha de entrada ou saída) conecta sempre dois símbolos, etc.

O primeiro passo na obtenção deste sistema computacional é separar os símbolos das linhas de conexão. Este processo de segmentação é crítico para o sucesso do sistema. Quanto melhor a segmentação, mais fácil será para as rotinas de classificação interpretar o desenho. Esta é a principal proposta deste trabalho: fazer uma boa segmentação da imagem. Para que isto ocorra, vários módulos foram desenvolvidos, tornando a tarefa do reconhecedor mais eficiente. A discussão de cada módulo é apresentada nos tópicos seguintes deste capítulo.

3.2 EXTRAÇÃO DOS COMPONENTES CONEXOS

3.2.1 OBJETIVO

A primeira tarefa a ser realizada é a separação dos caracteres que compõem as descrições dos elementos do desenho do diagrama de circuito, dos símbolos gráficos propriamente dito. Para que isto ocorra é necessário rotular (veja definições no Item 3.2.3 deste capítulo) a imagem e pesquisá-la a fim de se obter as regiões onde possivelmente se encontra um caracter ou conjunto de caracteres (*string*), os quais são definidos pelo tamanho do menor retângulo que circunda cada caracter (*bounding box*).

3.2.2 CONCEITOS BÁSICOS

Imagens de uma forma geral são representadas através de matrizes, cujos elementos representam os níveis de cinza que compõem a imagem e cujas dimensões definem o tamanho da imagem. Outro item importante é a resolução dada pelo número de pontos por polegada (*dots per inch* - dpi), em que a imagem foi capturada. Estes valores, dimensões e dpi, definem a quantidade de memória do computador necessária para o armazenamento da imagem. Em geral este número é razoavelmente grande, sendo que muitas vezes, microcomputadores de pequeno porte não comportam o processamento de tais imagens. Como exemplo, suponha uma imagem de dimensões 8.5'' x 11'' captada com 300 dpi. Esta imagem digitalizada em 2 níveis de cinza, necessita 2550 x 3300 pixels ou aproximadamente 1 Mbytes de memória para ser armazenada. Se a mesma imagem for digitalizada em 256 níveis

de cinza, a necessidade de memória aumenta em 8 vezes. Caso a imagem seja captada em 600 dpi, os números anteriores serão quadruplicados.

Esta introdução é para mostrar que quando se manuseia imagens, os processos a serem utilizados devem ser, na medida do possível, otimizados, pois caso contrário o usuário poderá não conseguir trabalhar com a imagem devido principalmente à falta de espaço necessário para armazená-la. Esta foi nossa principal preocupação durante o desenvolvimento deste sistema.

A técnica de **componentes conexos** (ou regiões) introduzida por [Rosenfeld e Kak 1976] é também chamada de **análise de regiões** e tem como objetivo simplificar uma imagem digital através da enumeração das suas regiões. Cada região é vista como um conjunto: o conjunto das posições dos pixels pretos conectados entre si. Em seguida definimos a noção de conexidade.

A adjacência entre pontos pode ser definida a partir da noção de vizinhança. A vizinhança tem 2 formas mais comuns de ser definida (veja Figura 3.2.1):

- vizinhança 4: os 4 pontos imediatamente ao norte, sul, leste e oeste do pixel em estudo formam a vizinhança;
- vizinhança 8: os 8 pontos imediatamente ao nordeste, este, sudeste, sul, sudeste, oeste, noroeste e norte do pixel em estudo formam a vizinhança;

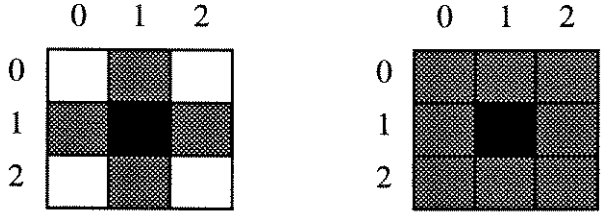


Figura 3.2.1 - Conectividade de pixels

Um ponto **p** é adjacente [Rosenfeld e Kak 1976] a um ponto **q** se **p** está na vizinhança de **q**. A adjacência será 4 ou 8 dependendo da vizinhança especificada.

Um caminho do ponto **p** para o ponto **q** é a seqüência de **n** pontos $\mathbf{p_i}$, tal que $\mathbf{p_i} = \mathbf{p}$ e $\mathbf{p_n} = \mathbf{q}$ e $\mathbf{p_i}$ é adjacente a $\mathbf{p_{i-1}}$ para todo $1 < i \leq n$, onde **n** é o tamanho do caminho. Pode-se definir caminho 4 ou 8 dependendo do tipo de adjacência utilizada. Os pontos **p** e **q** são as extremidades do caminho.

Dois pontos **p** e **q** são conexos num subconjunto **S** se são extremidades de um caminho em **S**.

Um conjunto **S** é um conjunto conexo ou região se qualquer dois pontos **p** e **q** de **S** são conexos em **S**.

Um subconjunto **C** de um conjunto **S** é um componente conexo de **S** se **C** é conexo e não existe outro subconjunto conexo contido em **S** e contendo **C**.

A Figura 3.2.2 mostra um exemplo com 3 regiões conexas.

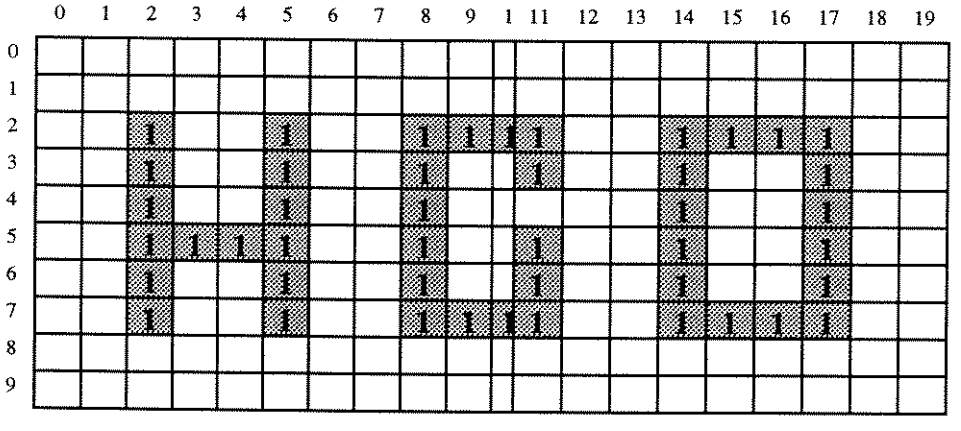


Figura 3.2.2 - Exemplo de regiões conexas numa imagem binária

Algoritmos que realizam a tarefa de rotulação de imagens são descritos em [Duda e Hart 1973]. Como nesta pesquisa trabalha-se apenas com imagens binárias, o problema é simplificado para encontrar o fundo (*background*) contra a imagem (*foreground*) que contém regiões conexas. Estas regiões conexas podem ser encontradas em qualquer parte da

imagem. Na Figura 3.2.2, os caracteres *HCO* são representados na forma matricial, de tal forma que os valores de pixel igual a 1 correspondem ao pixel aceso (*ON*) e os quadros com ausência de valores correspondem ao pixel apagado (*OFF*).

3.2.3 ALGORITMO BÁSICO

O algoritmo básico utilizado para se obter as regiões conexas numa imagem (rotulação) é aquele apresentado por [Rosenfeld e Kak 1976]. Cada região encontrada na imagem recebe uma identificação, chamada **rótulo** (*label*). Desta forma, todos os pontos pertencentes a uma componente *C* têm o mesmo rótulo, e nenhum ponto fora de *C* tem aquele rótulo.

A rotina que **rotula** os componentes da figura envolve pesquisa e propagação. A imagem é varrida linha a linha. Quando um pixel de imagem é encontrado, este toma o primeiro rótulo disponível. Seja este rótulo denominado *r*. O valor *r* é propagado, ou seja, a imagem é transformada de 1 para *r* enquanto naquela linha forem encontrados pixels adjacentes (**corrida**). Ao término da propagação de *r* se ainda houver outros conjuntos de pixels pertencentes à imagem, o processo é repetido com novo rótulo *r*, para cada nova corrida.

As Figuras 3.2.3 e 3.2.4 mostram respectivamente uma linha qualquer de uma imagem contendo os pixels que compõem esta imagem, e a identificação das corridas quando se atribui o rótulo inicial 5 para *r* .

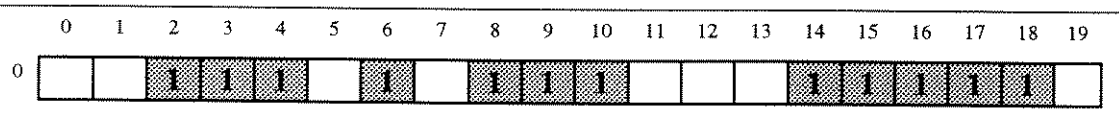


Figura 3.2.3 - Uma linha qualquer da imagem

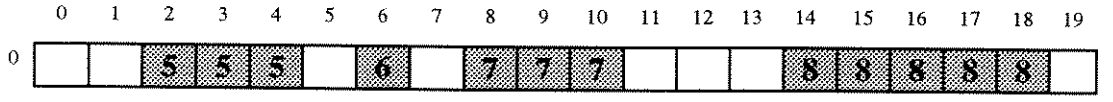


Figura 3.2.4 - Corridas rotuladas para rótulo inicial = 5

Para cada linha subsequente, cada corrida é comparada com as corridas da linha anterior, e as posições relativas entre estas corridas são estudadas.

Três situações são possíveis:

1. A nova corrida (refira-se à nova linha da imagem) não é adjacente a nenhuma corrida da linha anterior. Neste caso o rótulo para esta corrida é novo.
2. A nova corrida é adjacente a apenas uma corrida da linha anterior. Neste caso o rótulo desta nova corrida é o rótulo da corrida a qual ela é adjacente na linha anterior.
3. A nova corrida é adjacente a mais de uma corrida na linha anterior. Neste caso, o rótulo desta nova corrida é o rótulo da última corrida adjacente à ela na linha anterior, e as outras corridas da linha anterior que também são adjacentes a esta nova corrida, recebem uma **marca**, indicando que são corridas **equivalentes**. Elas serão re-rotuladas num segundo passo.

Quando todas as linhas da imagem tiverem suas corridas identificadas, inicia-se o segundo passo do algoritmo: identificar como iguais aquelas corridas marcadas na terceira etapa do primeiro passo do algoritmo. Isto significa que embora tenham rótulos diferentes, as corridas da linha anterior como são adjacentes a uma única corrida na linha subsequente, devem ter o mesmo rótulo comum a todas as corridas adjacentes.

A Figura 3.2.5 exemplifica a imagem da Figura 3.2.2 após o primeiro passo deste algoritmo.

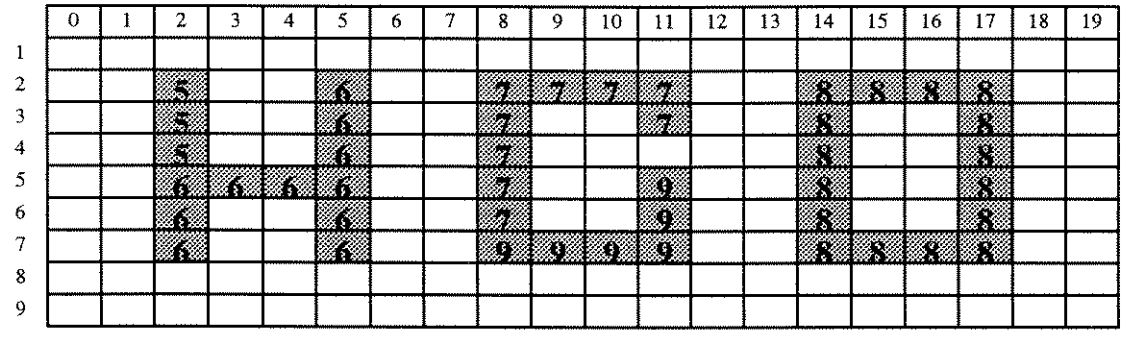


Figura 3.2.5 - Rotulação das corridas da imagem da Figura 3.2.2 ao final do primeiro passo do algoritmo

Veja por exemplo o topo da letra **H** e a parte de baixo da letra **C** ambos na Figura 3.2.5, onde tem-se respectivamente a corrida 6 comum à corrida 5 e a corrida 9 comum à corrida 7. Após a identificação desta situação, novos rótulos são estabelecidos para a identificação das corridas e consequentemente das regiões na imagem.

A Figura 3.2.6 mostra o processamento da segunda etapa do algoritmo aplicado à Figura 3.2.5. Uma vez identificadas todas as corridas adjacentes, tem-se por conseqüência **todas as regiões da imagem rotuladas**, visto que pixels com o mesmo rótulo estão 2 a 2 conectados.

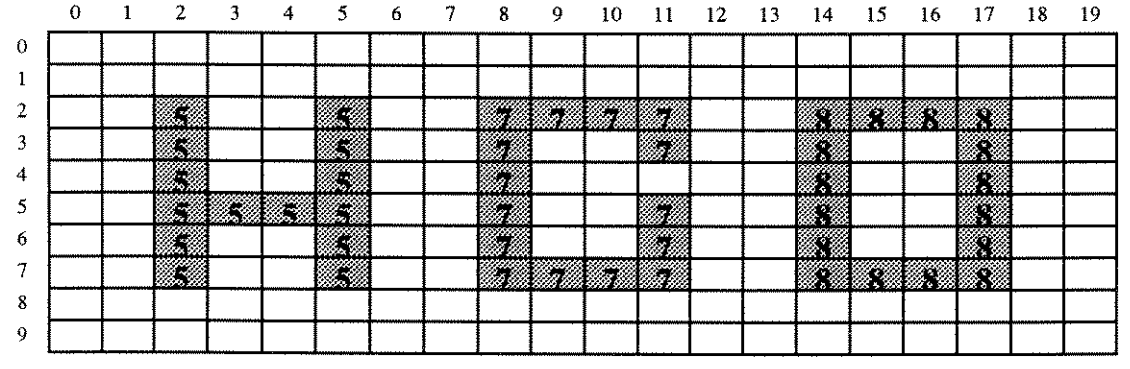


Figura 3.2.6 - Rotulação das corridas da imagem da Figura 3.2.5 ao final do segundo passo do algoritmo

3.2.4 ALGORITMOS ALTERNATIVOS

[Lumia e outros 1983] comentam o algoritmo de [Rosenfeld e Kak 1976] e apresentam três alternativas para a obtenção de componentes conexos, as quais são resumidas aqui.

Além destes artigos, para aqueles que trabalham com morfologia matemática, existe uma metodologia baseada em filas (método recursivo) apresentada por Luc Vincent em [Vincent 1992] e [Vincent 1993].

O algoritmo 1 em [Lumia e outros 1983] é o de [Rosenfeld e Kak 1976], apresentado no item anterior deste trabalho. Ele utiliza um par de conjuntos, de tal forma que um conjunto contenha os rótulos atuais das regiões e o outro contenha o menor rótulo equivalente. A imagem é processada de cima para baixo descendente (*top down*) e todas as equivalências são armazenadas em conjuntos. Um segundo passo é necessário com o objetivo de atribuir a cada rótulo armazenado nestes conjuntos, seus menores equivalentes.

O grande problema desta técnica, ocorre quando o número de regiões definidas é muito grande, dificultando o manuseio dos conjuntos de equivalência..

O algoritmo 2 em [Lumia e outros 1983] difere do primeiro pelo fato de não utilizar nenhuma tabela de equivalência. O algoritmo é desenvolvido em 3 passos: inicialmente uma matriz de rótulos com dimensões iguais à matriz de imagem é criada tal que para cada pixel da imagem diferente de zero, é criado um rótulo correspondente na matriz de rótulos. Então num segundo passo, um processo descendente (*top down*) ocorre. A cada pixel diferente de zero é atribuído o menor rótulo dentre os rótulos do conjunto formado pelo pixel e seus vizinhos. O terceiro passo é similar, sendo que a pesquisa é realizada em forma

ascendente. Os passos 2 e 3 são realizados alternativamente até que não haja mudanças nos rótulos.

O algoritmo 3 é um compromisso entre os 2 últimos algoritmos. Tal como o algoritmo 2, ele usa um processamento descendente seguido por um ascendente. Entretanto, não é necessário a utilização de iteração. Ele usa uma tabela de equivalência similar à utilizada no algoritmo 2, exceto que esta somente é usada para uma linha da imagem ao invés de toda a imagem, e portanto tem tamanho bem reduzido. A cada linha da matriz da imagem, a tabela de equivalência é criada e depois é destruída. Todo o processamento, realizado em 2 passos, tem como vantagem sobre os anteriores o fato de utilizar uma pequena tabela de equivalência e somente um passo descendente e um passo ascendente.

3.2.5 Implementação Adotada

Objetivando otimizar os métodos anteriores, empregamos um algoritmo de rotulação de imagens ligeiramente modificado. Na próxima seção é discutida a estrutura de dados utilizada e a seguir são apresentados os detalhes da implementação do algoritmo.

3.2.6 Estrutura de Dados

Para que o objetivo da extração de caracteres seja alcançado, é necessário estabelecer uma estrutura de dados que proporcione bom desempenho na execução das tarefas propostas. Como já mencionado, em imagens de diagramas de circuitos o volume de dados é

grande e portanto a estrutura de dados a ser utilizada para representá-las precisa ser suficientemente ágil e dinâmica a fim de manuseá-las no menor tempo possível.

Uma estrutura de dados possível para uma imagem é a de **matriz** como já citado anteriormente. O grande problema em se utilizar esta estrutura é a falta de flexibilidade no seu armazenamento, devendo na maioria dos casos, estar sempre ocupando a memória quase que integral da máquina. É uma estrutura muito estática. A estrutura utilizada nesta pesquisa é bem mais dinâmica e permite a utilização de algoritmos quase que universais para a busca e recuperação das informações: trata-se da estrutura de **árvore**, a qual pode ser vista como um **grafo** [Tremblay e Sorenson 1984]]. Nesta implementação uma árvore representa os componentes conexos encontrados na imagem e cada nó desta árvore tem uma estrutura tal como mostrado na Figura 3.2.7:

Figura 3.2.7 - Estrutura de um nó da árvore de componentes conexos

onde:

- LABEL: é o rótulo que identifica o componente conexo;
- X , Y: são as coordenadas do retângulo envoltório do componente conexo;
- PIXELS: é a quantidade de pixels no componente conexo;
- EQUIVAL: é o rótulo do componente conexo equivalente ao rótulo do componente conexo em estudo.

Quando a matriz da imagem é varrida no início do processo a fim de se extrair os componentes conexos, cada corrida obtida é acrescentada a uma árvore, formando no final do

processo a **árvore de corridas** daquela imagem, cujos nós têm a estrutura mostrada na Figura 3.2.7. As corridas equivalentes, são colocadas numa outra árvore, denominada **árvore de equivalentes**, a fim de que possa ser utilizada na segunda parte do processo de obtenção dos elementos conexos, sem conflitos com as corridas presentes na árvore de corridas.

Como retângulo envoltório entenda-se o retângulo imaginário definido pelas coordenadas do pixel mais à esquerda e mais acima do componente conexo e pelas coordenadas do pixel mais à direita e mais abaixo do componente conexo.

Por exemplo, o retângulo envoltório do componente conexo de rótulo 5 (letra H) na Figura 3.2.6, tem as seguintes dimensões: *X (2,5) e Y (2,7)* ou seja:

XMIN = 2 **e** *XMAX = 5*

YMIN = 2 **e** *YMAX = 7.*

Neste exemplo, o componente conexo é constituído de 14 pixels.

3.2.7 ALGORITMO

A estrutura do algoritmo desenvolvido nesta pesquisa para a rotulação de uma imagem é mostrada a seguir.

ALGORITMO rotulação

1. *buscar uma corrida de pixels na imagem*
2. *incluir corrida na árvore de corridas*
3. *atualizar árvore de equivalências*
4. *se imagem não terminou, voltar ao passo 1*
5. *associar árvore de corridas com árvore de equivalência*
6. *atualizar dimensões dos componentes conexos formados*
7. **FIM** ALGORITMO

Este algoritmo é similar ao apresentado por [Rosenfeld e Kak 1976], porém possui as seguintes vantagens:

- Não é criado outro conjunto para os rótulos da imagem como acontece nos algoritmos apresentados por [Lumia e outros 1983];
- Os rótulos dos elementos conexos encontrados na imagem bem como seus equivalentes, são armazenados em lista, cujas estruturas são mais dinâmicas e flexíveis que as tabelas.
- O processo é realizado em 2 passos: a busca dos componentes conexos na imagem e rotulação dos componentes conexos equivalentes através de listas.
- Rótulos já incluídos na lista de componentes conexos não são reincluidos, fazendo com que a lista seja sempre a menor possível.

A implementação da árvore de componentes conexos é realizada através de uma lista encadeada cujos nós têm a estrutura mostrada na Figura 3.2.7. A utilização deste tipo de estrutura deve-se ao fato dela proporcionar maior grau de mobilidade quando na inclusão, busca e deleção de nós nas árvores que compõem a imagem. O total de memória computacional utilizada não é estático pois depende do tamanho da imagem.

A imagem é processada uma linha a cada vez. Em cada linha os pixels seqüenciais são considerados uma única corrida. Cada corrida encontrada é incluída (se o rótulo atribuído a ela for novo) ou atualizada (se o rótulo atribuído a ela já existe) na lista de componentes conexos. Uma corrida recebe um novo rótulo se não for adjacente à nenhuma corrida anterior. Caso contrário, duas situações são possíveis: a corrida atual é adjacente à apenas uma corrida da linha anterior, e neste caso recebe o mesmo rótulo da corrida anterior; ou a corrida atual é adjacente à várias corridas da linha anterior, e neste caso recebe o rótulo da última corrida rotulada e as outras corridas são também equivalentes a ela. Neste último caso elas são

incluídas numa lista de componentes conexos para futura análise e atualização da imagem final. Nesta pesquisa está sendo utilizada a vizinhança 8 para a verificação de pixels vizinhos.

A grande diferença entre nossa implementação e as outras citadas, está na velocidade e otimização do processamento dos elementos da imagem. Nossa abordagem emprega listas de componentes (as menores possíveis), que por sua vez permitem a utilização de operações mais velozes tendo em vista que podem ser realizadas por apontadores internos de memória e ainda, o número de conjuntos a serem manipulados é menor que nos processos citados. No apêndice **A** é apresentado o algoritmo utilizado nesta pesquisa.

3.2.8 RESULTADOS

Como exemplo, analisemos a imagem da Figura 3.2.2, onde as regiões conexas elementares são apresentadas numa imagem binária (sintética). A lista de seus componentes conexos e respectivos equivalentes (sublistas) ao final do primeiro passo do algoritmo, são mostradas na Figura 3.2.8 através da representação gráfica da lista. Esta lista refere-se à imagem mostrada na Figura 3.2.5.

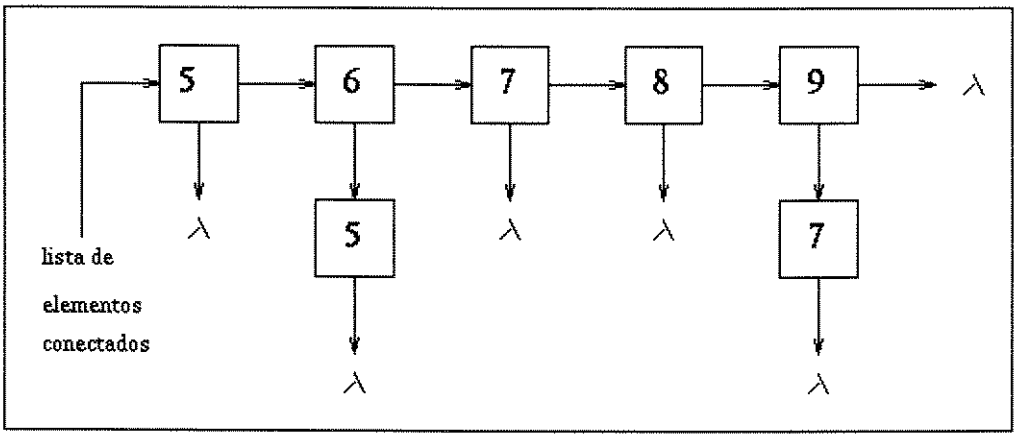


Figura 3.2.8 - Representação gráfica da lista dos componentes conexos e respectivos equivalentes da imagem da Figura 3.2.5

A saída final do algoritmo, com a lista de todos os **componentes conexos** já devidamente rotulados em toda a imagem (veja a Figura 3.2.6), é mostrada na Tabela 3.2.1.

Observar que para cada grupo de elementos conexos, tem-se as seguintes informações:

a) LABEL: identificação do grupo;

b) BOT: coordenadas do inicio do grupo;

c) TOP: coordenadas do fim do grupo;

d) PIXELS: número total de pixels no grupo;

LABEL	RETÂNGULO ENVOLVENTE				NÚMERO DE PIXELS
	Xmin	Ymin	Xmax	Ymax	
5	2	2	5	7	14
7	8	2	11	7	15
8	14	2	17	7	16

Tabela 3.2.1 - Lista final dos componentes conexos da imagem da Figura 3.2.6

3.2.9 EXTRAÇÃO DOS CARACTERES

A obtenção dos elementos conexos tem por objetivo final retirar da imagem os caracteres pertencentes a textos tais como identificação dos elementos, notas, esclarecimentos e outras informações de caráter não simbólico presentes na imagem.

Tratando-se de um diagrama de circuito, espera-se que este seja completamente conexo, ou seja, a menos dos caracteres pertencentes aos textos já mencionados, todos os demais elementos deverão estar conexos entre si. Partindo-se desta hipótese, empregamos esta técnica para extrair todos os caracteres presentes na imagem do circuito.

Em nossa abordagem, montamos a lista de todos os elementos conexos encontrados na imagem. Calculamos o tamanho da área do retângulo envolvente de cada elemento conexo encontrado. É natural supor-se que caracteres tenham um retângulo envolvente bem pequeno, e assim sendo, determinamos que todos os elementos conexos cujo retângulo envolvente tenha o tamanho menor que um determinado valor (parametrizado), seja considerado como sendo um caracter, e desta forma é eliminado da imagem. Os elementos conexos eliminados, são verificados a fim de saber-se se são caracteres isolados ou formam uma cadeia de caracteres. Isto é feito, analisando-se a distância entre os pontos médios de seus retângulos envolventes (centróides), alinhamento destes pontos, etc. Isto é importante de ser feito, pois estas cadeias são armazenadas num outro arquivo a fim de que se proceda seu reconhecimento através de um OCR.

Esta técnica apresenta como vantagem o fato de que todos os ruídos da imagem são efetivamente eliminados, mas tem também como desvantagem o fato de que alguns símbolos abertos (veja Item 3.4.1) podem ser parcialmente eliminados, como acontece principalmente com o símbolo terra.

Como exemplo, apresentamos na Figura 3.2.9 o circuito elétrico da Figura 2.3.2 com os caracteres eliminados utilizando-se a técnica discutida e desenvolvida para esta pesquisa.

Deve-se notar que apenas o símbolo de terra teve a última linha retirada, devido ao fato desta ter seu tamanho muito pequeno em relação ao restante do símbolo e portanto, foi

eliminada. Porém isto não é problema para o reconhecimento posterior do símbolo, tendo em vista a permanência de mais uma outra linha complementar do símbolo terra.

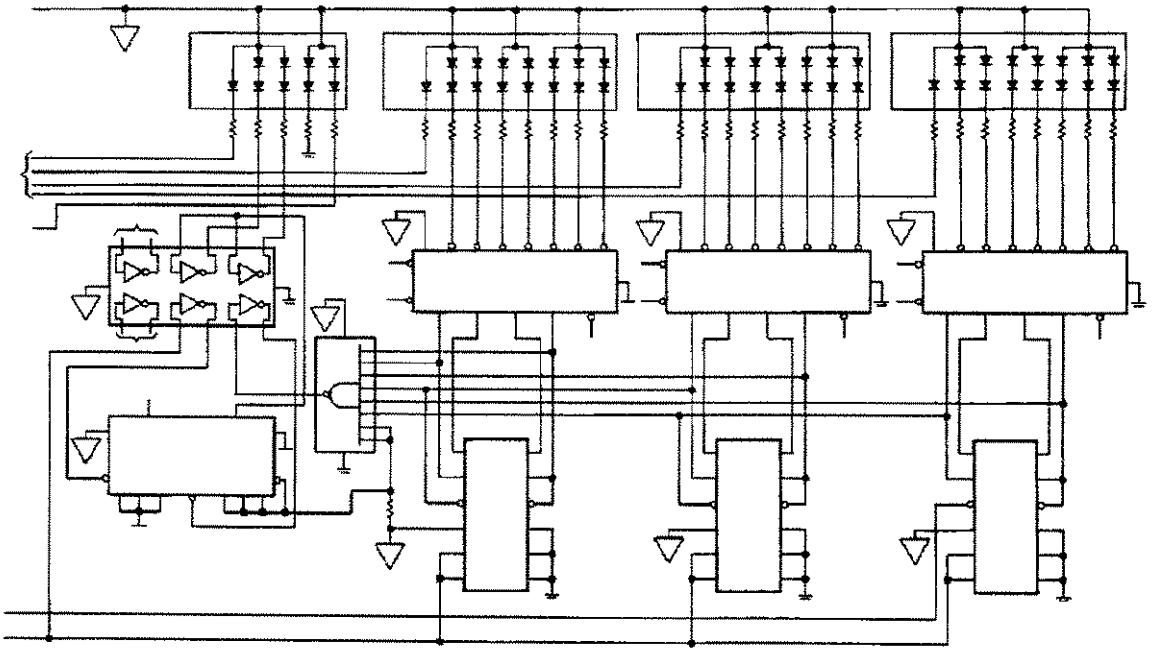


Figura 3.2.9 - Circuito da Figura 2.3.2 sem os caracteres

Outro exemplo é mostrado nas figuras seguintes: 3.2.10 mostra a imagem original completa de um diagrama de circuito elétrico, contendo todos caracteres de identificação, e a Figura 3.2.11 mostra a mesma imagem com todos os caracteres extraídos através do processo desenvolvido e empregado. Observar nesta figura a presença de linhas formando o traço da fração que identifica um capacitor. Estas linhas puderam ser eliminadas neste caso, tendo em vista a sua relação altura *x* largura e também devido ao fato de estarem isoladas.

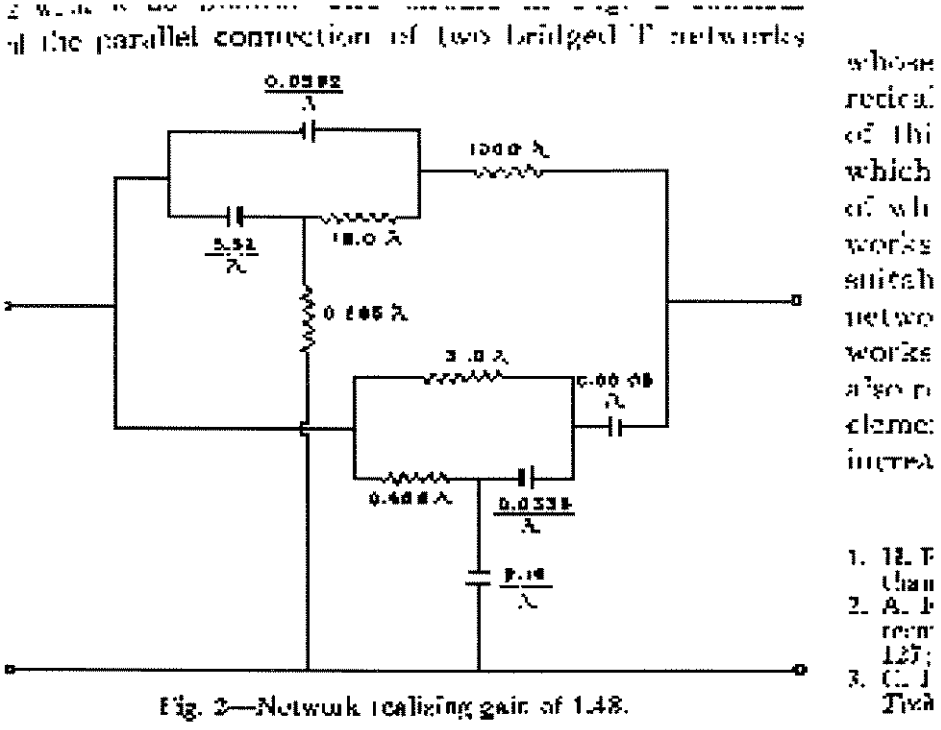


Fig. 2—Network realizing gain of 1.48.

Figura 3.2.10 - Trecho de uma imagem de circuito (TIFF 300 dpi)

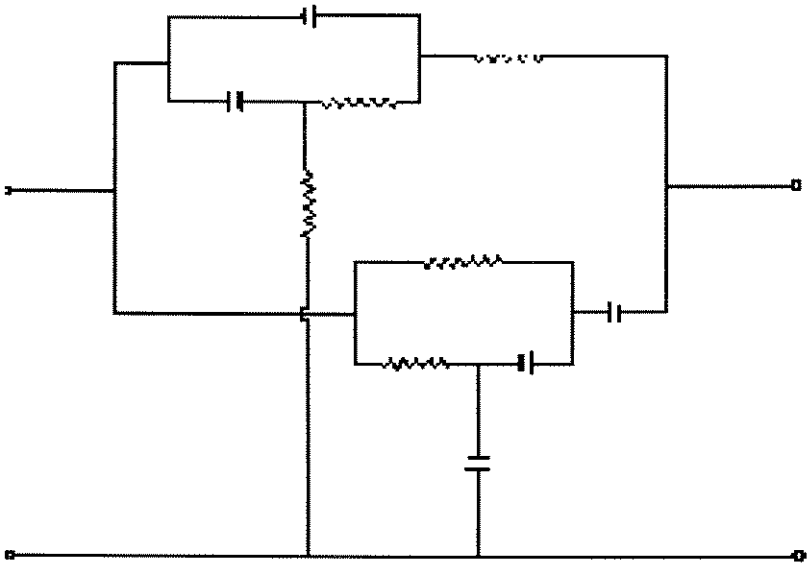


Figura 3.2.11 - Retirada dos caracteres da imagem da Figura 3.2.10 através de elementos conexos

3.3. VETORIZAÇÃO

3.3.1 OBJETIVO

Um diagrama de circuito elétrico completo geralmente contém 3 categorias de componentes essenciais: símbolos dos elementos que constituem o circuito, caracteres e linhas de interconexão. Os caracteres são retirados da imagem conforme descrito na etapa anterior desta pesquisa: a de extração dos elementos conexos; os símbolos do circuito e linhas de interconexão entre eles serão obtidas após a imagem ter sido transformada de *raster* para vetor. Este é o objetivo desta fase.

Portanto, o objetivo desta etapa é vetorizar a imagem de um diagrama de circuito elétrico após esta ter sido processada para a retirada dos textos usados na especificação de medidas e denominações dos elementos que a compõe.

3.3.2 DEFINIÇÃO DO PROBLEMA

Uma imagem digital por si só não tem muito significado. Esta imagem somente torna-se útil quando ela pode ser posta numa forma que possa ser manipulada por outras aplicações. Para que isto ocorra, é necessário colocar a imagem numa "forma padrão" para que as aplicações possam interpretá-la e usá-la. Programas gráficos em geral podem ser categorizados pela forma em que os dados de entrada, no caso uma imagem, podem ser armazenados e mostrados. Existem duas categorias de formatos para se representar uma imagem: ***raster*** e ***vetor*** [Kay e outros 1982]. O formato *raster* compreende a representação da imagem numa série de pixels que cobrem uma determinada área de vídeo. Imagens *raster*

são geralmente geradas por varreduras periódicas de elétrons sobre uma superfície própria e com um predeterminado padrão (uma câmera de vídeo por exemplo).

O formato *vetor* por outro lado, envolve o uso de segmentos de linhas ao invés de pixels para compor a imagem. Conexões e hierarquia são os pontos fortes de uma imagem vetorizada. É fácil de se determinar quais componentes formam um objeto na imagem. Exemplos destas aplicações são encontrados em programas CAD, onde a relação entre os elementos é muito importante, tendo em vista fatores como posicionamento e escala entre eles.

Os dois formatos raster típicos são **TIFF** (suportado pela Aldus, Microsoft, Hewlett-Packard, e muitos outros fabricantes) e **PCX/PCC** (suportado pela Zsoft's PC PaintBrush) enquanto que um exemplo de formato vetor é o **DXF** (suportado pela Autodesk Inc.).

Nesta pesquisa uma típica imagem gráfica (no formato TIFF) contendo um diagrama de circuito elétrico (Figura 2.3.2), pode ser constituida de linhas retas, linhas contínuas, linhas pontilhadas ou tracejadas, as quais devem ser extraídas e analisadas.

O problema aqui é desenvolver um algoritmo que aplicado a uma imagem binária, gere como resultado um arquivo contendo *vetores descritivos*, a partir dos quais a imagem binária original pode ser reproduzida. As formas dos objetos na imagem são obtidas pela combinação das linhas que a constitui, as quais podem ser representadas por características tais como **largura** e **esqueleto**. Uma das soluções possíveis para a obtenção da imagem descrita através de linhas, é a utilização de **afinamento**.

Afinamento é uma operação de processamento de imagem através da qual regiões da imagem, são reduzidos à linhas que se aproximam ao centro destas regiões (**esqueletos**). A finalidade do afinamento é reduzir os componentes da imagem às suas informações essenciais, tal que futuras análises e reconhecimento sejam facilitados.

Muitos são os métodos e algoritmos usados para o afinamento de uma imagem. Alguns exemplos são encontrados em: [Naccache e Shinghal 1984], [Zhang e Suen 1984], [Ammann e Angus 1985], [Pavlidis 1986], [Holt e outros 1987] e [Lam e outros 1992].

Na literatura em geral , [Lam 1992], há consenso que os métodos de afinamento devem ter certos requisitos para um bom desempenho: preservação das propriedades topológicas da imagem, isotropia, capacidade de ser reconstruido e boa velocidade de processamento.

Reconstrução, ou a habilidade de regenerar o padrão original a partir do esqueleto, é uma medida objetiva da exatidão com a qual o esqueleto está representando o padrão.

Isotropia, ou invariância à rotação, parece ser quase impossível para algoritmos iterativos de afinamento (aqueles que apagam sucessivas camadas de pixels na borda do padrão, até permanecer somente o esqueleto do mesmo). Em algoritmos iterativos seqüenciais (aqueles onde os pixels da borda são examinados numa ordem pré determinada), os resultados dependem da ordem na qual os pixels são examinados. Em algoritmos iterativos paralelos (aqueles onde os pixels são examinados para o apagamento, baseados nos resultados da iteração anterior), que removem um ou dois tipos de pontos da borda em cada subiteração (subconjunto de pixels da borda considerados aptos para remoção), o resultado depende da ordem das subiterações.

Conexidade e **topologia** em afinamento podem ser verificados através de vários métodos. Em algoritmos sequenciais é suficiente examinar uma janela 3 x 3 na vizinhança local do ponto de vista do número de cruzamentos (número de vezes que se passa do pixel branco para o preto, quando uma região é percorrida numa determinada ordem) por exemplo. Em algoritmos paralelos o problema é resolvido dividindo-se cada ciclo em

subiterações ou considerando-se a uma vizinhança 5x5 em cada subiteração.

Todos os algoritmos desenvolvidos para o afinamento visam de alguma forma contemplar estas propriedades, e por isto é inevitável a comparação entre eles.

No entanto, o grande problema com eles é que há sempre grande perda de informação por ocasião do afinamento, ou então **pós processamentos** são necessários para obter-se os vetores representativos da imagem. Também são muito sensíveis a ruídos, e alguns, à forma da imagem. Como exemplo, tem-se uma imagem de um circuito elétrico mostrada na Figura 3.2.11, a qual é mostrada na Figura 3.3.1 afinada e vetorizada pelo processo de [Harris 1982]. Esta figura é composta por 455 vetores.

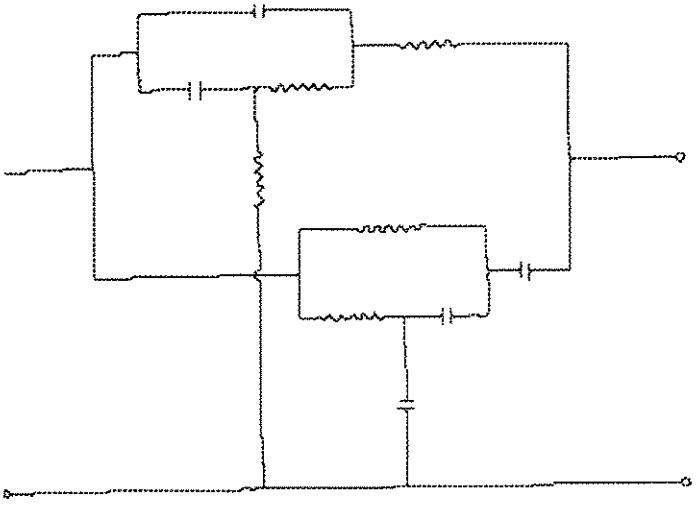


Figura 3.3.1 - Afinamento da imagem da Figura 3. 2.11 utilizando o processo de Harris

Assim como em outros métodos de afinamento, é comum obter-se como resultado, além das linhas reais, pequenas linhas ortogonais, tais como se fossem **espinhos** ou **perturbações** das linhas, as quais contribuem para que a representação da imagem tenha um número muito grande de pequenos segmentos, quando esta poderia ter apenas uns poucos segmentos para representá-la.

Estes espinhos e perturbações devem ser retirados no pós-processamento dos resultados obtidos do afinamento. Outro problema dos algoritmos de afinamento é que a grande maioria deles recebe como entrada a imagem *raster* e devolve como resultado uma imagem afinada também *raster*, o que impossibilita a utilização de mecanismos com o objetivo de selecionar os segmentos que compõem a imagem em estudo.

Tendo em vista estes problemas provenientes dos algoritmos de afinamento, desenvolveu-se neste trabalho uma metodologia para a vetorização da imagem binária, sem ter que afiná-la, passando-se diretamente da imagem *raster* para o conjunto de linhas que formam o diagrama de circuito. No problema de vetorização de um diagrama de circuito elétrico, esta etapa é muito importante, pois ela influencia na qualidade dos objetos estudados e também no seu reconhecimento. Tal algoritmo de vetorização tem como vantagens:

- é mais rápido para o objetivo a que se propõe (vetorizar diagramas de circuitos elétricos);
- é robusto à variação das larguras das linhas;
- vetoriza linhas cheias;
- vetoriza linhas tracejadas;
- produz um número pequeno de vetores.

Para este algoritmo funcionar, a imagem do circuito elétrico de entrada poderá possuir as seguintes entidades:

a) linhas retas interconectando símbolos;
b) linhas tracejadas (quando o tamanho de cada traço desta for maior que o retângulo envolvente que define um caracter, pois caso contrário a linha tracejada é eliminada por ocasião da eliminação dos elementos conexos);
c) finais de linhas;
d) símbolos formados por linhas retas;
e) símbolos abertos ou fechados;
f) símbolos e linhas com direções múltiplas de 45°;

g) cadeias de caracteres para identificação dos símbolos ou para especificação de valores de medida;
h) dobras de linhas em ângulos retos ou não (múltiplas de 45 graus);
i) cruzamentos de linhas ou junções T (simples ou soldados);

3.3.3 Metodologia Utilizada no Processo de Vetorização Proposto

O processo de vetorização da imagem, proposto nesta pesquisa, é realizado segundo a seguinte metodologia: como entrada à etapa de vetorização, considere-se uma imagem binária de um diagrama de circuito elétrico, com seus caracteres e textos retirados (como exemplo veja a Figura 3.3.1). Neste ponto tem-se também a lista dos elementos conexos, a qual será útil na definição das áreas de pesquisa na imagem. O processo de vetorização é realizado em dois passos: a) obtenção dos pontos característicos (extremos, junções ou dobras de linhas) e b) busca dos segmentos de linhas (vetores) que constituem a imagem.

A obtenção dos pontos característicos da imagem é realizada em 3 passos: a) enjanelamento de uma máscara de pesquisa na imagem; b) casamento de padrões entre máscara e imagem; e c) escolha da máscara que melhor representa a imagem nas regiões conectadas.

Uma lista de vértices é obtida com os dados das máscaras obtidas no passo anterior. Os vetores que constituem a imagem são obtidos através da análise da lista de vértices, a qual possui as direções dos possíveis segmentos de retas. Como resultado, obtém-se um conjunto de vetores contendo as coordenadas dos pontos extremos, os quais quando reagrupados, resconstituirão a imagem original de entrada.

3.3.4 Obtendo os Pontos característicos

São considerados **pontos característicos**, todos aqueles utilizados na busca de um segmento de reta da imagem. Por exemplo, na Figura 3.3.2 são mostrados os principais pontos característicos possíveis numa imagem de um diagrama de circuito elétrico.

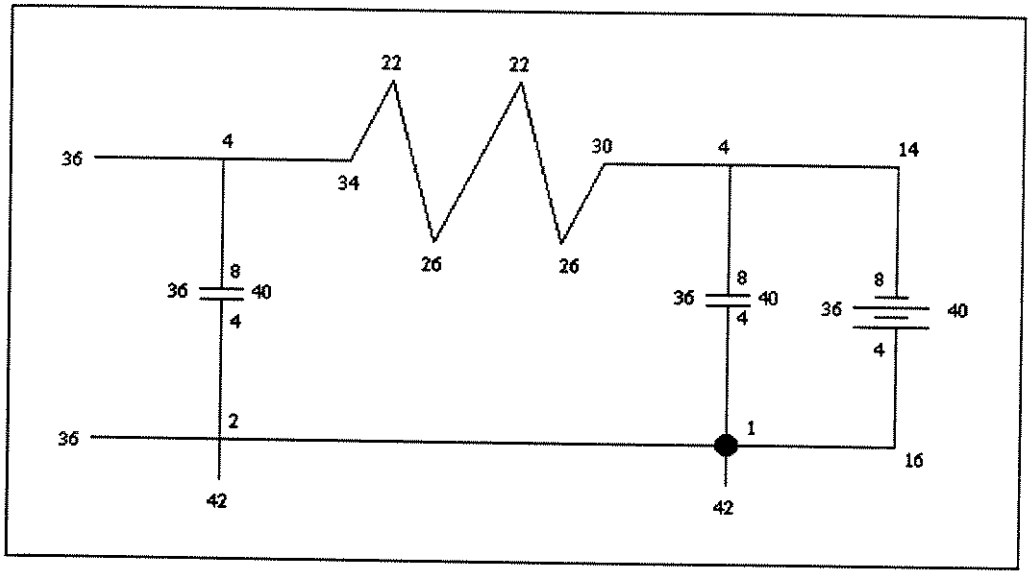


Figura 3.3.2 - pontos característicos num diagrama de circuito elétrico

Na Figura 3.3.2 os números correspondem aos pontos característicos, mostrados na Tabela 3.3.1: fim de segmento, junção (em cruz, em T), dobra de linha (com ângulo reto, ângulo agudo e ângulo obtuso).

Para determinar-se os pontos característicos na imagem binária, são utilizadas máscaras com dimensão 11 x 11 pixels (valor empírico), tais como mostrado nos modelos da Figura 3.3.3 e cujos *bitmaps* das posições iniciais dos modelos são mostrados na Figura 3.3.5. A escolha do tamanho das máscaras é importante e também difícil. Em geral, máscaras pequenas são sensíveis a ruídos, enquanto que máscaras grandes podem não detectar o elemento desejado.

Tipo	Descrição
1	junção cheia
2	junção em cruz
4	junção T
12	flexão com abertura = 90^0
20	flexão com abertura $< 90^o$
28	flexão com abertura $> 90^o$
36	fim de um segmento

Tabela 3.3.1 - Descrição dos pontos característicos possíveis num diagrama de circuitos elétricos - veja exemplo na Figura 3.3.2

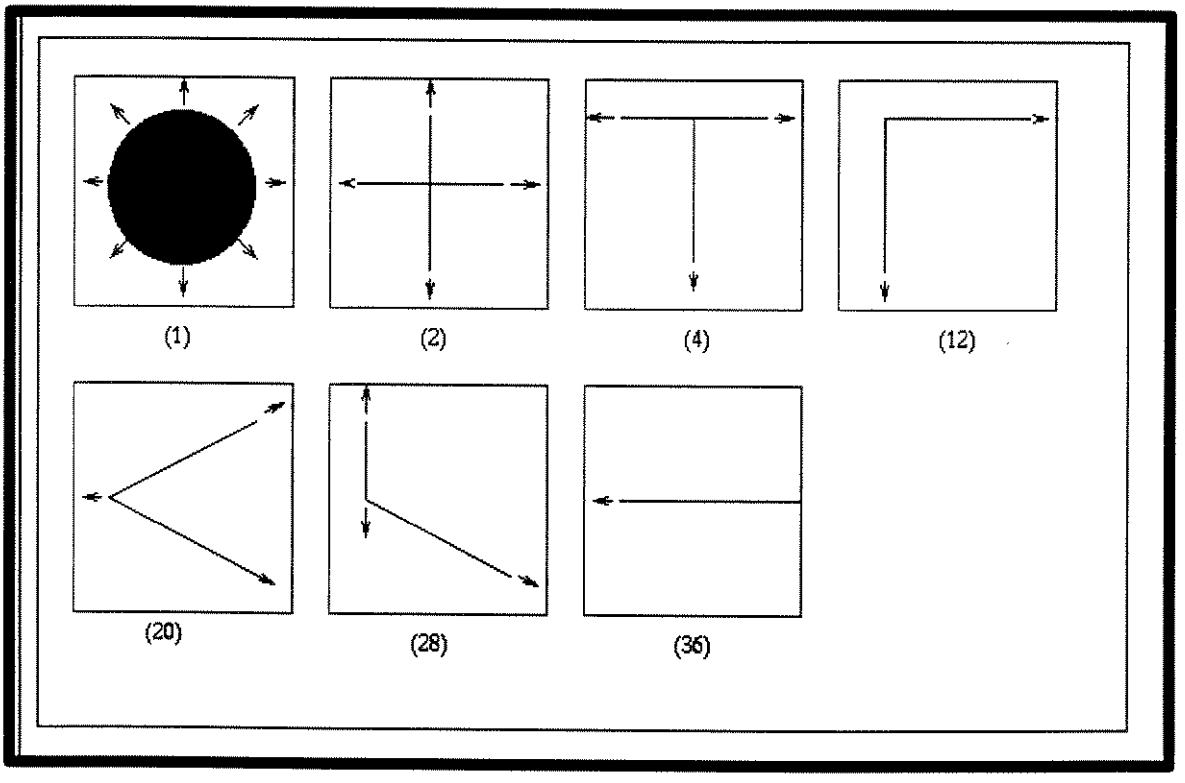


Figura 3.3.3 -Modelos para os pontos característicos

Observaçoes: 1) os números entre parênteses nos modelos apresentados na Figura 3.3.3, correspondem à identificação da máscara considerando-se todas as rotações possíveis (múltiplos de 45 graus ordenados no sentido horário); 2) as setas representam as direções de buscas de linhas a partir do pivô do modêlo representativo do ponto característico

Um tamanho ótimo para máscaras pode variar com a imagem ou com a parte da imagem que se deseja estudar. A fim de que este valor de tamanho de máscara fôsse achado, vários tamanhos foram testados nesta pesquisa, e chegou-se à conclusão que uma máscara com dimensões 11 x 11 pixels representa de forma satisfatória os elementos que se deseja segmentar na imagem. Mesmo assim, no sistema SRIDE, este valor é um parâmetro que permite ao usuário fazer novas experiências. Caso o usuário decida não modificar o tamanho das máscaras, este é automaticamente configurado para o valor 11 x 11 pixels.

Como exemplo, considere o modelo *(4)* da Figura 3.3.3, cuja máscara tem o *bitmap* mostrada na Figura 3.3.4.

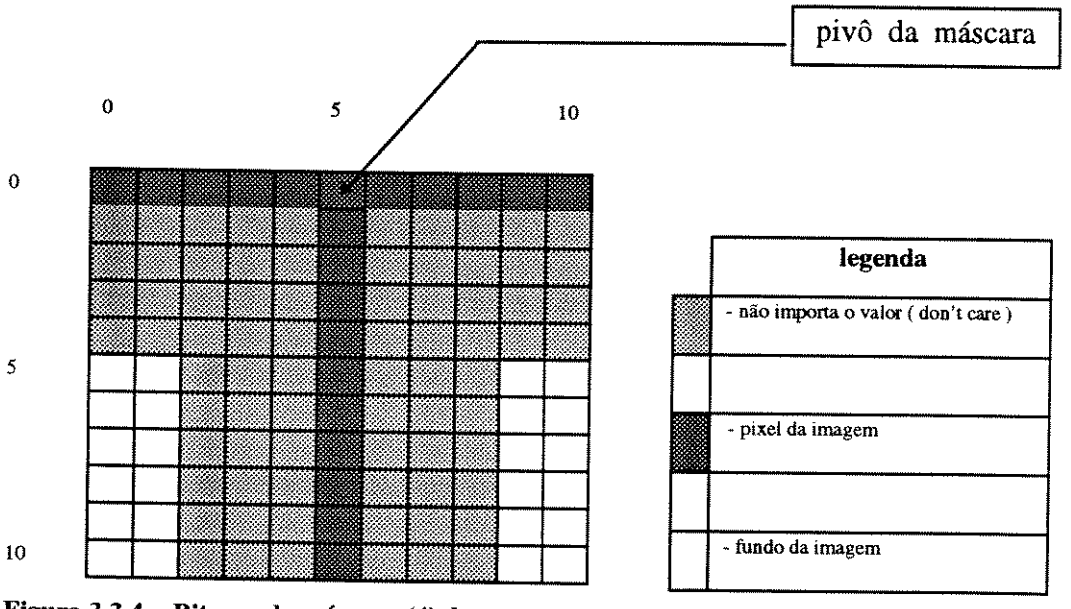
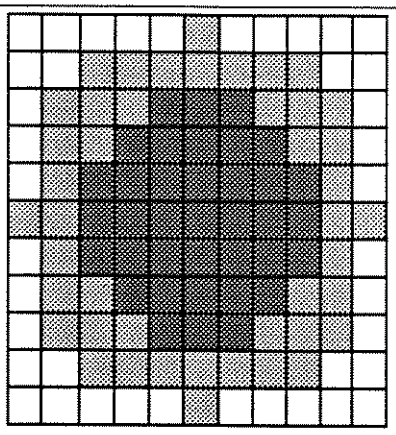
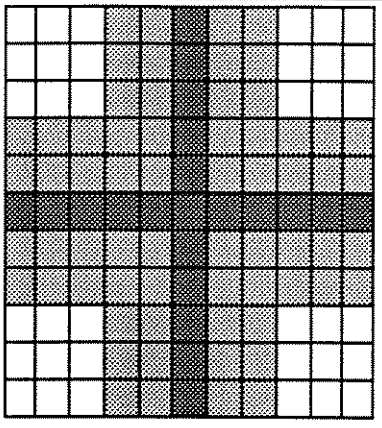
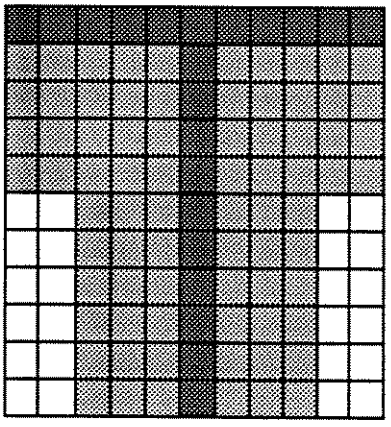


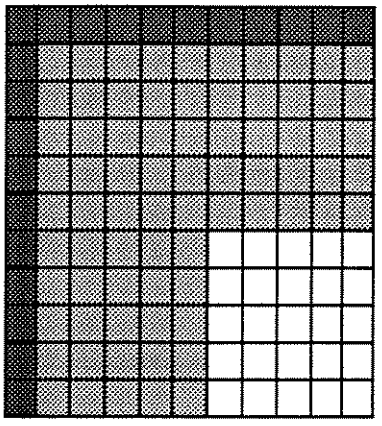
Figura 3.3.4 - Bitmap da máscara (4) do ponto característico T - veja Figura 3.3.3

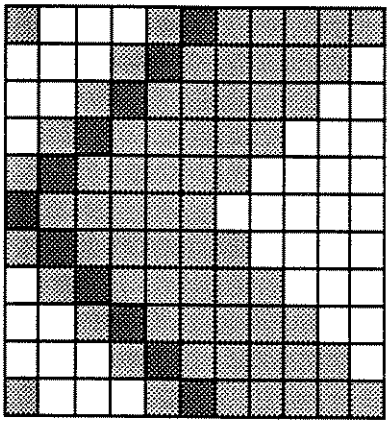
Cada máscara tem um ponto de referência chamado **pivô**, a partir do qual nasce um segmento de reta. Exceto na máscara *(36)* (e suas rotações) da Figura 3.3.3 onde o pivô é o cruzamento da linha com a borda direita da máscara, os pivôs das outras máscaras são os pontos de cruzamento de linhas dentro da máscara. No exemplo da Figura 3.3.4, o pivô é o pixel de coordenadas (5, 0).

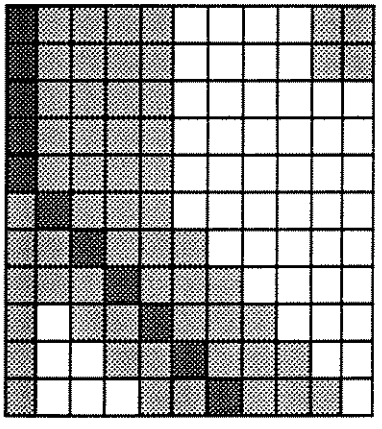












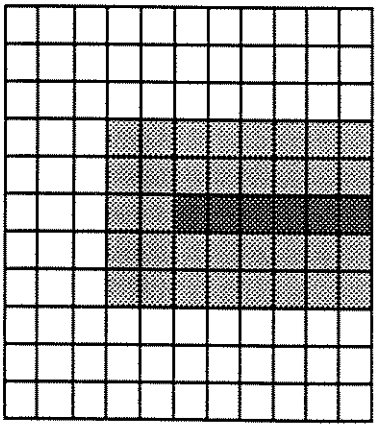


Figura 3.3.5 - Bitmap das máscaras dos modelos apresentados na figura 3.3.3
Veja legenda na figura 3.3.4

Quanto às rotações, as máscaras são pesquisadas dentro da imagem usando-se ângulos de 0 a 360 graus, com variação de 45 graus.

Como exemplo, seja a Figura 3.3.6, onde se vê uma máscara **T** (modelo 4 da Figura 3.3.3) navegando sobre uma imagem com o objetivo de detectar esta forma nesta imagem.

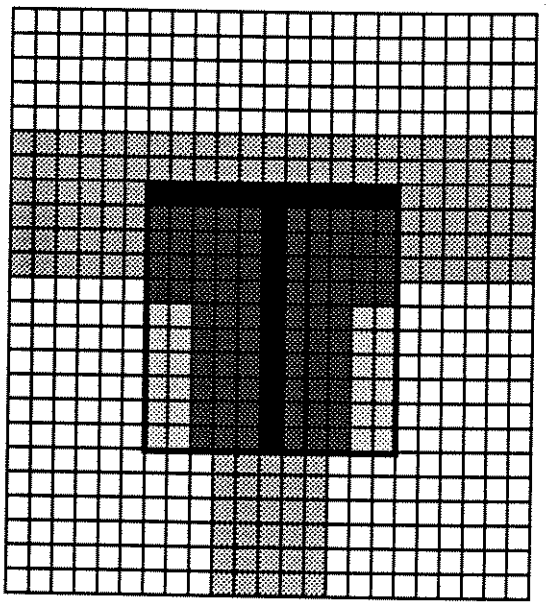


Figura 3.3.6 - Uma máscara navegando sobre uma imagem

Cada máscara, varre toda a imagem pixel a pixel. Esta operação é repetida para todas as 8 possíveis rotações (múltiplas de 45 graus) da máscara, exceto na máscara 1 que tem apenas 1 direção e na máscara 2 que tem 2 rotações. Nestas varreduras é verificado o casamento da máscara com a imagem. Este casamento é determinado pela perfeita coincidência dos pixels da imagem com os pixels da máscara, ou seja, pixels 1 da imagem casam com pixels 1 da máscara e o mesmo acontece com os pixels 0. A folga é determinada pela existência de pixels "don't care" na máscara, os quais não são considerados durante a verificação do casamento.

Havendo casamento de uma máscara com a imagem, o pixel da imagem que corresponde ao pivô da máscara é marcado com o número de identificação da máscara, não sendo modificado no futuro por outra máscara. Ao final da varredura de todas as máscaras, para uma determinada posição na imagem, as seguintes situações são possíveis:

a) houve casamento de apenas uma máscara;
b) houve o casamento de várias máscaras;
c) não houve casamento de nenhuma máscara.

Observa-se que na imagem da Figura 3.3.6, a máscara *(12)* da Figura 3.3.3 é encontrada várias vezes. Rotacionando-se esta máscara 90° no sentido horário obtém-se a máscara (*14*), a qual pode ser encontrada na imagem também várias vezes. Porém a máscara *(4)* é encontrada mais vezes que as outras máscaras, o que significa que esta é predominante naquela posição da imagem (veja exemplo real nas Figuras 3.3.16 e 3.3.17).

As coordenadas exatas do ponto característico assim determinado, são obtidas através da média aritmética de todas as coordenadas dos pontos da região onde ocorreu o casamento entre a imagem e a máscara predominante na região.

O resultado desta etapa é uma lista de todos os pontos característicos da imagem, contendo cada nó desta lista as seguintes informações:

1) identificação do nó: número inteiro que especifica cada ponto característico na imagem;
2) coordenadas do pivô do ponto característico com relação à origem da imagem;
3) tipo do ponto característico, de forma a identificá-lo como uma das máscaras da Figura 3.3.3.

3.3.5 Obtenção das Linhas Retas

O algoritmo aqui apresentado tem como objetivo, construir uma tabela contendo as coordenadas dos pontos iniciais e finais de todos os segmentos de retas encontrados na imagem binária de entrada através da aplicação do algoritmo de obtenção dos pontos característicos.

Suponha que de uma determinada imagem binária obtenha-se 2 pontos característicos situados topologicamente como mostrado na Figura 3.3.7. O ponto característico 1 corresponde ao modelo *(12)* da Figura 3.3.3, e o ponto característico 2 corresponde ao modelo (*4*), cujos pivôs são respectivamente os pontos P1 e P2. Pretende-se verificar a existência ou não de uma linha entre os pontos P1 e P2.

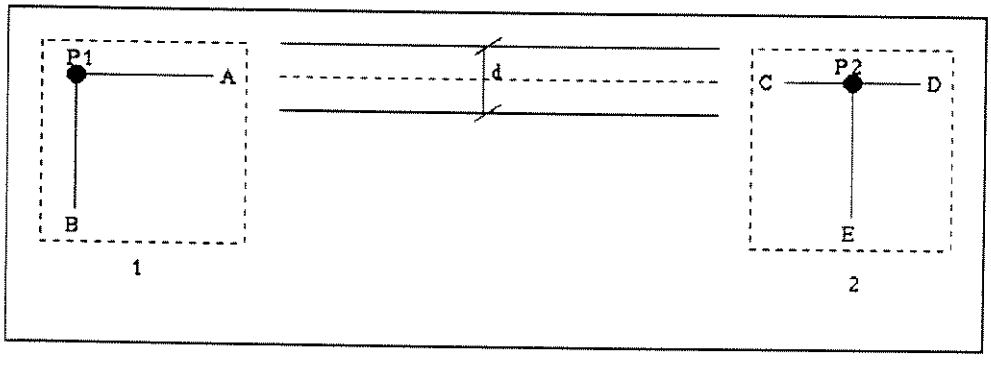


Figura 3.3.7 - Parte de uma imagem, mostrando 2 pontos característicos

De acordo com os tipos dos padrões envolvidos, tem-se a expectativa de se encontrar linhas entre os pontos **A** e **C** e linhas iniciando nos pontos **B**, ou **E** ou **D**. As direções disponíveis nesta pesquisa para a pesquisa de linhas, são mostradas na Figura 3.3.8.

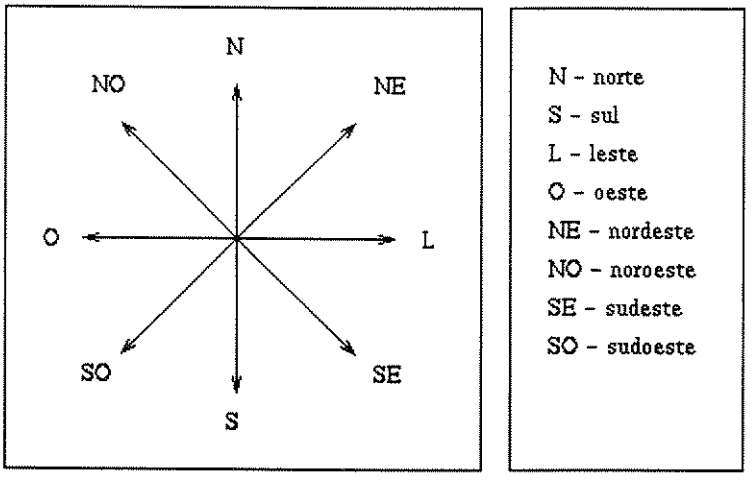


Figura 3.3.8 - Possíveis direções das linhas

Para o exemplo da Figura 3.3.7, a direção da pesquisa é a horizontal. Assim sendo, percorrendo-se do ponto **A** ao ponto **C**, deve-se traçar segmentos imaginários **L** ortogonais ao segmento **AC**, distantes *k* pixels um do outro. Veja o exemplo na Figura 3.3.9.

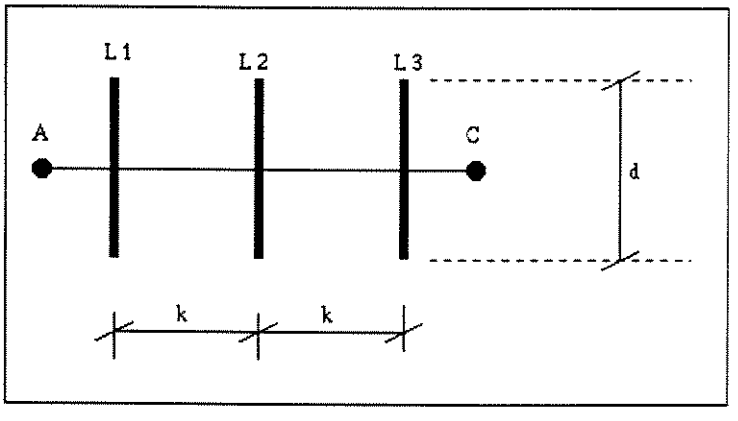


Figura 3.3.9 - Parâmetros envolvidos na pesquisa de um segmento de reta

Cada segmento imaginário $\mathbf{L}_n$ tem tamanho *d* e na hipótese de existir o segmento **AC**, *d* corresponde à largura máxima deste segmento. Tanto *d* quanto *k* são parâmetros definidos *a priori* com conhecimento do usuário. O parâmetro *k* pode ser igualado a 1 caso se deseje uma análise mais fina dos segmentos pesquisados. A medida que cada segmento $\mathbf{L}_n$ for

"traçado", verifica-se se são encontrados pixels de imagem sobre este segmento $\mathbf{L_n}$. Se for encontrado pelo menos 1 pixel num segmento $\mathbf{L_n}$, este é considerado existente e portanto, existe o pixel que compõem o segmento **AC** na posição *n* de **AC**.

Após todos os segmentos **L** entre os pontos **A** e **C** terem sido "traçados", as seguintes situações são possíveis:

1. em todas as posições de **AC** foram encontrados pixels da imagem, indicando desta forma a presença integral do segmento **AC**;
2. em alguns segmentos **L** não foram encontrados nenhum pixel da imagem. Esta situação indica que o segmento **AC** tem uma interrupção *g* (*gap*) nestas posições onde **L** foram "traçados" mas nenhum pixel da imagem foi encontrado sobre eles. Neste caso deve-se verificar as seguintes possibilidades:
   a) a interrupção é menor ou igual a um dado parâmetro *f*. Isto indica uma pequena falha na captura ou binarização da linha, podendo ser desconsiderada como interrupção. Veja a Figura 3.3.10 a seguir.

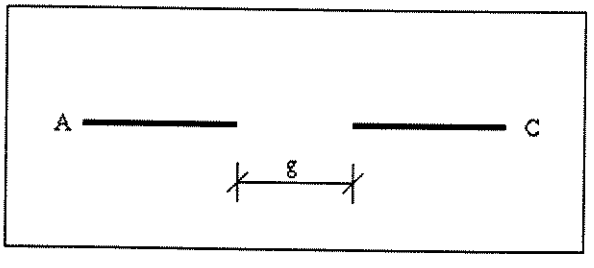


Figura 3.3.10 - Uma interrupção pequena (g < f)

   b) a interrupção *g* é maior que o parâmetro *f*. Isto significa que a falha deve ser considerada como tal, e desta forma a linha **AC** não é admitida como existente.

c) a existência de várias interrupções $f \leq g$ num intervalo constante indica a presença de linha tracejada (*dashed line*) no segmento **AC**. Veja Figura 3.3.11.

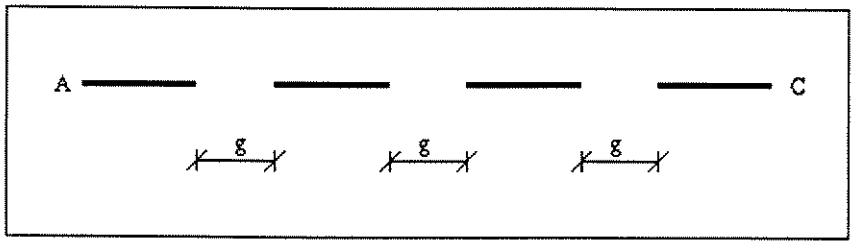


Figura 3.3.11 - Linha tracejada

3.3.6 DETALHES DA IMPLEMENTAÇÃO

O algoritmo de implementação do processo de vetorização descrito no item anterior e utilizado nesta pesquisa, é constituído dos seguintes passos:

algoritmo *vetorização*

1. Aquisição da Imagem

Todas as máscaras iniciais são lidas de arquivos ASCII sendo que suas rotações (múltiplas de 45 graus) são geradas pelo programa. Como exemplo, seja a máscara *(4)* da Figura 3.3.3, cujo arquivo de entrada é mostrado na Figura 3.3.12.

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

Figura 3.3.12 - Arquivo ASCII da modelo (4) da Figura 3.3.3

2. Armazenamento

O armazenamento das máscaras é feito em estruturas tais como a mostrada na Figura 3.3.13 a seguir.

Identif		Status		ndir	
dir1	dir2		dir3		dir4
Xcros			Ycros		
Largura			Altura		

Figura 3.3.13 - Estrutura de armazenamento de uma máscara

Os campos desta estrutura têm os seguintes significados:

a) Identif - corresponde à identificação da máscara

b) Status : = 0 - define a máscara fora de uso (usada para teste apenas);

= 1 - define a máscara em utilização.

c) ndir - corresponde à quantidade de direções válidas no próximo item.

d) dir1, dir2, dir3, dir4 - correspondem às direções das linhas encontradas nas máscaras a partir do ponto de referência (Xcros, Ycros). Numericamente elas têm os valores:

- 1 -norte
- 2 - sul
- 3 - leste
- 4 - oeste
- 5 - nordeste
- 6 - noroeste
- 7 - sudeste
- 8 - sudoeste

e) (Xcros, Ycros) - ponto de referência (pivô) da máscara, ou seja, ponto a partir do qual partem os segmentos de reta que compõem a máscara.

f) Largura e Altura - dimensões da máscara (em bits).

g) valores que constituem a imagem:

- 0 - representa o fundo da imagem
- 1 - representa um pixel da imagem
- -1 - não importa o valor (don't care).

Seja por exemplo a máscara *(4)* da Figura 3.3.3, cujo *bitmap* foi mostrado na Figura 3.3.12 e é repetido na Figura 3.3.14, em forma matricial.

Esta máscara é caracterizada pelos seguintes itens:

	0					5					10
0	1	1	1	1	1	1	1	1	1	1	1
	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
5	0	0	-1	-1	-1	1	-1	-1	-1	0	0
	0	0	-1	-1	-1	1	-1	-1	-1	0	0
	0	0	-1	-1	-1	1	-1	-1	-1	0	0
	0	0	-1	-1	-1	1	-1	-1	-1	0	0
	0	0	-1	-1	-1	1	-1	-1	-1	0	0
10	0	0	-1	-1	-1	1	-1	-1	-1	0	0

Figura 3.3.14 - Bitmap da máscara (4) da Figura 3.3.3 com a apresentação da estrutura de formação

- ponto de referência : (xcros, ycros) = (5, 0);
- segmento1 : (5, 0) , (0, 0) - direção 4 (oeste)
- segmento2 : (5, 0), (10, 0) - direção 3 (leste)
- segmento3 : (5, 0), (5, 10) - direção 2 (sul)
- identificação: número 4

3. Enjanelamento

A presença de cada máscara é verificada na imagem. A área de abrangência na matriz da imagem com fins de verificação é aquela definida pela lista dos elementos conexos, obtida na etapa anterior da pesquisa. Desta forma, ao invés de se pesquisar toda a imagem, somente a área que contém elementos é verificada. Seja por exemplo um trecho de imagem de um circuito elétrico, mostrado na Figura 3.3.15.

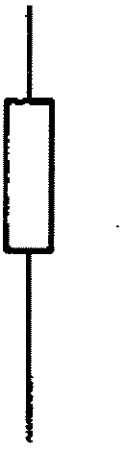


Figura 3.3.15 - uma imagem binária TIFF

- *bitmap* de parte desta imagem (o topo) é mostrado na Figura 3.3.16.

***** IMAGEM INICIAL *****

```
0000000000000000000000111000000000000000000
0000000000000000000000111000000000000000000
0000000000000000000000111000000000000000000
0000000000000000000000111000000000000000000
0000000000000000000000111000000000000000000
0000000000000000000000111000000000000000000
0000000000000000000000111000000000000000000
0000000000000000000000111000000000000000000
0000000000000000000000111000000000000000000
0000000000000000000000111000000000000000000
0000000000000000000000111000000000000000000
0000000000000000000000111000000000000000000
0000000000111111100011111111111111111100000
0000000001111111111111111111111111111110000
0000000001111111111111111111111111111110000
0000000001111001111000011111111111111110000
0000000001110000000000000000000000001110000
0000000001110000000000000000000000001110000
0000000001110000000000000000000000001110000
0000000001110000000000000000000000001110000
0000000001110000000000000000000000001110000
0000000001110000000000000000000000001110000
0000000001110000000000000000000000001110000
0000000001110000000000000000000000001110000
0000000001110000000000000000000000001110000
0000000001110000000000000000000000001110000
0000000001110000000000000000000000001110000
```

Figura 3.3.16 - Bitmap do topo da imagem na Figura 3.3.15

4. Casamento de Padrões

Cada máscara varre toda a imagem, pixel a pixel, buscando uma posição onde a máscara "casa" com a imagem, exatamente como acontece na operação morfológica denominada *HIT OR MISS* [Serra 1982] a qual pode ser usada para se detectar formas, isto é todos os 1s da imagem devem coincider com todos os 1s da máscara, o mesmo acontecendo para todos os 0s, e existem ainda os -1s, que significam o "don't care" que correspondem a não importa os valores das máscaras e da imagem pois não serão verificados. Encontrada uma máscara numa determinada posição da imagem cujos pixels 0 e 1 "casam"com a imagem, o pixel da imagem correspondente à posição

do pivô da máscara, recebe um rótulo igual ao rótulo de identificação da máscara. Isto significa que a posição de casamento da máscara na imagem corresponde ao seu ponto (Xcros, Ycros) de referência. Quando uma posição da imagem é ocupada por uma máscara, esta posição não é ocupada por outra máscara numa próxima pesquisa, ou seja, são atribuidas prioridades na pesquisa das máscaras na imagem (a ordem de prioridade é aquela mostrada na Figura 3.3.3). Por exemplo, a máscara *(8)* tem prioridade de pesquisa sobre a máscara *(18)* pois a primeira incorpora a segunda. Veja na Figura 3.3.17 o resultado desta etapa quando aplicado à imagem da Figura 3.3.16.

***** IMAGEM FINAL *****

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 01 01 01 01 01 01 00 00 00 01 01 18 18 18 01 01 01 01 01 01 01 01 01 14 14 00 00 00 00 00
00 00 00 00 00 00 00 00 00 12 12 12 01 01 01 01 01 01 01 01 01 08 08 08 01 01 01 01 01 01 01 01 01 14 14 14 00 00 00 00
00 00 00 00 00 00 00 00 00 12 12 12 01 01 01 01 01 01 01 01 01 08 08 08 01 01 01 01 01 01 01 01 01 14 14 14 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 01 00 00 01 01 01 01 00 00 00 00 18 01 01 01 01 01 01 01 01 01 14 14 14 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 00 00 00
```

Figura 3.3.17 - Bitmap da imagem da Figura 3.3.15 após a determinação das máscaras

Observação: a máscara 01 para ser encontrada é modificada inicialmente para 99, a fim de não se confundir com o pixel 01.

5. *Decisão*

Quando várias máscaras casam próximas umas das outras, formam uma região de máscaras e assim sendo, deve-se decidir qual máscara realmente representa aquela região (veja por exemplo na Figura 3.3.17, vários rótulos 12 formando uma região no canto esquerdo do desenho, indicando ali a presença do modelo 12 da Figura 3.3.3, o mesmo acontencendo com a região de rótulo 14 à direita da imagem, e ainda a região central onde aparecem os rótulos 8 e 18, indicando ali o casamento das máscaras 4 e 12, ambas rotacionas). Para isto toma-se como representante aquela que tem maior freqüência de ocorrência na região em estudo (ainda no exemplo da Figura 3.3.17, a máscara 8 é a representante final da região central, visto que ela tem 6 pixels contra 4 da máscara 18). Neste caso, a região final é constituída por todos os pixels onde as máscaras casam e são adjacentes, mas tem como valor representativo aquele da máscara com maior frequência de ocorrência.

6. *Estrutura de vértices*

Após todos os nós da lista de elementos conexos serem analisados para todas as máscaras, é gerada uma lista de vértices, onde a estrutura de cada nó é mostrada na Figura 3.3.18. Esta lista contém todos os vértices da imagem onde se detectou a presença de uma determinada máscara. Como mostrado na Figura 3.3.18, a estrutura de vértice guarda informações a respeito da máscara encontrada, em que posição

(centróide) isto ocorreu, direções, frequência e faixa de variação (com relação à referência do vértice) de ocorrência nas direções X e Y.

Identif		Status	ndir
dir1	dir2	dir3	dir4
X	Y	mark	times
Xmin	Xmax	Ymin	Ymax

Figura 3.3.18 - Estrutura de um vértice

- Identif, Status, ndir e dir1 ... dir4 - definem o grupo ao qual o vértice pertence. Estas informações são definidas pela máscara armazenada no vértice (veja Figura 3.3.13).

- X, Y - posição do vértice;
- mark - registra a ordem de ocorrência do vértice;
- times - número de vezes que o vértice ocorre na imagem nas vizinhanças do ponto (X, Y).
- Xmin, Xmax, Ymin, Ymax - faixa de variação do deslocamento do vértice tendo como referência o ponto (X, Y).

7. *Determinação dos vetores*

Com a lista de vértices definida, o último passo do algoritmo é buscar entre todos os vértices determinados, as possíveis linhas que os interligam. Esta busca é realizada de forma ordenada usando para isto as direções contidas em cada vértice como guias de busca.

8. Término

A saída do algoritmo é a lista dos segmentos de reta que compõem a imagem.

fim do algoritmo

3.3.7 RESULTADOS

Como exemplo, pode-se observar na Figura 3.3.19 os resultados da metodologia de vetorização desenvolvida nesta etapa da pesquisa, aplicada ao circuito da Figura 3.2.10 (após retirados os caracteres). Comparando-a com a Figura 3.3.1, vê-se que houve sensível melhora na representação da imagem com relação `a utilização de afinamento, pois os vetores são realmente contínuos e não se tem a presença dos "espinhos" nas linhas. Ainda comparando-se as duas figuras, tem-se **455** vetores desenhados na Figura 3.3.1 contra apenas **121** vetores na Figura 3.3.19, obtida pelo novo processo de vetorização desenvolvido para esta etapa da pesquisa.

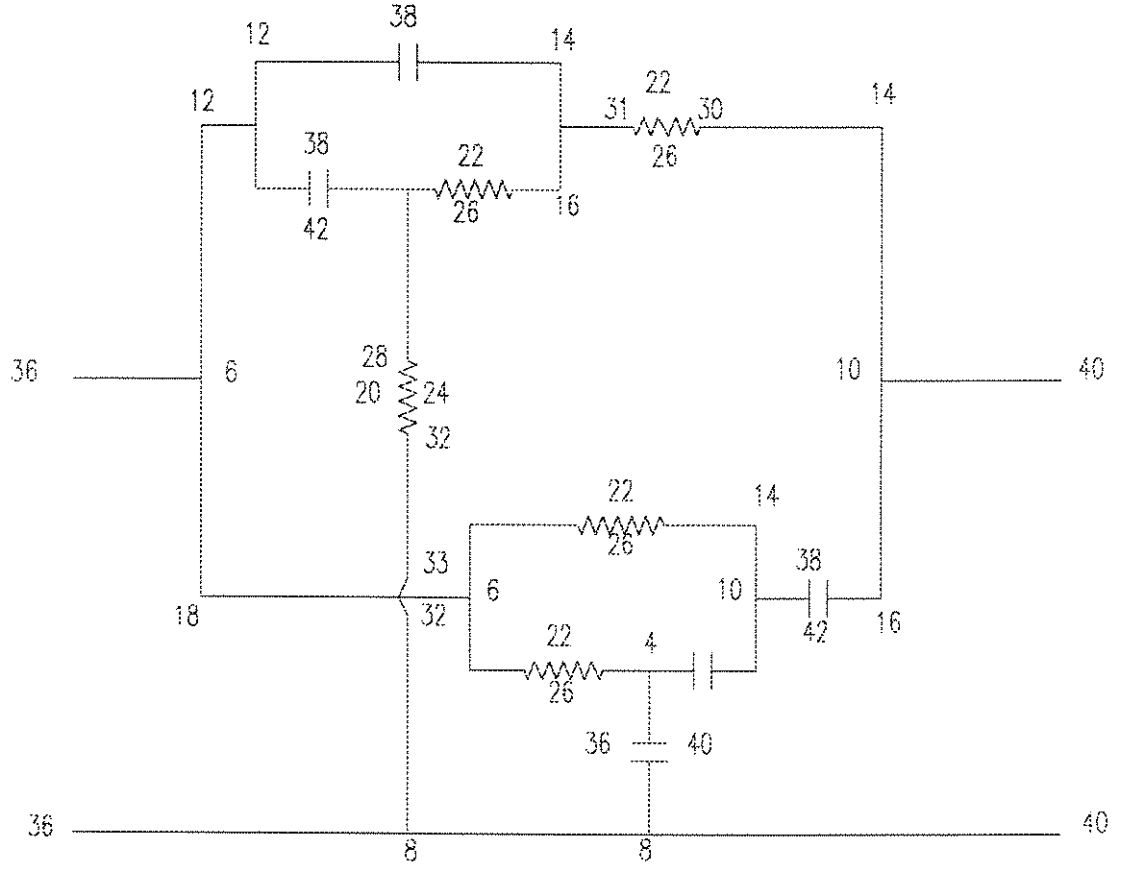


Figura 3.3.19 - Vetorização do circuito da Figura 3.2.10 apresentando os números de identificação de algumas das máscaras utilizadas para a segmentação dos vetores conforme Figura 3.3.3.

3.4 EXTRAÇÃO DE CARACTERÍSTICAS

3.4.1 OBJETIVO

Num diagrama de circuito elétrico, símbolos são conectados a outros símbolos formando uma malha bastante densa de segmentos, dificultando desta forma a correta identificação dos elementos que compõem esta imagem. Por outro lado, em diagramas de circuitos elétricos ou eletrônicos existem dois tipos básicos de símbolos [Okazaki e outros 1988]: **fechado** ou **laço** (*loop*) e **aberto**. Os símbolos fechados são aqueles formados por um ou mais polígonos (laços), enquanto que os símbolos abertos são constituídos de segmentos dispostos de tal maneira a não formarem laços fechados, tal como acontece com os símbolos de terra, capacitor e resistor.

Tendo em vista a minimização dos laços que compõem os símbolos fechados, optou-se nesta pesquisa primeiro pela segmentação dos símbolos abertos, pois uma vez detectados, fica facilitado a segmentação dos demais elementos do circuito. Em seguida faz-se a segmentação dos símbolos fechados através do estudo de suas características.

Este é o objetivo desta etapa desta pesquisa: *selecionar* e *extrair* as características mais importantes no reconhecimento de símbolos abertos e fechados, evidenciando as diferenças e similaridades entre eles. De uma forma geral, a extração de características é um processo usualmente associado à analise de segmentos de uma imagem [Duda e Hart 1973]. Estas características são usadas num classificador para distinguir cada segmento. Características podem ser usadas para descrever um objeto na imagem, uma coleção de objetos ou toda a imagem.

Um guia de como se extrair características pode ser encontrado em [Sklansky 1978] e seu resumo é mostrado a seguir.

a) Conhecer o usuário do sistema: a satisfação do usuário aumenta à medida que suas frustrações com o sistema são eliminadas. Portanto é importante saber do usuário o que e o quanto ele deseja do sistema para que este aperfeiçoe seu trabalho. Outro ponto importante é trazer para o sistema os conhecimentos do usuário sobre as características dos objetos em estudo.

b) Fazer o pré-processamento de forma criteriosa: o processo de digitalização, o tipo de digitalizador ou sistema ótico, filtros e histogramas usados no manuseio da imagem têm efeitos significativos no desempenho das rotinas de segmentação e extração de características.

c) Usar *hardware* conveniente: o processamento digital de imagens geralmente requer grande volume de dados e também muitos cálculos. Portanto especial atenção deve ser tomada quando da realização da extração das características, pois o *hardware* deve ser suficientemente capaz de manusear os dados sem produzir a sensação de desconforto para o usuário, como por exemplo muito tempo de processamento ou falta de espaço na memória de trabalho da máquina.

A maioria das técnicas utilizadas na extração de característica são baseadas em medidas de distâncias estatísticas [Sklansky 1978]. Extratores de características podem ser categorizados da seguinte forma:

a) descritores de forma (bordas);

b) descritores de textura;

c) descritores sintáticos (relações espaciais).

A abordagem destes descritores pode ser encontrada em [Sklansky 1978 e Bribiesca 1979].

Nesta pesquisa, um objeto pode ser definido como sendo um conjunto de partes da imagem que juntas possuem características determinadas. Por exemplo, um resistor pode ser representado por um conjunto de segmentos de retas dispostos fisicamente como um dente de serra, ou seja, segmentos seqüenciais e inclinados alternados, situados entre dois segmentos colineares (resistor R2 - Figura 3.4.1).

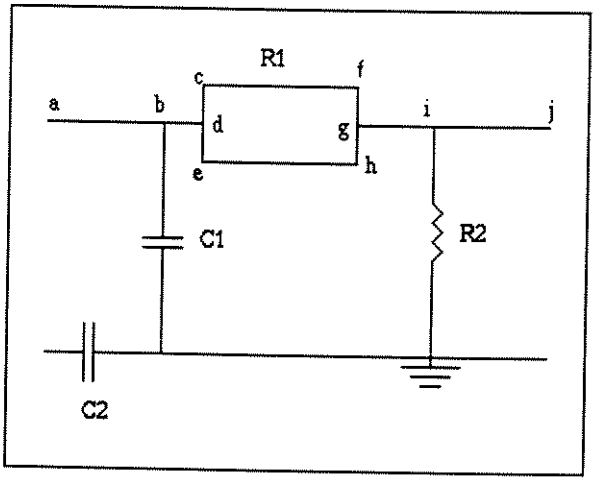


Figura 3.4.1 - características encontradas num diagrama de circuito elétrico

A seleção e extração de características estão intimamente ligadas à classificação dos objetos da imagem: quanto mais **discriminantes** forem as características, mais robusto será o classificador para determinar quais objetos estão presentes na imagem.

A saída da etapa de extração de características é a descrição da imagem em forma de atributos, a qual será usada na fase de classificação dos objetos.

3.4.2 SELEÇÃO DE CARACTERÍSTICAS NA IMAGEM

Um método de se identificar características discriminantes numa imagem é olhá-la como um ser humano o faria para obter informações precisas de modo a diferenciar um objeto de outro. Por exemplo, analisando-se a imagem de um circuito elétrico, como se pode diferenciar entre um capacitor e o símbolo de aterramento (Figura 3.4..1)? Para tomar-se a decisão correta é necessário estudar-se as características de cada objeto e verificar-se qual objeto tem características mais próximas daquelas que se quer reconhecer, ou seja, quão **similar** um objeto é do outro. Neste exemplo pode-se definir como características necessárias ao reconhecimento dos objetos, informações tais como a forma, o tamanho, a presença ou ausência de elementos próprios (particulares) do objeto. No exemplo acima, pode-se descrever o símbolo de aterramento por vários segmentos paralelos de tamanhos diferentes e o capacitor como dois segmentos paralelas iguais. Quanto mais próximas as características dos objetos em estudo, mais detalhados deverão ser os métodos de diferenciação entre eles, o que torna os algoritmos mais elaborados e voltados para atender objetos de sistemas específicos. É ainda muito mais difícil criar-se um sistema aberto capaz de caracterizar qualquer tipo de objeto.

Num sistema de análise de desenhos de engenharia deve-se determinar algumas características como importantes para a tarefa de reconhecimento, as quais são agrupadas num **vetor de características** C de um espaço de características N-dimensional, tal que:

$$C=\begin{bmatrix} C_1 \\ C_2 \\ . \\ . \\ . \\ C_n \end{bmatrix}$$

onde C_1, C_2, ..., C_n, são as características definidas para um determinado sistema.

Nesta pesquisa, as características mais importantes usadas para a classificação dos símbolos são:

a) tamanho dos segmentos que compõem a imagem:
segmentos que pertencem a símbolos têm tamanhos bem menores que os segmentos que interligam este símbolos;

b) distância entre os segmentos:
a distância entre os segmentos do símbolo de aterramento ou do capacitor é regular e uniforme no desenho, enquanto que entre segmentos quaisquer as distâncias são variadas;

c) números de laços:
símbolos sem laços (abertos) e com 1 ou mais laços (fechados) podem ser diferenciados por esta característica;

d) forma geométrica dos símbolos:
podem ser definidas pela presença de uma quantidade conhecida de ângulos específicos no símbolo. Conjuntos de segmentos paralelos e/ou ortogonais podem ajudar na definição de uma forma geométrica caracterizante de um símbolo.

Pela Figura 3.4.1 pode-se observar algumas das características anteriormente listadas nos itens a), b) c) e d) aplicadas a um desenho de um diagrama de um circuito elétrico. Por exemplo, pode-se notar que os capacitores **C1** e **C2** são formados por 2 segmentos paralelos de mesmo tamanho e o aterramento é formado por vários segmentos paralelos de tamanhos diferentes; o símbolo do resistor **R2** é constituído de vários segmentos formando um dente de serra; O símbolo do resistor **R1** é representado através de um retângulo, o qual apresenta segmentos formando ângulos retos e por conseqüência apresenta conjuntos de segmentos paralelos ortogonais. Por último deve-se notar a presença de segmentos que interconectam os elementos mencionados.

3.4.3 EXTRAÇÃO DE CARACTERÍSTICA NA IMAGEM

Quando se compara um objeto com outro, o objetivo é ter uma descrição **discriminante** do objeto, em termos de suas formas básicas representadas pelos seus atributos ou características, de tal forma que comparações errôneas sejam eficazmente rejeitadas e comparações corretas sejam realizadas sem ambigüidades. A natureza da descrição do objeto ou da imagem dependerá da aplicação, das **formas** dos objetos e do número de objetos a serem comparados.

Por exemplo, se o objetivo é reconhecer textos e gráficos, onde todos os gráficos consistem de longos segmentos, então características tais como tamanho das regiões ou comprimento dos contornos, podem ser usados como parâmetros discriminatórios. Para reconhecer a diferença entre quadrados, triângulos e círculos, pode-se aplicar métodos de descrição de curvaturas e dobras em segmentos. Como já mencionado, é impossível criar-se um sistema geral, próprio para reconhecer qualquer objeto. Portanto, cada aplicação deve ter características próprias a serem determinadas e extraídas da imagem. O importante a ser observado na metodologia empregada é que as características selecionadas devem ser apropriadas para a aplicação e devem garantir a discriminação dos objetos envolvidos. Para maiores detalhes sobre técnicas empregadas em descritores de formas, veja [Pavlidis 1978 e Marshall 1989].

Vê-se portanto que a distinção entre objetos que compõem uma imagem é realizada em função de informações de medidas extraídas de cada objeto. Estas medidas são chamadas de características (*features*) - [Schalkoff 1992], atributos ou propriedades. Daí a importância em selecionar quais características são importantes numa imagem, pois desta escolha dependerá a eficácia do sistema de reconhecimento. É muito importante que se escolha características relevantes à tarefa em execução, levando-se em conta também se elas são computacionalmente executáveis, conduzam a um bom resultado no reconhecimento e reduza o volume de dados sem perder informações vitais. Por último deve-se considerar também algumas restrições ou dificuldades nos

sistemas ou meios de medidas, o que pode limitar o conjunto de características extraídas.

Nesta pesquisa, os dados necessários para esta etapa são constituídos unicamente de segmentos de retas oriundos da vetorização da imagem do diagrama de circuito, os quais podem representar objetos abertos, objetos fechados ou simplesmente segmentos de interconexão entre objetos na imagem. Estes segmentos de retas são comparados contra regras pré-definidas que confirmam ou não a ocorrência dos objetos na imagem. Para os segmentos do conjunto de entrada que não formam laços, duas são as possibilidades: eles constituem os símbolos abertos ou então os segmentos de interconexão. Uma forma de minimizar o trabalho computacional envolvido na extração de características é obter em primeiro lugar as características dos símbolos abertos e em seguida as características dos símbolos fechados. Isto é simples de se perceber através do exemplo na Figura 3.4.2, pois uma vez detectados os segmentos que constituem os símbolos abertos (no caso, os resistores), estes são retirados da lista de segmentos que ainda precisam ser pesquisados no conjunto de entrada. Desta forma, se estes símbolos abertos formavam um falso laço, após a retirada deles, os falsos laços são desfeitos, evitando-se que estes sejam pesquisados no processo de segmentação dos laços.

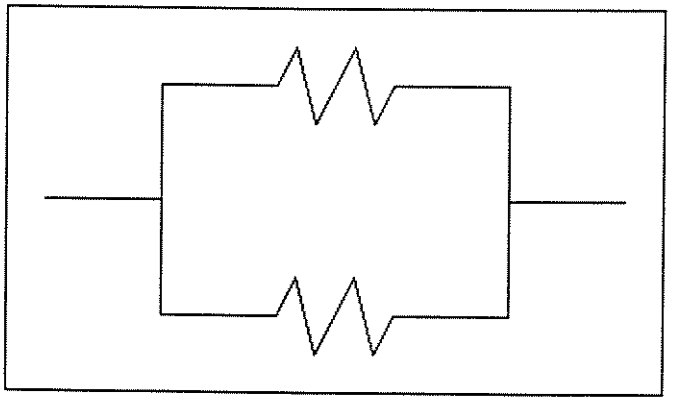


Figura 3.4.2 - Falso laço formado por símbolos abertos

Entre os segmentos restantes, buscam-se agora os objetos fechados (veja Item 3.5). Para cada objeto fechado, busca-se um laço que o compõe, e se este laço é confirmado como sendo

a ocorrência de um objeto fechado na imagem, todos os atributos daquele objeto confirmado são computados e armazenados numa estrutura de dados criada para este fim.

As características escolhidas formam uma representação eficiente do objeto pois é constituída apenas dos valores dos atributos sobre os laços, tais como por exemplo número de lados do laço, coordenadas dos extremos de cada segmento que o constitui, ângulo entre dois segmentos consecutivos, etc. As medidas que quantizam o vetor de características **C** dos objetos que compõem a imagem, são obtidas através da análise de cada segmento da imagem. Cada segmento de reta $\mathbf{r}_i$ é comparado com cada segmento $\mathbf{r}_k$ (onde $0 \leq i \leq n$, $0 \leq k \leq n$, $i \neq k$) da imagem e portanto muitas informações podem ser obtidas: ângulos entre eles, distâncias, direções, tamanhos, etc.

Tendo-se estas informações, para todos os segmentos de reta da imagem que formam laços, obtém-se todas as medidas necessárias à definição do vetor de características **C** para os objetos fechados da imagem.

Todas os segmentos restantes que não constituem objetos abertos nem fechados, são considerados como sendo candidatos aos segmentos de interconexão, isto por que há a necessidade de se verificar se os mesmas estão conectados a algum símbolo ou ponto de conexão do circuito em estudo. Caso isto não ocorra, existe a probabilidade da existência de um erro, pois não é comum a presença de segmentos completamente desconectadas num circuito elétrico.

3.4.4 Reconhecimento dos Elementos Abertos

Todo o procedimento descrito no item anterior teve por fim obter as características dos objetos que constituem a imagem e assim sendo deve-se ter uma forma de armazenar os resultados obtidos nesta fase para serem utilizados na classificação dos objetos extraídos.

A estratégia utilizada nesta pesquisa para a extração de características dos símbolos que constituem o desenho de um diagrama de circuito elétrico, foi baseada no fato de que cada objeto no circuito é constituído de um conjunto de segmentos cujos pontos delimitantes já foram extraídos durante a fase de vetorização. Durante a fase de extração de características, obtém-se as medidas que caracterizam os objetos na imagem. Como os objetos a serem extraídos podem ser abertos ou fechados, e como os objetos abertos têm características conhecidas *a priori*, o que não acontece com os objetos fechados, decidiu-se nesta pesquisa, por priorizar a extração de características dos objetos abertos. O processo empregado obtém não somente as características, mas também já classifica o objeto cujo símbolo é aberto.

Uma vez reconhecido, o símbolo aberto é armazenado numa **lista de símbolos reconhecidos**, cuja estrutura é mostrada na Figura 3.4.3 a seguir.

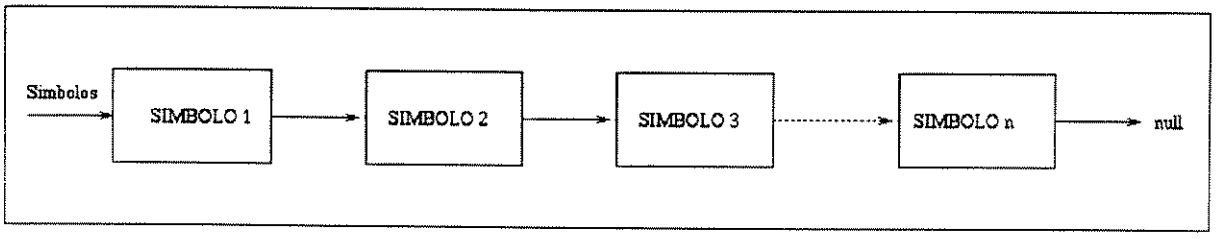


Figura 3.4.3 - Lista final dos símbolos reconhecidos num desenho

A estrutura de cada símbolo armazenado nesta lista é constituída das seguintes informações:

- quantidade de entradas;
- quantidade de saídas;
- coordenadas dos pontos de entrada;
- coordenadas dos pontos de saída;
- tipo do símbolo conforme descrições na Tabela 3.4.1
- valor numérico que o símbolo representa.

tipo	nome	unidade	símbolo
1	resistor	ohm	R
2	capacitor	farad	F
3	gerador	volts	V
4	terra		
5	transistor NPN		NPN
6	transistor PNP		PNP

Tabela 3.4.1 - tipo dos elementos abertos

A lista de símbolos reconhecidos será então utilizada na criação da *net list* do circuito em estudo, a qual será usada por exemplo no Spice e no CAD para futuras análises.

Descritores sintáticos provêm descrições das relações espaciais entre objetos numa imagem e descrições das relações entre características. Tais descritores podem variar do mais simples ao mais complexo, dependendo é claro do objeto descrito. Eles diferem principalmente em quanto conhecimento *a priori* tem-se sobre a forma do objeto que se quer descrever. Nesta pesquisa, decidiu-se por representar o conhecimento dos símbolos abertos, por regras fixas no *software*. A princípio parece ser este procedimento bastante desvantajoso, visto que a representação dos objetos não pode ser flexível. No entanto, como se conhece *a priori* a comunidade de símbolos abertos possíveis de serem utilizados num diagrama de circuito, esta desvantagem desaparece frente ao fato de se ter que construir um *software* bastante sofisticado para reconhecer um símbolo qualquer. Assim sendo, as características dos símbolos abertos resistor, capacitor, terra, gerador e transistor, são obtidas através dos conhecimentos *a priori*, descritos conforme as regras mostradas a seguir para cada um deles. Qualquer novo símbolo pode ser facilmente incluído no sistema, através da inclusão de sua respectiva rotina contendo as regras descritivas do mesmo.

A seguir são apresentadas as regras para identificação dos símbolos abertos reconhecidos pelo sistema SRIDE.

a) RESISTOR

- obter segmentos seqüenciais de tal forma que cada segmento se conecte apenas com sua sucessora e seu antecessor, não havendo portanto junções com mais de 2 segmentos em qualquer dos vértices formados pelos segmentos em questão;
- o ângulo entre qualquer dois segmentos consecutivos deverá ser diferente de 180 graus;
- o conjunto de segmentos seqüenciais deverá estar entre dois segmentos horizontais (resistor horizontal) ou verticais (resistor vertical);
- para resistor horizontal, os valores de **x** de cada vértice de segmento que o compõe, deverão também ser seqüenciais e os valores de **y** deverão ser alternados, formando desta forma um "dente de serra" horizontal;
- o mesmo acontece com o resistor vertical, trocando-se o **x** pelo **y** no item anterior.

b) CAPACITOR, GERADOR e TERRA:

- Obter os segmentos onde um de seus vértices não se conectem com nenhum outro segmento;
- dentre os segmentos obtidas, obter aquelas que são colineares com um vértice comum;

- dentre os segmentos colineares, obter aquelas cujos pares correspondentes sejam paralelos (usar como referência a mesma mediatriz: segmento que passa ortogonalmente entre as dois segmentos colineares que formam o par de segmentos colineares. Exemplo: os segmentos **CE** e **DG** na Figura 3.4.4 são as mediatrizes dos capacitores considerando-se os extremos dos segmentos que os compõem);

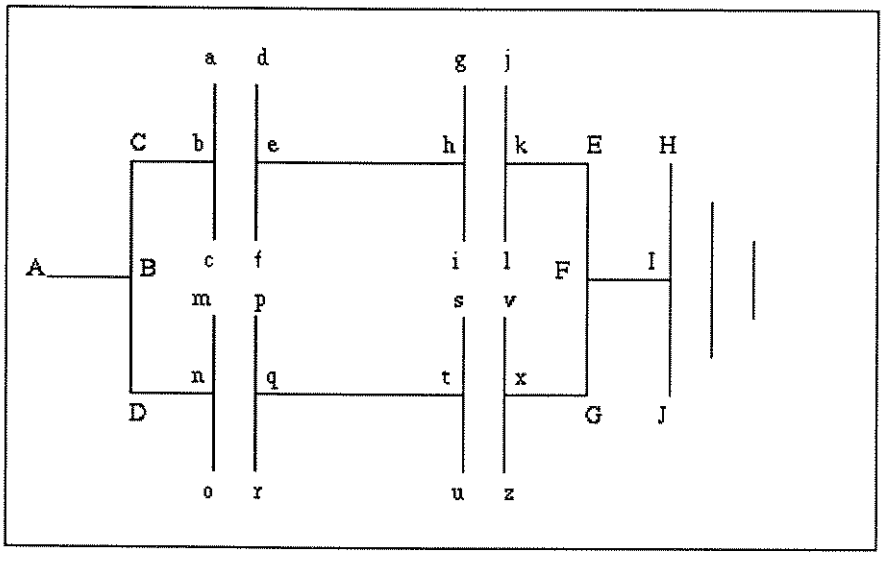


Figura 3.4.4 - Exemplo de um trecho de circuito apresentando elementos abertos

Exemplos:

Inicialmente, nestas condições, os pares paralelos são:

ab, de, gh, jk

bc, ef, hi, kl, HI

mn, pq, st, vx, IJ

no, qr, tu, xz

- Considerando-se apenas os pares dos segmentos colineares, tem-se:

  ac, df, gi, jl

  mo, pr, su, vz

  HJ

- agrupá-las de tal forma que cada grupo represente um conjunto de segmentos colineares 2 a 2 e paralelas entre si, tendo como referência a mesma mediatriz; Exemplo:

  grupo1 : **ac, df, gi, jl** - mediatriz: **CE**

  grupo 2: **mo, pr, su, vz** - mediatriz: **DG**

- Seja **L** um segmento formado por outros dois segmentos colineares, conforme descrito anteriormente. Seja **T** outro segmento formado da mesma forma. Para cada segmento **L** não marcada de cada grupo, obter qual é o segmento **T** mais próxima daquele grupo, desde que não exista nenhum segmento entre eles.

- Se encontrado um par de segmentos **L** e **T** nestas condições, e se os tamanhos deles forem iguais, tem-se um capacitor. Se os tamanhos forem diferentes, tem-se um gerador. Se não for encontrado um par para o segmento **L** em estudo, possivelmente ela é origem do símbolo terra. Para se obter as outros segmentos que constituem este símbolo, buscar dentre os segmentos não agrupados, aqueles que são paralelas a **L**, tendo como referência a mediatriz de **L**, que têm tamanhos menores que **L**, que sejam isolados (nenhum outro segmento é conectado à elas) e que sejam consecutivas (nenhum outro segmento deverá estar entre eles).

c) TRANSISTORES

- obter grupos com 4 segmentos colineares;
- dentre os grupos de 4 segmentos colineares obtidos anteriormente, verificar se o vértice entre o segundo e o terceiro segmento tem um segmento conectado à 90 graus dos segmentos colineares;

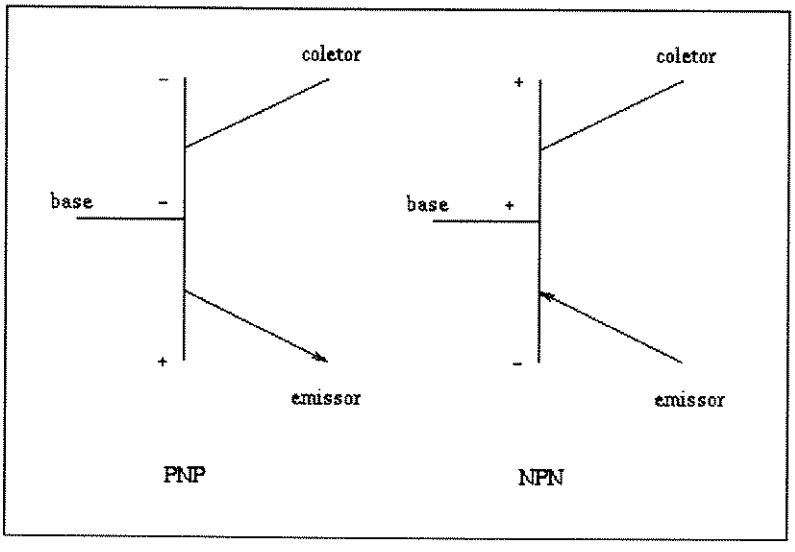


Figura 3.4.5 -Tipos de transistores

- verificar se os vértices entre o primeiro e segundo segmento e entre o terceiro e quarto segmento, têm um segmento conectado com ângulo diferente de 90 graus dos segmentos colineares. Os valores destes ângulos devem ser próximos entre si;
- determinar o tipo do emissor através da detecção da seta que o constitui;

Todas os segmentos que pertençam a símbolos reconhecidos, devem ser **marcados** a fim de se otimizar a busca dos outros símbolos. Desta forma, segmentos já utilizados num símbolo não serão utilizados em outros, e portanto não devem ser pesquisados novamente.

3.4.5 Elementos Fechados

Reconhecidos os símbolos abertos, restam ainda os símbolos fechados e os segmentos de interconexão entre os símbolos. Como os objetos fechados podem representar símbolos de diversos tipos, estes não podem ser especificados com conhecimento *a priori* de suas características, como ocorre com os símbolos abertos. Desta forma, o processo aqui utilizado cria um banco contendo as informações das características de cada laço encontrado na imagem. Os objetos deste banco são depois classificados conforme descrito no próximo item deste capítulo. Cada elemento do banco corresponde às características obtidas de um objeto na imagem em estudo e contém as seguintes informações :

C_1 - quantidade de laços;

C_2 - quantidades de vértices;

C_3 - quantidade de laços colados (laços com pelo menos um lado comum);

C_4 - quantidade de ângulos = 90° ;

C_5 - quantidade de ângulos > 90° ;

C_6 - quantidade de ângulos < 90° ;

C_7 - quantidade total de segmentos;

C_8 - quantidade de segmentos colineares;

C_9 - quantidade de segmentos ortogonais;

C_{10} - quantidade de segmentos paralelos;

C_{11} - quantidade de portas de entradas ou saídas.

Esta é a estrutura de cada elemento do banco de dados que armazena os resultados da extração de características da imagem, para os elementos fechados. Como exemplo, seja parte da imagem da Figura 3.4.1 (o resistor R1) repetida aqui na Figura 3.4.6.

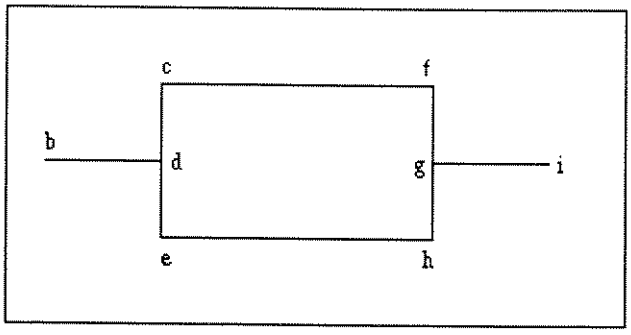


Figura 3.4.6 - Resistor R1 da Figura 3.4.1 mostrando vetores que o compõe

Após a obtenção dos laços e a extração de características, tem-se o seguinte vetor de características para esta imagem:

C_1 = q-laços = 1 - **cfghedc**

C_2 = q-vértices = 6 - **c, f, g, h, e, d**

C_3 = q-laços-att = 0

C_4 = q-ang=90 = 4 - **c, e, f, h**

C_5 = q-ang>90 = 2 - **d, g**

C_6 = q-ang<90 = 0

C_7 = q-segmentos = 6 - **cf, fg, gh, he, ed, dc**

C_8 = q-lin-colin = 2 - correspondem aos conjuntos de segmentos que têm uma das extremidades comum ao outro segmento com ângulo = 180^0.

São eles os segmentos: **cd** e **de; fg** e **gh**

C_9 = q-lin-orto = 8 - são os conjuntos de segmentos que têm uma das extremidades comum ao outro segmento com ângulo = 90^0.

São os segmentos:

cd com **cf**

de com **eh**

eh com **de** e **gh**

gh com **eh**

fg com **cf**

cf com **cd** e **fg**

os segmentos **bd** e **gi** não pertencem ao laço; são considerados segmentos de interconexão;

C_{10} = q-lin-paral = 3 - são os conjuntos de segmentos que têm em comum a mesmo segmento ortogonal à elas e que passa pelas suas mediatrizes.

São os segmentos: **cd** e **fg**

de e **gh**

cf e **eh**.

C_{11} = q-ports = 2 - são os vértices do objeto comuns aos segmentos de interconexão.

São eles: **d**, **g**

Estas informações serão utilizadas na próxima e última fase do processo de reconhecimento: a classificação dos elementos fechados envolvidos na imagem.

3.5 LAÇOS

3.5.1 OBJETIVO

O objetivo desta etapa é pesquisar a existência de símbolos fechados na imagem, os quais orientarão na classificação de símbolos fechados pertencentes ao desenho em estudo. Estes símbolos fechados são obtidos a partir da lista de segmentos de retas que compõem a imagem. Uma vez identificados os símbolos fechados, os segmentos de retas restantes, aqueles que não fazem parte de nenhum símbolo fechado, são posteriormente identificados na fase de reconhecimento como possíveis linhas de interconexão.

3.5.2 DEFINIÇÃO DO PROBLEMA

Nesta pesquisa, a busca de laços visa a associação dos mesmos aos símbolos disponíveis no universo de símbolos dos desenhos de diagramas de circuitos elétricos que se deseja reconhecer.

Esta associação de símbolos pode ser feita por exemplo, através do casamento de formas (*template matching*) ou através da comparação de características oriundas do objeto em estudo. Estes aspectos são mostrados em outro item deste trabalho, na fase de reconhecimento. O importante a ser observado é que os laços são uma boa forma de se reconhecer a maioria das imagens provenientes de diagramas de circuitos elétricos, visto que estas são constituídas em grande parte por eles.

Basicamente, laços são compostos por um ou mais **laços primitivos**. Um laço é primitivo dentro de um símbolo fechado se ele não é a união de dois ou mais polígonos deste

símbolo, como por exemplo, um triângulo, retângulo, um paralelogramo, etc. Na Figura 3.5.1 tem-se exemplos de laços num trecho de diagrama de circuito elétrico.

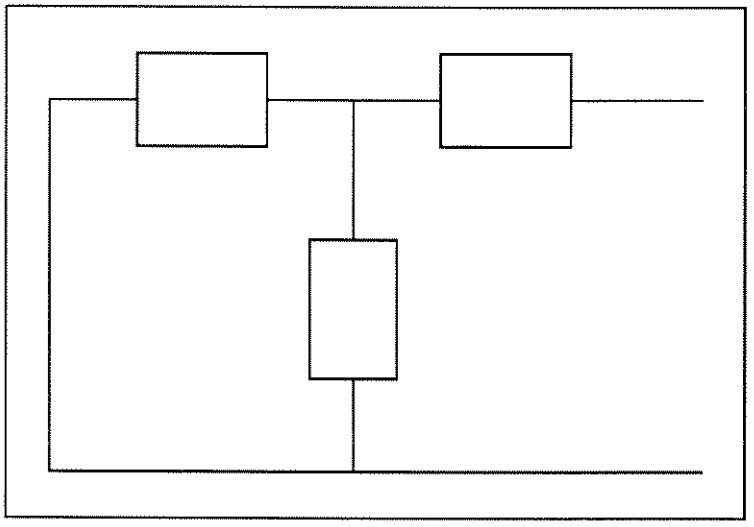


Figura 3.5.1 - Um trecho de diagrama de circuito elétrico contendo laços

Um aspecto importante na identificação de laços numa imagem é a independência destes com relação às transformações geométricas, ou seja, o tamanho, a direção e a sua posição na imagem, as quais não devem ser fixas. Nesta pesquisa pode-se ter variações em tamanhos, 8 direções são disponíveis e o posicionamento do laço é livre na imagem.

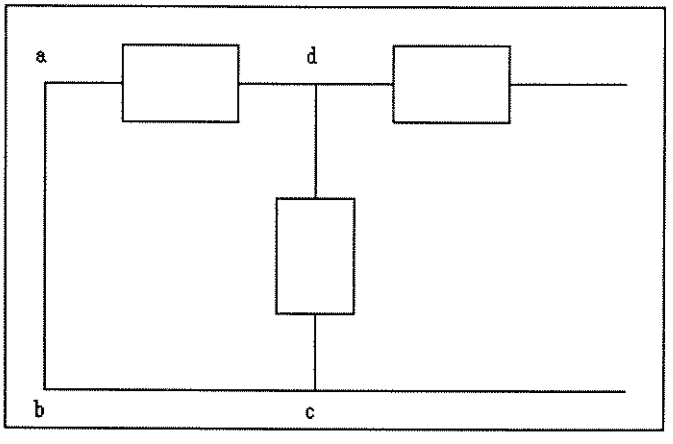


Figura 3.5.2 - Mostrando falsos laços num trecho de circuito elétrico

Outra tarefa é minimizar o problema de se detectar falsos laços (aqueles que não fazem partem de um símbolo), ou seja, aqueles formados quando linhas conectando os símbolos interceptam-se. Na Figura 3.5.2, vê-se um falso laço formado pelos segmentos entre os pontos **a, b, c, d,** os quais apenas interconectam os símbolos do circuito elétrico.

Por último, existe o problema de se obter a quantidade mínima de laços pertencentes a um símbolo. Por exemplo na Figura 3.5.3, o desenho mostrado é constituído de 8 linhas e 6 laços podem ser definidos, no entanto, somente 3 laços seriam suficientes para se definir o mesmo desenho.

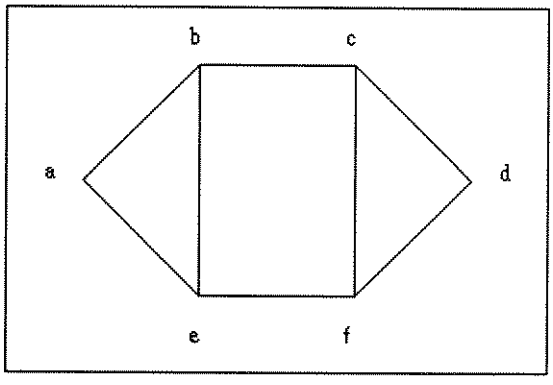


Figura 3.5.3 - Mínimos laços

Pelo processo geral de obtenção de laços, os laços obtidos são:

$laço_1$ = { a, b, c, f, e, a }

$laço_2$ = { b, c, d, f, e, b }

$laço_3$ = { a, b, c, d, f, e, a }

$laço_4$ = { a, b, e, a }

$laço_5$ = { b, c, f, e, b }

$laço_6$ = { c, d, f, c }

No entanto, apenas os laços 4, 5 e 6 são necessários para definir o mesmo desenho. Os laços 1, 2 e 3 são redundantes. O processo de exclusão dos laços redundantes é chamado de obtenção de **laços com mínima redundância**, os quais podem ser então definidos como aqueles necessários e suficientes para descrever o desenho.

3.5.3 Estratégia Utilizada

A seguinte estratégia é utilizada nesta pesquisa para se obter os laços numa imagem de diagrama de circuito:

a) a entrada de dados é a lista de vetores obtida na etapa de vetorização, excluídos os vetores já utilizados no reconhecimento dos símbolos abertos. Esta lista de vetores é denominada aqui de **lista inicial de vetores R,** ou seja um par de pares de coordenadas.

b) a seguir obtém-se os vértices (ponto de junção entre duas ou mais retas) formados pelas retas **R** que compõem a imagem. Estes vértices são armazenados numa lista denominada **lista inicial de vértices V.** Os elementos desta lista serão chamados **nós.**

c) tendo-se a lista inicial de vértices **V** e a lista inicial de vetores **R**, cria-se o grafo **RR** representativo da imagem, isto é, transforma-se a lista de vetores em lista de arestas, ou seja, um par de vetores. A lista **R** é então destruída, pois não é mais necessária.

d) processando-se a lista inicial de vértices **V** e o grafo **RR** da imagem, obtém-se a lista de vértices que formam os polígonos fechados. Estes vértices são agrupados formando a **lista de laços L.**

e) neste ponto, vários laços foram obtidos, mas nem todos representam verdadeiramente um objeto pois podem ser redundantes. É necessário então obter-se os **laços mínimos** que constituem a lista final de laços que pertencem à imagem em estudo.

f) identificam-se os segmentos de retas que não fazem parte de nenhum laço.

3.5.4 OBTENÇÃO DOS VÉRTICES

A entrada de dados é constituída da **lista inicial de retas R** obtida na etapa anterior, a de vetorização da imagem. Seja por exemplo um trecho de circuito eletrônico representado pelos seus vetores conforme mostrado na Figura 3.5.4 (os números correspondem a ordem de entrada das retas):

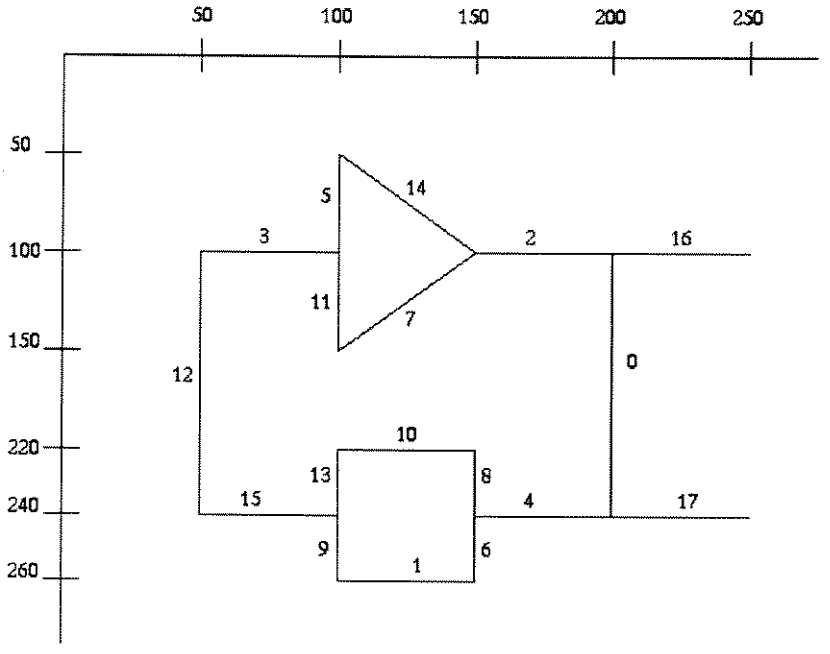


Figura 3.5.4 - Imagem obtida com vetores dados na fase de vetorização

Um vértice é o ponto de junção entre duas ou mais retas ou ainda o ponto extremo de uma reta. O objetivo desta etapa é a obtenção destes pontos através da pesquisa na lista inicial de retas **R**, dadas em qualquer ordem. Os vértices obtidos através do procedimento explanado a seguir são armazenados numa lista denominada **lista inicial de vértices V.**

Para cada nó da lista **R**, que representa uma reta do circuito, executar os seguintes procedimentos:

a) extrair os vértices inicial e final da reta que compõe este nó.

b) verificar se cada vértice extraído no item anterior pertence ou não à lista de vértices **V**. Caso já pertença nada a fazer. Se o vértice não pertence ainda à lista **V**, duas situações são possíveis:

- ele é próximo a um vértice da lista **V**;
- ele não é próximo a nenhum vértice da lista **V**.

Esta proximidade é medida através de um valor parametrizado (*snap*) dado pelo usuário. Desta forma:

p = True se $d(P_1 P_2) \leq S$, onde:

p = proximidade

S = parâmetro snap

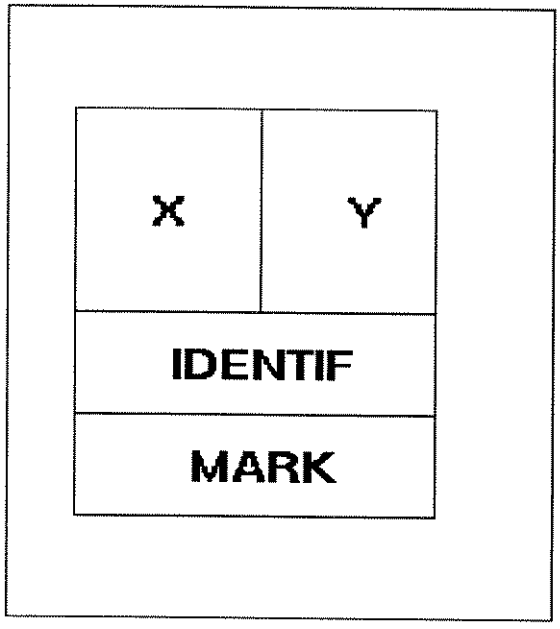
$d(P_1 P_2)$ = distância Euclidiana entre os pontos $P_1 (x_1, y_1)$, que é o ponto em estudo e $P_2 (x_2, y_2) \in \mathbf{V}$

Se **p = True**, considera-se que ponto P_1 já esteja presente na lista **V** pois está próximo à $P_2 \in \mathbf{V}$, e neste caso atualizam-se as coordenadas de P_2 como sendo a média ponderada da distância Euclidiana entre P_1 e P_2. Caso contrário, o vértice P_1 é incluído na lista **V** como sendo um novo nó.

c) Cada vértice incluído na lista inicial de vértices **V** tem a estrutura mostrada na Figura 3.5.5.

d) Tendo-se a lista de vértices **V** com as coordenadas e identificações de todos os vértices, a lista inicial de retas **R** é transformada no grafo **RR** representativo da imagem, cuja estrutura de cada elemento é mostrada na Figura 3.5.6. A lista inicial de retas **R** é então destruída. O grafo **RR** é obtido, varrendo-se

seqüencialmente a lista de vértices **V**. Para cada reta **i** de **R** toma-se um dos seus vértices **k**, o que é a raiz das arestas em **RR** formados pelas retas em **R** que têm em comum o vértice **k**.



(x , y) = coordenadas do vértice

identif = identificação da posição do vértice dado pela posição na seqüência em cada vértice é incluido na lista;

mark = campo não usado nesta fase de construção da lista.

Figura 3.5.5 - Estrutura dos elementos da lista de vértices V

Exemplo: a Tabela 3.5.1 mostra a lista **R** da Figura 3.5.8. Tomando-se a primeira reta, encontramos os nós **a** e **b**. Como o grafo está vazio, o nó **a** é colocado como raiz. Analisando-se o restante da lista R (Tabela 3.5.1) encontramos as retas **a-e** e **a-v**, fazendo com que sejam criados em **RR** os arcos **ab**, **ae** e **av**. Veja na Figura 3.5.9, o resultado desta metodologia aplicada a todos os nós da Tabela 3.5.1.

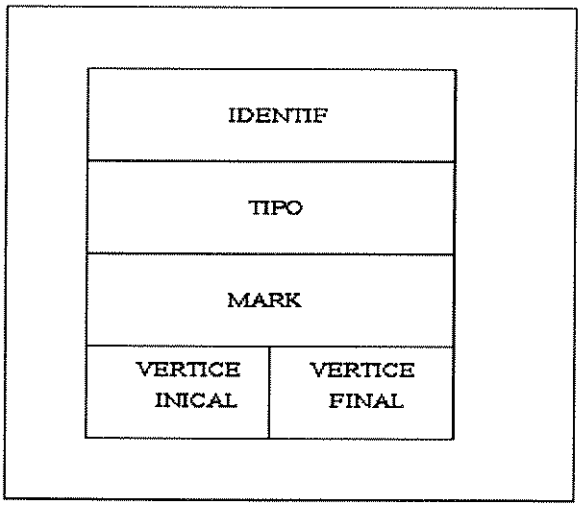


Figura 3.5.6 - Estrutura do grafo da imagem contendo os campos de identificação dos vértices que compõem cada reta

O algoritmo a seguir mostra o desenvolvimento do procedimento de obtenção dos vértices para esta pesquisa:

ALGORITMO vértices

entrada: lista inicial de retas **R** = { r_i }

onde:

$i = 0, 1, 2, \ldots\ldots n_r$

n_r = número de retas

$r = (P_1 , P_2)$: segmento de reta entre os pontos P_1 e P_2

$P = (x, y)$: ponto de coordenadas (x, y)

saída: lista de vértices **V** = { v_j }

grafo RR = { rr_i } constituído pelas retas da lista **R** definidas pelos índices de identificação dos respectivos vértices extremos.

onde:

$j = 0, 1, 2, \ldots\ldots nv$

nv = número de vértices

v = (P, índice , k) , tal que :

P - ponto de coordenadas (x, y) do vértice

índice - identificação do vértice dada pela letra que representa a ordem sequencial em que ele é incluído na lista;

k - número de pontos que contribuíram para o cálculo de P;

rr = (t, v) - segmento de reta (arco do grafo) formado pelos vértices **t** e **v** da lista de vértices **V** .

inicio: V = { } lista de vértices vazia

nv = 0 número de vértices

Processo:

Para cada reta r_i em R faça

Se P_{1i} ou P_{2i} é igual a algum nó de V

então

nada a fazer (vértice já pertence à lista de vértices)

senão

Se P_i é próximo a algum nó v_k em V

então (próximo)

v_k é atualizado pela média ponderada entre v_k e P_i

senão (novo)

insere P_i em V

incrementa nv

fim-se

fim-se

fim-para

fim-algoritmo

Para o exemplo dado na Figura 3.5.4, tem-se como resultado a lista de vértices mostrada na Figura 3.5.7

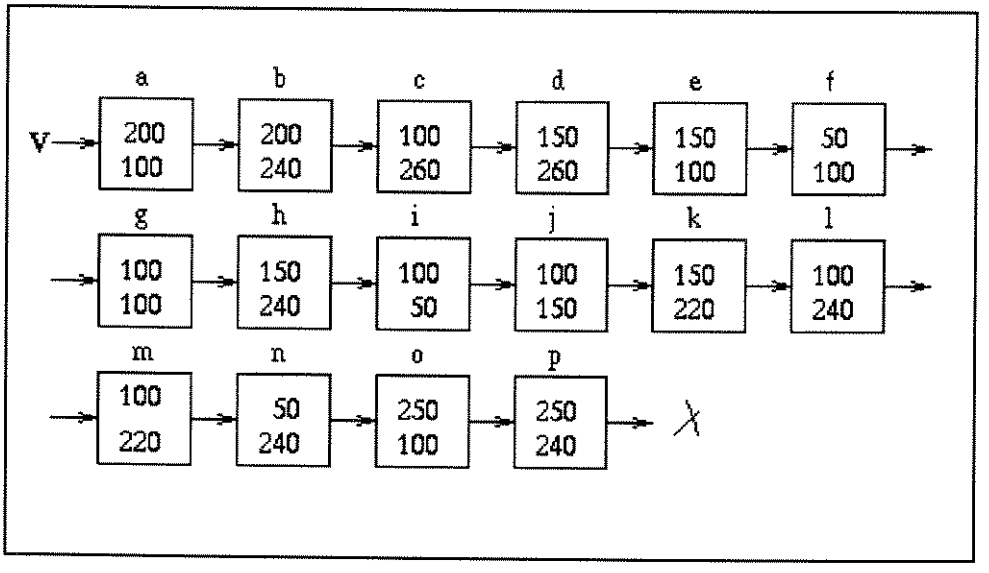


Figura 3.5.7 - Lista dos vértices iniciais obtidos com as retas da lista R para a imagem da Figura 3.5.4

Associando-se estes vértices à imagem da Figura 3.5.4, obtém-se a imagem mostrada na Figura 3.5.8. Deve-se observar que como as retas são processadas aleatoriamente, na ordem em que aparecem na lista **R**, os vértices obtidos são identificados de acordo com a ordem das retas processadas e a lista de retas **R** pode ser escrita conforme mostrado na Tabela 3.5.1. O grafo **RR** desta lista é mostrado na Figura 3.5.9.

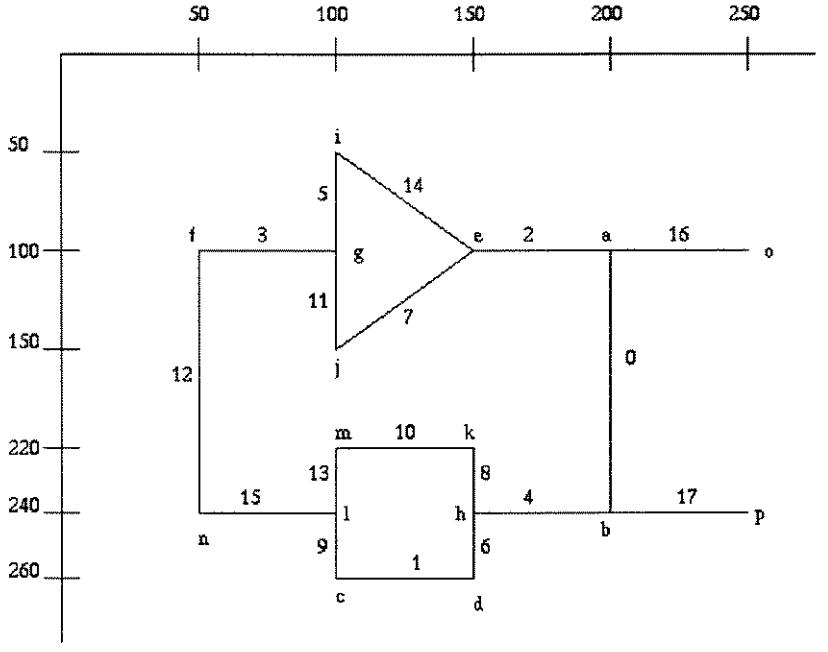


Figura 3.5.8 - Identificação dos vértices V da imagem da Figura 3.5.4

3.5.5 OBTENÇÃO DOS LAÇOS

Nesta etapa o objetivo é obter os polígonos fechados formados pelas retas da imagem. Uma das formas de se obter estes laços é através da análise do grafo **RR** mostrado na Figura 3.5.9. Empregou-se aqui a técnica chamada *depth-first search* através da qual cada nó do grafo é "visitado", e através de procedimentos recursivos é verificado se o nó visitado é encontrado num caminho conectado, fechando desta forma o laço.

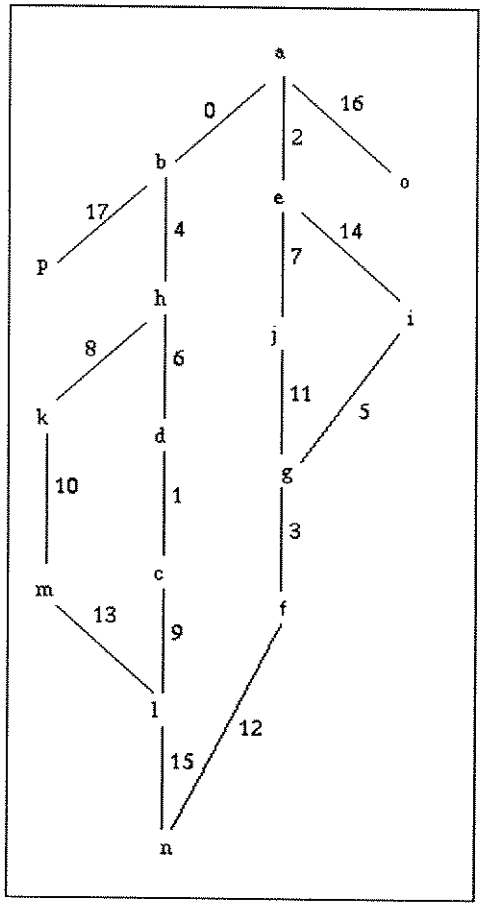


Figura 3.5.9- Representação do grafo RR da imagem da Figura 3.5.8

reta	vért	vértic
0	a	b
1	c	d
2	a	e
3	f	g
4	b	h
5	i	g
6	d	h
7	j	e
8	k	h
9	c	l
10	m	k
11	j	g
12	f	n
13	m	l
	i	e
15	l	n
16	a	o
17	p	b

Tabela 3.5.1- lista RR das retas da imagem da Figura 3.5.8

O método aqui empregado para a obtenção de laços analisa todos os vértices do grafo **RR**, conforme descrito no seguinte procedimento:

a) computa-se a quantidade de retas no grafo **RR**.

b) para garantir a independência da ordem em que os vértices aparecem nas retas,

para cada reta **r** $\in$ **RR**, acrescenta-se a **RR** a reta **rv** reversa de **r**.

Seja **r**: (x_1, y_1) (x_2, y_2)

então **rv**: (x_2, y_2) (x_1, y_1)

Esta técnica aplicada nesta pesquisa, resultou no algoritmo mostrado a seguir:

ALGORITMO laços

entrada: $RR = \{ rr_i \}$

onde:

$i = 0, 1, \ldots\ldots, nr$

nr = número de retas

rr = (t, u) - segmento de reta entre os vértices **t** e **u**

saída: $L = \{ l_{k,j} \}$

onde:

$j = 0, 1, \ldots\ldots, nl$

nl = número de laços finais

$k = 0, 1, \ldots\ldots, nk_j$

nk_j = número de vértices no laço j

$l_{k,j}$ = v : vértice k de L que compõe o laço j.

etapa 1) início:

1.1) faz-se $RR = RR \cup \{ rv_i \}$

onde: para todo $rr = (t, u) \in RR$

tem-se: rv = (u, t) - reta reversa

$i = 0, 1, \ldots\ldots, nr$

1.2) lista inicial (base) de laços:

$L = \{ l_{0,n} \}$ (o primeiro nó de cada laço)

onde:

$n = 0, 1, \ldots\ldots, nv$

nv = número de vértices na lista de vértices V

$l_{0,n} = v_n$: vértice **n** da lista de vértices V

etapa 2) criação das sublistas dos laços (nós vizinhos)

$m_t = 0$

Para cada reta $rr_i = (t, v)$ em RR

$l_{mt,t} = v$

onde $0 < m_t < nk_t$

incrementa m_t

etapa 3) retirando vértices cujas sublistas possuem apenas um nó:

Para cada nó $l_{0,j}$ de L

Se $nk_j = 1$

então

a sublista do vértice j de L possui apenas 1 vértice e portanto este vértice é retirado de toda a lista L em todas suas instâncias.

etapa 4) faz $R = \lambda$: a lista é destruída

etapa 5) obter os laços.

fim-algoritmo

Para o exemplo da Figura 3.5.8 o grafo **RR** compor-se-á então dos seguintes elementos ($RR = r \bigcup rv$):

RR = { (a, b), (c, d), (a, e), (f, g), (b, h), (i, g), (d, h), (j, e),
(k, h), (c, l), (m, k), (j, g), (f, n), (m, l), (i, e), (l, n),
(a, o), (p, b), (b, a), (d, c), (e, a), (g, f), (h, b), (g, i),
(h, d), (e, j), (h, k), (l, c), (k, m), (g, j), (n, f), (l, m),
(e, i), (n, l), (o, a), (b, p)
}

A lista de laços a ser formada pelas retas que constituem a imagem tem uma estrutura tal como mostrada na Figura 3.5.10.

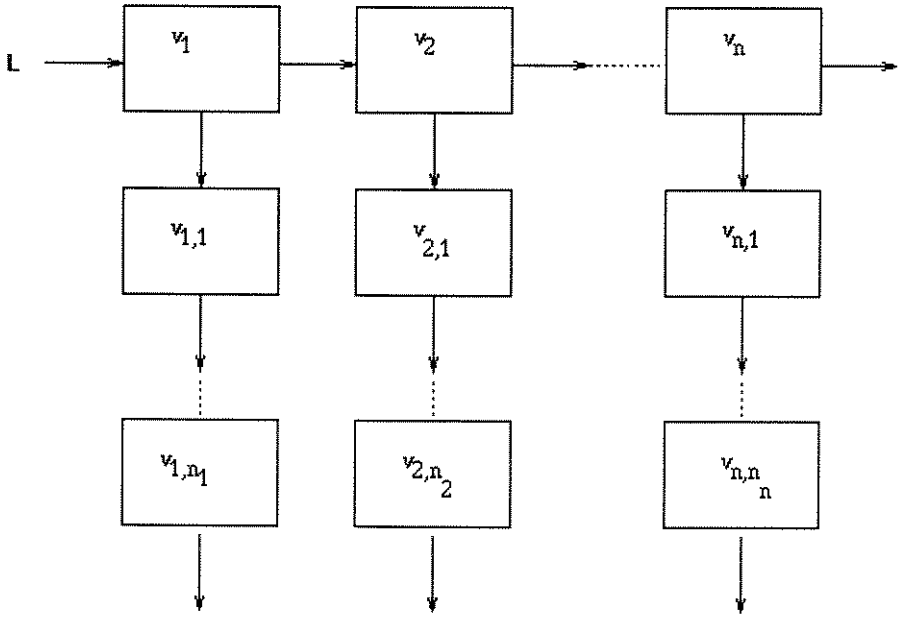


Figura 3.5.10 - Estrutura da lista de laços L

Na verdade, esta estrutura é composta de uma lista de listas, tal que a lista base contendo os nós raízes (v_1, v_2, . . . v_n) (veja Tabela 3.5.1), é constituída pelos vértices da lista de vértices **V**, e cada sublista ($v_{n,1}$ . . . $v_{n,\,nn}$) é constituído pela lista de vértices que formam o laço que têm como origem o nó base (raiz) daquela sublista.

Assim sendo, para o exemplo da Figura 3.5.8 a lista inicial **L** de laços é a seguinte:

{ a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p }

c) Cada sublista constitui um laço derivado do respectivo nó (vértice) raiz na lista base **L** de laços. Sua obtenção é realizada pesquisando-se o grafo **RR** de retas cada vez que se adiciona um nó base em **L**. Desta forma, um vértice x é incluído na sublista de um nó base **y** da lista **L**, se o par (y, x) $\in$ **RR**. Utilizando-se este procedimento no exemplo da Figura 3.5.8, obtém-se a lista **L** completa mostrada na Tabela 3.5.2 .

Neste ponto a lista **R** é destruída pois não é mais necessária à continuação do procedimento.

nó base	sublista
a	b, e, o
b	h, a, p
c	d, l
d	h, c
e	a, j, i
f	g, n
g	f, i, j
h	b, d, k
i	g, e
j	e, g
k	h, m
l	n, c, m
m	k, l
n	f, l
o	a
p	b

Tabela 3.5.2 - lista inicial de laços obtidos do grafo RR

d) sublistas com menos de 2 elementos não podem caracterizar laços, assim sendo elas e os respectivos nós bases são retirados da lista de laços **L**. É o caso dos nós **o** e **p** no exemplo anterior. Ao se deletar os nós bases de **L** deve-se também deletar suas ocorrências nas demais sublistas. No exemplo, os nós bases **a** e **b** contém os nós bases **o** e **p** respectivamente deletados anteriormente e portanto são rearranjados. Logo, a nova lista de laços **L** tem agora a configuração mostrada na Tabela 3.5.3.

nó base	sublista
a	b, e
b	h, a
c	d, l
d	h, c
e	a, j, i
f	g, n
g	f, i, j
h	b, d, k
i	g, e
j	e, g
k	h, m
l	n, c, m
m	k, l
n	f, l

Tabela 3.5.3 - lista de laços obtida após rearranjo da lista de laços inicial

e) uma das formas disponíveis para se detectar os laços é pesquisar cada vértice da lista base de laços e recursivamente buscar nas sublistas o vértice que fecha laço com o vértice base em estudo. Veja no apêndice B um exemplo desta metodologia aqui apresentada.

f) Os laços formados a partir do vértice **a** são:

laço 0 : { a, b, h, d, c, l, n, f, g, i, e, a }

laço 1 : { a, b, h, d, c, l, n, f, g, j, e, a }

laço 2 : { a, b, h, k, m, l, n, f, g, i, e, a}

laço 3 : { a, b, h, k, m, l, n, f, g j, e, a }

g) Um laço é encontrado quando o vértice em estudo aparece como integrante de uma sublista cuja profundidade [1] é > 1.

[1] número de elementos na sublista

h) A pesquisa é encerrada num nó quando este nó é antecessor da sublista em estudo mas não é o nó em estudo.

i) Cada laço encontrado deve ser verificado para saber-se se ele é ou não cíclico e portanto repetido. Por exemplo, o laço **a, b, c, d, e** é igual ao laço **c, d, e, a, b** e portanto, deve ser desconsiderado da lista de laços.

j) O processo é repetido para todos os nós da lista **L**, podendo no entanto ser otimizado, ou seja, para cada nó pesquisado, os vértices vizinhos de sua sublista não entram como nós de pesquisas nos próximos passos. Assim sendo no nosso exemplo anterior, tem-se a ordem de pesquisa mostrada na Tabela 3.5.4.

Ou seja, uma vez pesquisado o vértice **a**, os nós **b** e **e** não precisam ser pesquisados pois estão ligados ao vértice **a**. Portanto, o próximo vértice a ser pesquisado é o nó **c**, cuja sublista é **d, l** os quais não vão ser pesquisados nos próximos passos, e assim por diante. Desta forma pesquisa-se apenas os vértices **a, c, f, h, i, j, m**.Como resultado deste algoritmo, obtém-se a seguinte lista de laços:

nó base	sublista
a	b, e
c	d, l
f	g, n
h	k, b, d
i	e, g
j	g, e
m	k, l

Tabela 3.5.4 - Lista de pesquisa otimizada

0) { i, e, j, g, i }

1) { c, l, m, k, h, d, c }

2) { a, b, h, k, m, l, n, f, g, j, e, a }

3) { a, b, h, k, m, l, n, f, g, i, e, a }

4) { a, b, h, d, c, l, n, f, g, j, e, a }

5) { a, b, h, d, c, l, n, f, g, i, e, a }

k) Os laços formados por estas listas são mostrados na Figura 3.5.11.

Compare-a com a Figura 3.5.8, a fim de localizar os laços obtidos.

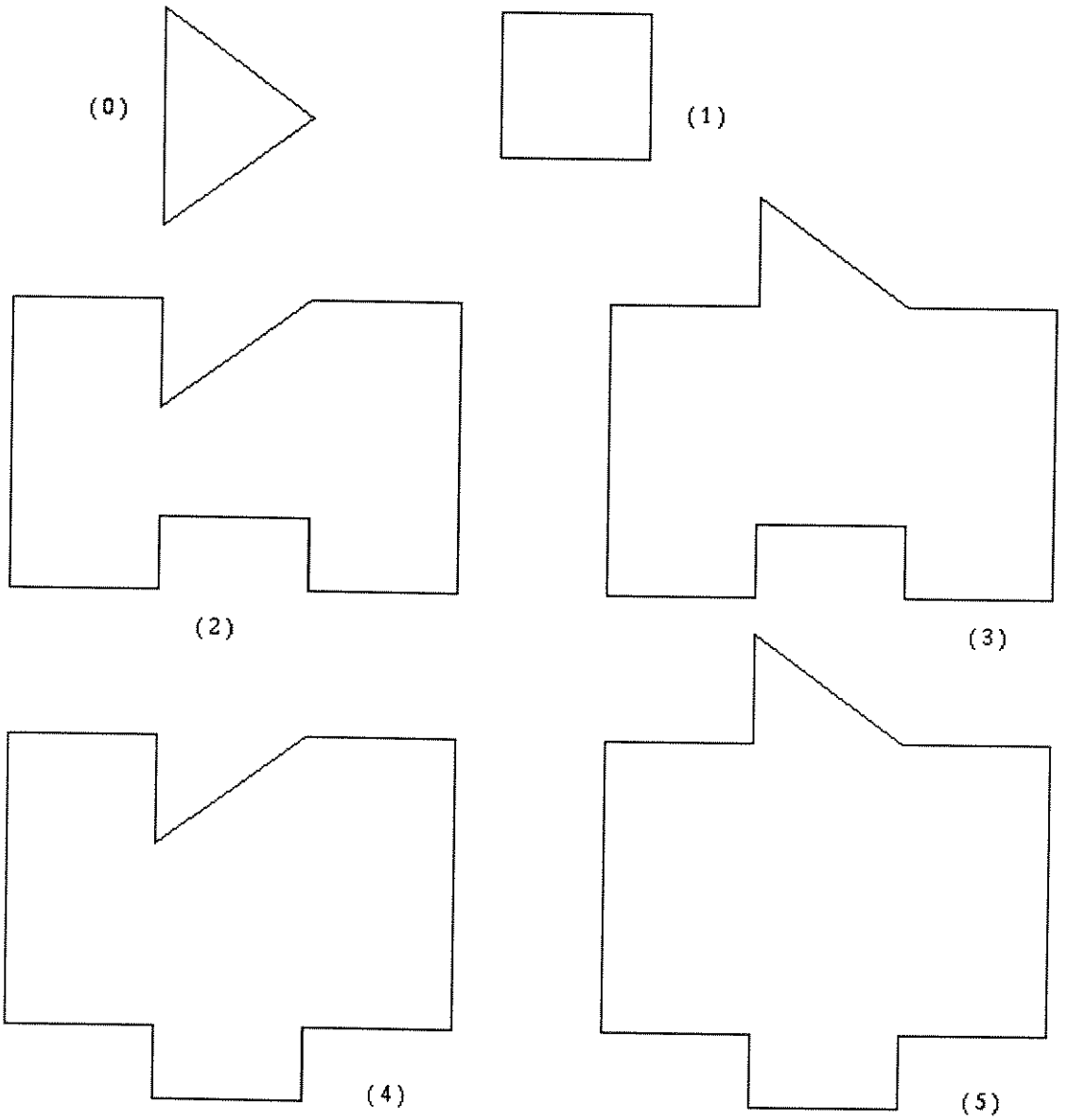


Figura 3.5.11 - Laços obtidos da imagem da Figura 3.5.8

3.5.6 Identificação dos Laços Primitivos

O processo de obtenção de laços apresentado no item anterior resolve o problema da segmentação dos laços, mas tem algumas desvantagens: é lento pois o trabalho computacional envolvido é muito extenso e também não minimiza os laços obtidos. Como exemplo, vê-se na Figura 3.5.11 que apenas os laços (0), (1) e (2) são necessários à definição da imagem e os demais são redundantes.

[Franklin e Akman 1986] apresentam um processo para tratamento de superfícies escondidas. Adaptamos as idéias envolvidas em tal processo na criação do algoritmo seguinte cujo objetivo é otimizar o processo de obtenção dos laços numa imagem.

- Computa-se a quantidade de vértices na lista de vértices **V**.
- Ordenam-se os vértices em ordem crescente das abcissas X ou no caso daqueles que tenham abcissas iguais, ordená-los em ordem crescente das ordenadas Y
- Chamou-se esta lista de vértices ordenados **VO.**
- Na estrutura de vértices mostrada na Figura 3.5.5, o campo *mark* agora recebe o valor da ordem do nó na lista **VO**.
- A lista base da lista inicial de laços **L** (Tabela 3.5.2) agora é formada com os vértices ordenados **VO** e não mais na ordem sequencial de sua identificação.
- Rearranjando-se esta lista (retirando-se as sublistas com apenas 1 elemento e suas propagações) tem-se a lista de laços **L** mostrada na Tabela 3.5.5.

posição	nó base	sublista
0	f	g, n
1	n	f, l
2	i	g, e
3	g	f, i, j
4	j	e, g
5	m	k, l
6	l	n, c, m
7	c	d, l
8	e	a, j, i
9	k	h, m
10	h	b, d, k
11	d	h, c
12	a	b, e
13	b	h, a

Tabela 3.5.5 - Lista de vértices ordenados VO provenientes dos nós da Figura 3.5.7

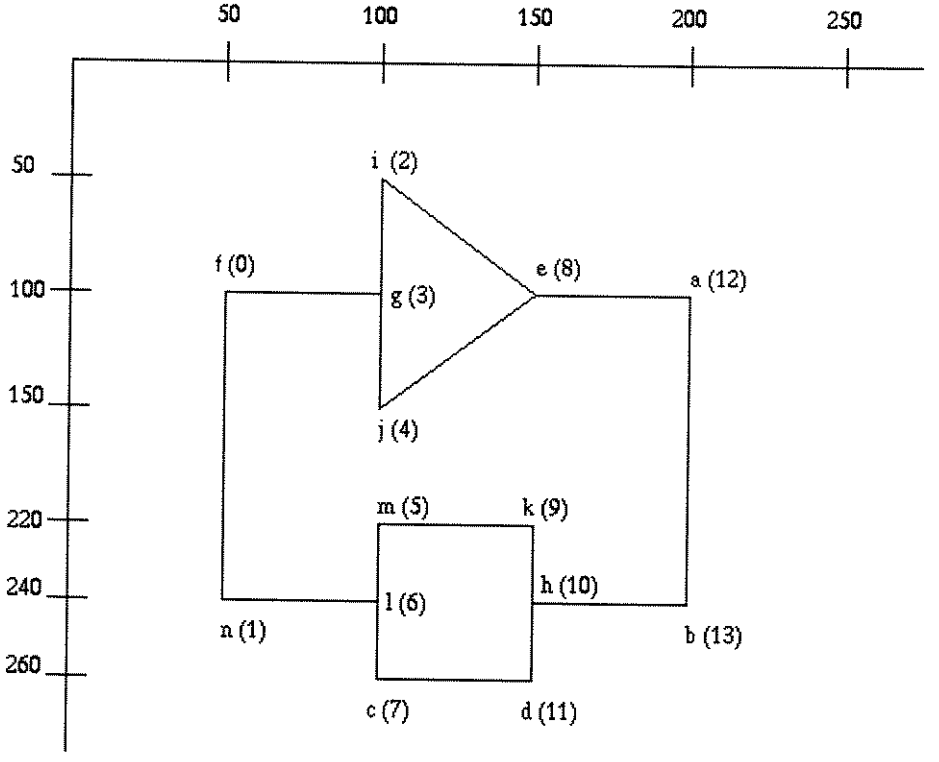


Figura 3.5.12 - A posição dos nós quando ordenados

Observe na Figura 3.5.12 a posição de cada nó (colocada entre parêntesis) conforme dada na Tabela 3.5.5 após a ordenação dos vértices.

- A sublista de cada nó base na Tabela 3.5.5 deve ser ordenada em ordem crescente do ângulo entre a aresta formada pelo nó base e cada elemento da sublista, tendo como referência o nó base. Por exemplo, na posição 0 da Tabela 3.5.5 tem-se o nó base **f** cuja sublista é formada pelos vértices **g** e **n**. As coordenadas destes vértices são respectivamente: **f** (50, 100), **g** (100, 100) e **n** (50, 240). Se traçarmos uma reta entre o nó **f** e nó **g** esta teria um ângulo de 0 graus, em relação à horizontal, e uma reta entre o nó **f** e o nó **n** teria um ângulo de 270 graus.

nós bases ordenados	sublista ordenada
f	g, n
n	l, f
i	g, e
g	i, f, j
j	e, g
m	k, l
l	m, n, c
c	d, l
e	a, i, j
k	m, h
h	b, k, d
d	h, c
a	e, b
b	a, h

Tabela 3.5.6 - Lista de laços L com as sublistas ordenadas

Esta sublista de nós ordenados é obtida tomando-se o nó base (pivô) e no sentido anti-horário, listar todos os nós ligados a este nó base, iniciando-se por aquele de menor ângulo.

Logo, o ângulo entre o nó base **f** e o vértice **g** é 0 graus e o ângulo entre o nó base **f** e o vértice **n** é 270 graus. Daí, a ordem desta sublista é **g**, **n**. A Tabela 3.5.6 mostra a lista de laços **L** ordenada conforme as regras apresentadas.

Observando-se a Tabela 3.5.6 e a Figura 3.5.12, vê-se que cada nó base tem sua sublista formada pelos vértices que se ligam a ele tomados no sentido anti-horário, iniciando-se por aquele com menor ângulo. Da mesma forma que no processo anterior, os laços são formados pesquisando-se a lista de nós bases da lista **L** e suas respectivas sublistas. Cada vez que um vértice base é encontrado numa sublista após "perseguir" as sublistas de **L**, um laço é fechado. Seja **L** a lista de laços tal como mostrada na Tabela 3.5.6. Seja b_i o i-ésimo nó base de **L** e s_{ik} o k-ésimo nó da sublista de b_i. Inicialmente, todos os nós das sublistas estão disponíveis.

ALGORITMO otimização

1) para todo nó **i** da lista de vértices V, toma-se o primeiro nó b_i que ainda possui nós disponíveis na sua sublista; se não houverem mais nós b_i o processo terminou ($0 < i < nv$);

2) para todo nó **k** da sublista do nó b_i em **V**, toma-se o primeiro nó s_{ik} disponível ($0 < k < nk_i$);

3) acrescenta-se os vértices b_i e s_{ik} à lista L_j, j = 0, ..., n, onde **n** é o número de vértices formados; o primeiro nó b_i em L_j é denominado pivô daquele laço;

4) s_{ik} é marcado como indisponível;

5) busca-se em **L**, $b_r = s_{ik}$;

6) busca-se na sublista de b_r em **V** o primeiro vértice disponível s_{rt} tal que s_{rt} venha depois de b_i, ou seja $s_{r,\,t-1} = b_i$; acrescenta-se s_{rt} à L_j e marca-o como indisponível; a pesquisa na sublista é cíclica, ou seja, se o próximo nó for *nil* volta-se ao primeiro nó da mesma (*wrap-around*);

7) se s_{rt} = pivô, o conjunto L_j é encerrado sendo este formado pelos elementos lá contidos; incrementa-se j e volta-se ao passo 1; caso contrário, $s_{rt} \neq$ pivô, faça $b_i = b_r$ e $s_{ik} = s_{rt}$. e volta-se ao passo 5.

fim algoritmo

Tomando-se como exemplo a lista da Tabela 3.5.6 que corresponde à imagem da Figura 3.5.12, e aplicando-se o algoritmo desenvolvido e apresentado, obtém-se os seguintes laços:

$laço_0$	{ f, g, j, e, a, b, h, k, m, l, n, f }
$laço_1$	{ f, n, l, c, d, h, b, a, e, i, g, f }
$laço_2$	{ i, e, j, g, i }
$laço_3$	{ m, k, h, d, c, l, m }

Tabela 3.5.7- Laços obtidos da imagem da Figura 3.5.12

Exemplo para o laço 0:

- pela Tabela 3.5.6, o primeiro nó base pivô disponível é o nó **f**, cuja sublistas é composta pelos nós **g, n.** Inicialmente todas as sublistas estão disponíveis, logo, tem-se para este primeiro nó a seguinte inicialização para o laço formado por ele:

{ f, g } - o nó **g** agora está indisponível;

- o próximo nó base de estudo é o nó **g** cuja sublista é : (**i, f, j**);

- primeiro nó disponível da sublista de **g** e que vem após o nó base anterior **f**: nó **j**

Logo, **j** está agora indisponível e é acrescentado ao laço em formação:

{ f, g, j }

- próximo nó base: **j** (**e, g**)

- primeiro nó disponível da sublista de **j** que vem após o nó base anterior **g**: nó **e**

Logo, **e** está indisponível e é acrescentado ao laço em formação:

{ f, g, j, e }

- próximo nó base: **e** (**a, i, j**)

- primeiro nó disponível: **a**

- logo, **a** fica indisponível , e o laço em formação fica:

{ f, g, j, e, a }

- próximo nó base : **a** (**e, b**)

- primeiro nó disponível: **b**

laço em formação: **{ f, g, j, e, a, b }**

e assim por diante, até fechar o laço, ou seja, quando o primeiro nó disponível de uma sublista for igual ao primeiro nó do laço em formação.

O processo é repetido até que todos os laços tenham sido encontrados, ou seja, até não houver mais nenhum nó disponível nas sublistas dos nós bases.

Estes laços correspondem respectivamente aos laços (2), (5), (0) e (1) da Figura 3.5.11. Claramente tem-se como vantagens a obtenção de todos os laços necessários com custo computacional muito menor devido à simplicidade do algoritmo aplicado.

3.5.7 Laços Envoltórios

O fato de se ordenar os vértices na lista base de laços e suas respectivas sublistas faz com que os laços obtidos sejam compostos de vértices sequenciais, ordenados e obedeçam ao sentido horário ou anti-horário. O conjunto de laços no sentido horário formam os **laços**

primitivos que compõem a imagem enquanto que os laços no sentido anti-horário correspondem aos laços envoltórios da imagem e portanto são redundantes neste caso.

Veja a Tabela 3.5.7 e a Figura 3.5.12. Os laços 0, 2 e 3 estão todos no sentido horário e correspondem aos laços mínimos encontrados naquela imagem, enquanto que o laço 1 é percorrido no sentido anti-horário, indicando uma envoltória. O algoritmo para determinar se 3 pontos consecutivos **A**, **B** e **C** são percorridos no sentido horário ou anti-horário, é dado por [Sedgewick 1990] e foi adaptado aqui para definir o sentido dos laços obtidos com os vértices que constituem uma imagem.

Sejam os vértices: $A (x_a, y_a)$

$B (x_b, y_b)$

$C (x_c, y_c)$

e as distâncias:

$$dx_1 = x_b - x_a$$

$$dy_1 = y_b - y_a$$

$$dx_2 = x_c - y_a$$

$$dy_2 = y_c - y_a$$

O objetivo é determinar qual é o sentido dos pontos obtidos, partindo-se do ponto **A** em direção ao ponto **C**. Duas são as possibilidades:

a) sentido horário - como mostrado na Figura 3.5.13

b) sentido anti-horário - como mostrado na Figura 3.5.14.

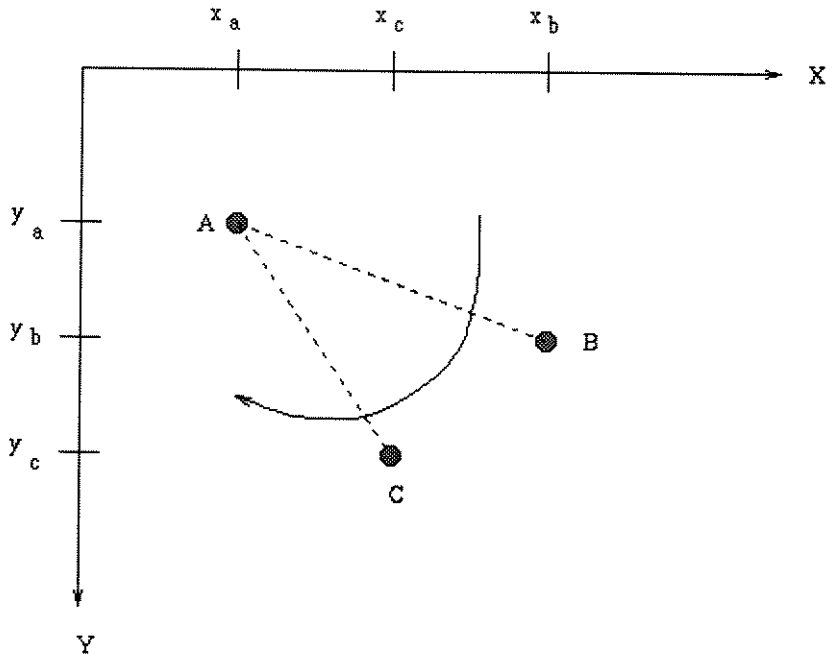


Figura 3.5.13- Pontos no plano XY - sentido horário

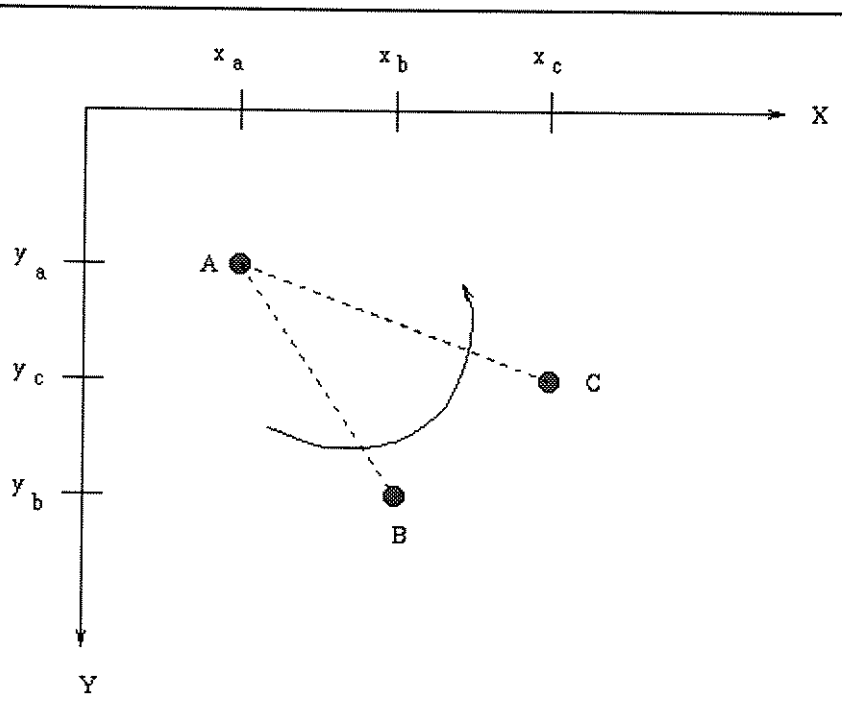


Figura 3.5.14 - Pontos no plano XY - sentido anti horário

As inclinações das retas **AB** e **AC** são dadas por $m_1 = dy_1 / dx_1$ e $m_2 = dy_2 / dx_2$ respectivamente. Portanto, se $m_2 > m_1$ o caminho percorrido do ponto **A** ao ponto **B** ao ponto **C** é no sentido horário (Figura 3.5.13) e anti-horário caso contrário (Figura 3.5.14).

Para evitar-se problemas com retas verticais e também o cálculo de inclinações, multiplicam-se as equações para o cálculo de inclinação por $dx_1 . dx_2$, obtendo-se as seguintes regras para verificação do sentido de percorrimento de 3 pontos quaisquer no espaço bidimensional:

- se $dx_1 . dy_2 > dy_1 . dx_2$

  então eles são percorridos no sentido horário; caso contrário, considere sentido anti horário;

- Para pontos colineares verticais, se $dx_1 . dx_2 < 0$

  então eles estão no sentido anti horário;

- Para pontos colineares verticais, se $dy_1 . dy_2 < 0$

  então eles estão no sentido anti horário;

Para se obter o sentido dos laços obtidos com os segmentos de reta provenientes da vetorização da imagem, aplica-se os conceitos anteriores a cada 3 nós consecutivos b_i, b_{i+1} e b_{i+2} da lista $laço_k$, onde $i = 0, 1, 2, ... n_k - 2$ e n_k é a quantidade de nós na lista $laço_k$.

Desta forma, os laços da Tabela 3.5.7 são classificados em anti horários (1) e horários (0, 2 e 3). Descartando-se os laços cujos sentidos são anti horários tem-se como resultado final os laços mostrados na Tabela 3.5.8 a seguir:

$laço_0$	{ f, g, j, e, a, b, h, k, m, l, n, f }
$laço_1$	{ i, e, j, g, i }
$laço_2$	{ m, k, h, d, c, l, m }

Tabela 3.5.8 - Laços finais obtidos da imagem 3.5.12

3.6 CLASSIFICAÇÃO

3.6.1 OBJETIVO

Uma vez extraídos da imagem e classificados os símbolos abertos, resta classificar os laços obtidos como um símbolo fechado novo ou já existente. Este é o objetivo desta parte desta pesquisa.

3.6.2 INTRODUÇÃO

Considere por exemplo a imagem de uma parte de um diagrama de circuito elétrico, tal como mostrado na Figura 3.6.1, a seguir:

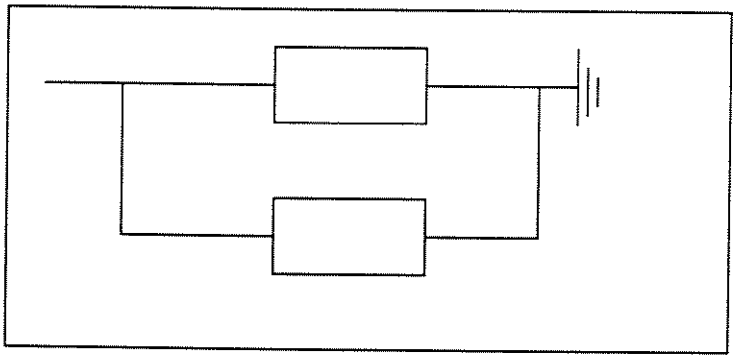


Figura 3.6.1 - Parte de um circuito elétrico

Deveria ser possível após uma classificação simples, termos como resultado a identificação dos componentes: resistor e aterramento. Uma classificação completa deveria relacionar todos os elementos presentes na cena, e poderia resultar numa descrição tal como por exemplo: "dois resistores em paralelo e este conjunto em série a um aterramento".

Frente a esta descrição tem-se o que se entende por reconhecimento de padrões existentes na imagem por intermédio da classificação dos objetos pertencentes a esta imagem. Este reconhecimento pode ser realizado através de duas metodologias:

a) classificação de padrões, onde o objetivo é atribuir um objeto a uma das possíveis classes, usando como ferramenta básica a teoria da decisão estatística;

b) reconhecimento sintático (ou estrutural) de padrões, onde se procura uma descrição do objeto, em termos das relações entre seus componentes, utilizando-se principalmente a teoria de linguagens formais quando possível.

Nesta pesquisa usou-se o reconhecimento estrutural durante a fase de extração das características dos objetos abertos componentes da imagem. Não se usou nenhuma linguagem formal, mas sim as relações estruturais dos objetos que se deseja reconhecer.

Para os objetos fechados, utilizou-se o modelo estatístico, e neste caso o problema de classificação automática constitui-se basicamente de particionar o **espaço de atributos ou características** (*feature space*) em regiões, sendo que cada região corresponde a uma categoria ou classe de objetos. Idealmente, estas partições deveriam ser de tal forma que os resultados obtidos do classificador sejam sempre corretos. Quando isto não puder ocorrer, o ideal é minimizar a probabilidade de erro, ou, se alguns erros são mais caros que outros, o custo médio dos erros.

Existem dois tipos básicos de métodos de classificação estatística: **supervisionado** e **não supervisionado**. Nos métodos supervisionados, o usuário supervisiona o processo através da seleção inicial de alguns objetos de cada possível classe. Destes, o algoritmo de classificação determina as características de cada classe e então associa cada objeto da imagem a uma das classes definidas, baseado em alguma regra estatística pré-estabelecida. Nos

métodos não supervisionados, as classes são determinadas pelo usuário quando este associa cada classe a um **agregado** (*cluster*) de objetos da imagem.

A seguir são mostrados alguns métodos clássicos de classificação e finalmente é descrito o método de classificação utilizado nesta etapa desta pesquisa.

3.6.3 A CLASSIFICAÇÃO NO RECONHECIMENTO DE PADRÕES

O objetivo de um classificador é distinguir objetos entre vários objetos [Duda e Hart 1988 e Schalkoff 1999]. A habilidade discriminatória do classificador provém do fato de que diferentes padrões (diferentes classes de padrões) são compostos de diferentes atributos ou características com valores numéricos distintos. Daí pode-se dizer que um classificador atribui os objetos de uma determinada cena a uma das k classes pré-especificadas, baseado na extração das suas características ou seus atributos significantes e no processamento ou análise destes atributos.

Deste modo, é importante que se tenha inicialmente um conjunto de medidas (atributos), as quais devem ser representativas dos objetos que se quer reconhecer e também devem se adaptar ao método de classificação que se deseja utilizar. Nesta pesquisa, estes atributos já foram obtidos na fase anterior a esta e estão armazenados no vetor de característica descrito no Item 3.4.5.

O processo de reconhecimento pode ser melhorado tanto com o emprego de modelos matemáticos sofisticados, como pela escolha de melhores características para uma dada aplicação.

Reconhecimento é basicamente um processo de associação de objetos à classes. Um dos enfoques mais simples para o problema do reconhecimento é o conceito de classificação de padrões por funções de distâncias. A motivação para o uso dessas funções como ferramentas de classificação vem do fato de que o caminho natural para o estabelecimento de uma medida de similaridade entre vetores de padrões, os quais são pontos no espaço de características, é o da determinação de suas distâncias.

Uma classe *w* de padrões é constituida de um conjunto de padrões similares. A chave para os sistemas de reconhecimento de padrões é identificar atributos (características) que formam um conjunto de boas medidas de similaridade. Por exemplo, se as características de um padrão $\underline{x}$ é muito similar a outros padrões pertencentes à classe *w*, pode-se intuitivamente classificar $\underline{x}$ como pertencente à classe *w*.

Regiões de decisões são definidas pela partição do espaço de características e estão associadas a cada classe resultante do processo de classificação. A fim de que estas regiões sejam únicas numa atribuição de decisão, elas precisam cobrir todo o espaço dos números reais e serem disjuntas (não sobrepostas). A borda de uma região de decisão é chamada **fronteira de decisão.** No caso multidimensional linear, estas fronteiras são **hiperplanos** definidos pelas funções de decisão ou funções discriminantes. Com estas informações, a classificação de um vetor de características $\underline{x}$ de um objeto desconhecido fica muito simples: determina-se a região de decisão $\underline{R}$ na qual $\underline{x}$ se situa, e atribui-se a $\underline{x}$ a classe correspondente à região $\underline{R}$. Desta forma, padrões com características comuns ou pertencendo à mesma classe, são colocados na mesma região. O uso de vetores de características para representar padrões e hiperplanos para particionar o espaço de características é usualmente referenciado como decisão teórica ou enfoque discriminante. Embora esta estratégia de decisão seja simples, a determinação das regiões de decisão é um grande desafio.

Como exemplo seja a Figura 3.6.2 onde são apresentadas duas classes de padrões num espaço bidimensional de características $\underline{x}_1$ e $\underline{x}_2$.

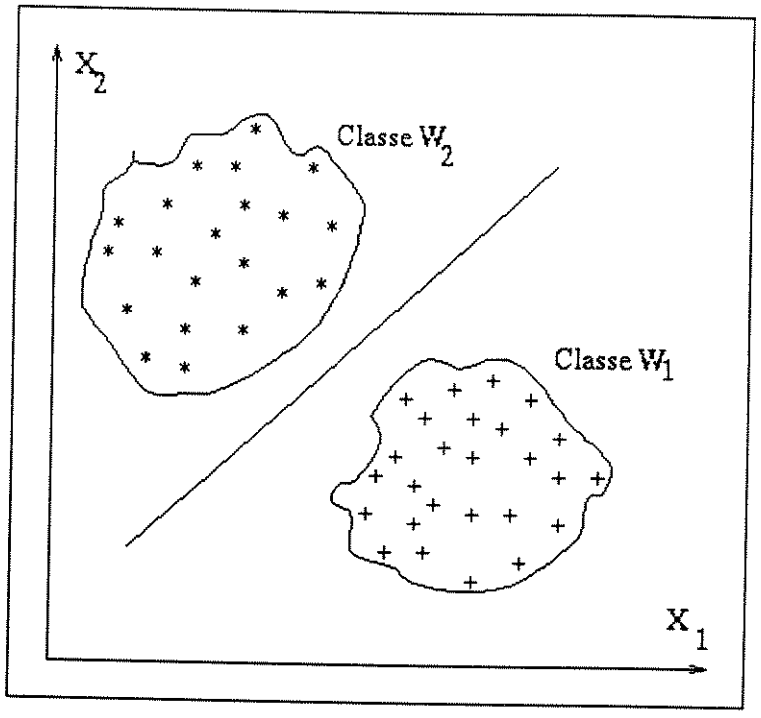


Figura 3.6.2 - Duas classes de padrões num espaço bidimensional

Nesta ilustração, as classes w_1 e w_2 poderiam assumir por exemplo, elementos (padrões) de um diagrama de circuito elétrico tais como resistor e aterramento respectivamente. Cada padrão é caracterizado por duas medidas, como por exemplo: número de retas que o compõem e o ângulo entre as retas. Portanto, os vetores de padrões terão a forma $x = (x_1, x_2)^t$, onde x_1 representa o número de retas que compõem cada padrão e x_2 o ângulo entre as retas no padrão. Cada vetor do padrão pode ser considerado como um ponto num espaço bidimensional.

Se por exemplo todos os símbolos de resistores possuem mais de 6 retas e todos os símbolos de aterramento possuem menos, estas informações podem ser usadas para classificar um conjunto de objetos de um circuito elétrico. Devido ao fato de que alguns resistores possuírem menos de 6 retas e alguns aterramentos possuírem mais, o critério não

permitiria uma perfeita discriminação. O método portanto pode falhar. Continuando com o exemplo deve-se selecionar primeiro uma melhor quantidade de retas como limiar (*threshold*) para representar um símbolo. Para isto, deve-se obter informações estatísticas sobre a distribuição da quantidade de retas que formam os símbolos de resistores e de aterramento.

Examinando-se a distribuição de probabilidade de quantidade de retas, pode-se localizar um valor que permita a separação ótima das duas classes. Entretanto, pode-se ainda incorrer em dois tipos de erros: um resistor pode ser classificado como aterramento ou vice versa. Um bom limiar deve ser tal que minimize o número de erros esperados.

Tem-se então que a classificação não é perfeita usando-se apenas uma única característica, daí a necessidade de introduzir-se outras. Por exemplo, pode-se incluir o ângulo entre as retas que formam os símbolos e escolher um segundo limiar para a discriminação dos símbolos. As duas medidas (quantidade de retas e ângulos entre elas) podem ser combinadas, por exemplo, pela média da quantidade de retas esperada para ambos os símbolos. Conclui-se pelo exemplo mostrado, que os métodos de classificação não devem ser adotados sem uma pré-seleção das medidas a serem utilizadas. Os resultados da classificação só serão bons se as medidas selecionadas também o forem.

Como o objetivo da classificação é minimizar a probabilidade de classificações falsas baseadas no conhecimento *a priori* das probabilidades de ocorrências de certos padrões e das probabilidades condicionais de suas ocorrências, dado um vetor de características, os padrões de classe w_1 podem ser separados da classe w_2 por uma função discriminante g ($\underline{x}$) linear (Figura 3.6.3 a) ou não linear (Figura 3.6.3 b, c).

No caso linear, g ($\underline{x}$) é escrita da seguinte forma:

$$g(\underline{x}) = c_1 \underline{x}_1 + c_2 \underline{x}_2 + c_0$$

onde os coeficientes c_0 , c_1 e c_2 são os pesos e $\underline{x}_1$ e $\underline{x}_2$ são os valores das características. Assim $g(\underline{x})$ pode ser usada como uma função de decisão nas seguintes condições: um padrão $\underline{x} \in w_1$ se $g(\underline{x}) < 0$ e $\underline{x} \in w_2$ em caso contrário.

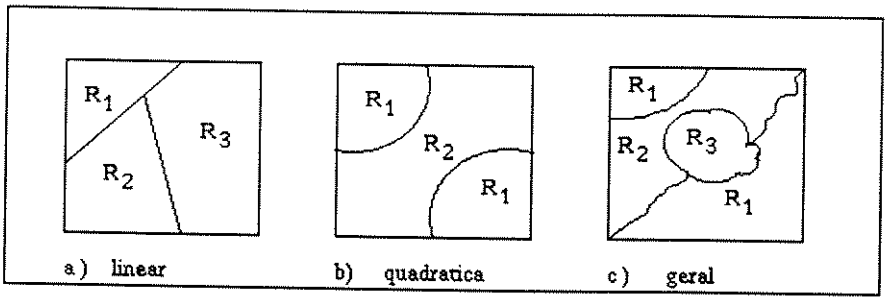


Figura 3.6.3 - Modelos de regiões de decisão

De uma forma geral, funções discriminantes lineares, denotadas por $g_i(\underline{x})$, $i \in [1, k]$ são usadas para particionar o espaço $\underline{R}$ em k classes, de acordo com a regra de decisão mostrada anteriormente porém escrita de outra forma: atribuir $\underline{x}$ à classe w_m (correspondente à região R_m), onde $g_m(\underline{x}) > g_i(\underline{x})$ para todo $i \in [1, k]$ e $i \neq m$. Quando $g_r(\underline{x}) = g_s(\underline{x})$ tem-se uma fronteira de decisão, e aí a decisão deve ser feita ao acaso.

A mais simples e intuitiva estratégia utilizada para a classificação de padrões é aquela que utiliza o cálculo da distância entre o padrão que se quer classificar e as classes disponíveis. A Figura 3.6.4 mostra um exemplo deste classificador: o da **distância mínima**.

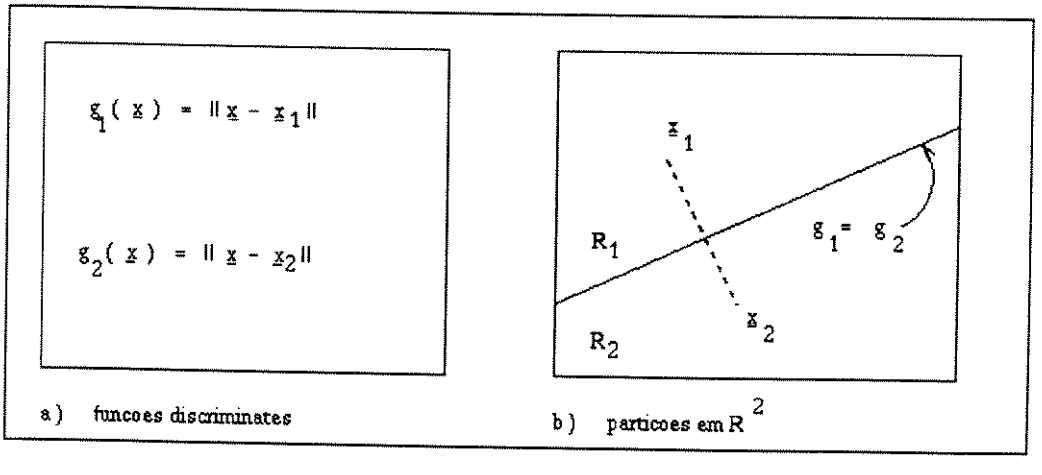


Figura 3.6.4 - Funções discriminantes (lineares) e as correspondentes regiões de decisão

Neste caso o classificador computa a distância entre o padrão $\underline{x}$ cuja classe é desconhecida, e cada classe do conjunto de classes (equações na Figura 3.6.4 a - para o caso de duas classes), atribuindo-se a $\underline{x}$ a classe w cuja distância fôr menor, ou seja, $\underline{x}$ é atribuído à classe w_i se $g_i < g_j$ para todo $i \neq j$. Considerando-se a distância como sendo uma medida de similaridade, pode-se dizer: quanto menor a distância Euclideana entre os padrões, maior é a similaridade entre eles.

Esta idéia pode ser generalizada da seguinte forma: considere um conjunto de amostras $s = \{ s_1, s_2, \ldots, s_n \}$, cuja classificação é desejada. A regra de classificação da **distância mínima** atribui a um padrão desconhecido $\underline{x}$ a classe de seu vizinho mais próximo. Esta regra pode ser escrita da seguinte forma:

$s_i \in \{ s_1 , s_2 , \ldots , s_n \}$ é o vizinho mais próximo de $\underline{x}$ se :

$$d (s_i , \underline{x}) = \min \{ d (s_j , \underline{x}) \} \ , \ j \in [1, n]$$

e **d** é qualquer medida de distância definida no espaço de padrões. Pode-se ainda estender esta regra para os k vizinhos mais próximos a $\underline{x}$, usando-se a classe que for maioria para os k vizinhos mais próximos para a classificação de $\underline{x}$. Esta regra é conhecida como classificação do tipo **k vizinhos mais próximos** *(the k nearest neighbours)*, o qual se torna **distância mínima** quando $k = 1$.

Se *a priori* conhece-se as informações sobre os padrões em estudo de tal forma que com poucas características extraídas do objeto, pode-se reconhecer a classe a que ele pertence, então o sistema de classificação pode ser projetado através da utilização de propriedades estatísticas [Schalkoff 1992]. Desta forma, pode-se representar a probabilidade a priori $P (w_i)$ a probabilidade que um objeto pertença à classe w_i , antes de que qualquer medida tenha sido feita.

Dada uma medida x feita sobre o objeto em estudo, define-se a probabilidade a

posteriori $P\ (\ w_i / x\ )$ como sendo a **probabilidade condicional** de que o objeto com valor x esteja na classe w_i. Define-se também a **densidade de probabilidade condicional** $P\ (\ x / w_i\ )$ como sendo uma função das características que representam a probabilidade de que um membro da classe w_i tenha o valor x. Isto implica na existência de uma função $P\ (\ x / w_i\ )$ diferente para cada valor de w.

A classificação estatística de imagens é formalizada pela teoria estatística de decisão. Segundo tal teoria, os atributos constantes do vetor x são variáveis aleatórias. Supõe-se que a função densidade de probabilidade associada a cada classe w_i, seja conhecida, assim como as probabilidades *a priori* de cada classe.

O método estatístico de classificação de padrões mais conhecido é baseado na regra de Bayes, o qual faz uso destas probabilidades [Duda e Hart 1976].

$P\ (\ w_i\ )$ é chamada **probabilidade a priori** e reflete o conhecimento *a priori* da distribuição estatística dos objetos presentes na cena antes desta ser vista. Isto quer dizer, ao se decidir classificar um objeto conhecendo-se apenas suas probabilidades *a priori*, então a seguinte regra de decisão poderia ser usada: decidir por w_1 se $P(\ w_1\ ) > P\ (\ w_2\ )$, caso contrário decidir por w_2 . Por exemplo, conhecendo-se para cada classe, a probabilidade *a priori* $P\ (\ w_i\ )$, a densidade de probabilidade condicional $P\ (\ x / w_i\ )$ e o valor da medida da característica $\underline{x}$, como estas medidas podem levar a uma correta classificação? A resposta é dada pela fórmula de Bayes:

$$P\ (\ w_i / x\ ) \quad = \quad \frac{P\ (\ x / w_i\ )\ .\ P\ (\ w_i\ )}{\sum_{j=1}^{k} P\ (\ x / w_j\ )\ P\ (\ w_j\ )} \qquad \textbf{Equação 3.6.1}$$

para $i \in [\ 1, k\ ]$. (onde k é o número de classes disponíveis).

A fórmula de Bayes mostra como buscar a probabilidade *a posteriori* $P (w_i / x)$ em função da probabilidade *a priori* $P (w_i)$ e da distribuição da característica x de cada classe, ou seja, após as medidas, deve ser possível usar as probabilidades condicionais a fim de se aumentar o conhecimento sobre os membros das classes. A chamada formulação bayesiana pode envolver também custos de se tomar uma decisão por uma classe w_1 quando a verdadeira é w_2. Desse modo, sendo observado o vetor de atributos x, existe um risco médio ao se tomar uma decisão. O objetivo final é tomar uma decisão que minimize o risco médio sobre a distribuição dos atributos. Nestas condições, a minimização do risco médio é feita se, para cada vetor x de atributos observado, decidir-se pela classe w_i que maximize a probabilidade *a posteriori* $P (w_i / x)$ [Duda & Hart, 1973], [Bow, 1984], [Fukunaga, 1972], [Niblack, 1986], [Schalkoff, 1992] e [Watanable, 1985].

[Duda e Hart 1973] mostram que a classificação de objetos na Figura 3.6.2 é dada então pela seguinte regra: decida por w_1 se:

$$P (w_1 / x) > P (w_2 / x) \qquad \textbf{Equação 3.6.2}$$

caso contrário, decida por w_2. Esta é conhecida como **regra de Bayes**.

A regra de Bayes usada como classificador pode ter mais fatores tais como o uso de mais de uma característica e o uso de mais classes. Substituindo-se a regra de Bayes dada na equação 3.6.1 na equação 3.6.2, tem-se:

$$P (x / w_1) . P (w_1) > P (x / w_2) P (w_2) \qquad \textbf{Equação 3.6.3}$$

como condição para atribuir à classe w_1 o objeto que tem característica x.

Desta forma, pode-se considerar o processo de classificação como o cálculo da função discriminante $P (x / w) P (w)$ para cada vetor de atributos x observado e, pela distribuição deste vetor à classe w_i que forneça a máxima função discriminante. A Figura 3.6.5 mostra um diagrama de blocos representativo deste classificador.

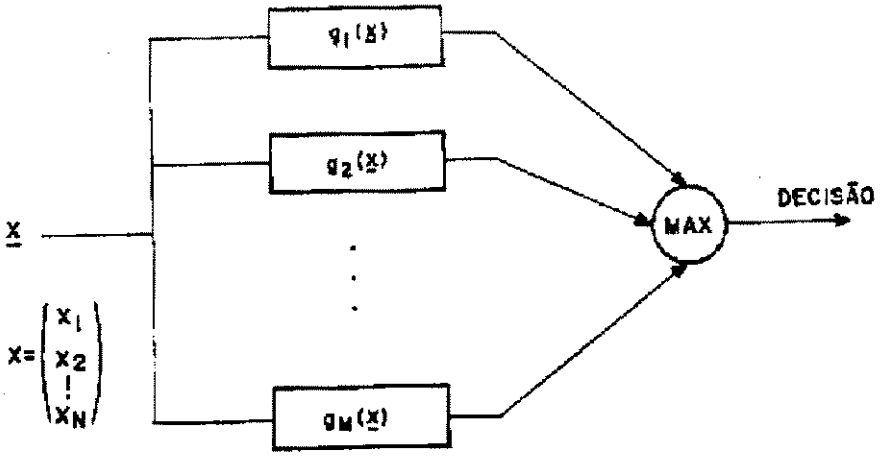


Figura 3.6.5 -Classificador de decisão por funções discriminantes

Quando se admite que as probabilidades *a priori* $P(w_i)$, $i \in [1, N]$ sejam iguais a 1/N, as funções discriminantes resumem-se às funções densidade de probabilidade condicionais $P(x/w_i)$.

Este classificador recebe o nome de **classificador de erro mínimo** ou **máxima verossimilhança** (*maximum likelihood*).

Resumindo: este método de classificação atribui a característica $\underline{x}$ à classe w_i quando a probabilidade $P(w_i/x)$ é máxima. Isto requer o conhecimento da probabilidade $P(w_i/x)$ para todo i. Não se tem inicialmente estes valores, mas pelo teorema de Bayes eles podem ser calculados utilizando-se $P(x/w_i)$ e $P(w_i)$, as quais também não se conhecem, mas podem ser estimadas. Pode-se estimar $P(x/w_i)$ tomando o histograma normalizado da classe w_i e estimando-se $P(w_i)$ pela frequência de ocorrência da classe w_i. Como o denominador $P(x)$ na regra de Bayes é o mesmo para todo i, este pode ser ignorado na seleção do máximo de $P(w_i/x)$.

A implementação do classificador pela equação 3.6.3 eqüivale a uma partição do espaço N-dimensional de vetores de atributos x em regiões. No caso de duas classes o cálculo

da probabilidade de erro de decisão pode ser feito observando-se que há duas maneiras de se cometer um erro, isto é, o vetor x provém da classe w_1 e decidiu-se pela classe w_2, ou vice versa [Duda & Hart, 1973]. Veja na Figura 3.6.6 um exemplo para um caso unidimensional.

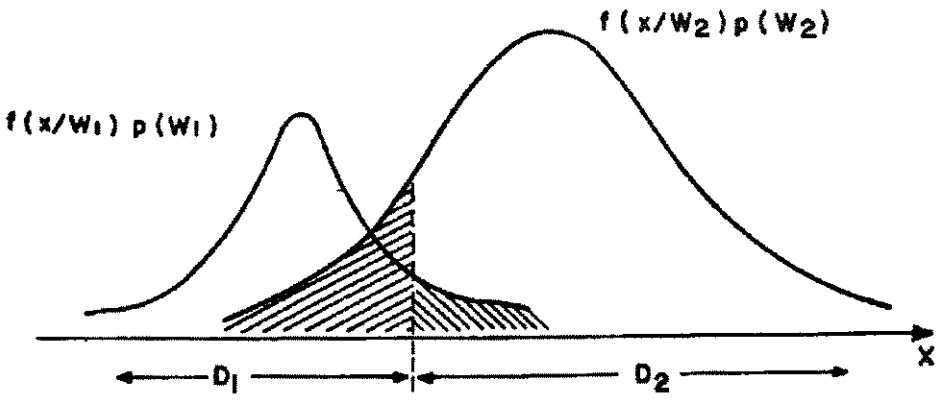


Figura 3.6.6 - Cálculo das probabilidades de erro no caso gaussiano unidimensional

Neste caso o classificador dividiu o espaço em duas regiões: D_1 e D_2. Como estas regiões são arbitrariamente escolhidas, a probabilidade de erro não é tão pequena quanto se desejaria que fosse. O erro é dado pela soma da área sob $f(x/W_2)\,p(W_2)$ ao longo de D_1 e a área sob $f(x/W_1)\,p(W_1)$ ao longo de D_2. Movendo-se a linha de decisão para a esquerda, o erro de classificação vai ser reduzido e, consequentemente a probabilidade de erro também é reduzida. Em geral, se $p(x/w_1)\,P(w_1) > p(x/w_2)\,P(w_2)$ é vantagem ter-se x na região D_1 de tal forma que menores quantidades irão contribuir para o erro. Isto é exatamente o que faz a regra de decisão de Bayes.

No caso multidimensional o cálculo da probabilidade de erro torna-se bem mais complicado.

A seguir é apresentada uma seqüência de passos necessários à classificação de uma imagem usando este método:

1) Selecionam-se as áreas de treinamento para cada classe candidata. Destas áreas conhecem-se (ou assume-se) os objetos da classe. Em geral, quanto mais dados no conjunto de treinamento da classe, melhor a representação do histograma da classe.

2) Calcula-se o histograma de dimensão N para o conjunto de treinamento (veja próximo tópico deste capítulo) de cada classe, onde N é o número de características; obtém-se o histograma normalizado $H_i\ (\ x\ )$. Utiliza-se este como estimativa das funções de densidade de probabilidade $P\ (\ x\ /\ w_i\ )$, obtendo-se a probabilidade de um objeto ter o valor x dado que ele pertence à classe w_i .

3) Estimam-se as probabilidades a priori $P\ (\ w_i\ )$ e estas s para escalar as probabilidades $P\ (\ x\ /\ w_i\ )$, obtendo-se $P\ (\ x\ /\ w_i\ )\ .\ P\ (\ w_i\ )$, ou seja $P\ (\ w_i\ )\ H_i\ (\ x\ )$. Esta é a probabilidade de um objeto ter o valor x dado que ele pertence à classe w_i vezes a probabilidade do objeto estar na classe w_i .

4) Utiliza-se a equação de Bayes (3.6.1) para classificar cada objeto da imagem. Na prática, o denominador é desconsiderado, pois é o mesmo para todas as classes. Para classificar-se o objeto, deve-se atribui-lo à classe w_i onde $P\ (\ w_i\ /\ x\ )$ é máximo.

Os passos 1), 2) e 3) são realizados somente uma vez, enquanto que o passo 4) é realizado para cada objeto da imagem. O termo $P\ (\ w_i\ /\ x\ )$ é normalmente considerado como sendo função de x, e é uma probabilidade.

Este algoritmo apresenta a idéia básica para a classificação chamada **não paramétrica** onde $P\ (\ x\ /\ w_i\ )$ é estimado pelo histograma do seu conjunto de treinamento. No entanto, existe um problema para sua implementação: no passo 2, $P\ (\ x\ /\ w_i\ )$ é calculado de

um histograma normalizado N dimensional da imagem, o que requer um conjunto muito grande de dados para ser significativo. Este problema pode ser evitado, representando-se $P\ (\ x\ /\ w_i\ )$ não como um histograma N dimensional, mas sim como funções para as quais somente poucos parâmetros seriam necessários. Este método é chamado **classificador paramétrico**. Os parâmetros envolvidos devem ser então estimados ou calculados, e a representação mais comum de $P\ (\ x\ /\ w_i\ )$ é a distribuição Gaussiana, na qual para o caso de uma dimensão, os parâmetros que precisam ser estimados são a média e o desvio padrão da distribuição.

3.6.4 TREINAMENTO E APRENDIZADO

Na classificação de padrões, quanto melhores (mais discriminantes) forem as informações usadas, mais eficiente o sistema ficará. No item anterior mostrou-se que a classificação dependia do conhecimento das densidades de probabilidade que caracterizam as classes. No entanto, esta condição nem sempre é satisfeita, as informações nem sempre estão disponíveis ou são difíceis de serem extraídas. Daí uma das formas de lidar-se com este problema é estudar algumas amostras de padrões cujas características e atributos são perfeitamente conhecidos *a priori*. O conjunto de padrões onde atributos típicos, ou classes ou estruturas são conhecidas, formam um banco de dados chamado **conjunto de treinamento** (*training set*), que contém uma representação particular dos padrões que se quer classificar. De uma forma geral, o conjunto de treinamento contém informações significativas de como associar os dados de entrada com as decisões de saída tais como classificações ou descrições estruturais. O problema, então, é achar um meio de usar as informações disponíveis para classificar uma imagem.

Uma estratégia para este problema é aquela que usa amostras para estimar as probabilidades e densidades de probabilidades desconhecidas, e estes resultados estimados são utilizados como se fossem os verdadeiros valores. Num problema típico de classificação de padrões, a estimação das probabilidades *a priori* não é muito difícil. Entretanto, a estimação das densidades de probabilidades é mais complicada. O número de amostras disponíveis é sempre menor do que o necessário, e o problema fica mais complicado quando a dimensionalidade do vetor de característica x é grande.

Treinamento é intuitivamente associado com aprendizado. O conjunto de treinamento é usado para possibilitar o sistema **aprender** informações relevantes, tais como parâmetros estatísticos, características chaves, estruturas. Um sistema de aprendizado deve adaptar sua estrutura interna a fim de sempre obter a melhor resposta, baseado em medidas prévias de desempenho. Estas medidas de desempenho podem ser a diferença entre a resposta real do sistema e a resposta desejada. O conceito geral de aprendizado está relacionado com técnicas de erro-correção empregadas no aprendizado biológico (animal ou humano), o qual se baseia no fato de que são necessários alguns experimentos até que o aprendizado esteja completado. [Schalkoff 1990] mostra uma curva típica de aprendizado a qual resume este conceito. Veja-a na Figura 3.6.7.

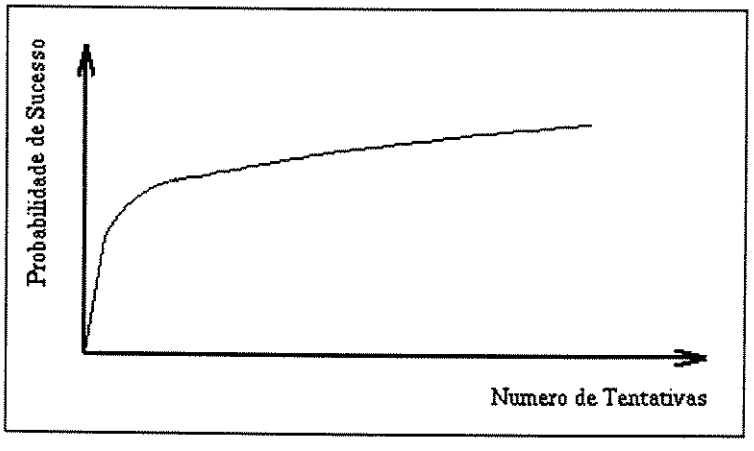


Figura 3.6.7 - Curva típica de aprendizado

As técnicas utilizadas no treinamento e aprendizado podem ser resumidas da seguinte forma [Mascarenhas & Velasco, 1984]:

a) aprendizado supervisionado

Quando a forma funcional das densidades é conhecida, mas não seus parâmetros, tem-se um problema de aprendizado, o qual pode ser caracterizado por um processo de estimação dos parâmetros desconhecidos. Quando existem amostras disponíveis de classificação já definida, tem-se então um problema de aprendizado supervisionado. Se os parâmetros a serem estimados forem não aleatórios, as técnicas de estimação por máxima verossimilhança podem ser empregadas. No entanto, para os parâmetros aleatórios, deve-se empregar as técnicas de estimação bayesianas.

b) aprendizado não supervisionado

No aprendizado não supervisionado, as amostras no conjunto de treinamento não são rotuladas e a solução para o problema de classificação só é possível se o número de classes e suas respectivas probabilidades *a priori* são conhecidas, assim como a forma das densidades $P(x/w_i)$, exceto por seus parâmetros [Duda & Hart, 1973].

c) técnicas não paramétricas

Quando a forma das densidades de probabilidade não é conhecida, as formas paramétricas dificilmente descreverão com fidelidade as densidades encontradas na prática. Neste caso, deve-se então utilizar as técnicas não paramétricas, as quais envolvem a estimação das densidades condicionais de cada classe a partir de amostras. Outras técnicas estimam diretamente as probabilidades *a posteriori* das classes, que são usadas como funções discriminantes, sem passar pelas densidades condicionais. Estes procedimentos estão relacionados com o método de classificação conhecida mais próxima (vizinho mais próximo).

Em geral, utilizando-se as técnicas não paramétricas, troca-se a necessidade de se conhecer a forma funcional das densidades pela necessidade de um número maior de amostras.

A tarefa de reconhecimento de padrões para proceder o aprendizado inclui as seguintes atividades:

1. faz-se um estudo completo das possíveis classes de padrões possíveis em seus dados;
2. determina-se a disponibilidade dos dados de características ou medidas;
3. deve-se considerar o sistema computacional disponível, principalmente quando se fala a respeito de precisões de cálculos;
4. deve-se considerar a disponibilidade do conjunto de treinamento;
5. deve-se estudar as técnicas de reconhecimento de padrões disponíveis;
6. desenvolve-se um sistema de reconhecimento simulado;
7. treina-se o sistema;
8. simula-se o desempenho do sistema;
9. interage-se com os passos anteriores até que se encontre o desempenho desejado.

3.6.5 Método de Treinamento Adotado

Quando a única informação possível para se projetar o classificador consiste em amostras não rotuladas, a classificação por **aglomerados** *(clusters)* deve ser utilizada para se obter as densidades de probabilidade das amostras de padrões disponíveis. As técnicas desenvolvidas para este fim baseiam-se freqüentemente na minimização de um critério derivado de uma medida de similaridade entre amostras.

O ponto essencial deste algoritmo é construir os aglomerados usando-se medidas de distâncias, as quais são um tipo de medida de similaridade cujo objetivo principal é particionar o aglomerado de amostras num espaço N-dimensional.

Para o sistema SRIDE foi adotado um método de treinamento onde inicialmente todos os aglomerados já conhecidos pelo usuário, são armazenados num **banco de conhecimentos.** Estes aglomerados contém os objetos (símbolos) dos quais o usuário já tem conhecimento *a priori* de suas características e portanto já podem ser **classificados**. Cada aglomerado representa uma classe de objetos ou símbolos possíveis num diagrama de circuitos.

Logo, para se proceder ao treinamento inicial do sistema, este assume que a entrada (constituída do vetor de atributos) de cada um dos k símbolos disponíveis corresponde a um novo símbolo (quando comparado seu vetor de características com os outros já existentes no banco de conhecimentos e não for encontrado nenhum igual a ele) e portanto, exige a intervenção do usuário especialista para definir sua identificação e por conseqüência, sua classificação. Nessas condições, tem-se:

$$\underline{x}_i = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_n \end{bmatrix}$$ onde:

$i \in [\, 1, k \,]$

n : dimensão do espaço de atributos

$\underline{x}$: vetor de atributos

k : número de amostras.

Se durante a entrada destes k símbolos houver símbolos com medidas diferentes, mas classificados como iguais, estes formarão os aglomerados de símbolos de mesma classe.

Ao final desta entrada de dados tem-se os m aglomerados de símbolos (correspondentes às m classes), tal que cada aglomerado W_j , $j \in [\ 1,\ m\ ]$, possui r_j símbolos representativos daquela classe j .

Supondo-se $P\ (\ w_i\ )$ iguais, isto é, o número de símbolos de cada classe é o mesmo, e supondo-se também que a distribuição $P\ (\ \underline{x}\ /\ w_i\ )$ é gaussiana com atributos estatisticamente independentes, tem-se que a matriz de covariância associada a cada classe i é uma matriz diagonal Σ_i dada por:

$$\Sigma_i = \begin{bmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \sigma_n^2 \end{bmatrix}$$

onde o centro de cada classe i é dado por:

$$\underline{\mu}_i = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \ldots \\ \mu_n \end{bmatrix}$$

o estimador de máxima verossimilhança é dado por:

$$\underline{\mu}_i = \frac{1}{r_i} \sum_{j=1}^{r_i} \underline{x}_{ij}$$

onde :

$i \in [\ 1,\ m\ ]$ { corresponde à classe }

r_i : número de amostras da classe i

$\underline{x}_{ij}$: é a j-ésima amostra $\underline{x}$ da classe i

Logo,

$$\mu_{is} = \frac{1}{r_i} \sum_{j=1}^{r_i} x_{ijs}$$

$s \in [\ 1, n\ ]$ { corresponde ao atributo da amostra }

n : número de atributos

O estimador das variâncias é dado por:

$$\underline{\sigma}_i^2 = \frac{1}{r_i - 1} \sum_{j=1}^{r_i} \left(\underline{x}_{ij} - \underline{\mu}_i \right)^2$$

onde:

$$\underline{\sigma}_i^2 = \begin{bmatrix} \sigma_{i_1}^2 & 0 & 0 & 0 \\ 0 & \sigma_{i_2}^2 & 0 & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \sigma_{i_n}^2 \end{bmatrix}$$

e portanto:

$$\sigma_{is}^2 = \frac{1}{r_i - 1} \sum_{j=1}^{r_i} \left(x_{ijs} - \mu_{is} \right)^2$$

Ao final desta etapa duas situações são possíveis para cada uma das classes i no banco de conhecimentos:

a) $r_i \geq t$

b) $r_i < t$

Onde t (valor empírico) é a quantidade mínima de símbolos necessários à classe para considerá-la como **treinada.** Isto significa que no caso a) o símbolo já está treinado, e no caso b) o símbolo ainda está em **treinamento**.

Uma vez o símbolo tenha sido treinado, ele passa a ser representado apenas pela sua média μ e seu desvio padrão σ. Enquanto em treinamento, são guardados os r_i vetores de atributos dos símbolos pertencentes à classe i.

Esta é a primeira fase de vida do sistema de classificação, a qual produz um banco de conhecimentos inicial, através do treinamento e aprendizagem dos símbolos fechados disponíveis.

Após o treinamento ter sido considerado completado (isto significa que não há mais mudanças na classificação de x), pode-se deixar o sistema trabalhar livremente no processo de classificação de um número qualquer de padrões.

Este algoritmo tem como vantagens, o fato de ser simples e eficiente, não são necessários muitos cálculos, os padrões podem ser processados na ordem em que aparecem e não há necessidade de se especificar *a priori* um número de classes. Por outro lado, o método é mais eficiente quando os aglomerados são compactos e bem separados e ainda o método é altamente dependente do valor de distância que determina o raio da região do aglomerado.

3.6.6 Sistema de Classificação Adotado

O classificador paramétrico baseado na regra de Bayes discutido anteriormente é provavelmente o melhor e o mais usado para classificar-se uma imagem, quando se conhece as distribuições de probabilidades. No entanto, para sistemas onde a distinção entre as classes é marcante, como ocorre com este projeto, o investimento dispendido na aplicação de tal classificador não se justifica. A idéia é aprimorar a etapa anterior, ou seja, a segmentação, de tal forma a se obter as características dos objetos da imagem em grupos bem definidos, possibilitando maior facilidade na distinção entre elas.

Desta forma, pode-se usar um método classificatório menos sofisticado, mas que apresenta resultados também confiáveis, visto que na fase de segmentação obtém-se os vetores

representativos da imagem, que submetidos ao processo de extração de características proporcionam valores que definem aglomerados de padrões distintos.

Nesta fase do processo o sistema começa a ter vida própria, ou seja, deve receber as imagens contendo símbolos e classificá-los.

Para cada um dos símbolos fechados apresentados ao sistema, calcula-se a distância de Mahalanobis entre o símbolo apresentado e as classes existentes. O objetivo é classificar o símbolo numa das classes existentes ou caso isto não seja possível, criar nova classe. As seguintes situações são possíveis:

a) o símbolo pertence a uma classe já treinada e portanto, ele é classificado nesta classe;

b) o símbolo pertence a uma classe em treinamento e neste caso, ele é agregado àquela classe, sendo que os valores de r , μ e σ da classe devem ser atualizados.

c) o símbolo não pertence a nenhuma das classes treinadas e então o usuário é chamado para atribuir a ele uma identificação. Duas situações são possíveis:

c.1) a classe atribuída ao símbolo é nova e neste caso o símbolo passa ter o mesmo tratamento de uma classe em treinamento, como ocorreu na primeira fase do classificador durante seu treinamento inicial. Atribui-se à esta nova classe o valor da variância da classe mais próxima.

c.2) a classe atribuída ao símbolo já está treinada, significando desta forma que o símbolo representa de forma diferente um elemento do diagrama elétrico.

Desta forma tem-se um sistema de classificação flexível, ou seja, atende a novas classes quando estas aparecem. Se o número de amostras para cálculo de $\underline{\mu}$ for maior que t então supõe-se que a classe está conhecida (treinada). Caso contrário, ela estará em treinamento e a cada novo elemento que for classificado no critério usual, participará na atualização do novo $\underline{\mu}$, $\underline{\sigma}$ e $\underline{R}$, onde este último representa **raio máximo** do agregado.

O classificador usa como critério a **distância mínima de *Mahalanobis*** (Duda & Hurt, 1972), dada por:

$$\left\| \underline{x} - \underline{\mu} \right\|_{\Sigma}^{2} = \left[\underline{x} - \underline{\mu} \right]^{t} \Sigma^{-1} \left(\underline{x} - \underline{\mu} \right)$$

Como Σ é diagonal, tem-se que:

$$\left\| \underline{x} - \underline{\mu} \right\|_{\Sigma}^{2} = \sum_{i=1}^{n} \left(\frac{x_i - \mu_i}{\sigma_i} \right)^{2}$$, onde i é cada atributo.

A classificação se faz pela obtenção da distância mínima de Mahalanobis entre o objeto em estudo e as classes já classificadas, desde que a seguinte regra seja também observada:

$$\left\| \underline{x} - \underline{\mu} \right\|_{\Sigma}^{2} < \alpha$$

onde α é um valor, geralmente igual a 3.0, de tal forma a atingir toda população do agrupamento em estudo (isto porque com a hipótese gaussiana, 3 desvios padrões abrangem mais ou menos 95% da população).

3.6.7 DESEMPENHO DO CLASSIFICADOR

De uma forma geral, um classificador é avaliado pela sua probabilidade de erro de classificação [Fukunaga, 1972]. No caso de múltiplas classes envolvendo a média de vários atributos, o cálculo direto da probabilidade de erro torna-se muito difícil. A solução seria estimar esta probabilidade a partir do desempenho do classificador. No entanto, este é um procedimento otimista, visto que se está usando as próprias amostras de treinamento. Daí, torna-se usual dividir as amostras de classificação conhecida em área de treinamento e área de testes, e então avaliar o classificador sobre essas últimas.

Dado um conjunto de amostras de classificação conhecidas, tem-se então o problema de como dividi-las em amostras de treinamento e de testes. Se muitas amostras forem selecionadas para treinamento e poucas para teste, a avaliação de desempenho do classificador será pobre. Caso a estratégia oposta seja adotada, o projeto do classificador poderá ficar comprometido. Quando se dispõe de um pequeno número N de amostras, é preferível fazer a estimativa da probabilidade de erros em N passagens, em cada uma das quais uma observação é mantida para teste e as restantes N - 1 são usadas para treinamento. A disponibilidade de um número finito de amostras para treinamento pode afetar significativamente o desempenho do classificador, tendo sido observado que, nessa situação, o desempenho do classificador não aumenta indefinidamente com o aumento do número de atributos [Mascarenhas & Velasco, 1984].

No capítulo seguinte, são mostrados os resultados obtidos do classificador adotado neste sistema na classificação de várias imagens.

CAPÍTULO 4

4. RESULTADOS

4.1 INTRODUÇÃO

O sistema computacional SRIDE (Sistema Reconhecedor de Imagens de Diagramas Elétricos) desenvolvido para a execução desta pesquisa incorpora todos os módulos mencionados no capítulo anterior, ou seja, extração dos caracteres, segmentação, vetorização, obtenção dos laços e classificação. Eles estão escritos em linguagem C e foram implementados no sistema operacional MS-DOS e algumas partes no sistema operacional LINUX, tendo em vista facilitar o manuseio de imagens de maior porte.

4.2 ORGANIZAÇÃO DO SISTEMA

O sistema computacional SRIDE está organizado de forma que todos os módulos são independentes mas podem ser gerenciados por um único módulo de comando, cujo objetivo é integrar todas as partes do sistema, para o reconhecimento de uma imagem de um diagrama de circuito elétrico ou eletrônico. Considera-se ainda que este sistema pode ser utilizado para a execução de diversas aplicações, como as descritas a seguir.

4.2.1 Aquisição da Imagem

As imagens de entrada são obtidas através de um digitalizador (*scanner*) de mesa, e são armazenadas em preto e branco, com resolução de 300 dpi e em tamanhos variáveis (limitados fisicamente a um retângulo de aproximadamente 2000 x 1000 pixels). O *software* para aquisição e armazenamento da imagem é produzido pela própria empresa que fabrica o digitalizador e ele controla tamanho e níveis de binarização da imagem. Tendo em vista que o uso do formato TIFF (veja apêndice C) é bastante difundido na comunidade científica devido a sua grande eficiência, este foi o formato escolhido para o manuseio das imagens do sistema SRIDE.

O digitalizador pode também fornecer a imagem não binarizada e em níveis de cinza, e desta forma o usuário do sistema SRIDE pode fazer uso do módulo de binarização, discutido no próximo item, para proceder a binarização da imagem de entrada com valores de limiarização escolhidos por ele.

4.2.2 Histograma e Binarização da Imagem

Para que partes ou toda a imagem possam ser binarizadas e visualizadas, é necessária a execução das operações de segmentação da imagem, separando a imagem do objeto, da imagem do fundo. Neste caso, é o usuário quem decide pelo valor de limiar *(threshold)* a ser utilizado na binarização da imagem. O processo é bastante dinâmico pois o sistema permite a imediata iteração do usuário com o processo; para auxiliar o usuário que decisão tomar quanto ao valor do limiar de binarização, o sistema fornece imediatamente a imagem obtida com o valor do limiar escolhido.

4.2.3 Visualização da Imagem

Com a finalidade de se verificar o padrão de qualidade das imagens capturadas e armazenadas em memória auxiliar para serem colocadas à disposição dos demais módulos do SRIDE, a qualquer momento, o sistema permite a visualização das mesmas no monitor de vídeo ou ainda sua impressão, usando padrão PostScript .Os módulos SHOWTIF e PRINTIF são responsáveis respectivamente pela visualização e impressão das imagens no sistema SRIDE.

4.2.4 Desenhos Editados

Destinado ao treinamento e à formação de pessoal na área de Visão Artificial, este módulo permite que os usuários desenhem objetos bidimensionais e os incorporem ao conjunto de imagens sintéticas estudadas pelo sistema. Dessa forma, os desenhos realizados com o uso deste recurso podem ser utilizados para testes do sistema. Um conjunto básico de funções gráficas é colocado à disposição do usuário, com o objetivo de aumentar sua produtividade. O desenho é, portanto, feito sem a necessidade de programação, basta apenas seguir um menu de operações disponíveis no editor de desenhos desenvolvido para este fim no módulo EDITGRAF.

4.2.5 Extração de Caracteres

O módulo de extração de caracteres pode ser usado para o tratamento de diversas imagens, não só as de circuitos elétricos. Veja exemplos nas figuras a seguir.

Table 5-6. Precentage of Chips.

	table 1		table 2		table 3	
	BLACK HAT	GRAY HAT	BLACK HAT	GRAY HAT	BLACK HAT	GRAY HAT
colored chips	5	3	6	9	11	12
white chips	6	4	3	5	9	9
% colored	45	43	67	64	55	57

277

Figura 4.2.1 - Imagem de uma tabela completa Tiff

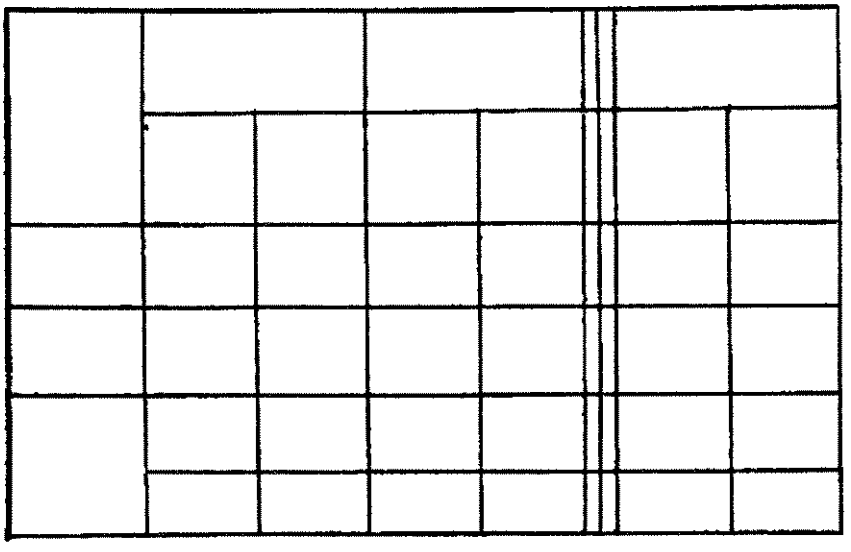


Figura 4.2.2 - Tabela da Figura 4.2.1 com os caracteres extraídos

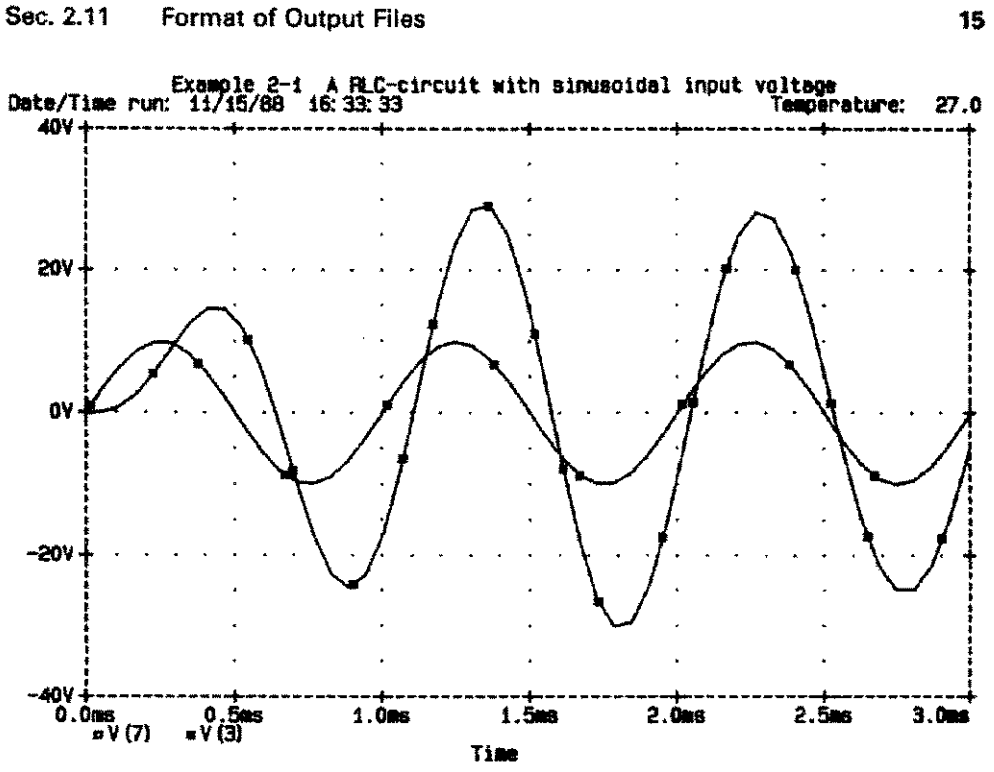


Figure 2.2 Transient response for Example 2.1

Figura 4.2.3 - Exemplo de um gráfico completo

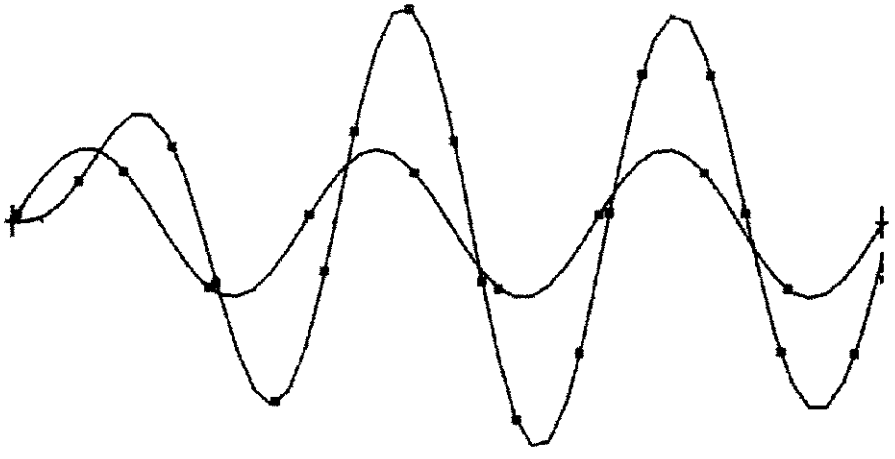


Figura 4.2.4 - Gráfico da Figura 4.2.3 com os caracteres extraídos

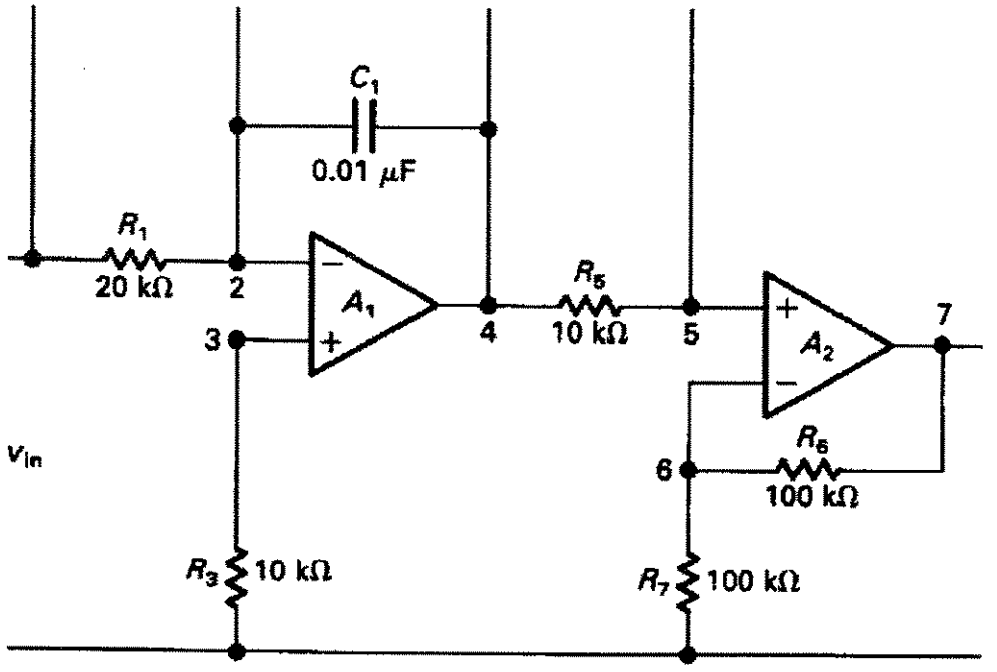


Figura 4.2.5 - Exemplo de um circuito elétrico completo

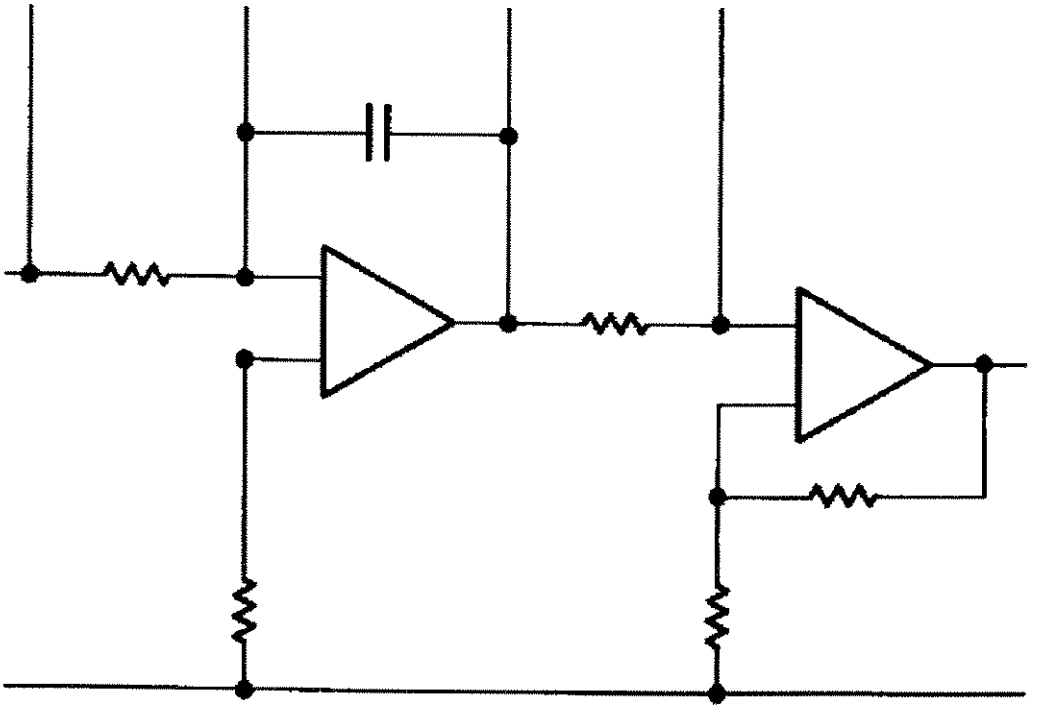


Figura 4.2.6 - Circuito da Figura 4.2.5 com os caracteres extraídos

Esta fase é muito útil durante o pré processamento de imagens onde os caracteres não são necessários, ou para "limpeza" da imagem ou ainda para o reconhecimento dos caracteres envolvidos na imagem através de um OCR especializado.

No caso do sistema SRIDE, temos integrado a ele o software Omnipage, que toma como entrada, o arquivo TIF contendo os caracteres extraídos da imagem pelo sistema SRIDE para proceder seu reconhecimento.

Pode-se melhorar a eficiência do sistema, integrando-se o resultado obtido pelo reconhecimento dos caracteres através do OCR, ao arquivo DXF (veja apêndice D) que representa a imagem com o formato utilizado pelo AUTOCAD, obtendo-se assim a imagem completa do diagrama de circuito constituída dos elementos e suas respectivas identificações, todos devidamente reconhecidos.

4.2.6 Vetorização

O módulo de vetorização, o qual converte imagens *raster* em vetores, também pode ser aplicado em outras imagens além daquelas relacionadas com diagramas de circuitos elétricos. O algoritmo aplicado é aquele explanado no item 3.3.3 e também foi desenvolvido na linguagem C e está disponível nas versões UNIX e DOS. Como nesta pesquisa as imagens são estáticas, a velocidade do processamento não foi considerado como fator preponderante no julgamento da eficiência do algoritmo e sim a qualidade dos resultados obtidos, embora nos exemplos estudados a velocidade da resposta foi bastante satisfatória. Veja os exemplos nas Figuras 4.2.7 e 4.2.8 a seguir, respectivamente a vetorização da tabela apresentada na Figura 4.2.2 e a vetorização do circuito elétrico apresentado na Figura 4.2.6. O resultado da

vetorização é armazenado num arquivo ASCII o qual é transformado (pelo próprio SRIDE) para o formato DXF e fica pronto para ser usado num CAD. No apêndice D, encontram-se vários outros exemplos.

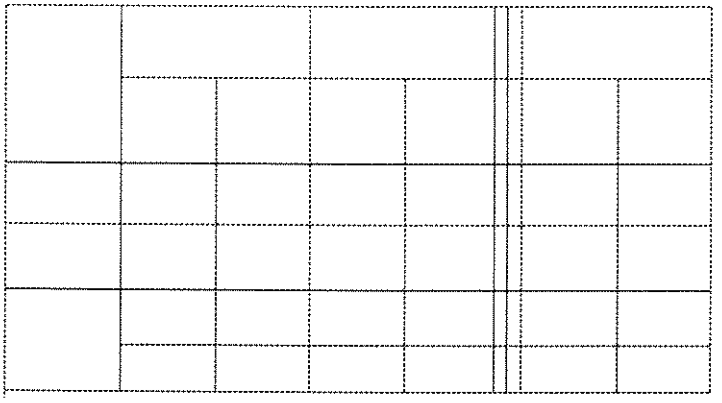


Figura 4.2.7 - Vetorização da tabela da Figura 4.2.1

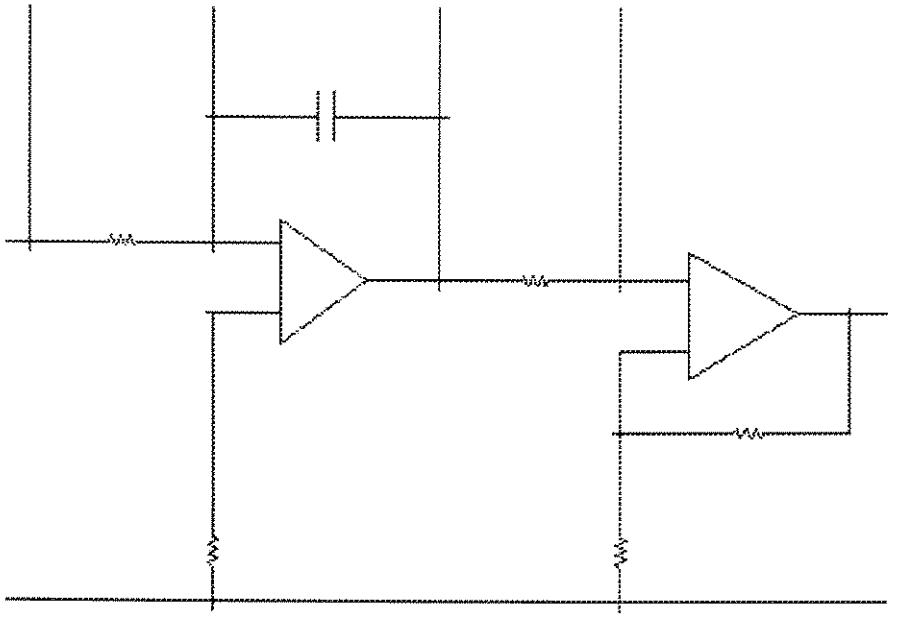


Figura 4.2.8 - Vetorização do circuito da Figura 4.2.5

Como já mencionado anteriormente, as imagens *raster* utilizadas no processo de vetorização, já passaram pelo processo de extração dos caracteres, visto que estes não são de interesse nesta fase de tratamento da imagem pelo sistema SRIDE.

A grande vantagem deste módulo é poder usá-lo para obter imagens vetoriais a partir de imagens *raster* e usar estas imagens vetoriais num AUTOCAD, após transformá-las para o formato eletrônico compatível. Esta tarefa também é realizada pelo sistema SRIDE.

Desta forma, a imagem do desenho é capturada via um digitalizador, armazenada eletronicamente na memória do computador no formato TIF e, então é transformada para o formato DXF, possibilitando que o usuário através de um CAD faça alterações em parte ou em todo o desenho.

4.2.7 Classificação

O módulo de classificação é específico para a finalidade a que se propõe, no presente pesquisa deve classificar símbolos num diagrama de circuito elétrico ou eletrônico. Após a vetorização da imagem do diagrama de circuito, é realizada a segmentação dos símbolos abertos conforme as regras mostradas em 3.4.4. Os vetores que constituem estes símbolos são retirados da lista inicial de vetores (formada por todos os vetores da imagem) e tem-se então uma imagem tal como a mostrada na Figura 4.2.9 de nosso exemplo. Como se vê na figura, apenas os símbolos fechados e as linhas de interconexão estão presentes no desenho. Observe também que os falsos laços formados pelos símbolos abertos foram desfeitos quando estes foram retirados do desenho. Para a classificação dos símbolos fechados, os laços presentes no circuito são detectados (veja-os na Figura 4.2.10 para o nosso exemplo)

conforme explanado em 3.5 e os vetores de características para estes laços são montados conforme as regras estabelecidas em 3.4.5.

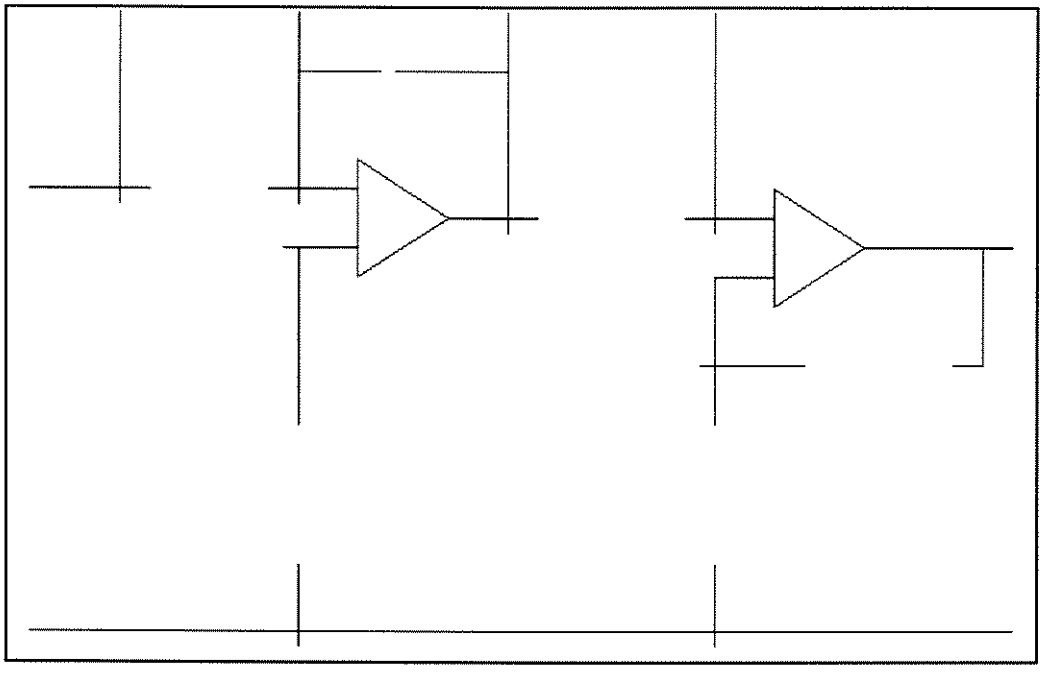


Figura 4.2-9 - Segmentação dos símbolos abertos do circuito da Figura 4.2.8

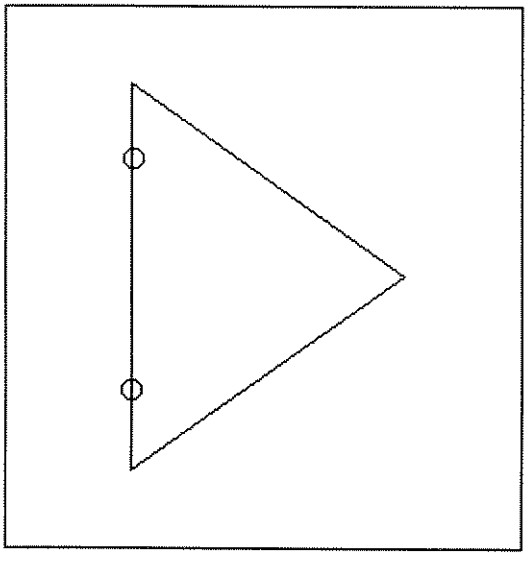


Figura 4.2-10 -

Laço presente no circuito da Figura 4.2.9

observar que:

- este laço aparece 2 vezes na imagem
- os círculos representam os pontos entrada no símbolo.

A classificação dos símbolos se dá usando-se o critério da distância mínima de Mahalanobis (conforme mostrado em 3.6.5) entre os laços em estudo e as classes já classificadas. Neste exemplo em particular, o sistema classifica os laços obtidos no desenho como sendo "amplificador operacional" , visto que têm 2 entradas e 1 saída.

Todas as demais linhas restantes no desenho são classificadas como linhas de interconexão, pois ou são linhas onde os dois extremos conectam símbolos, ou são linhas onde um extremo conecta símbolos e outro extremo é final de linha.

Ao final desta tarefa, a lista do circuito (*net list*) reconhecido é gerada e é enviada ao programa comercial Pspice para análise topológica do circuito. Se algum erro ocorrer, o usuário é chamado a intervir. Caso contrário a saída gráfica é gerada e enviada a um CAD para ser manuseada pelo usuário.

C0	2	9	capacitor
R1	12	19	resistor
R2	25	33	resistor
R3	43	50	resistor
R4	58	65	resistor
R5	68	75	resistor
E0	23	41	amplificador
E1	37	85	amplificador

Net list gerada pelo SRIDE para o circuito da Figura 4.2.9

(veja Figura 4.4.1 para referência dos vértices)

4.3 SOFTWARES COMERCIAIS INTEGRADOS

Para maior eficiência dos resultados obtidos integramos ao sistema SRIDE alguns softwares comerciais, cujos objetivos são descritos nos próximos itens.

4.3.1 Turbo CAD

Turbo CAD é um *software* desenvolvido pela IMSI LIMITED, San Rafael, CA, EUA, que tem por objetivo processar um mini CAD, com muitos dos recursos encontrados num CAD de grande porte. Como o sistema SRIDE apenas utiliza CAD para correções simples dos desenhos obtidos através do reconhecimento de diagramas, este CAD não precisa ser de grande abrangência e com vários recursos pois, muitos dos quais não precisam ser usados pelo sistema SRIDE. Assim sendo, optou-se por um produto simples, de fácil implementação e que realiza as tarefas mais comuns que um CAD normalmente realiza , sem deixar de ser compatível com todos os outros CADs disponíveis no mercado. Tem como vantagem adicional, o fato de poder ser implementado e utilizado em qualquer microcomputador com um mínimo de recursos gráficos.

Características:

- importa e exporta em formatos DXF, HPGL e ASCII;
- suporta impressoras e *ploters* comuns;
- algumas operações comuns:
  - edita linhas, faz transformações geométricas;
  - permite *zoom* e *pan*;

- escolhe penas, cores e tipos de caracteres;
- etc.

- possui 256 *layers*, várias fontes de textos e várias janelas de desenho;
- etc.

Observação: todos os desenhos vetorizados apresentados no apêndice E se utilizam deste software para sua impressão.

4.3.2 Pspice

O *software* Pspice foi desenvolvido pela MicroSim Corporation para ser rodado em microcomputadores. É um simulador de circuitos similar ao SPICE desenvolvido pela Universidade da Califórnia. Pspice é constituído de um PROBE, que é um processador gráfico muito útil na plotagem dos resultados da simulação efetuada pelo Pspice (veja exemplo na Figura 4.2.3).

Em cada simulação, Pspice gera um arquivo relatório do estado do circuito em estudo. É exatamente este arquivo que o sistema SRIDE utiliza para verificar se o diagrama em estudo está ou não correto do ponto de vista das conexões elétricas. Encontrados erros, o sistema alerta o usuário que deverá tomar as providências necessárias para corrigi-los.

Características:

- avalia os efeitos de variações em elementos tais como resistores, transistores, etc.;

- avalia os efeitos de ruídos, no circuito;
- permite análise de Fourier;
- avalia os efeitos dos elementos não lineares na *perfomance* do circuito;
- não é iterativo e não suporta um método iterativo de solução. Se os elementos de um circuito são especificados a saída pode ser prevista. Por outro lado, se a saída é especificada, Pspice não pode ser usado para sintetizar os elementos do circuito.

4.3.3 Cadastro dos Elementos da Imagem

Este módulo mantém o controle e permite a visualização das imagens dos objetos reconhecidos e armazenados num banco de dados de imagens próprio para este fim. Este procedimento facilita operações de adição, exclusão e identificação de elementos de imagens pelo usuário no banco de dados de elementos.

4.3.4 Apresentação do Sistema SRIDE

O menu principal de apresentação do sistema SRIDE é mostrado na Figura 4.3.1 a seguir:

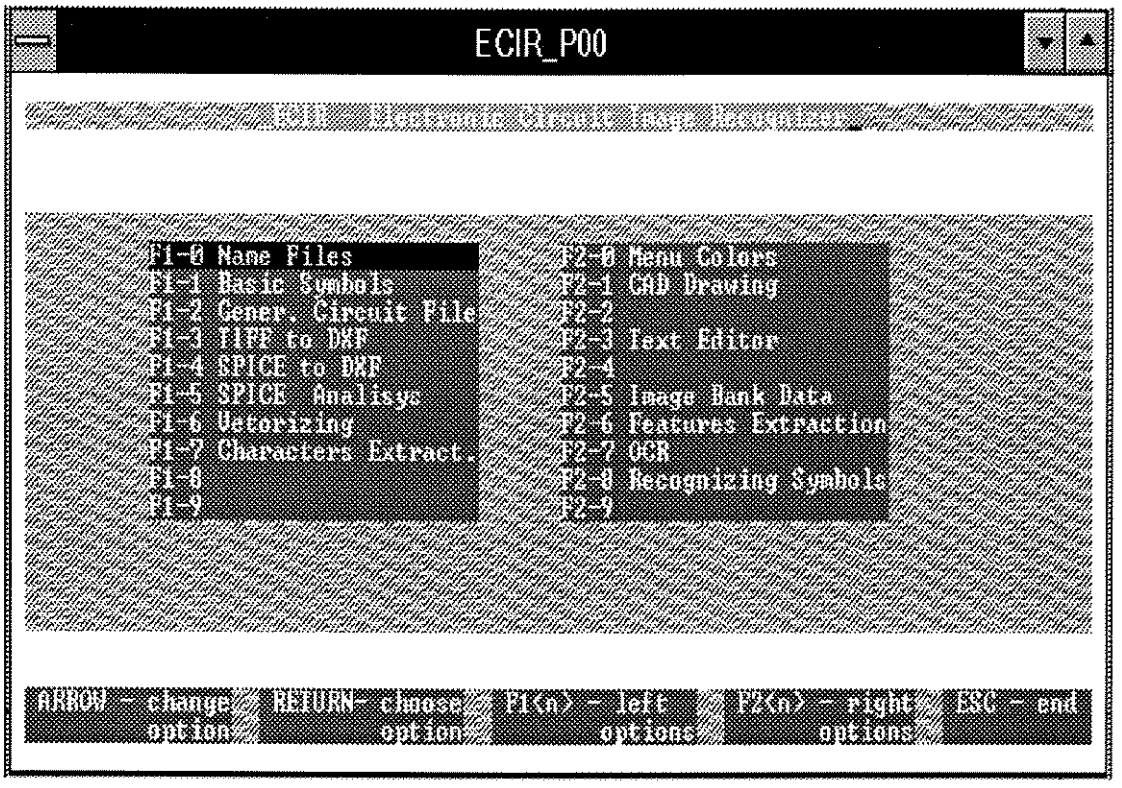


Figura 4.3-1 - Menu principal do sistema SRIDE

Módulos que compõem o sistema SRIDE e que são acessados pelo seu menu principal mostrado na Figura 4.3.1

- teste do circuito via Pspice;
- transformações de vários tipos de arquivos;
- acesso ao CAD;
- definição do ambiente gráfico disponível;
- vetorização;
- OCR;
- extração de caracteres da imagem;

- extração de características dos elementos;
- reconhecimento dos elementos da imagem;
- visualização do banco de elementos;

4.4 RESULTADOS OBTIDOS

Com o propósito de se testar o sistema, selecionou-se um total de 11 (onze) imagens de diagramas de circuitos elétricos. Veja-os no apêndice E. A Tabela 4.6.1 apresenta as principais características do conjunto de dados. Pode-se notar através desta tabela e das imagens, que estas apresentam símbolos em várias situações, o que representa um bom conjunto de teste para o sistema.

O sistema foi testado com todo o conjunto de imagens apresentado no apêndice E. Observar que o objetivo do mesmo é separar os símbolos da imagem, por isto sua eficiência pode ser medida em ambas as formas: em termos de símbolos e em termos de linhas de conexão.

Para efeito de apresentação dos resultados obtidos, o circuito da Figura 4.2.5, cuja vetorização está na Figura 4.2.8, é repetido aqui na Figura 4.4.1 mostrando alguns dos vértices detectados e numerados.

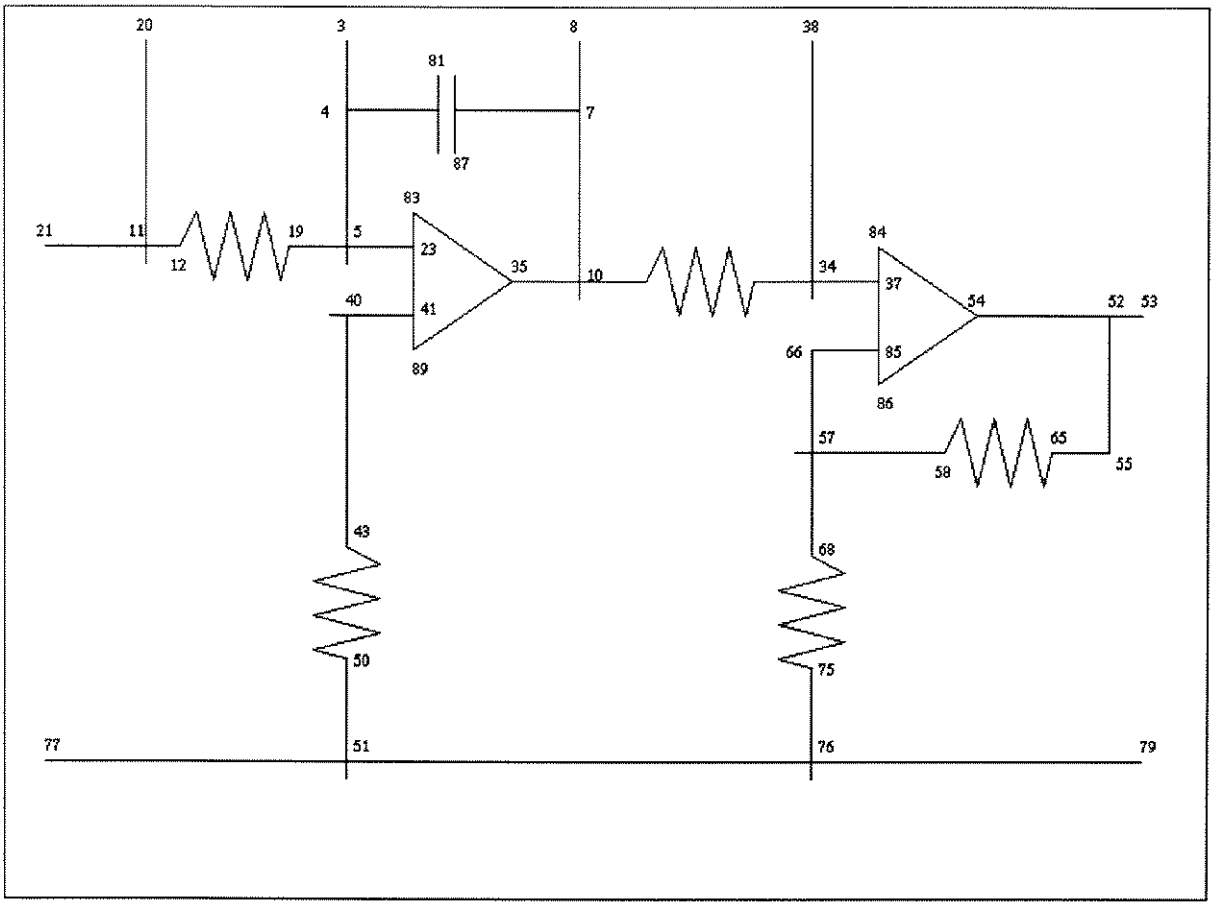


Figura 4.4-1 - Diagrama de circuito com os vértices identificados

Após a vetorização deste circuito, a lista dos elementos reconhecidos gerados pelo Sride, é mostrada a seguir:

*CIRCUITO DE TESTE

C0 2 9 1,000F

R1 12 19 1,000ohm

R2 25 33 1,000ohm

```
R3  43  50  1,000ohm
R4  58  65  1,000ohm
R5  68  75  1,000ohm
E0   23  41  (1,000 1,0000) 1.000
E1   37  85  (1,000 1,0000) 1,000
.END
```

Nesta lista, o nome do elemento é representado por uma letra, que determina o código do símbolo (**R** : resistor, **C** : capacitor, **V** : fonte de tensão, **D** : diodo, **A** : curto circuito, etc.) mais um número que representa a ordem na qual o elemento foi definido.

Utilizando-se o PSPICE verifica-se o circuito através da análise dos resultados produzidos por este analisador. Estes resultados são armazenados num arquivo de saída, o qual é analisado pelo sistema SRIDE. Encontrando erros, ele alerta o usuário de tais erros. Por último, é gerado uma lista DXF, que é o formato para ser utilizado em sistema CAD, o qual é chamado pelo programa SRIDE a fim de proporcionar ao usuário do sistema, mais um meio de verificação do sistema reconhecido, além de automatizar o armazenamento e manutenção do desenho em estudo. Esta lista DXF é gerada a partir da lista do circuito *(net list)* do PSPICE, proporcionando assim, absoluta certeza de que o circuito originado do processamento pelo sistema SRIDE está ou não correto.

No apêndice F é mostrado o resultado final completo para o circuito CIRC1, cujo diagrama é mostrado no apêndice E..

4.5 INTERAÇÃO MANUAL DOS AJUSTES FINAIS

O objetivo final do sistema é obter o completo reconhecimento do circuito elétrico o qual está em estudo. Por reconhecimento, entende-se a identificação de todos os símbolos da imagem e a determinação das coordenadas dos extremos de todas as linhas de interconexão.

O reconhecimento de um desenho é necessário por várias razões. Em primeiro lugar, o resultado da segmentação pode não ser totalmente correto. Erros podem ocorrer e quando isto acontece, eles precisam ser corrigidos. Segundo, os símbolos segmentados necessitam ser identificados com nomes próprios indicativos do que eles representam. Isto envolve a utilização de uma biblioteca de símbolos, programada com uma série de regras (características dos símbolos) de tal forma que a pesquisa de novos símbolos e a decisão de aceitá-los ou não, são baseadas nesta biblioteca.

Desde que o sistema SRIDE é designado para o reconhecimento de circuitos elétricos, é necessário designar a biblioteca de símbolos também para este domínio.

Os símbolos nesta pesquisa são representados em duas formas: **estrutural** (onde as características principais do símbolo são definidas num vetor de características) e **sintática** (onde as características do símbolo são definidas através de regras de formação). A forma estrutural é utilizada nos símbolos fechados, os quais possuem formas imprevisíveis num circuito. A forma sintática é utilizada na representação dos símbolos abertos, cuja variedade é conhecida no seu domínio. É importante observar que ambas as formas de representação, devem ser imunes às transformações geométricas dos símbolos que representam.

Outra grande vantagem destas representações é a possibilidade de usar-se o sistema para outros domínios de conhecimento, bastando para isto, mudar os critérios representativos dos símbolos para o novo domínio.

Com relação às linhas de interconexão, o maior problema para seu reconhecimento está situado no ponto onde elas se conectam. A decisão fica por conta das regras estabelecidas para o desenho em estudo. Nesta pesquisa, considerou-se como um ponto de conexão, aquele constituído de uma junção com um ponto em evidência sobreposto. O simples cruzamento de linhas significa que elas não se conectam entre si.

Nessa pesquisa, os símbolos abertos são reconhecidos em primeiro lugar, seguidos pelos símbolos fechados e por último as linhas de interconexão, as quais são "perseguidas" com base nas informações dos símbolos anteriormente reconhecidos.

4.6 RESULTADOS

Através da pesquisa bibliográfica apresentada anteriormente neste trabalho, constata-se a inexistência de estatísticas que mostrem a eficiência relativa dos diferentes métodos utilizados no reconhecimento de circuitos elétricos. Algumas justificativas para tal omissão podem ser consideradas [Alves 1992]:

- a simples apresentação da modelagem matemática pode não permitir avaliar a eficiência computacional do método;
- a aplicação de um modelo matemático para a extração de uma característica particular, muitas vezes, gasta mais tempo computando operações auxiliares do que computando aquelas definidas no próprio modelo;

- a implementação é feita em computadores de diferentes portes, tornando difícil a comparação, já que este dado é omitido na maioria das publicações;
- também, diferentes linguagens de programação são utilizadas nas implementação dos algoritmos. Sabe-se que a eficiência do código gerado é dependente não somente da implementação do compilador, mas também da seleção do nível de compilação feita pelo usuário.

Desta forma, comparações entre a forma de implementação dos algoritmos utilizados pelo SRIDE e outras publicadas na literatura, não podem ser feitas com rigor.

As Tabelas 4.6.1 e 4.6.2 sumarizam respectivamente os dados de 11 (onze) circuitos de testes e a eficiência do sistema para essas imagens e podem servir como referência inicial para futuros trabalhos. Pode-se observar pela leitura do capítulo 3 que o projeto do sistema SRIDE é modular, sendo seu *software* inteligível. Possui os atributos de modificabilidade e tem bom grau de portabilidade, na medida em que pode ser instalado em qualquer sistema operacional DOS ou UNIX (exceto os softwares comerciais agregados à versão DOS). A abrangência do sistema está no fato de que ele pode suportar vários circuitos. Para suportar um certo símbolo, entretanto, é necessária a existência de um conjunto adequado de componentes que possam ser utilizados para a extração de características de um símbolo do circuito elétrico, objetivando seu reconhecimento.

DESENHO	DIMENSÕES		Nº DE VETORES	Nº DE SÍMBOLOS		
	ALTURA	LARGURA	AFINAMENTO	ABERTO	FECHADO	TOTAL
circ1	441	568	590	20	0	20
circ2	458	832	522	12	0	12
circ3	483	540	474	10	0	10
circ4	692	872	803	16	0	16
circ5	580	336	285	4	2	6
circ6	754	400	708	12	6	18
circ7	527	744	515	6	2	8
circ8	390	584	359	9	0	9
circ9	722	731	3195	17	53	70
circ10	495	880	685	17	0	17
circ11	1084	1981	6234	39	28	67

Tabela 4.6.1 - Características estruturais dos desenhos utilizados para testes veja-os no apêndice E

DESENHO	Nº DE VETORES		Símbolos reconhecidos corretamente		TEMPO DE PROCESSAMENTO (segundos, milesegundos)		
	SRIDE	redução %	Nº	%	Pre Processamento	Valorização	Reconhecimento
circ1	200	66	20	100	2 , 550	8 , 796	70 , 523
circ2	123	76	12	100	3 , 25	10 , 780	41 , 5
circ3	106	80	10	100	4 , 274	11 , 931	37 , 107
circ4	159	80	15	94	3 , 974	14 , 606	46 , 10
circ5	51	82	6	100	1 , 470	6 , 395	19 , 31
circ6	168	76	18	100	2 , 646	9 , 928	56 , 371
circ7	92	82	8	100	2 , 809	11 , 526	27 , 151
circ8	89	75	9	100	1 , 889	7 , 872	30 , 12
circ9	743	76	70	100	8 , 210	16 , 916	90 , 143
circ10	181	73	16	94	3 , 401	10 , 893	56 , 10
circ11	1013	79	67	100	10,332	21,234	120,123

Tabela 4.6.2 - Resultados obtidos pelo sistema SRIDE para alguns circuitos de teste

4.7 DISCUSSÃO

Como pode se observar na Tabela 4.6.2, o algoritmo de segmentação foi um sucesso. A maioria das linhas de conexão foram reconhecidas na imagem do diagrama. Isto é resultado da grande eficiência no método de vetorização proposto e utilizado nesta pesquisa, minimizando-se desta forma as tarefas futuras do algoritmo de segmentação.

A utilização de pontos característicos é talvez o ponto mais forte na metodologia empregada para a vetorização de diagramas de circuitos elétricos, pois como visto nas tabelas, minimiza espaço e tempo de computação. Embora não tenhamos resultados de tempos de processamento de outros trabalhos, podemos notar pelos nossos resultados que o tempo dispendido para o sistema obter os vetores que o compõe é diretamente proporcional à quantidade de linhas presentes no diagrama, assim sendo podemos concluir que os resultados mostram que nosso método pode detectar os vetores de uma imagem binária com precisão satisfatória e com razoável economia de tempo e memória.

Além da obtenção dos vetores, há a tarefa de reconhecimento dos elementos que constituem este diagrama elétrico, envolvendo aí as características próprias que regem o comportamento de tais diagramas. Como a maior responsabilidade do sistema está na vetorização, a fase de classificação e correspondente reconhecimento pôde ser realizada com a utilização de métodos mais simples, sem prejuízo da precisão dos resultados obtidos.

Como nosso objetivo maior foi gerar uma descrição completa do desenho de circuito a partir de sua imagem binária, utilizando para isso um eficiente método de vetorização, pelos resultados obtidos julgamos esta missão cumprida, visto que todos os símbolos foram identificados, suas interconexões também o foram, e isto gerou para cada exemplo apresentado sua correspondente *net list*, a qual foi prontamente utilizada no programa

supervisor de topologia de circuitos elétricos (Pspice). Além disto, criamos um banco de símbolos que pode, para um determinado domínio, estar sempre pronto para receber novos símbolos (tal como numa biblioteca de símbolos), o que pode propiciar a troca de domínio pela simples troca dos conhecimentos envolvidos nas características destes símbolos.

Por último, pelos testes realizados com o sistema notamos uma notável ajuda ao usuário proporcionada pelo sistema, pois agora ele está trabalhando para o reconhecimento de circuitos, achar erros, e transformar desenhos em papel para formato eletrônico. Tudo isto num curto espaço de tempo e com razoável precisão. Apenas por experiência, experimentamos fazer esta tarefa manualmente, ou seja, desenhar um diagrama e vetorizá-lo através ou de uma mesa digitalizadora ou um programa que contém biblioteca de símbolos. O resultado é impressionantemente mais longo, sem contar que neste caso não há o reconhecimento dos elementos que participam do circuito.

CAPÍTULO 5

5. CONCLUSÕES

5.1 SUMÁRIO E CONTRIBUIÇÕES

Símbolos conectados através de segmentos de linhas fazem parte da grande maioria dos desenhos técnicos empregados em engenharia ou desenhos utilizados no planejamento e administração. Por isso acreditamos ser de grande importância o sistema aqui desenvolvido com o objetivo de lidar com estes desenhos.

Assim como nos trabalhos mais antigos que manuseiam desenhos de uma forma geral, também empregamos aqui no desenvolvimento do sistema SRIDE métodos de baixo nível, tais como separador de caracteres dos desenhos, reconhecimento de linhas tracejadas, etc. No entanto, desenvolvemos a tarefa de reconhecimento através de uma segmentação independente de domínio seguida por uma rotulação e verificação de erros dependentes de domínio. Esta conotação, os algoritmos de vetorização e obtenção de laços, o algoritmo de reconhecimento, suas implementações e resultados formam o elenco de contribuições deste trabalho. O algoritmo de classificação de símbolos segmentados empregado aqui, representa apenas uma das formas de como os resultados de uma segmentação podem ser processados para um reconhecimento de alto nível de um desenho. Outras metodologias, baseadas inteiramente em diferentes princípios, tais como redes neurais ou sistemas baseados em regras,

poderiam ser utilizados. O ponto importante aqui é que nossa segmentação independente de domínio, proporciona uma representação intermediária para futuros processamentos que tenham diferentes objetivos e técnicas.

Existem muitas aplicações para o sistema desenvolvido nesta pesquisa, algumas imediatas, outras a longo prazo. Em muitas situações há um grande número de desenhos baseados em papel e este sistema pode na maioria das vezes, convertê-los de forma eficiente numa forma eletrônica, própria para desenhos, tal como o CAD. A representação CAD pode ser muito útil em várias formas. Pode ser usada para fazer um inventário de partes ou símbolos em desenhos complexos por exemplo, além de manusear de forma eficiente os desenhos assim armazenados. Associados a estas vantagens, temos ainda o fato de que o sistema SRIDE faz um reconhecimento automático dos elementos simbólicos, num determinado conhecimento específico, no caso o de circuitos elétricos, agilizando de modo muito significativo o trabalho desenvolvido pelo seu usuário.

A metodologia empregada em nossa pesquisa difere de outras citadas nos artigos da bibliografia, em duas significantes maneiras. Os esquemas empregados nos outros métodos são designados para domínio específico. Nosso trabalho pode ser aplicado para outros domínios que empreguem símbolos como representantes, isto é, desenhos constituídos de símbolos conectados por segmentos de retas. Por isto, nossa abordagem é mais abrangente que as outras, até pelo menos a vetorização dos elementos. A segunda diferença está na forma de como usamos o conhecimento de alto nível no processo de reconhecimento. Na maioria dos métodos, este conhecimento está sempre estreitamente ligado a todos os níveis do sistema. Em nossa pesquisa, ele é modular e estritamente restrito à unidade de reconhecimento. Por isto, pode-se esperar que com a mudança apenas do banco de dados que contém o conhecimento do domínio, o sistema atue em outros tipos de desenhos.

O sistema SRIDE ora apresentado, incorpora todas as funções propostas no objetivo inicial desta pesquisa. Foi desenvolvido na linguagem C baseado parte no sistema operacional Unix e parte no sistema operacional DOS. A maioria de seus módulos são independentes, mas no final, o sistema SRIDE os gerencia através de um módulo principal contendo todo o menu de tarefas.

Como resultado deste trabalho tem-se um protótipo de um sistema reconhecedor de símbolos de um diagrama de circuitos elétricos a partir de sua captura via um digitalizador, o que certamente ajudará em muito as tarefas de manuseio com tais documentos. Porém, muitas atividades podem ser melhoradas neste protótipo: refinar várias operações, incorporar novas funções e introduzir novas atividades às funções já existentes, são algumas delas.

5.2 LIMITAÇÕES E RECOMENDAÇÕES

Concentramos nossos esforços no desenvolvimento e implementação dos algoritmos referentes à vetorização, segmentação e reconhecimento dos símbolos no desenho. Entretanto, os algoritmos de baixo nível também tiveram que ser empregados e muitas vezes desenvolvidos e implementados, principalmente aqueles referentes à separação dos caracteres dos gráficos, obtenção de linhas e obtenção dos pontos característicos do desenho. Todas as rotinas foram extensamente testadas e embora funcionem bem para a maioria dos desenhos apresentados, o sistema apresenta suas limitações, tendo em vista o caráter extenso do trabalho aqui desenvolvido.

Nos preocupamos apenas em símbolos constituídos por segmentos de retas, o que nem sempre acontece num diagrama de circuito. Alguns circuitos apresentam símbolos

constituídos de linhas curvas. Métodos para a segmentação deste tipo de símbolo deveriam ser estudados e implementados.

O desenho final reconhecido é apresentado ao usuário num CAD, através do qual este desenho pode ser manuseado, alterado e armazenado. Atualmente o sistema SRIDE envia as informações pertinentes ao reconhecimento ao CAD, mas ainda não recebe de volta estas alterações realizadas pelo usuário. Isto poderia ser uma inovação importante no trabalho do usuário.

Símbolos muito pequenos, principalmente o símbolo terra, devido à parametrização de alguns fatores no sistema, são parcialmente perdidos durante a fase de segmentação. Maior utilização do sistema poderia tornar mais eficiente a estimação destes parâmetros e o emprego de melhor algoritmo de segmentação para estes pequenos objetos, poderiam produzir um conjunto de descritores de linhas que removam as distorções enquanto preservam os detalhes dos pequenos objetos.

As regras para obtenção dos símbolos abertos foram implementadas numa linguagem determinística. Por enquanto não apresentaram nenhum problema durante esta fase de treinamento e desenvolvimento do sistema. No entanto, um sistema constituído de um conjunto de regras implementadas numa forma heurística poderia tornar o sistema mais flexível para o usuário.

O sistema SRIDE utiliza-se do PSPICE para validar o circuito segmentado e reconhecido. Pode utilizar este programa inclusive para simular o circuito eletricamente. No entanto, poderíamos pensar na implementação de regras específicas para o domínio do reconhecimento em teste, que façam esta verificação antes de liberarmos a saída do SRIDE para outros sistemas.

APÊNDICE A

Rotulação da Imagem

No item 3.2.7 apresentamos a definição e aplicação do algoritmo de ROTULAÇÃO o qual é mostrado a seguir.

ALGORITMO *rotulação*

Passo 1: Obter os componentes conectados equivalentes

{ Inicializar a lista de componentes conectados }

LC $\leftarrow \lambda$ (lista de componentes conectados vazia)

{ Inicializar a lista de equivalentes }

EQ $\leftarrow \lambda$ (lista de equivalentes vazia)

{ Analisar cada uma das NL linhas da imagem }

Para L = 1 **até** NL **faça**

{ busca das corridas da linha L }

J $\leftarrow$ 0

(NP = número de colunas da matriz da imagem)

Para I = 1 **até** NP **faça**

Se IMAGEM (L, I) = 1

então

Se I = 1

então

J $\leftarrow$ J + 1

INIC (J) $\leftarrow$ 1

```
                senão
                  Se IMAGEM ( L, I - 1 ) = 0
                    então
                      J ← J + 1
                      INIC ( J ) ← I
                  fim-se
              fim-se
            senão
              Se I > 1 e IMAGEM ( L, I - 1 ) = 1
                então
                  FIM ( J ) ← I - 1
              fim-se
          fim-se
        fim-para
        Se IMAGEM ( L, NP ) = 1
          então
            FIM ( J ) ← NP
        fim-se
        NS ← J
        { Analisar cada corrida da linha L }
        Para J = 1 até NS faça
          { Inicializar a lista auxiliar de componentes conectados }
          LA ← λ
          { Atualizar a lista auxiliar de pixels rotulados adjacentes }
          I ← INIC ( J ) - 1
          Se I < 0
            então
              I ← 0
          fim-se
          F ← FIM ( J ) + 1
```

```
            Se F > NP
              então
                F ← NP
            fim-se
            Para Q = I até F faça
            LA ← { IMAGEM ( N, Q ) / IMAGEM ( N, Q ) > 1 e
                        N = L - 1  e  L > 1  e  IMAGEM ( N, Q ) ∉ LA }
            fim-para

           { Analisar cada um dos pixels da corrida J linha L }
           { Atribuir novo rótulo ao pixel LP }
           Se LA = λ
              então
                 M ← novo rótulo
                 { Atualizar a lista de componentes conectados }
                 LC ← { M }   { incluiu M à LC }
              senão
                 M ← último { IMAGEM ( N, Q ) / IMAGEM ( N, Q ) ∈ LA }
                 { Atualizar a lista de equivalentes }
                  Para X = IMAGEM ( N, Q ) / X ∈ LA e X ≠ M faça
                    EQ ← { ( X ) }
                  fim-para
                  LC ( M ) ← EQ    { incluiu a lista EQ como sublista do nó M em LC }
            fim-se
            Para P = INIC ( J ) até FIM ( J ) faça
              IMAGEM ( L, P ) ← M
            fim-para
        fim-para
    fim-para
```

Passo 2: Obter as classes equivalentes.

Para X ∈ LC / ∃LC (X) **faça**
  M ← menor LC (X)
fim-para
Para L = 1 **até** NL **faça**
  Para P = 1 **até** NP **faça**
    X ← IMAGEM (L, P)
    Se X ∉ LC
     então
       X ←M / X ∈ LC (M)
       IMAGEM (L, P) ← X
     fim-se
  fim-para
fim-para
FIM ALGORITMO

APÊNDICE B

Obtenção dos Laços

Afim de se entender o desenvolvimento do algoritmo de obtenção dos laços apresentado no item 3.5.5, é apresentado aqui um exemplo para a obtenção dos laços da imagem da figura 3.5.8, onde a seguinte convenção é utilizada:

n: a, b

n: corresponde ao vértice em estudo

a, b: corresponde à sublista de vértices do vértice n

a^f : corresponde ao fim da pesquisa para o vértice a.

Inicia-se a pesquisa pelo primeiro vértice disponível na lista de laços **L**, no caso o vértice **a**, o qual tem como sublista os vértices **b** e **e** (veja a tabela 3.5.4), repetida aqui:

nó base	sublista
a	b, e
c	d, l
f	g, n
h	k, b, d
i	e, g
j	g, e
m	k, l

Logo:

inicio: a: b, e => inicio da pesquisa
b: h, a^f => ligação entre os vértices **a** e **b**
h: b^f, d, k
d: h^f, c
c: d^f, l
l: n, c^f, m
n: f, l^f
f: g, n^f
g: f^f, i, j
i: g^f, e
e: a, j, i^f => **laço 6 (figura 3.4.11)**
j: e^f, g^f
j: e, g^f
e: a, j^f, i^f => **laço 5**
m: l^f, k
k: h^f, m^f
k: h^f, m
m: l, k^f
l: n, c, m^f
n: f, l^f
f: g, n^f
g: f^f, i, j
i: g^f, e
e: a, j, i^f => **laço 4**
j: e^f, g^f
j: e, g^f
e: a, j^f, i => **laço 3**
i: g^f, e^f
c: d, l^f
d: h^f, c^f

e: a, j, i
a: => ligação entre os vértices **a** e **e**;
Neste caso, estamos iniciando a pesquisa do laço partindo do vértice **a** em direção ao vértice **e**, o qual evidentemente é o sentido inverso do laço obtido no passo anterior (de **a** para **b**). Esta é uma forma de se economizar tempo: se a sublista do nó em estudo possui apenas 2 nós (a grande maioria) , fazer a pesquisa para apenas um dos nós da sublista. O outro corresponde ao sentido inverso do laço e portanto é desnecessário. Quando a sublista tem mais de 2 nós, como o vértice **h** da figura 3.4.9, então esta regra não se aplica, devendo-se então pesquisar toda a sublista.

fim a ====> fim da pesquisa do primeiro nó

APÊNDICE C

Formato Tiff

O formato eletrônico para armazenamento de imagens TIFF (Tag Image File Format) foi desenvolvido e suportado por Aldus Developer's Desk e Microsoft Windows Marketing Group, ambas nos EUA. Tem como plataformas todos os ambientes computacionais disponíveis atualmente e tem como principais vantagens: pode ser aplicado em grande variedade de imagens; é independente de máquina, sistema operacional e hardware gráfico. Pode manusear imagens coloridas, preto e branco, e em níveis de cinza, e além disto pode suportar vários tipos de compressão. Por consequência TIFF é uma das melhores escolhas para o armazenamento e manuseio de imagens em geral. Tem também algumas desvantagens, e a principal delas é que o desenvolvimento de programas para o manuseio do formato TIFF é bastante trabalhoso, tendo em vista que o programador deve atender todos os requisitos possíveis para o tratamento das imagens neste formato, de forma a prover perfeita troca de imagens entre os diversos sistemas.

A estrutura deste formato pode ser resumida da seguinte forma:

Um arquivo TIFF pode conter várias imagens. Cada imagem é armazenada em 3 níveis de hierarquia: cabeçalho, tags, dados; O nível mais alto, o cabeçalho, contém 3 entidades: um código indicando a ordem dos bytes (menos significativo para o mais

significativo ou vice versa); um código indicando o arquivo como TIFF; um ponteiro para Diretório de Arquivos de Imagens (IFD).

O IFD contém uma série de ponteiros, os quais possuem os endereços onde os vários campos de dados estão armazenados no arquivo. Desta forma, qualquer imagem pode ter partes armazenadas em qualquer parte do arquivo, cabendo ao leitor do formato "seguir" os ponteiros correspondentes para obter as informações necessárias à sua localização. A última entrada do IFD deve ser o endereço do próximo IFD que corresponde a uma nova imagem. Geralmente, utiliza-se uma imagem em cada arquivo, e então este valor se iguala a zero.

Cada ponteiro do IFD possuem uma marca (Tag), que é um código que identifica o tipo do campo que se está tratando. Por exemplo, o código 256 corresponde a largura da imagem. O usuário tem à disposição uma série de tags para o manuseio das imagens. A lista completa dos tags com seus respectivos tamanhos e significados, pode ser encontrada no manual técnico deste formato, fornecido pela Aldus.

Alguns destes códigos (os mais comuns) são listados a seguir:

258 - bits per sample - define imagem binária ou em nível de cinzas;

259 - compression - define o tipo de compressão utilizado;

257 - image length - define a altura da imagem;

256 - image width - define a largura da imagem;

273 - stripoffset - aponta para o início da imagem;

etc.

APÊNDICE D

Formato DXF

O formato DXF (Drawing Interchange Format) foi desenvolvido pela Autodesk Inc, EUA, e tem por objetivo manusear imagens designadas para o Auto CAD.Também pode ser utilizada na maioria dos ambientes computacionais disponíveis e apresenta como vantagem o fato de que como o AutoCAD da Autodesk é um programa muito popular para desenvolvimento gráfico, este formato tornou-se também muito conhecido na área de engenharia pois é muito prático no tratamento de vetores e consequentemente desenhos de engenharia em geral.

O arquivo DXF onde a imagem é armazenada geralmente é escrito em ASCII e portanto pode ser utilizado por qualquer editor de texto. Existe também a versão binária deste arquivo, o qual evidentemente é bem mais rápido para ser processado.

DXF é considerado mais uma linguagem ou um meta arquivo gráfico do que propriamente um formato eletrônico. A exata posição ou ordem dos dados no arquivo não são importantes. Tal como numa linguagem, o contexto em que os termos aparecem é que é importante; os códigos com mesmo valor podem ter diferentes significados, dependendo do contexto em que aparecem no arquivo. Por exemplo, o código 10 descrevendo uma linha tem sentido completamente diferente quando descreve um círculo, e assim por diante.

Cada arquivo DXF é constituido de pares chamados grupos. Cada grupo tem seu código e seu nome; é nos grupos onde os dados são armazenados.

O arquivo DXF é então organizado da seguinte forma:

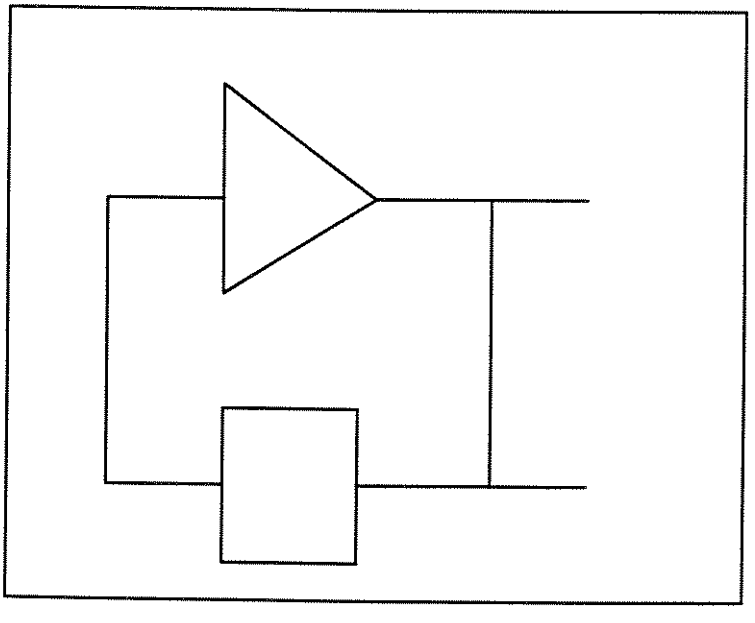
a) Cabeçalho: contém informações gerais, nada relacionadas com os dados propriamente ditos, e sim informações sobre autores, dimensões, textos em geral. Pode ser omitida;

b) Tabelas: define as constantes universais utilizadas no desenho, tais como aquelas relacionadas com layers, ângulos, distâncias, sistemas de coordenadas e estilos usados na representação do desenho. Também pode ser omitida;

c) Blocos: define os grupos de entidades pelos nomes e dados;

d) Entidades: define as entidades do desenho propriamente dito, ou seja, possue os elementos de desenho e suas coordenadas; as entidades disponíveis mais populares para representação são: ponto, linha, circulo, arco de circulo, sólido, polyline, vértices, 3D, etc.

Como exemplo, seja o seguinte desenho CAD e a seguir um trecho (apresentado em colunas) do correspondente arquivo DXF:



```
0
SECTION
 2
HEADER
 9
$LTSCALE
 40
   10
 9
$LIMMIN
 10
   0.0
 20
   0.0
 9
$LIMMAX
 10
  1117.6
 20
   863.6
 9
$EXTMIN
 10
   0.0
 20
   0.0
 9
$EXTMAX
 10
  1117.6
 20
   863.6
 9
$CLAYER
  8
1
  9
$CECOLOR
 62
   7
 9
$CELTYPE
  6
CONTINUOUS
 0
ENDSEC
 0
SECTION
 2
TABLES
 0
TABLE
 2
LTYPE
 70
   10
 0
LTYPE
 2
CONTINUOUS
 70
   64
 3
Solid line
 72
   65
 73
    0
 40
```

```
DASHED
 70
    0
 3
__ . __ . __ . __ . __ . __
 72
   65
 73
    6
 40
1.25
 49
0.5
 49
-0.25
 49
0.0
 49
-0.25
 49
0.0
 49
-0.25
  0
LTYPE
  2
    9
 70
    0
  3
close
 72
   65
 73
    2
 40
0.1
 49
0.05
 49
-0.05
  0
ENDTAB
0
TABLE
  2
LAYER
 70
  256
  0
LAYER
  2
1
 70
    0
 6
CONTINUOUS
 62
1
  0
LAYER
  2
5
 70
    0
 6
CONTINUOUS
 62
5
  0
ENDTAB
```

```
TABLE
  2
STYLE
 70
    8
  0
STYLE
  2
TXT
 70
0
 40
0.0
 41
1.0
 50
0.0
 71
0
 42
1.0
  3
TXT
  4
  0
STYLE
  2
ITALIC
 70
0
 40
0.0
 41
1.0
 50
0.0
 71
0
 42
1.0
  3
ITALIC
  4
  0
STYLE
  2
SIMPLEX
  4
  0
STYLE
  2
SCRIPTC
 70
0
 40
0.0
 41
1.0
 50
0.0
 71
0
 42
1.0
  3
SCRIPTC
  4

  0
STYLE
  2
GOTHICE
```

```
0
 40
0.0
 41
1.0
 50
0.0
 71
0
 42
1.0
 3
GOTHICE
 4
 0
ENDTAB
 0
TABLE
 2
VIEW
 70
   0
 0
ENDTAB
 0
ENDSEC
 0
SECTION
 2
ENTITIES
 0
LINE
 8
5
 6
CONTINUOUS
 62
   1
 10
1.5865720251E+02
 20
7.0350704961E+02
 11
2.8698288100E+02
 21
7.0350704961E+02
 0
LINE
 8
5
 6
CONTINUOUS
 62
   1
 10
2.8931607516E+02
 20
7.6889712794E+02
 11
2.8931607516E+02
 21
6.4037180157E+02
 0
LINE
 8
5
 6
CONTINUOUS
 62
   1
 10
 70
 20
6.4037180157E+02
 11
4.1530855950E+02
 21
7.0350704961E+02
 0
LINE
 8
5
 6
CONTINUOUS
 62
   1
 10
4.1764175365E+02
 20
4.1263394256E+02
 11
2.8698288100E+02
 21
4.1263394256E+02
 0
LINE
 8
5
 6
CONTINUOUS
 62
   1
 10
2.8698288100E+02
 20
4.1263394256E+02
 11
2.8698288100E+02
 21
5.4115926893E+02
 0
LINE
 8
5
 6
CONTINUOUS
 62
   1
 10
1.6099039666E+02
 20
4.8027885117E+02
 11
2.8698288100E+02
 21
4.8027885117E+02
 0
LINE
 8
5
 6
CONTINUOUS
 62
   1
 10
5.4363423800E+02
 20
7.0350704961E+02
 11
5.4363423800E+02
 21
4.8027885117E+02
0
ENDSEC
0
EOF
```

APÊNDICE E

Vetorização

A seguir são mostrados alguns dos diagramas de circuitos utilizados para testes neste projeto. Todos foram capturados com 300 dpi num *scanner* HP preto e branco, e em seguida as suas imagens tiveram seus caracteres retirados. Por último são mostrados os diagramas vetorizados pelo sistema SRIDE, utilizando-se o processo descrito no item 3.2.

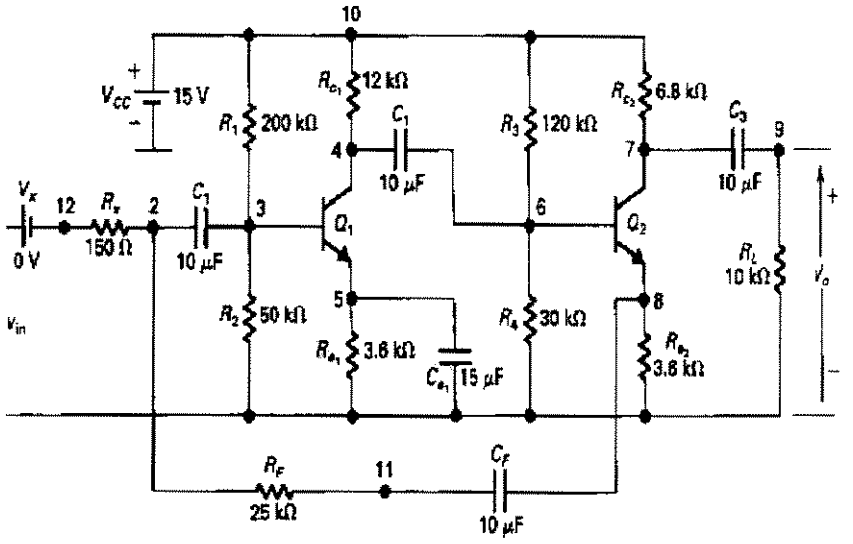


Figure 8.18 Two-stage BJT amplifier with shunt-series feedback

Figura E. 1

CIRC1 - Imagem original Tiff

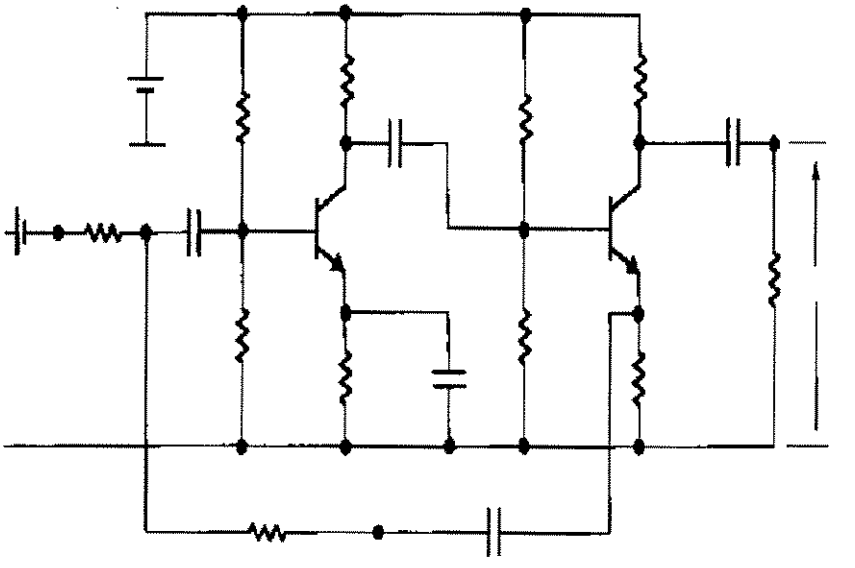


Figura E. 2

CIRC1 - Circuito sem os caracteres - tiff

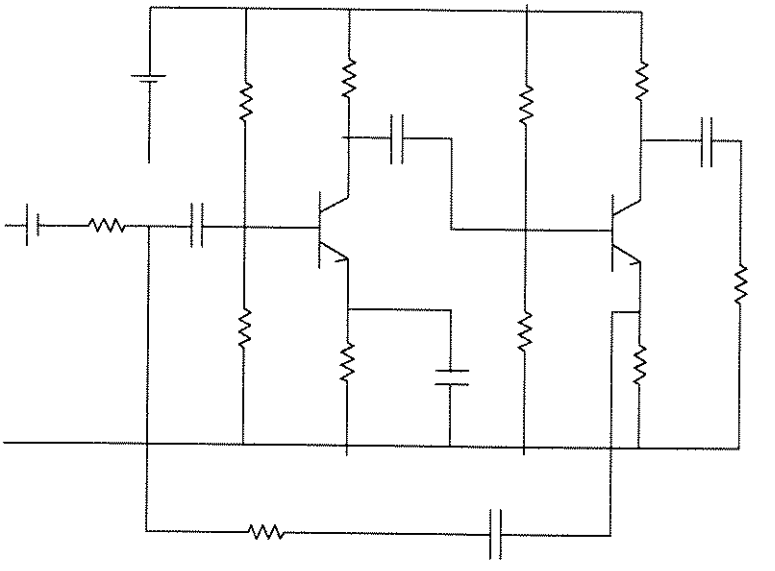


Figura E. 3

Imagem vetorizada pelo Sride

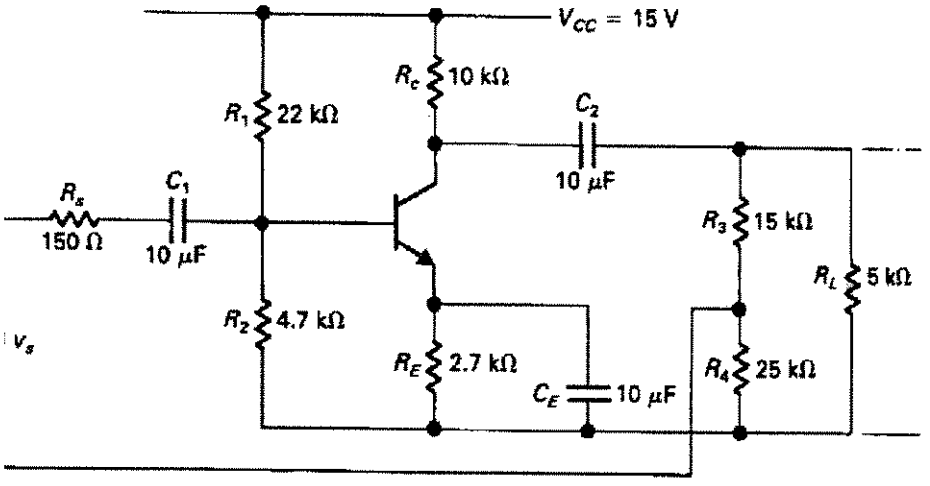


Figura E. 4

CIR2 - Imagem original Tiff

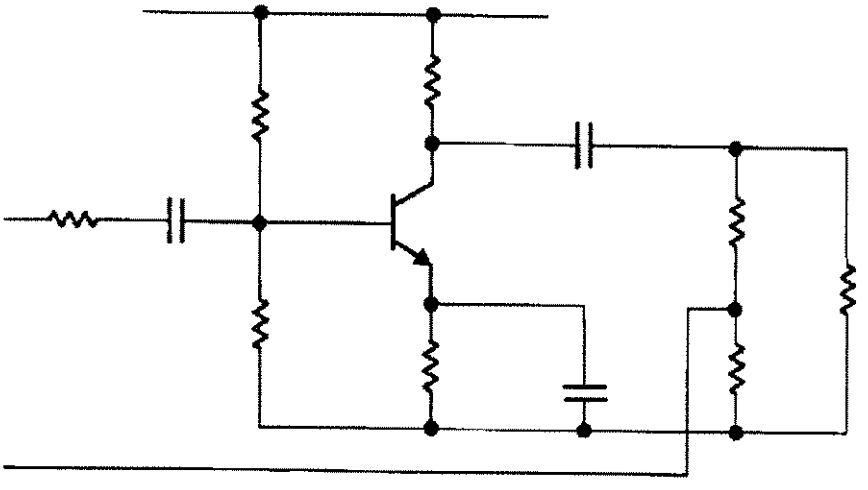


Figura E. 5

CIRC2 - Circuito sem os caracteres - Tiff

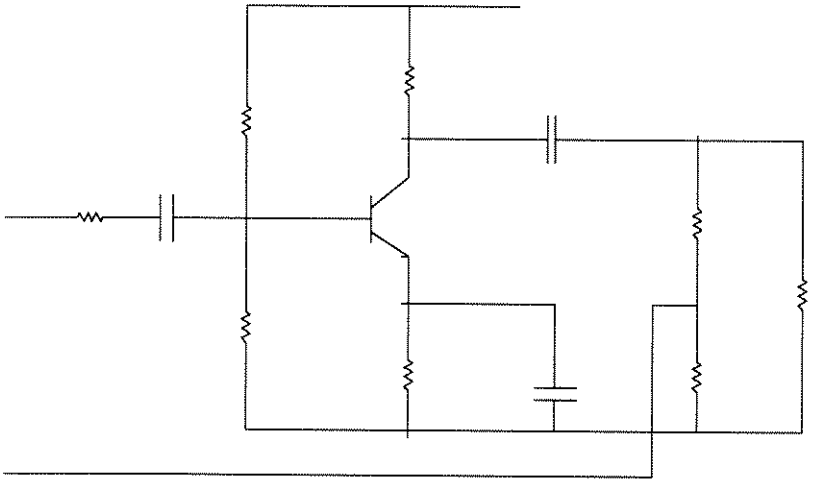


Figura E. 6

Imagem vetorizada pelo Sride

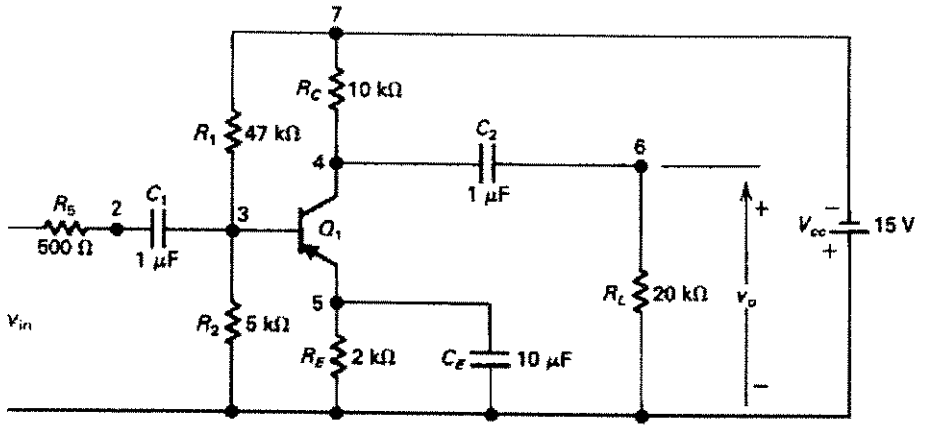


Figure 8.7 Bipolar transistor amplifier circuit

Figura E. 7

CIRC3 - Imagem original Tiff

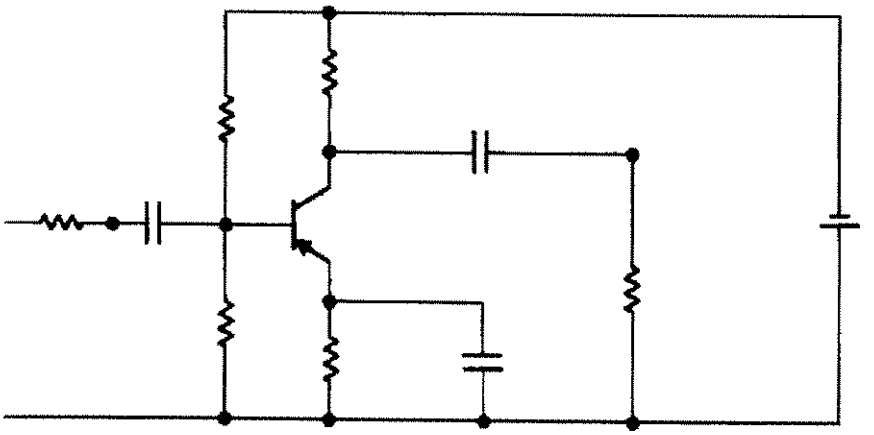


Figura E. 8

CIRC3 - Circuito sem os caracteres

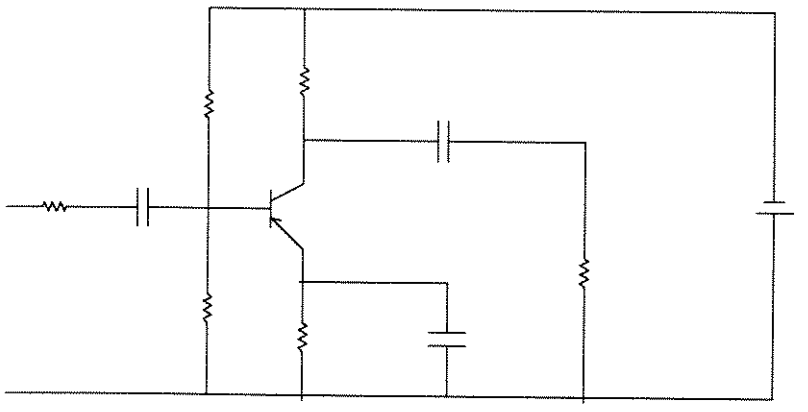


Figura E. 9

Imagem vetorizada pelo Sride

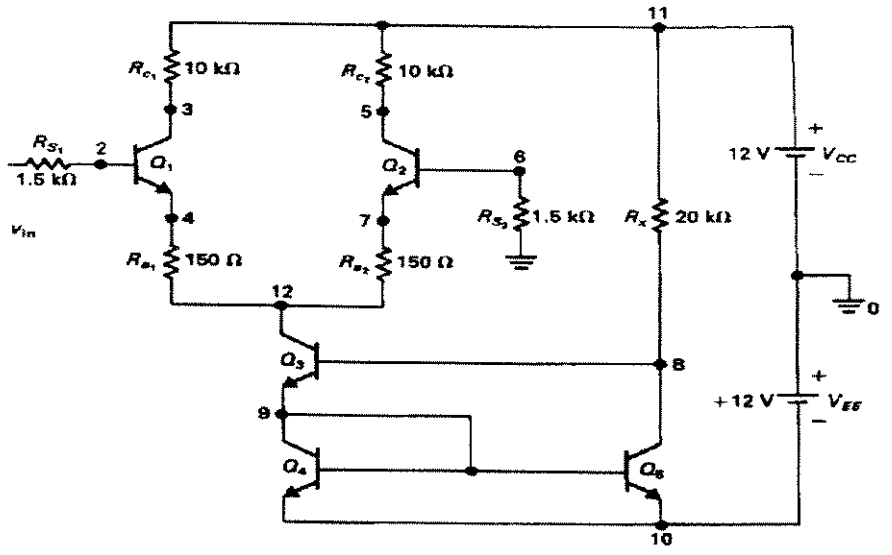


Figura E. 10

CIRC4 - Imagem original Tiff

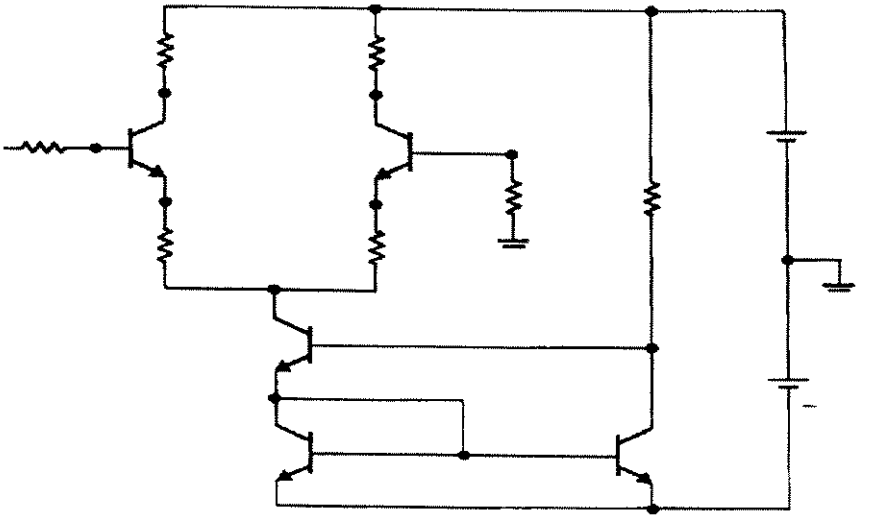


Figura E. 11

CIRC4 -Circuito sem os ca-racteres

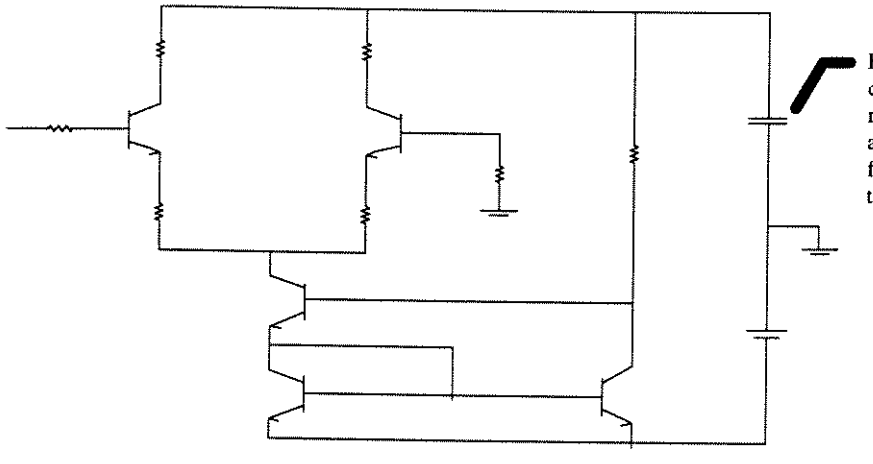


Figura E. 12 - Imagem vetorizada pelo Sride

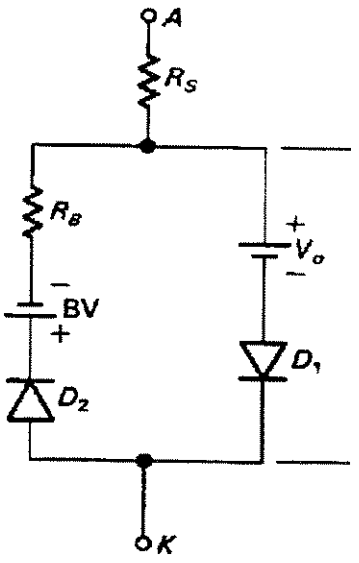


Figura E. 13

CIRC5 - Imagem original Tiff

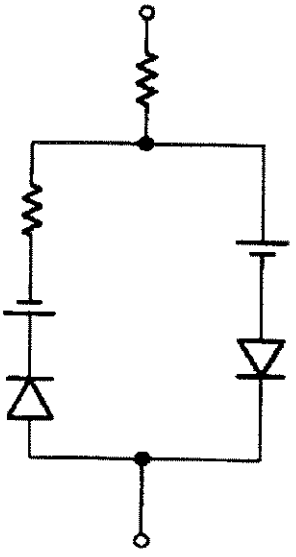


Figura E. 14

CIRC5 Circuito sem os caracteres

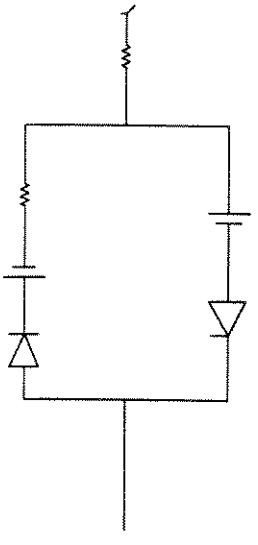


Figura E. 15

Imagem vetorizada pelo Sride

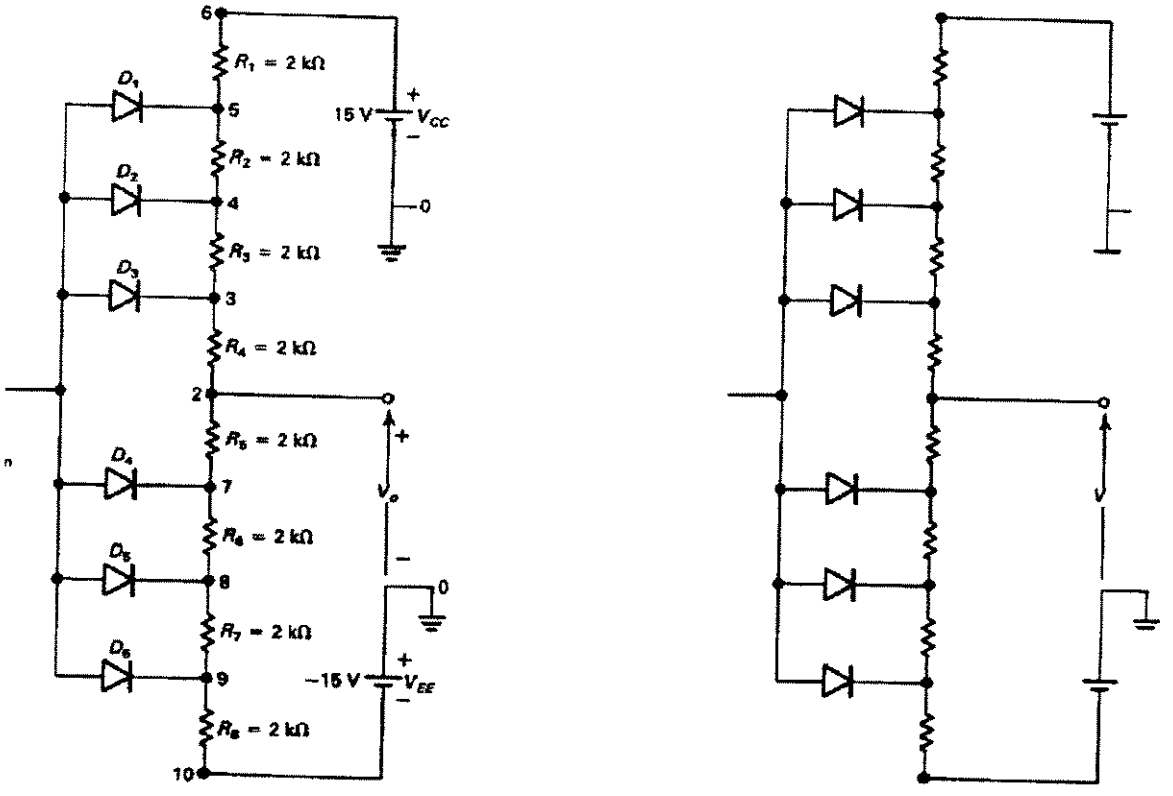


Figura E. 16 - CIRC6 - Imagem original e imagem sem caracteres - ambas Tiff

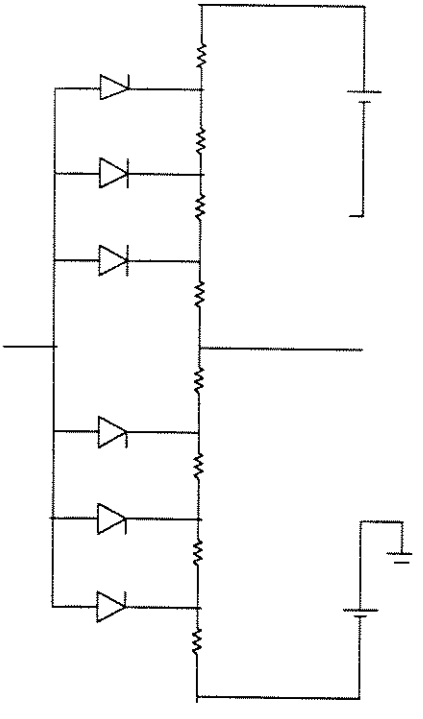


Figura E. 17

Imagem vetorizada pelo Sride

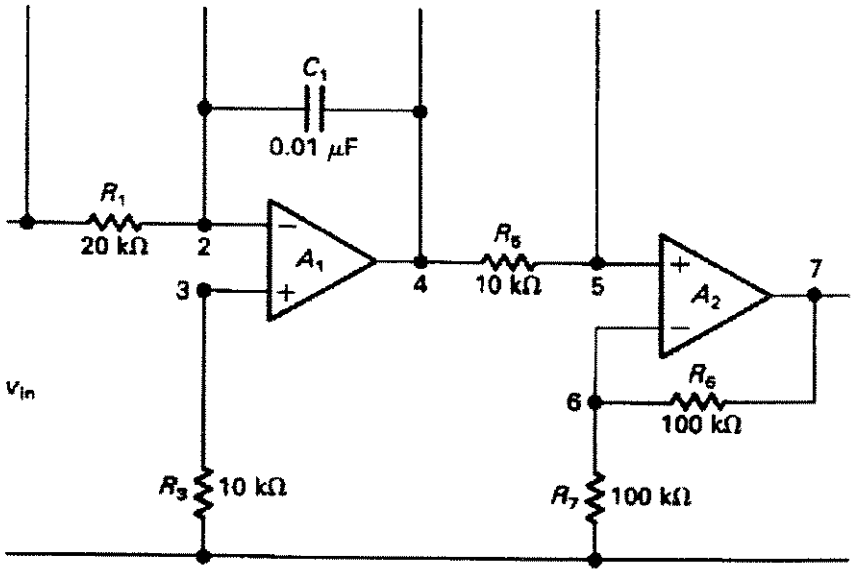


Figura E. 18

CIRC 7 - Imagem original Tiff

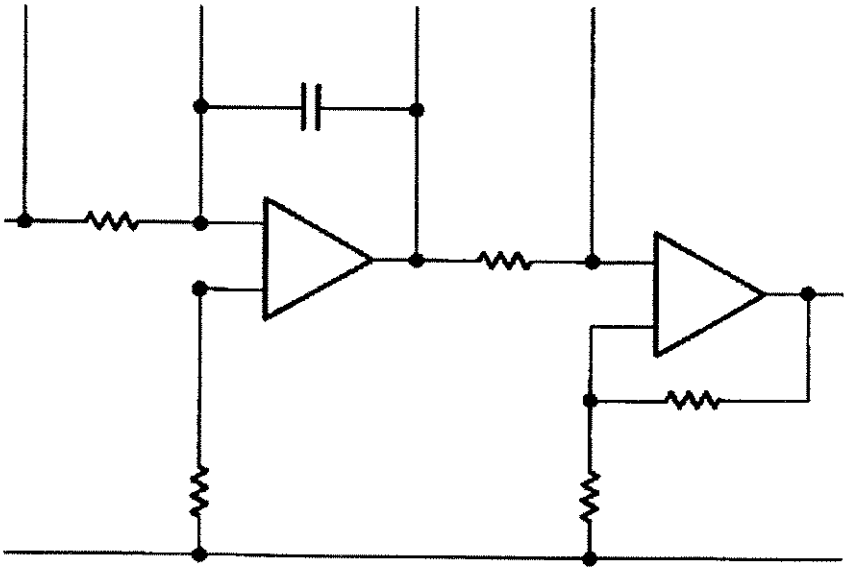


Figura E. 19

CIRC 7 - Circuito sem os caracteres

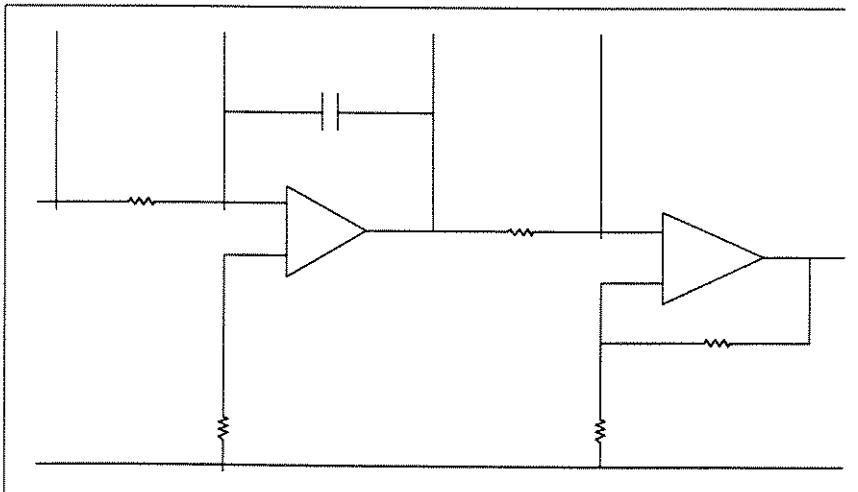


Figura E. 20

Imagem vetorizada pelo Sride

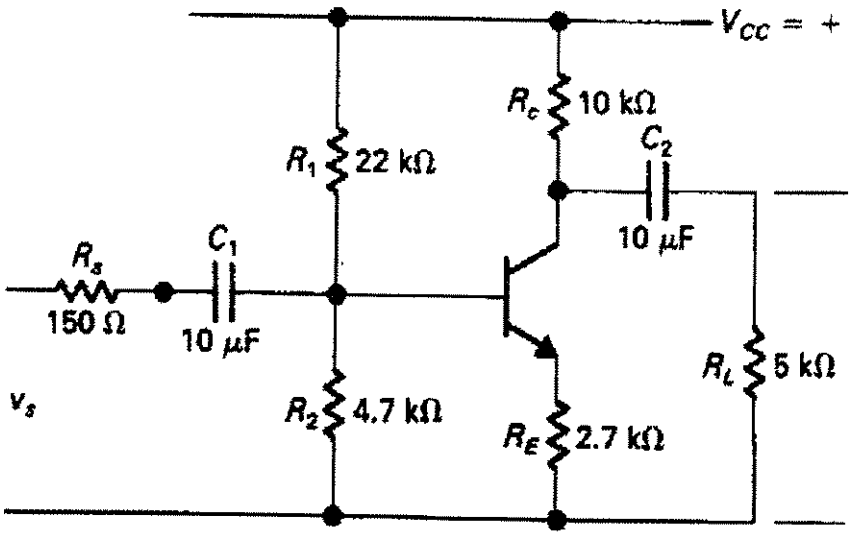


Figura E. 21

CIRC 8 - Imagem original Tiff

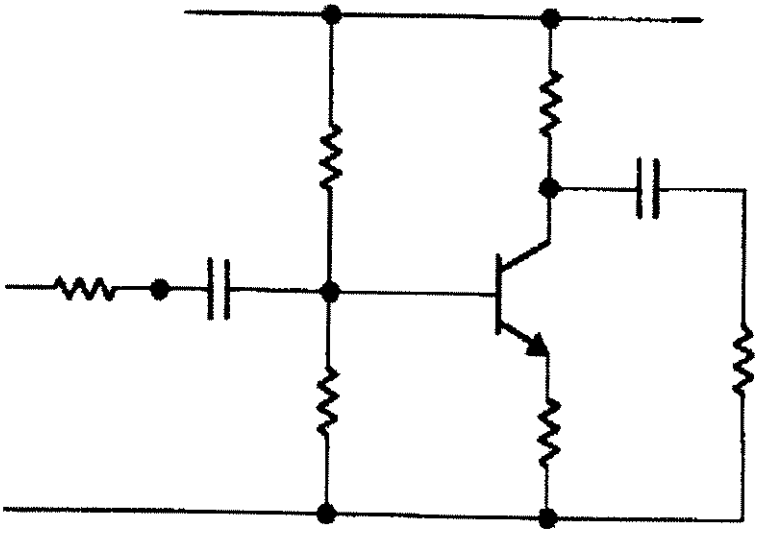


Figura E. 22

CIRC 8 - Circuito sem os caracteres

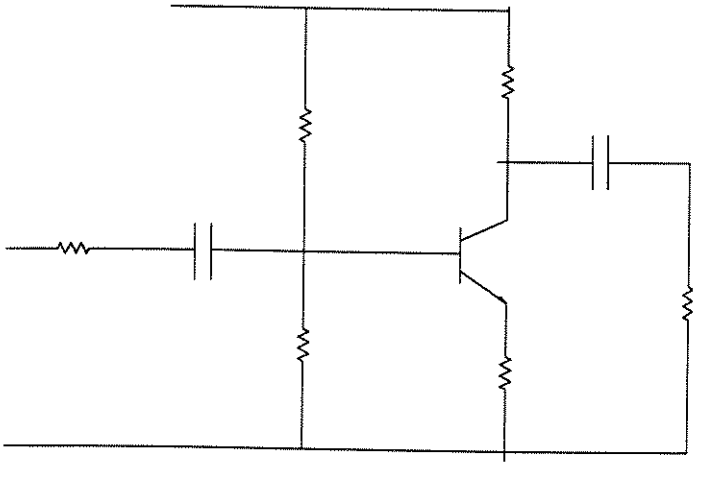


Figura E. 23 - Imagem vetorizada pelo Sride

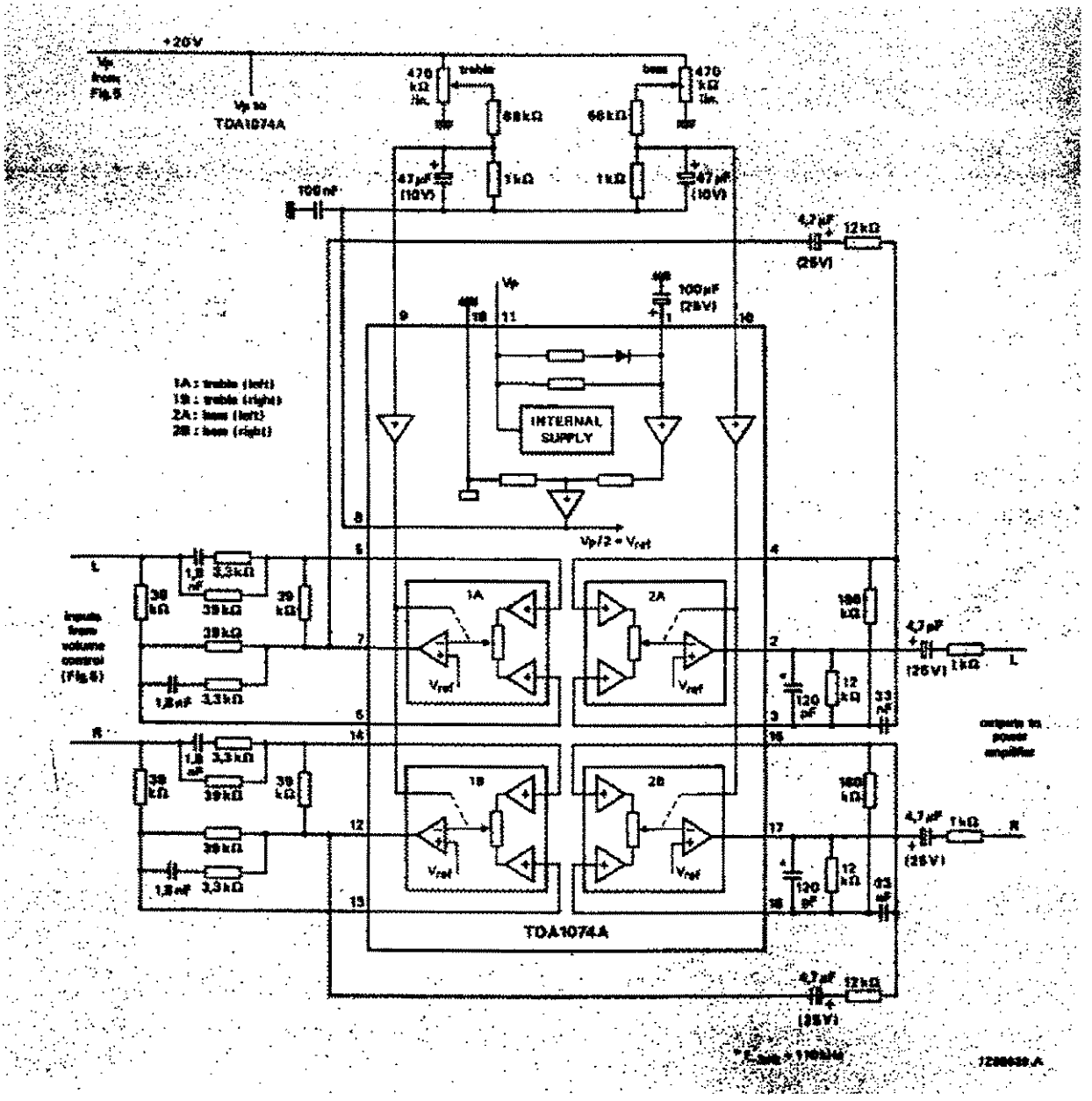


Figura E. 24 - CIRC 9 - Imagem original Tiff

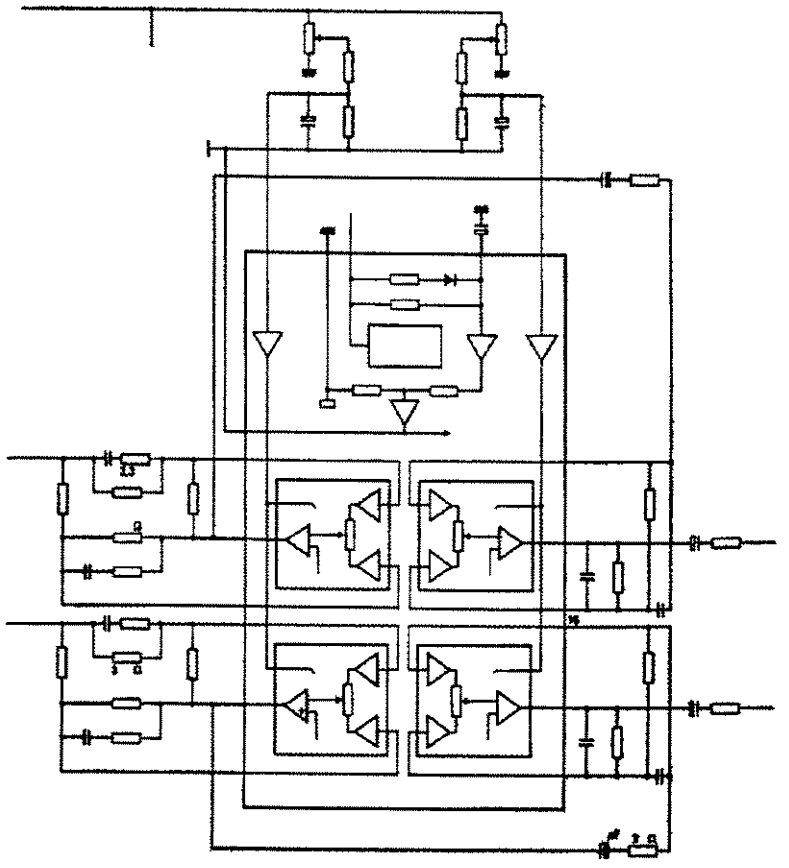


Figura E. 25

CIRC 9 - Circuito sem os caracteres

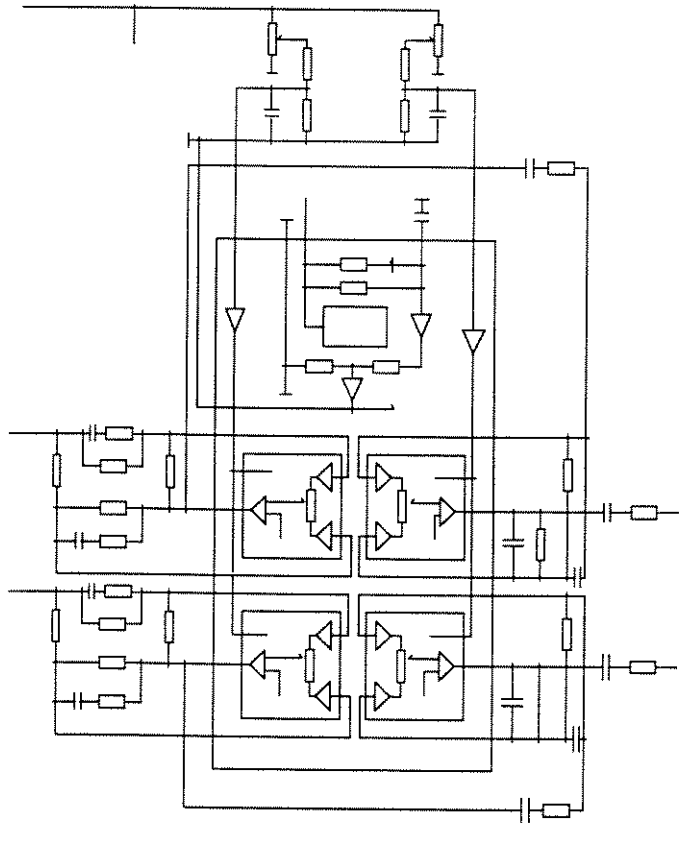


Figura E. 26

Imagem vetorizada pelo SRIDE

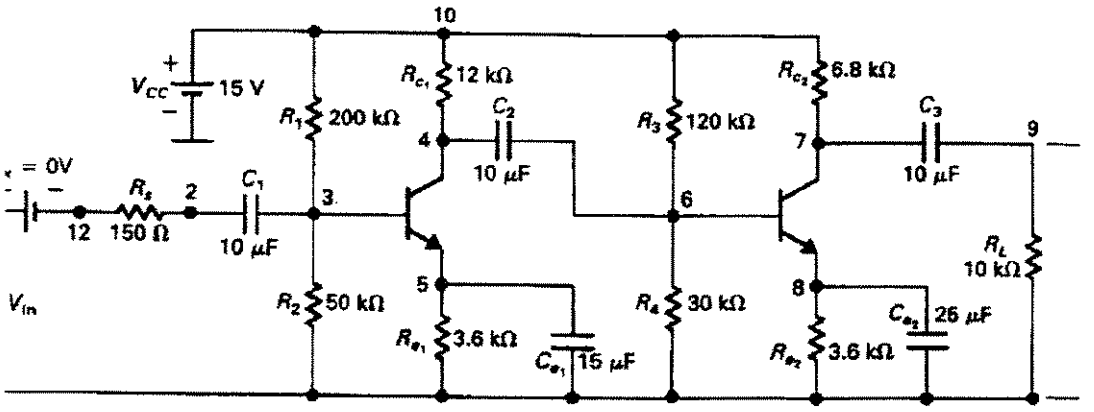


Figure 8.16 Two-Stage BJT Amplifier

Figura E. 27 - CIRC 10 - Imagem original Tiff

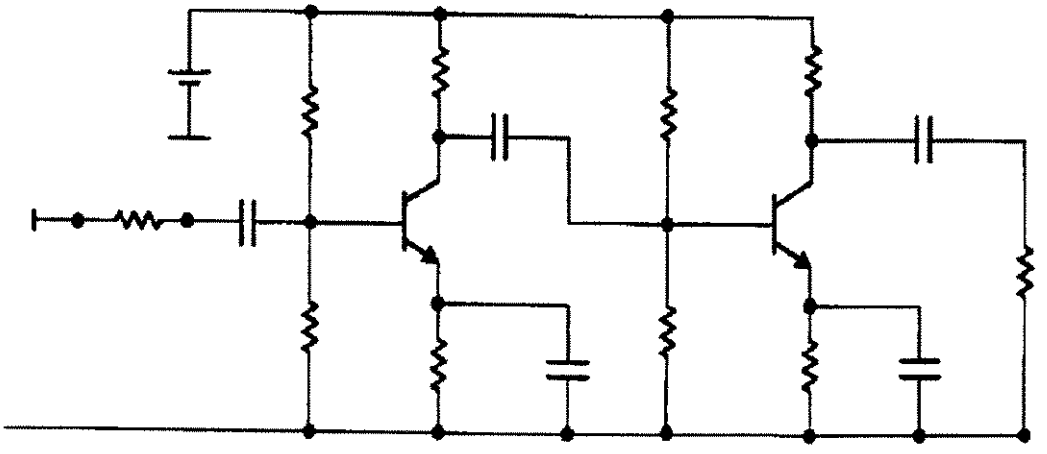


Figura E. 28 - CIRC 10 Circuito sem os caracteres

Erro: reconheceu capacitor ao invés de fonte de tensão

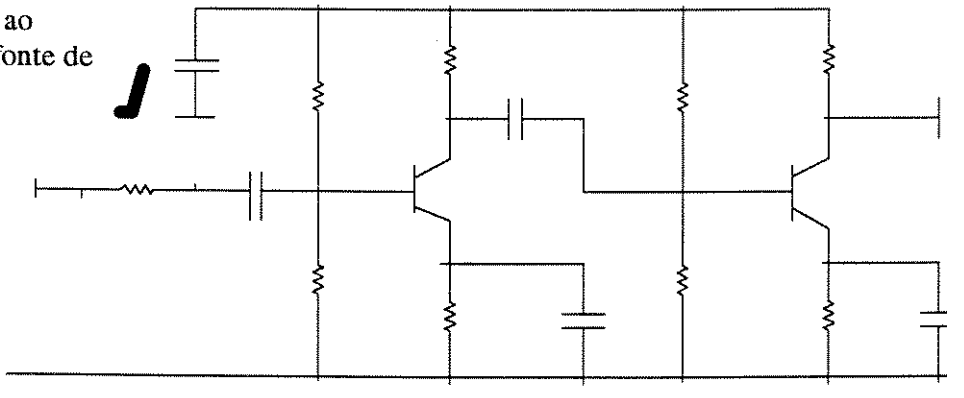


Figura E. 27 - Imagem vetorizada pelo Sride

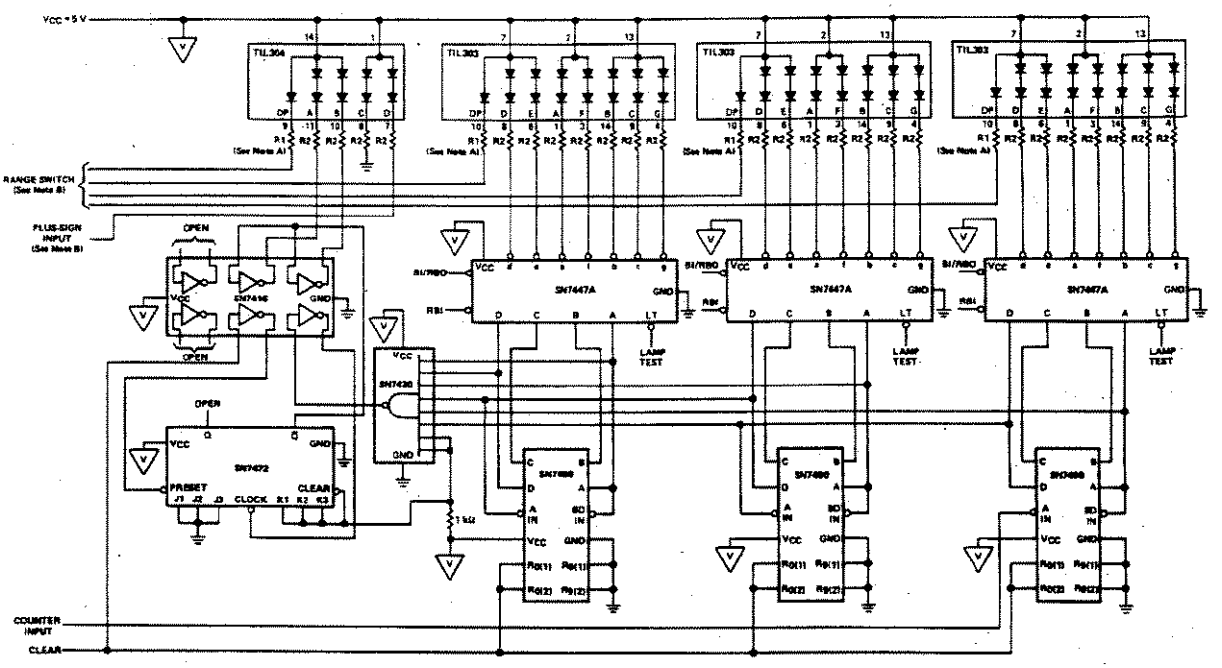


Figura E.30 - CIRC11 - Imagem original Tiff

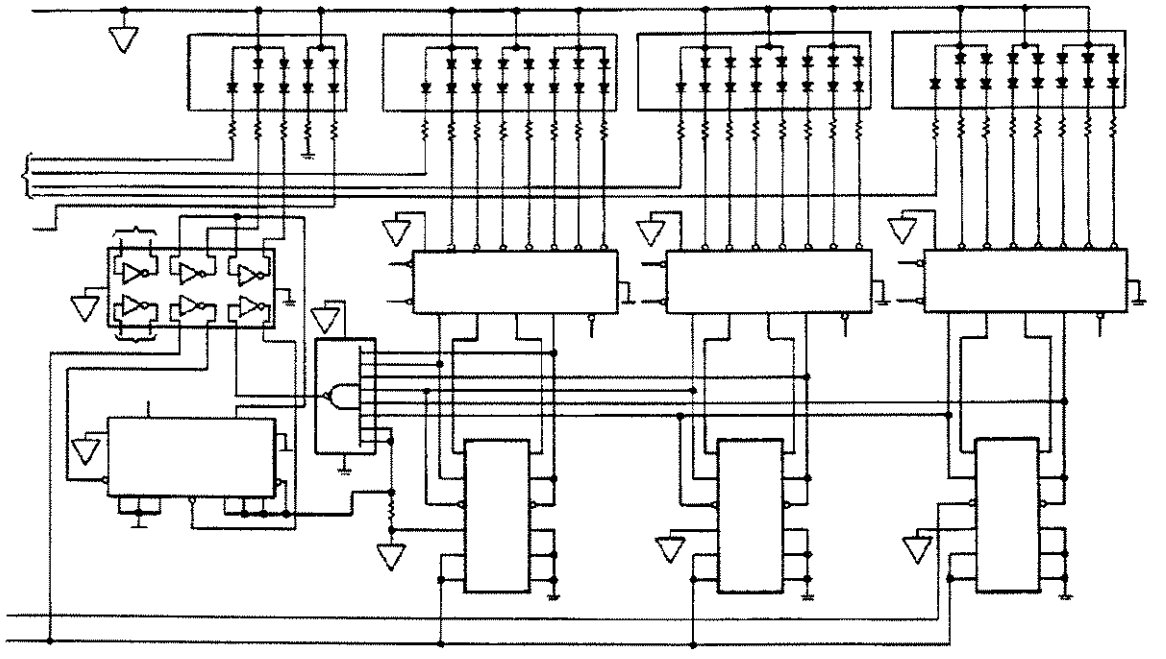


Figura E.31 - CIRC11 - circuito sem os caracteres

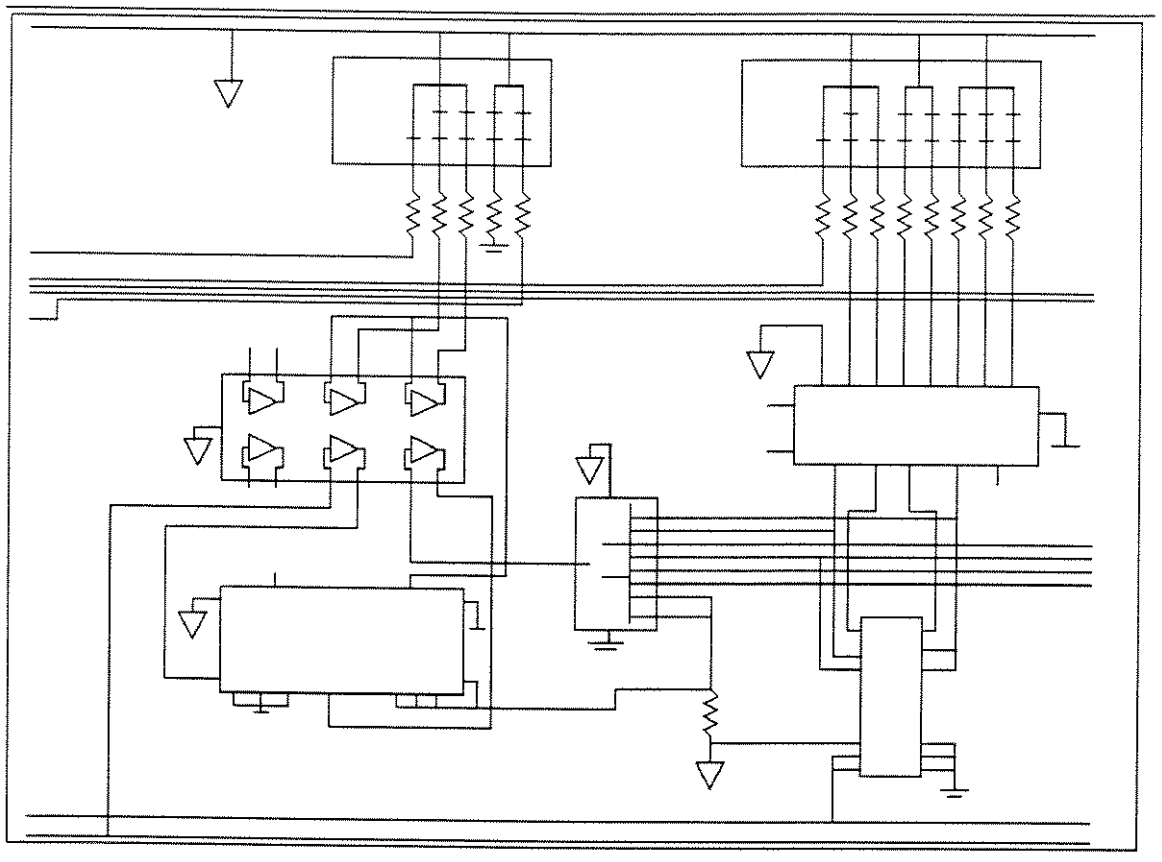
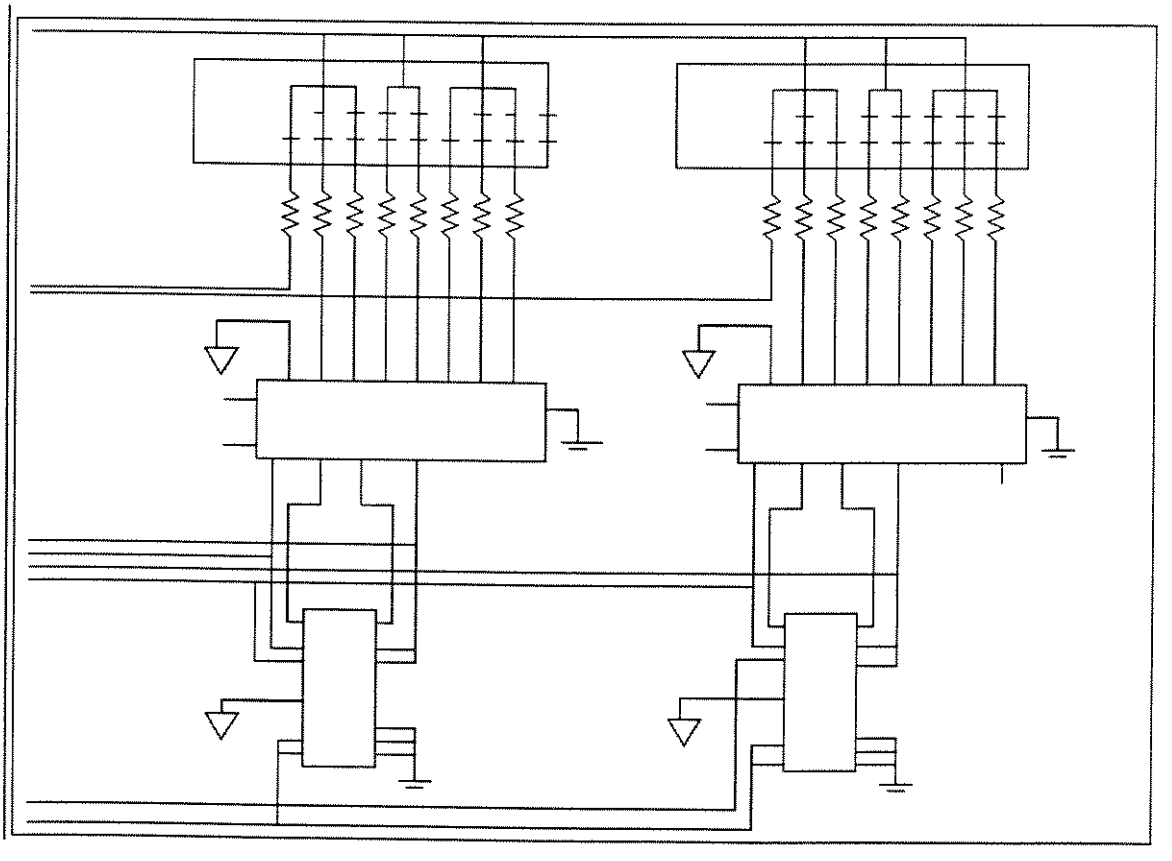


Figura E.32 - imagem vetorizada pelo Sride



APÊNDICE F

Resultados obtidos na análise do circuito CIRC1

Apresentamos neste apêndice as tabelas completas geradas pelo sistema SRIDE aplicado no diagrama de circuito CIRC1 cujo desenho foi apresentado na figura E.1 do apêndice E.

Após a vetorização deste circuito, a lista de vértices e a lista de vetores gerados pelo SRIDE são mostradas a seguir:

LISTA DOS VÉRTICES DO DIAGRAMA: CIRC1

Esta lista e' constituída das informações:

.numero do vértice;
.coordenadas do vertice;
.grau (quantidade de linhas que acessam o vértice);
.vértice equivalente:
-1: indica nenhum equivalente;
-n: indica que o vértice n e' seu equivalente;

0) 227 24 - 3 - -1
1) 346 24 - 4 - -1
2) 155 92 - 2 - -1
3) 159 94 - 2 - -1
4) 490 204 - 2 - -1
5) 490 293 - 2 - -1
6) 423 208 - 3 - -1
7) 423 230 - 2 - -1
8) 427 232 - 2 - -1
9) 159 24 - 3 - -1
10) 96 24 - 2 - -1
11) 494 198 - 2 - -1
12) 486 202 - 2 - -1
13) 159 159 - 4 - -1
14) 223 61 - 2 - -1
15) 231 65 - 2 - -1
16) 223 69 - 2 - -1
17) 494 182 - 2 - -1
18) 486 186 - 2 - -1
19) 231 73 - 2 - -1
20) 163 72 - 2 - -1
21) 155 76 - 2 - -1
22) 163 88 - 2 - -1
23) 296 159 - 2 - -1
24) 346 159 - 4 - -1
25) 346 70 - 2 - -1
26) 346 94 - 2 - -1
27) 227 55 - 2 - -1
28) 231 57 - 2 - -1
29) 350 72 - 2 - -1
30) 342 76 - 2 - -1
31) 350 80 - 2 - -1
32) 342 92 - 2 - -1
33) 346 210 - 2 - -1
34) 163 80 - 2 - -1
35) 346 234 - 2 - -1
36) 346 293 - 4 - -1
37) 346 297 - 1 - -1
38) 159 70 - 2 - -1
39) 350 212 - 2 - -1
40) 155 84 - 2 - -1
41) 494 190 - 2 - -1
42) 486 194 - 2 - -1
43) 419 236 - 2 - -1
44) 342 216 - 2 - -1
45) 350 220 - 2 - -1
46) 342 224 - 2 - -1
47) 342 232 - 2 - -1
48) 350 228 - 2 - -1
49) 227 178 - 3 - -1
50) 219 180 - 1 - -1
51) 227 208 - 3 - -1
52) 131 159 - 3 - -1
53) 58 155 - 2 - -1
54) 62 163 - 2 - -1
55) 66 155 - 2 - -1
56) 70 163 - 2 - -1
57) 159 210 - 2 - -1
58) 159 234 - 2 - -1
59) 159 293 - 3 - -1
60) 163 212 - 2 - -1
61) 155 216 - 2 - -1
62) 163 220 - 2 - -1
63) 155 224 - 2 - -1
64) 163 228 - 2 - -1
65) 155 232 - 2 - -1
66) 96 159 - 3 - -1
67) 124 159 - 3 - -1
68) 22 159 - 3 - -1
69) 39 159 - 2 - -1
70) 56 159 - 2 - -1
71) 80 159 - 2 - -1
72) 74 155 - 2 - -1
73) 78 163 - 2 - -1
74) 0 159 - 1 - -1
75) 15 159 - 3 - -1
76) 96 293 - 4 - -1
77) 404 293 - 4 - -1

78) 296 293 - 3 - -1
79) 423 293 - 3 - -1
80) 404 208 - 2 - -1
81) 404 346 - 2 - -1
82) 427 240 - 2 - -1
83) 419 244 - 2 - -1
84) 427 248 - 2 - -1
85) 419 252 - 2 - -1
86) 423 254 - 2 - -1
87) 227 209 - 0 - 51
88) 227 230 - 2 - -1
89) 227 254 - 2 - -1
90) 227 293 - 4 - -1
91) 231 232 - 2 - -1
92) 223 236 - 2 - -1
93) 231 240 - 2 - -1
94) 223 244 - 2 - -1
95) 231 248 - 2 - -1
96) 223 252 - 2 - -1
97) 227 299 - 1 - -1
98) 0 293 - 1 - -1
99) 96 346 - 2 - -1
100) 96 66 - 3 - -1
101) 106 66 - 1 - -1
102) 84 66 - 1 - -1
103) 90 70 - 1 - -1
104) 96 70 - 3 - -1
105) 100 70 - 1 - -1
106) 96 120 - 1 - -1
107) 296 247 - 3 - -1
108) 308 247 - 1 - -1
109) 296 208 - 2 - -1
110) 286 247 - 1 - -1
111) 296 255 - 3 - -1
112) 423 103 - 3 - -1
113) 464 103 - 3 - -1
114) 470 103 - 3 - -1
115) 490 103 - 2 - -1
116) 256 89 - 1 - -1
117) 256 103 - 3 - -1
118) 227 103 - 4 - -1
119) 223 103 - 1 - -1
120) 464 119 - 1 - -1
121) 464 89 - 1 - -1
122) 470 119 - 1 - -1
123) 470 89 - 1 - -1
124) 256 119 - 1 - -1
125) 404 137 - 1 - -1
126) 404 149 - 3 - -1
127) 404 159 - 3 - -1
128) 404 168 - 3 - -1
129) 404 184 - 1 - -1
130) 404 150 - 0 - 126
131) 423 140 - 2 - -1
132) 423 178 - 3 - -1
133) 416 180 - 1 - -1
134) 208 159 - 3 - -1
135) 208 149 - 3 - -1
136) 227 140 - 2 - -1
137) 208 168 - 3 - -1
138) 208 169 - 0 - 137
139) 208 184 - 1 - -1
140) 208 137 - 1 - -1
141) 208 148 - 0 - 135
142) 124 172 - 1 - -1
143) 124 145 - 1 - -1
144) 324 331 - 1 - -1
145) 324 346 - 3 - -1
146) 162 346 - 2 - -1
147) 186 346 - 2 - -1
148) 255 346 - 2 - -1
149) 164 342 - 2 - -1
150) 168 350 - 2 - -1
151) 172 342 - 2 - -1
152) 176 350 - 2 - -1
153) 180 342 - 2 - -1
154) 184 350 - 2 - -1
155) 324 361 - 1 - -1
156) 423 24 - 2 - -1
157) 346 23 - 0 - 1
158) 346 20 - 1 - -1
159) 286 255 - 1 - -1
160) 308 255 - 1 - -1
161) 423 55 - 2 - -1
162) 427 57 - 2 - -1
163) 419 61 - 2 - -1
164) 427 65 - 2 - -1
165) 419 69 - 2 - -1
166) 423 79 - 2 - -1
167) 427 73 - 2 - -1
168) 419 77 - 2 - -1
169) 490 180 - 2 - -1
170) 227 79 - 2 - -1
171) 223 77 - 2 - -1
172) 131 172 - 1 - -1
173) 263 103 - 3 - -1

174) 263 119 - 1 - -1
175) 296 103 - 2 - -1
176) 263 89 - 1 - -1
177) 131 145 - 1 - -1
178) 331 346 - 3 - -1
179) 15 172 - 1 - -1
180) 15 146 - 1 - -1
181) 22 152 - 1 - -1
182) 22 166 - 1 - -1
183) 331 361 - 1 - -1
184) 331 331 - 1 - -1
185) 342 84 - 2 - -1
186) 350 88 - 2 - -1
•

Esta lista de vértices é gerada pelo SRIDE logo após a vetorização do diagrama de circuito. Da mesma forma é também gerada a lista dos vetores componentes do circuito, mostrada a seguir.

TA DAS LINHAS
TITUINTES DO DIAGRAMA
DE CIRCUITO: CIRC1

Esta lista e' constituída das seguintes informações:

. ordem da linha;
. vértice inicial e vértice final;
. angulo da linha;

0) 0 1 - 0.00000
1) 2 3 - 153.43495
2) 4 5 - 90.00000
3) 6 7 - 90.00000
4) 7 8 - 153.43495
5) 9 0 - 0.00000
6) 10 9 - 0.00000
7) 11 12 - 26.56505
8) 3 13 - 90.00000
9) 14 15 - 153.43495
10) 15 16 - 26.56505
11) 17 18 - 26.56505
12) 16 19 - 153.43495
13) 20 21 - 26.56505
14) 12 4 - 153.43495
15) 22 2 - 26.56505
16) 23 24 - 0.00000
17) 1 25 - 90.00000
18) 26 24 - 90.00000

19) 27 28 - 153.43495
20) 28 14 - 26.56505
21) 25 29 - 153.43495
22) 29 30 - 26.56505
23) 30 31 - 153.43495
24) 32 26 - 153.43495
25) 24 33 - 90.00000
26) 21 34 - 153.43495
27) 35 36 - 90.00000
28) 36 37 - 90.00000
29) 9 38 - 90.00000
30) 33 39 - 153.43495
31) 34 40 - 26.56505
32) 41 42 - 26.56505
33) 8 43 - 26.56505
34) 40 22 - 153.43495
35) 39 44 - 26.56505
36) 44 45 - 153.43495
37) 45 46 - 26.56505
38) 38 20 - 153.43495
39) 47 35 - 153.43495
40) 48 47 - 26.56505
41) 49 50 - 14.03624
42) 49 51 - 90.00000
43) 52 13 - 0.00000
44) 53 54 - 116.56506
45) 54 55 - 63.43494
46) 55 56 - 116.56506
47) 13 57 - 90.00000
48) 58 59 - 90.00000
49) 57 60 - 153.43495

50) 60 61 - 26.56505
51) 61 62 - 153.43495
52) 62 63 - 26.56505
53) 63 64 - 153.43495
54) 64 65 - 26.56505
55) 65 58 - 153.43495
56) 46 48 - 153.43495
57) 66 67 - 0.00000
58) 68 69 - 0.00000
59) 69 70 - 0.00000
60) 71 66 - 0.00000
61) 70 53 - 63.43494
62) 56 72 - 63.43494
63) 72 73 - 116.56506
64) 73 71 - 63.43494
65) 74 75 - 0.00000
66) 66 76 - 90.00000
67) 36 77 - 0.00000
68) 78 36 - 0.00000
69) 77 79 - 0.00000
70) 80 77 - 90.00000
71) 77 81 - 90.00000
72) 79 5 - 0.00000
73) 43 82 - 153.43495
74) 82 83 - 26.56505
75) 83 84 - 153.43495
76) 84 85 - 26.56505
77) 85 86 - 153.43495
78) 51 88 - 90.00000
79) 89 90 - 90.00000
80) 88 91 - 153.43495
81) 91 92 - 26.56505
82) 92 93 - 153.43495
83) 93 94 - 26.56505
84) 94 95 - 153.43495
85) 95 96 - 26.56505
86) 96 89 - 153.43495
87) 90 78 - 0.00000
88) 59 90 - 0.00000
89) 90 97 - 90.00000
90) 76 59 - 0.00000
91) 98 76 - 0.00000
92) 76 99 - 90.00000
93) 100 101 - 0.00000
94) 10 100 - 90.00000
95) 102 100 - 0.00000
96) 103 104 - 0.00000
97) 104 105 - 0.00000
98) 104 106 - 90.00000
99) 107 108 - 0.00000
100) 109 107 - 90.00000
101) 110 107 - 0.00000
102) 111 78 - 90.00000
103) 112 113 - 0.00000
104) 114 115 - 0.00000
105) 116 117 - 90.00000
106) 118 117 - 0.00000
107) 119 118 - 0.00000
108) 113 120 - 90.00000
109) 121 113 - 90.00000
110) 114 122 - 90.00000
111) 123 114 - 90.00000
112) 117 124 - 90.00000
113) 125 126 - 90.00000
114) 126 127 - 90.00000
115) 127 128 - 90.00000
116) 128 129 - 90.00000
117) 126 131 - 25.34617
118) 128 132 - 152.24146
119) 132 133 - 15.94539
120) 13 134 - 0.00000
121) 135 136 - 27.75854
122) 137 49 - 152.24146
123) 135 134 - 90.00000
124) 134 137 - 90.00000
125) 137 139 - 90.00000
126) 140 135 - 90.00000
127) 67 142 - 90.00000
128) 143 67 - 90.00000
129) 80 6 - 0.00000
130) 144 145 - 90.00000
131) 99 146 - 0.00000
132) 147 148 - 0.00000
133) 148 145 - 0.00000
134) 146 149 - 63.43494
135) 149 150 - 116.56506
136) 150 151 - 63.43494
137) 151 152 - 116.56506
138) 152 153 - 63.43494
139) 153 154 - 116.56506
140) 154 147 - 63.43494
141) 145 155 - 90.00000
142) 1 156 - 0.00000
143) 1 158 - 90.00000
144) 159 111 - 0.00000
145) 111 160 - 0.00000

146) 161 162 - 153.43495
147) 162 163 - 26.56505
148) 163 164 - 153.43495
149) 132 6 - 90.00000
150) 164 165 - 26.56505
151) 166 112 - 90.00000
152) 165 167 - 153.43495
153) 167 168 - 26.56505
154) 168 166 - 153.43495
155) 169 17 - 153.43495
156) 0 27 - 90.00000
157) 170 118 - 90.00000
158) 171 170 - 153.43495
159) 118 136 - 90.00000
160) 52 172 - 90.00000
161) 173 174 - 90.00000
162) 173 175 - 0.00000
163) 115 169 - 90.00000
164) 176 173 - 90.00000
165) 177 52 - 90.00000

166) 178 81 - 0.00000
167) 75 179 - 90.00000
168) 180 75 - 90.00000
169) 181 68 - 90.00000
170) 68 182 - 90.00000
171) 51 109 - 179.16968
172) 86 79 - 90.00000
173) 178 183 - 90.00000
174) 184 178 - 90.00000
175) 18 41 - 153.43495
176) 175 23 - 90.00000
177) 31 185 - 26.56505
178) 185 186 - 153.43495
179) 186 32 - 26.56505
180) 42 11 - 153.43495
181) 112 131 - 90.00000
182) 156 161 - 90.00000
183) 19 171 - 26.56505
184) 24 127 - 0.00000
•

Após a vetorização, o sistema SRIDE obtém os laços formados pelos vetores constituintes do circuito, e com eles as características dos elementos formados por estes vetores. Por último, procede-se o reconhecimento dos elementos encontrados após a classificação destes elementos dentre os *templates* armazenados num banco de conhecimento. A fim de que se proceda a verificação do correto reconhecimento destes elementos, o sistema SRIDE faz uso do PSPICE, o [illegible] uma análise topológica e elétrica do sistema reconhecido. Para isto, SRIDE cria automaticamente, após o reconhecimento dos símbolos, uma lista (*net list*) contendo a relação dos elementos constituintes do diagrama. Esta lista tem a seguinte forma:

NOME DO ELEMENTO	vertice inicial	vértice final	valor	unidade ou modelo

Assim sendo, para o circuito aqui mostrado como exemplo, obteu-se a seguinte *net list:*

```
* circ1.net
* net list entrada para o Pspice

C0     53   68    1.000F
V1     69   76    1.000V
V2     105  101    1.000V
C3     108  112    1.000F
C4     114  115    1.000F
C5     118  174    1.000F
C6     146  179    1.000F
R7     4    39    1.000ohm
R8     5    170   1.000ohm
R9     8    87    1.000ohm
R10    26   27    1.000ohm
R11    28   171    1.000ohm
R12    34   36    1.000ohm
R13    58   59    1.000ohm
R14    71   72    1.000ohm
R15    89   90    1.000ohm
R16    147  148    1.000ohm
R17    162  167    1.000ohm
Q18    133  128   132 MQNPN
Q19    50   135   137 MQNPN
A20    1    2     0.000
A21    5    6     0.000
A22    7    8     0.000
A23    10   1     0.000
A24    11   10    0.000
A25    4    14    0.000
A26    24   25    0.000
A27    2    26    0.000
A28    27   25    0.000
A29    25   34    0.000
A30    36   37    0.000
A31    37   38    0.000
A32    10   39    0.000
A33    50   52    0.000
A34    53   14    0.000
A35    14   58    0.000
A36    59   60    0.000
A37    67   68    0.000
A38    69   70    0.000
A39    70   71    0.000
A40    72   67    0.000
A41    75   76    0.000
A42    67   77    0.000
A43    37   78    0.000
A44    79   37    0.000
A45    78   80    0.000
A46    81   78    0.000
A47    78   82    0.000
A48    80   6     0.000
A49    52   89    0.000
A50    90   91    0.000
A51    91   79    0.000
A52    60   91    0.000
A53    91   98    0.000
A54    77   60    0.000
A55    99   77    0.000
A56    77   100    0.000
A57    11   101    0.000
A58    105  0     0.000
A59    110  108    0.000
A60    112  79    0.000
A61    113  114    0.000
A62    115  116    0.000
A63    119  118    0.000
A64    120  119    0.000
A65    14   135    0.000
A66    81   7     0.000
A67    100  147    0.000
A68    148  149    0.000
A69    149  146    0.000
A70    2    157    0.000
A71    2    159    0.000
A72    133  7     0.000
A73    167  113    0.000
A74    1    28    0.000
A75    171  119    0.000
A76    119  137    0.000
A77    174  176    0.000
A78    116  170    0.000
A79    179  82    0.000
A80    52   110    0.000
A81    87   80    0.000
A82    176  24    0.000
A83    113  132    0.000
A84    157  162    0.000
A85    25   128    0.000
A86    75   99    0.000
.model MQNPN npn
.END
•
```

Nesta lista, o nome do elemento é representado por uma letra, que determina o código do símbolo (**R** : resistor, **C** : capacitor, **V** : fonte de tensão, **D** : diodo, **A** : curto circuito, etc) mais um número que representa a ordem na qual o elemento foi definido.

A verificação da correteza do circuito através do PSPICE é obtida através da análise dos resultados produzidos por este analisador. Estes resultados são armazenados num arquivo de saída, o qual é analisado pelo sistema SRIDE. Encontrando erros, ele alerta o usuário de tais erros.

```
**** 03/19/97 17:40:45 ******** Evaluation PSpice (September 1991)

* NEWSPICE FILE

****     CIRCUIT DESCRIPTION

*******************************************************************************

C0 4 5 1.000F
V1 0 5 1.000V
V2 0 1 1.000V
C3 50 5 1.000F
C4 113 115 1.000F
C5 119 24 1.000F
C6 148 5 1.000F
R7 4 1 1.000ohm
R8 5 115 1.000ohm
R9 5 5 1.000ohm
R10 1 24 1.000ohm
R11 1 119 1.000ohm
R12 24 5 1.000ohm
R13 4 5 1.000ohm
R14 0 5 1.000ohm
R15 50 5 1.000ohm
R16 5 148 1.000ohm
R17 1 113 1.000ohm
Q18 5 24 113 MQNPN
Q19 50 4 119 MQNPN
.model MQNPN npn
.END
```

```
**** 03/19/97 17:40:45 ******** Evaluation PSpice (September 1991) **********

 * NEWSPICE FILE


 ****    BJT MODEL PARAMETERS

******************************************************************************
         MQNPN
         NPN
      IS  100.000000E-18
      BF  100
      NF   1
      BR   1
      NR   1


**** 03/19/97 17:40:45 ******** Evaluation PSpice (September 1991) **********

 * NEWSPICE FILE

 **    SMALL SIGNAL BIAS SOLUTION      TEMPERATURE =   27.000 DEG C
******************************************************************************

 NODE   VOLTAGE    NODE   VOLTAGE    NODE   VOLTAGE    NODE   VOLTAGE


(  1)      -1.0000     (  4)    -1.0000     (  5)    -1.0000     ( 24)    -1.0000

( 50)      -1.0000     (113)    -1.0000     (115)    -1.0000     (119)    -1.0000

(148)      -1.0000


   VOLTAGE SOURCE CURRENTS
   NAME       CURRENT

   V1        -1.000E+00
   V2         2.220E-16

   TOTAL POWER DISSIPATION   1.00E+00  WATTS

      JOB CONCLUDED

      TOTAL JOB TIME            .22
```

Por último, é gerado uma lista DXF, que é o formato para ser utilizado em sistema CAD, o qual é acessado pelo sistema SRIDE a fim de proporcionar ao usuário do sistema, mais um meio de verificação do sistema reconhecido, além de automatizar o armazenamento e manutenção do desenho em estudo. Esta lista DXF é gerada a partir da *net list* do PSPICE, proporcionando assim, absoluta certeza da correteza ou não do sistema originado do processamento pelo sistema SRIDE.

BIBLIOGRAFIA

1). K. **Abe**, Y. **Azumatani**, M. **Mukouda** e S. **Suzuki**, "Discrimination of Symbols, Lines and Characters in Flowchart Recognition", *Proceedings of 8th International Conference on Pattern Recognition*, pp. 1071-1074, (1986).

2). S. B. **Alves**, "Visart - Visão Artificial para Uso Industrial: Um sistema abrangente", Tese de Doutorado, INPE-5379-TDI/474, (1992).

3). C. J. **Ammann** e A. G. S.**Angus**, "Fast Thinning Algorithm for Binary Images", *Image and Vision Computing*, Vol. 3 , # 2, pp. 71-79, (1985).

4). D. H. **Ballard** e C. M. **Brown**, "Computer Vision", *Prentice Hall*, (1982).

5). G. J. F. **Banon** e J. **Barrera**, "Bases da Morfologia Matemática para a Análise de Imagens Binárias", IX Escola de Computação, Recife, (1994).

6). H. **Bley**, "Segmentation and Preprocessing of Electrical Schematics Using Picture Graphs", *Computer Vision, Graphics, and Image Processing*, Vol. 28, pp. 271-288, (1984).

7). S. T. **Bow**, "Pattern Recognition - Applications to Large Data-Set Problems", *Marcel Dekker, Inc.* , (1984).

8). H. **Bunke**, "Atributed Programmed Graph Grammars and Their Application to Schematic Diagram Interpretation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 4, # 6, pp. 574-582, (1982).

9). H. **Bunke**, "Experience with Several Methods for the Analysis of Schematic Diagrams", *Proceedings of 6th International Conference on Pattern Recognition*, pp. 710-712, (1982).

10). H. **Bunke**, "Automatic Interpretation of Lines and Text in Circuit Diagrams", *Pattern Recognition Theory and Applications*, pp. 297-310, (1982).

11). H. **Bunke** e G. **Allerman**, "Understanding of Circuit Diagrams by Means of Probabilistic Relaxation", *Signal processing*, Vol. 4, pp. 169-180, (1982).

12). T. P. **Clement**, "The Extraction of Line-Structured Data from Engineering Drawings", *Pattern Recognition*, Vol. 14, #1-6, pp. 43-52, (1981).

13). D. **Dori**, "A Syntatic/Geometric Approach to Recognition of Dimensions in Engineering Machine Drawings", *Computer Vision, Graphics, and Image Processing*, Vol. 47, pp. 271-291, (1989).

14). R. O. **Duda** e P. E. **Hart**, "Pattern Classification and Scene Analysis", *Jon Willey & Sons*, (1973).

14) M. **Ejiri**, T. **Miyatake**, H. **Matsushima**, S. **Kakumoto** e S. **Shimada**, "Automatic Recognition of Design Drawings and Maps", *Proceedings of 7th International Conference on Pattern Recognition*, pp. 1296-1305, (1984).

15) C. S. **Fahn**, J. F. **Wang**, e J. Y. **Lee**, "A Topology-Based Component Extractor for Understanding Electronic Circuit Diagrams", *Computer Vision, Graphics, and Image Processing*, Vol. 44, pp. 119-138, (1988).

16) A. J. **Filipski** e R. **Flandrena**, "Automated Conversion of Engineering Drawings to CAD Form", *Proceedings of IEEE*, Vol. 80, # 7, pp. 1195-1209, (1992).

17) W. R. **Franklin** e V. **Akman**, "Reconstructing Visible Regions From Visible Segments". *Bit*, #26, pp. 430-441, (1986).

18) Y. **Fukada**, "A Primary Algorithm for the Understanding of Logic Circuit Diagrams", *Proceedings of 6th International Conference on Pattern Recognition*, pp. 706-709, (1982).

19) Y. **Fukada**, "A Primary Algorithm for the Understanding of Logic Circuit Diagrams", *Pattern Recognition*, Vol. 17, # 1, pp. 125-134, (1984).

20) K. **Fukunaga**, "Introduction to Statistical Pattern Recognition", *Academic Press*, (1972)

21) M. **Furuta**, N. **Kase** e S. **Emori**, "Segmentation and Recognition of Symbols for Handwritten Piping & Instrument Diagram", *Proceedings of 7th International Conference on Pattern Recognition*, pp. 626-629, (1984).

22) R. C. **Gonzalez** e J. T. **Tou**, "Pattern Recognition Principles", *Addison-Wesley Publishing Co*, (1974).

23) F. C. A. **Groen** e R. J. V. **Munster**, "Toppology Based Analysis of Schematic Diagrams", *Proceedings of 7th International Conference on Pattern Recognition*, pp. 1310-1312, (1984).

24) R. M. **Haralick** e D. **Queeney**, "Understanding Engineering Drawings", *Computer Vision, Graphics, and Image Processing*, Vol. 20, pp. 244-258, (1982).

25) J. F. **Harris**, J. **Kittler** , B. **Llewelly**, e G. **Preston**, "A Modular System for Interpreting Binary Pixel Representations of Line-Structered Data", *Pattern Recognition Theory and Applications*, pp.311-351, (1982).

26) C. M. **Holt**, A. **Stewart**, M. **Clint** e R. H. **Perrott**, "An Improved Parallel Thinning Algorithm", *Communications of the ACM*, Vol. 30, # 2, pp. 156-160, (1987).

27) M. **Karima**, K. S. **Sadhal** e T. O. **McNeil**, "From Paper Drawings to Computer Aided Design", *IEEE Computer, Graphics, and Applications*, Vol. 5, # 2, pp. 27-39, (1985).

28) R. **Kasturi**, S. T. **Bow**, W. **El-Masri**, J. R. **Gattiker**, J. **Shah** e U. B. **Mokate**, "A System for Interpretation of Line Drawings", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, #10, pp. 978-992, (1990).

29) D. C. **Kay** e J. R. **Levine**, "Graphics File Formats", *Mc Graw-Hill*, (1982).

30) T. Y. **Kong** e A. **Rosenfeld**, "Digital Topology: Introduction and Survey", *Computer Vision, Graphics, and Image Processing*, Vol. 48, pp. 357-393, (1989).

31) C. P. **Lai** e R. **Kasturi**, "Dectection of Dimension Sets in Engineering Drawings", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, #16, pp. 848-855, (1994).

32) L. **Lam**, S. W. **Lee** e C. Y. **Suen**, "Thinning Methodologies - A Comprehensive Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, #9, pp. 869-885, (1982).

33) X. **Lin**, S. **Shimotsuji** , M. **Minoh** e T. **Sakai**, "Efficient Diagram Understanding With Characteristic Pattern Detection", *Computer Vision, Graphics, and Image Processing*, Vol. 3, pp. 84-1062, (1985).

34) R. **Lumia**, L. **Shapiro** e O. **Zuniga**, "A New Connected Components Algorithm for Virtual Memory Computer", *Computer Vision, Graphics, and Image Processing*, Vol. 22, pp. 287-300, (1983).

35) N. D. A. **Mascarenhas** e F. R. D. **Velasco**, "Processamento Digital de Imagens", *IV Escola de Computação, USP*, (1984).

36) S. **Marshall**, "Review of Shape Coding Techniques", *Image and Vision Computing*, Vol 7, #4, pp. 281-294, (1989).

37) T. **Masui**, S. **Chimizu**, M. **Yoshida** e T. **Abe**, "Recognition System for Design Chart Drawn on Section Paper", *Proceedings of 5th International Conference on Pattern Recognition*, pp. 127-130, (1980).

38) N. J. **Naccache** e R. **Shinghal**, "SPTA: A Proposed Algorithm for Thinning Binary Patterns", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 14, # 3, pp. 409-418, (1984).

39) V. **Nagasamy** e N. **Langrana**, "Engineering Drawing Processing and Vectorization System", *Computer Vision, Graphics, and Image Processing*, Vol. 49, pp. 379-397, (1990).

40) W. **Niblack**, "An Introduction to Digital Image Processing", *Prenticce Hall* , (1982).

41) A. **Okazaki**, T. **Kondo**, K. **Mori**, S. **Tsunekawa** e E. **Kawamoto**, "An Automatic Circuit Diagram Reader with Loop-Structured-Based Symbol Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, #3, pp. 331-341, (1988).

42) T. **Pavlidis**, "A Review of Algorithms for Shape Analysis", *Computer Vision, Graphics, and Image Processing*, Vol. 7, pp. 243-258, (1978).

43) T. **Pavlidis**, "Vector and Arc Enconding of Graphics and Text", *Proceedings of 6th International Conference on Pattern Recognition*, pp. 610-613, (1982).

44) T. **Pavlidis**, "A Vectorizer and Feature Extractor for Document Recognition", *Computer Vision, Graphics, and Image Processing*, Vol. 35, pp. 111-127, (1986).

45) M. H. **Rashid**, "SPICE for Circuits and Electronics Using Pspice", *Prentice Hall*, (1990).

46) A. **Rosenfeld** e A. C. **Kak**, "Digital Picture Processing", *Academic Press*, (1976).

47) T. **Sato** e A. **Tojo**, "Recognition and Understanding of Hand-Drawn Diagrams", *Proceedings of 6th International Conference on Pattern Recognition*, pp. 674-677, (1982).

48) R. **Schalkoff**, "Pattern Recognition - Statistical, Structural and Neural Approaches", *John Willey & Sons*, (1992).

49) R. **Sedgewick**, "Algorithms in C", *Addison-Wesley Pub. Co.* , (1990).

50) J. **Serra**, "Image Analysis and Mathematical Morphology", Academic Press, (1982).

51) S. **Shimizu**, S. **Nagata**, A. **Inoue** e M. **Yoshida**, "Logic Circuit Diagram Processing System", *Proceedings of 6th International Conference on Pattern Recognition*, pp. 717-719, (1982).

52) J. **Sklansky**, "Image Segmentation and Feature Extraction", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 8, # 4, pp. 237-247, (1978)

53) M. **Takagi**, T. **Konishi** e M. **Yamada**, "Automatic Digitizing and Processing Method for the Printed Circuit Pattern Drawings", *Proceedings of 6th International Conference on Pattern Recognition*, pp. 713-716, (1982).

54) J. P. **Tremblay** e P. G. **Sorenson**, "An Introduction to Data Structures with Applications", *Mc Graw-Hill*, (1984).

55) P. W. **Tuinenga**, "SPICE - A Guide and Electronics Using Pspice", *Prentice Hall*, (1992).

56) P. **Vaxiviere** e K. **Tombre**, "Celesstin: CAD Conversion of Mechanical Drawings", *Proceedings of IEEE Transasctions on Computer*, pp. 46-54, (1989).

57) L. **Vincent**, "Morphological Grayscale Reconstruction in Image Analysis: Applications and Efficient Algorithms", IEEE Int Coputer Vision and Pattern Recognition, pp. 633-635, (1992).

58) L. **Vincent**, "Morphological Grayscale Reconstruction: Definition, Efficient Algorithm and Applications in Image Analysis", IEEE Trans on Image Processing, Vol. 2, #2, pp.176-201, (1993).

59) T. Y. **Zhang** e C. Y. **Suen**, "A Fast Parallel Algorithm for Thinning Digital Pictures", *Communications of the ACM*, Vol. 27, # 3, pp. 236-242, (1984).

60) Y. **Yu**, A. **Samal** e S. **Seth**, "Automatic Segmentation of Engineering Drawings with Connections", *Pattern Recognition*, Vol. 27, pp. 391-404, (1994).

61) S. **Watanabe**, "Pattern Recognition - Human and Mechanical", *John Willey & Sons, Inc.*, (1985).