UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

PRÉ-PROCESSAMENTO E NÍVEL MÉDIO DE CLASSIFICAÇÃO PARA UM SISTEMA DE LEITURA AUTOMÁTICA DE SÍMBOLOS MUSICAIS

por

Francesco Artur Perrotti

Orientador

Prof. Dr. Roberto de Alencar Lotufo

Tese apresentada à Faculdade de Engenharia Elétrica da Universidade de Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

ÍNDICE

RESUMO

Este trabalho é uma contribuição ao projeto de um sistema de Reconhecimento Ótico de Símbolos Musicais (OMR). A partir da imagem digitalizada de uma partitura o OMR interpreta e reconhece os símbolos musicais. Nestes sistemas a delimitação dos símbolos constitue uma das principais dificuldades encontradas pelos pesquisadores.

É proposta uma metodologia para a delimitação dos símbolos utilizando-se de configurações de pontos considerados críticos e vetores na imagem.

O trabalho enfatiza a parte de pré-processamento da imagem digitalizada, a extração de atributos dos símbolos e a primeira fase de classificação. No pré-processamento foi desenvolvido um novo algoritmo de afinamento e de vetorização de imagens binárias.

As soluções propostas foram reunidas em um sistema modular de programas desenvolvidos em linguagem C.

ABSTRACT

This work is a contribution to the design of a Optical Music Recognition System (OMR). Given a digitized image of a partitur, the OMR interprets and recognizes the musical symbols. In these system, the symbol delimitation is one of the main dificulties found by the reseachers.

It is proposed a methodology for the symbol delimitation problem using the critical points configuration found in the image.

The work gives emphasis in the digitized image pre-processing, symbol attribute extrection and the first part of classification process. In the pre-processing, it was developed a new thinning and a new vectoring algorithm.

The solutions proposed here were collected together in a modular system of programs written in the C language.

1. Introdução

A tendência de redução de preços dos equipamentos de informática e em especial dos digitalizadores de imagens (*scanners*, por exemplo) e microcomputadores vem tornando o processamento de imagens cada vez mais acessível ao usuário leigo. Ao mesmo tempo, as placas de expansão para processamento de sinais sonoros são facilmente conectadas a pequenos sistemas de computação e ficam cada vez mais baratas e com mais recursos. Assim, pode-se prever que em breve a maioria dos usuários que tiver necessidade de processamento sonoro, terá acesso ao hardware necessário. Já com o software, o processo de barateamento caminha em um ritmo muito mais lento. São ainda poucos e caros os softwares desenvolvidos como ferramentas para músicos, compositores, maestros ou mesmo editores de partituras, mas a crescrente popularização da microinformática tende a inverter essa situação.

O Reconhecimento de Padrões tem sido objeto de pesquisa de muitos autores devido a infinidade de aplicações em que pode ser utilizado. Entre elas, o reconhecimento de caracteres (OCR) vem ganhando impulso rapidamente. Apesar da grande quantidade de trabalhos em OCR, há relativamente poucos que se ocupam do reconhecimento de partituras e símbolos musicais (OMR - Optical Music Recognition). Por este motivo, o reconhecimento de símbolos musicais é uma área em que não existem soluções definitivas, mas sim uma quantidade de caminhos inexplorados. Este trabalho explora um destes caminhos e descreve suas etapas de processamento. Para isso foi desenvolvido um sistema que implementa as soluções propostas e permite a visualização dos resultados em cada etapa.

A notação musical, assim como a escrita de qualquer língua, vem evoluindo ao longo dos séculos, adquirindo novos símbolos e gerando dialetos regionais. Os muitos instrumentos musicais têm características diferentes e cada um deles permite técnicas de execução variadas. Com o tempo surgiram símbolos que são usados apenas para um ou um grupo de instrumentos. A imensa variedade de símbolos criados, seja por hábitos regionais, ou seja para melhor registrar uma determinada técnica (como arpejo, trinado, uso de pedais, arrasto e etc), torna inviável a criação de um sistema de reconhecimento capaz de reconhecer completamente qualquer partitura. Os sistemas de reconhecimento musical desenvolvidos até o

momento são bastante específicos e nenhum deles é capaz de ler todas as partituras de todos os instrumentos. Na verdade poucos músicos são capazes deste feito. Apesar disso, existe um subconjunto de símbolos que podem ser considerados padrão e são entendidos por todos os músicos alfabetizados em música em todos os países que adotam a notação ocidental.

1.1 Características do problema

Cada alfabeto de símbolos tem características próprias que devem ser levadas em conta durante o processo de reconhecimento. Os ideogramas chineses por exemplo, têm algumas características que não são encontradas no alfabeto latino e vice-versa. Por esta razão, uma metodologia que mostre ser eficiente para um alfabeto, pode não o ser para outro. Quase sempre um sistema de reconhecimento tem embutido algum conhecimento das características do alfabeto. Textos em alfabetos latinos por exemplo, geralmente são escritos em linhas horizontais. Já as linhas de um texto chinês são verticais. A organização espacial dos textos varia com o alfabeto.

Na notação musical, uma linha de texto é organizada em duas dimensões. A sequência temporal dos símbolos é dada linearmente pela dimensão horizontal de modo parecido com as linhas de um texto em alfabeto latino, mas a dimensão vertical é crucial para o reconhecimento. O mesmo símbolo em alturas diferentes tem significados diferentes. Além disso, existe a possibilidade de vários símbolos ocuparem a mesma coordenada temporal, como ocorre com os acordes onde várias notas que devem ser executadas ao mesmo tempo e às vezes com durações diferentes. Algo semelhante ocorre em relação à acentuação dos caracteres latinos. Letra e acento são lidos ao mesmo tempo, como partes do mesmo símbolo. A diferença é que embora a letra e o acento possam ser considerados um único símbolo sem aumentar o alfabeto demasiado, o mesmo não acontece com os acordes. Não é viável considerar um acorde como um símbolo por que neste caso o número de símbolos seria virtualmente infinito.

Além da organização espacial do texto, existem diferenças na topologia e morfologia dos símbolos. Os caracteres latinos nos estilos mais comuns são formados por linhas finas e de espessura razoavelmente constante. Já em uma partitura a maior parte dos símbolos representam notas musicais e estes símbolos têm sempre uma

circunferência em uma das extremidades que geralmente é cheia. As notas que dividem o mesmo tempo são normalmente unidas por uma barra que em determinadas condições pode ser confundida com as linhas do pentagrama. Uma característica desta barra de ligação é sua espessura, que geralmente é de 3 a 5 vezes maior que a espessura das linhas do pentagrama. Geralmente a espessura das linhas de uma imagem de caracteres não é importante para seu reconhecimento por ser constante. Assim boa parte dos trabalhos em OCR afinam a imagem desprezando a informação da espessura. Já para o reconhecimento de partituras, a espessura dos símbolos pode ser um fator determinante e deve ser preservada.

A principal característica dos textos em notação musical é a presença obrigatória do pentagrama. São 5 linhas horizontais que atravessam toda a extensão de uma linha de texto. Estas linhas conectam quase todos os símbolos dificultando o processo de segmentação. A conexão de um símbolo com os próximos (ligatura) é um problema que tem preocupado muitos autores na área de OCR. Pode ocorrer nos caracteres devido a ruído na imagem ou intencionalmente na impressão do texto. O que é um possível problema para o OCR, é uma característica do OMR. O processo de reconhecimento deve levar em conta que os símbolos estão sempre conectados pelo pentagrama.

A delimitação dos símbolos é um dos principais problemas enfrentados pelos sistemas de OMR. Vários fatores colaboram para dificultar esse processo. Como todos os símbolos estão conectados pelo pentagrama, a sua conectividade não é informação muito relevante. Além disso o tamanho dos símbolos varia muito. O retângulo envolvente de alguns símbolos é muito maior que o de outros. A variação no tamanho dos símbolos é tão grande que Prerau em sua tese de doutorado [BLO 92] usa o tamanho do retângulo envolvente como parâmetro no primeiro nível de classificação dos símbolos e consegue uma redução considerável no número de símbolos candidatos para cada retângulo (3 a 5 símbolos por retângulo segundo o autor). Finalmente, a posição do símbolo varia na dimensão vertical e não existe espaçamento fixo na dimensão horizontal. Por estes motivos, os sistemas de OMR dedicam boa parte do processamento à delimitação dos símbolos.

Em compensação, a presença do pentagrama introduz na imagem uma informacão importantíssima para o reconhecimento: a

escala dos símbolos. A relação entre a distância entre as linhas e o tamanho dos símbolos pode ser considerada constante nas partituras impressas. Segundo [BLO 92] esta afirmação não foi rigorosamente testada mas é considerada verdadeira pela quase totalidade de autores que trabalham com OMR. Em quase todos os trabalhos e também neste, a distância entre as linhas é o fator de normalização da escala dos símbolos.

Alguns autores removem o pentagrama da imagem depois que é localizado. A remoção do pentagrama é desejável porque facilita a delimitação dos símbolos, além de reduzir a variação na morfologia dos símbolos. Por outro lado, sua retirada implica em problemas de difícil solução. Símbolos podem ser fragmentados ou mutilados em regiões decisivas para o reconhecimento. Um problema típico que ocorre [BLO 92] é o desaparecimento de regiões do símbolo que tangenciam as linhas do pentagrama.

Roach e Tatem [ROA 88] descrevem um método de remoção do pentagrama que minimiza a fragmentação dos símbolos através de vetores associados a cada pixel ativo da imagem. Cada vetor tem sua origem em um pixel ativo e aponta para o pixel ativo conectado e mais distante dentro de uma janela. Os autores criaram uma gramática para o processamento dos vetores que é capaz de localizar e remover o pentagrama. Embora apresente ótimos resultados, o método não resolve o problema do desaparecimento da curvas tangentes às linhas.

A rígida gramática musical permite que algumas ambiguidades sejam resolvidas pelo contexto de outros símbolos já reconhecidos. Símbolos com morfologia parecida como o sustenido e o bequadro dificilmente podem ser trocados sem ferir a gramática da notação musical. As barras de compasso também fornecem meios de verificar a exatidão do reconhecimento, já que entre duas barras de compasso existe sempre o mesmo número de tempos. Outros símbolos ocorrem sempre na mesma altura em relação ao pentagrama como as claves, fórmulas de compasso, barras de repetição entre outros.

1.2 Sistemas de Reconhecimento de Padrões em imagens

Tipicamente os sistemas de reconhecimento de padrões em imagens normalmente têm 3 etapas distintas: pré-processamento, extração de características e classificação. As operações de processamento de imagens podem ser classificadas em operações de baixo, médio e alto nível segundo o grau de abstração dos dados. Operações de baixo nível manipulam dados organizados na forma de pixels, bytes e bits. Nas operações de médio nível os dados têm significados um pouco mais abstratos como pontos, retas, arcos, vetores, centros geométricos e outras primitivas. Finalmente as operações de alto nível exigem conhecimento do assunto da imagem e manipulam configurações de primitivas (pontos, retas, etc) para extrair informações.

O pré-processamento inclui a aquisição da imagem e operações que colocam os dados adquiridos em um formato mais apropriado para os objetivos do sistema. Os algoritmos usados nesta fase realizam operações de baixo/médio nível que manipulam os dados com pouco ou nenhum conhecimento de seu significado e por isso podem ser usados em muitas aplicações fora do Reconhecimento de Padrões. Os dados em geral não representam parte significativa da informação contida na imagem (letras, símbolos, texto, etc). Exemplos típicos desta fase de processamento são as operações de filtragem, afinamento, vetorização, extração de contorno, cálculo de transformadas e histogramas, filtros e outros processos. Roach e Tatem [ROA 88] citam as vantagens do uso de conhecimento sobre os símbolos musicais durante a fase de pré-processamento.

Durante a fase de extração de características são identificados e calculados atributos da imagem e dos objetos. Nesta fase de processamento já é necessário algum conhecimento do assunto da imagem. Os atributos têm que representar características que permitam diferenciar os símbolos do alfabeto e por este motivo os algoritmos usados nesta fase têm embutido conhecimento de características gerais do alfabeto. Os dados têm um nível maior de abstração e as operações que os manipulam são de médio/alto nível. O resultado desta fase são dados como centros de massa, retângulos envolventes, centros geométricos, configurações de linhas e pontos, etc. Os atributos podem ser globais usando a imagem inteira como domínio ou regionais onde o domínio é uma região da imagem. Estes

atributos são usados para classificar os objetos da imagem.

A etapa de classificação exige conhecimento de cada símbolo do alfabeto. Este conhecimento é codificado no programa ou armazenado em um banco de dados. Os atributos calculados na fase de extração de características são analisados e os símbolos são reconhecidos em função desta análise. Esta fase de processamento envolve operações de alto nível e é altamente específica para os objetivos do sistema. Em geral cada autor desenvolve um método diferente.

1.3 Método proposto

A idéia central do método apresentado por este trabalho é reconhecer os símbolos através da configuração dos pontos considerados críticos na imagem. São pontos críticos os pontos terminais, pontos de cruzamento e pontos de inflexão das curvas. Para facilitar a identificação destes pontos a imagem é afinada e vetorizada. Durante a fase de extração de características são calculados os atributos de cada ponto crítico identificado. Estes atributos podem ser usados para alimentar o banco de dados durante o treinamento, ou para o reconhecimento na etapa de classificação.

O treinamento supervisionado é feito fornecendo ao sistema imagens acompanhadas da lista de símbolos que contém e as coordenadas dos retângulos envolventes. O sistema, depois de calcular os atributos de todos os pontos críticos, associa a cada símbolo todos os pontos que estão dentro de seu retângulo envolvente. Os pontos, ao serem armazenados no banco de dados, recebem como coordenadas sua posição em relação ao retângulo do símbolo. Em seguida são calculados e armazenados os atributos dos símbolos da imagem.

Na fase de classificação, cada ponto é associado a uma lista dos pontos mais próximos no banco de dados. Cada ponto no banco de dados está associado a um símbolo e um retângulo envolvente. A idéia é detectar concentrações de pontos concordantes, ou seja pontos que se referem a um mesmo símbolo e retângulo envolvente. A figura 1 ilustra as etapas de processamento do sistema.

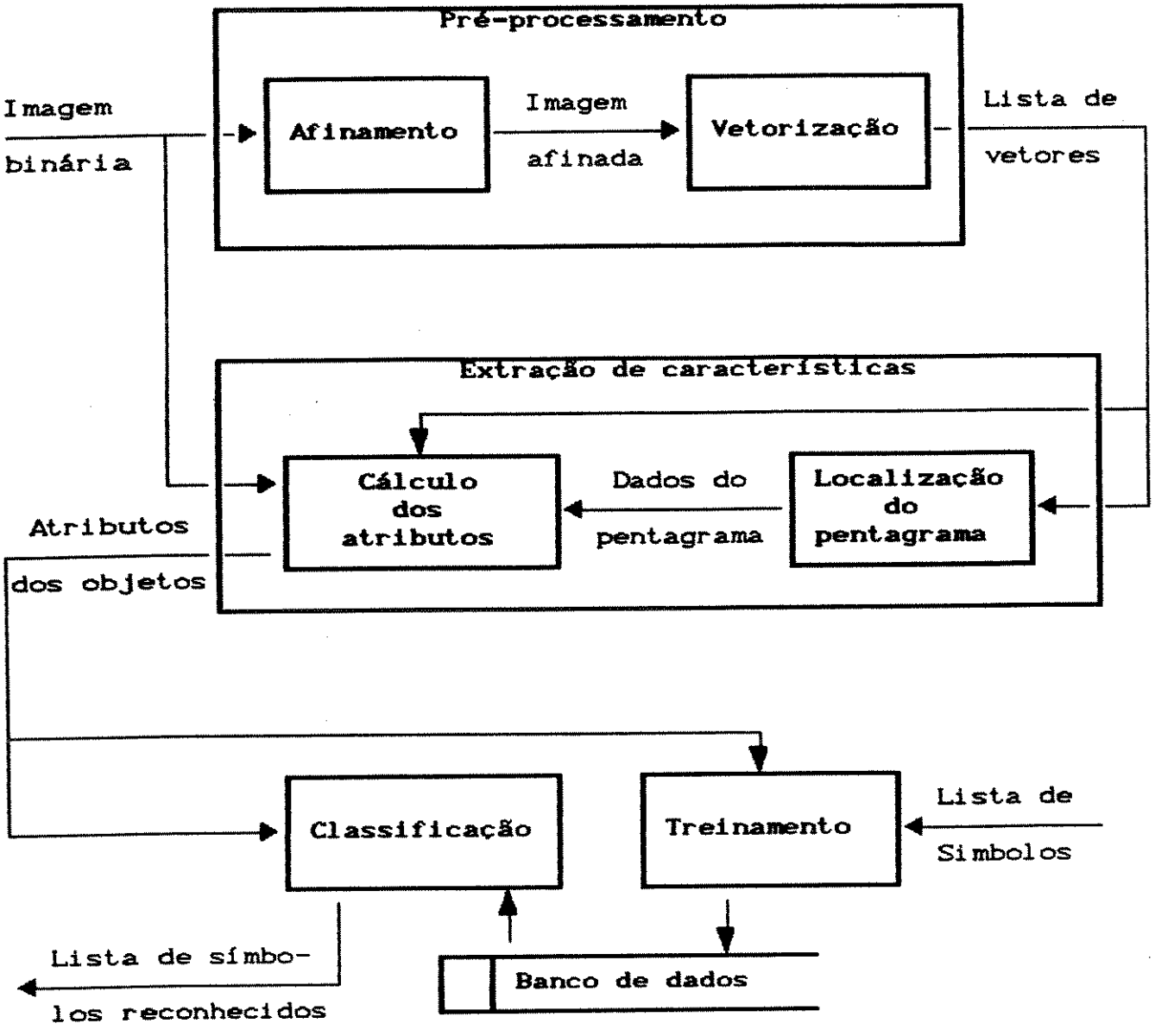


Fig 1 - Diagrama das etapas de processamento do sistema.

1.3.1 Pré-processamento

A fase de pré-processamento inclui a aquisição da imagem, limiarização, corte, afinamento, extração do chain-code, vetorização e filtragem. O objetivo é fornecer ao módulo de extração de características a lista de pontos críticos da imagem.

Afinamento é o processo de redução da espessura dos objetos. Para o afinamento, foi desenvolvido no decorrer deste trabalho um algoritmo inédito baseado no mapa de vizinhança da

imagem. Por ser rápido e paralelo pode ser aproveitado em diversas aplicações. Também prevê parâmetros para ajuste de sensibilidade à ruído. Como o algoritmo não garante linhas com um pixel de largura, um segundo algoritmo é usado para retirar os pixels excedentes.

Depois do afinamento, as linhas do esqueleto são transformadas em segmentos de retas, ou vetores. Para isso foi criado um algoritmo baseado no chain-code do esqueleto. O algoritmo é capaz de detectar as mudanças de direção do chain-code. O processo de vetorização fornece os pontos críticos do esqueleto. Estes pontos serão usados adiante para o reconhecimento dos símbolos.

1.3.2 Localização do pentagrama

A localização do pentagrama consiste em identificar sua posição na imagem e determinar seu espaçamento. A partir da imagem vetorizada, um filtro é usado para obter apenas os vetores aproximadamente horizontais, ou seja, aqueles que têm inclinação próxima de zero. Com esta lista de vetores, o pentagrama é localizado através da projeção horizontal dos vetores e a distância entre as linhas é estimada. A inclinação média do pentagrama também é obtida nesta etapa. Uma vez obtida a distância entre as linhas, todas as medidas na imagem são feitas usando a distância entre as linhas como unidade de comprimento. Desta maneira, elimina-se a necessidade de tratar o fator de escala dos símbolos. A desvantagem é que isso obriga o uso de números com ponto flutuante para armazenar as coordenadas dos pontos e símbolos, o que pode tornar o processamento mais demorado. Para evitar os problemas decorrentes da remoção do pentagrama, ele é mantido na imagem e considerado parte dos símbolos.

Devido à grande extensão das linhas do pentagrama alguns autores usam a projeção horizontal diretamente sobre a imagem binária. O histograma apresenta picos bem definidos nas posições em que as linhas ocorrem. No caso deste sistema, optou-se por projetar os vetores (e não os pixels) no histograma porque o volume de dados diminui consideravelmente.

1.3.3 Cálculo de atributos

Para cada ponto crítico da lista, são calculadas várias funções regionais. Estas funções têm como domínio uma região em torno do ponto que está sendo calculado e levam em consideração a quantidade e posição dos outros pontos dentro da região e a posição, inclinação e comprimento dos vetores. São sempre inversamente proporcionais à distância do ponto considerado de modo a dar maior peso às características mais próximas do ponto. A idéia é associar cada configuração de vizinhança dos pontos a símbolos no banco de dados.

1.3.4 Treinamento e criação do banco de dados

Um banco de dados é criado a partir de imagens acompanhadas da lista de símbolos correspondente construída manualmente. O sistema calcula então os atributos dos pontos da imagem e armazena o resultado no banco de dados.

1.3.5 Reconhecimento

O primeiro nível de reconhecimento é feito associando cada ponto crítico da imagem a vários pontos no banco de dados. A idéia nesta etapa é detectar concentrações de pontos associados ao mesmo símbolo e produzir os retângulos envolventes de candidatos a símbolos. O segundo nível usa a distância dos atributos dos retângulos envolventes aos símbolos no banco de dados para estimar um peso proporcional ao número de pontos críticos que concordam com o candidato a símbolo.

Um terceiro nível de reconhecimento é recomendado para resolver ambiguidades que surgem quando dois candidatos têm peso parecido e estão sobrepostos. Neste nível poderia ser usado conhecimento sobre a ortografia e gramática dos símbolos.

1.4 Organização deste trabalho

No capítulo 2, um algoritmo de afinamento é proposto e descrito. Este algoritmo faz parte de uma das etapas de processamento do sistema. Foi colocado em destaque porque tem aplicação imediata em vários problemas ligados ao reconhecimento de

padrões e absorveu boa parte do tempo empregado na elaboração deste trabalho. O capítulo 3 descreve o processo desenvolvido para a vetorização dos esqueletos. A extração de características é descrita no capítulo 4 e o capítulo 5 apresenta a classificação dos símbolos. O capítulo 6 discute o banco de dados utilizado. A construção de candidatos é explicada no capítulo 7 e o capítulo 8 mostra os resultados obtidos. Finalmente as conclusões são apresentadas no capítulo 9.

Um breve resumo das noções básicas da notação musical foi incluído no Apêndice A. O Apêndice B inclue um resumo dos programas desenvolvidos.

2. Afinamento

Durante a elaboração deste trabalho, vários algoritmos de afinamento foram estudados e cogitados. A pesquisa forneceu base para a elaboração de um novo algoritmo de afinamento. Os resultados iniciais obtidos incentivaram um estudo mais profundo do algoritmo o que permitiu melhorar a eficiência do processo, consumindo porém parte significativa do tempo disponível. Também por ser um algoritmo com aplicações em muitos problemas, foi colocado em destaque em um capítulo a parte. Este capítulo é uma adaptação do artigo publicado em [PER 92] e não exige a leitura dos outros capítulos.

2.1 Introdução

Considerável esforço tem sido empregado na busca de algoritmos de afinamento que sejam rápidos e eficientes [CHI 87, KWO 88, NAC 84]. Em particular os algoritmos paralelos estão merecendo a atenção de um número cada vez maior de autores [DAV 81, ZHA 84, SUZ 87, GUO 89] graças ao crescente uso de sistemas de processamento paralelo aplicado à computação de imagens. Infelizmente não é possível obter algoritmos completamente paralelos utilizando operadores com suporte 3x3 [GUO 89] e vários trabalhos procuram contornar este problema utilizando suportes maiores [CHI 87], passos (sub-ciclos) [ZHA 84, SUZ 87, GUO 89], ou partição da imagem [HAL 89].

O algoritmo apresentado aqui é:

- Totalmente paralelo com suporte 3x3 no processamento do mapa de vizinhança
- Utiliza operadores isotrópicos
- Resulta esqueleto de boa qualidade próximo do eixo mediano
- Possui parâmetros para controlar a sensibilidade a ruído na imagem.

O algoritmo não garante que as linhas do esqueleto tenham sempre um pixel de espessura. Como discutido na seção 2.2, tal garantia implica na perda de isotropia do algoritmo.

2.2 Afinamento e Esqueletos

2.2.1 Conceitos básicos

- A imagem é binária, com pixels tendo valores 0 ou 1.
- Dado o pixel p, um pixel é vizinho-8 de p se tem uma aresta ou um vértice em comum com p (ver figura 4).
- Um pixel é vizinho-4 de p se tem uma aresta em comum com p.
- Um pixel é considerado ativo ou parte do objeto se o seu valor for 1 e desativado ou pertencente ao fundo quando seu valor for zero.
- Considera-se imagem com conexão 8-4, ou seja, os pixels dos objetos são conectados por vizinhança-8 e os pixels do fundo por vizinhança-4.
- Um algoritmo de afinamento é isotrópico se o espelhamento da imagem (vertical, horizontal ou ambos) e rotacões de múltiplos de 90 graus não alteram o esqueleto gerado.

Sinha [SIN 87] define o esqueleto de um caracter como sendo o que resta deste após a remocão da informação sobre sua espessura. Podemos definir o afinamento de imagens como sendo o processo de redução da espessura dos objetos de uma imagem. No afinamento, a idéia é erodir as bordas dos objetos preservando suas propriedades topológicas de conectividade até que sobrem apenas linhas nas medianas.

Segundo Guo e Hall [GUO 89], um algoritmo de afinamento paralelo deve tentar atingir os seguintes objetivos:

- A conectividade dos objetos e do fundo da imagem é preservada no esqueleto.
- As curvas e uniões das curvas do objeto devem estar representadas no esqueleto.
- As curvas do esqueleto devem ser tão finas quanto possível e próximas da mediana das regiões dos objetos.
- O afinamento deve ser realizado no menor número de iterações possível.

O'Gorman [OGO 90] acrescenta ainda que a posição aproximada dos pontos terminais das linhas deve ser mantida.

Vale notar que estes objetivos não são suficientes para definir um único esqueleto para cada objeto possível [NAC 84]. O

fato é que não existe uma definição formal de esqueleto, o que permite variações no esqueleto gerado. Diferentes algoritmos de afinamento aplicados a um mesmo objeto podem produzir esqueletos diferentes e ainda atingir os objetivos acima.

2.2.2 Algoritmos de afinamento

Tem havido uma grande proliferação de algoritmos de afinamento utilizando uma variedade de métodos. De maneira geral, para chegar ao esqueleto, os algoritmos podem corroer a imagem de maneira iterativa [CHI 87, ZHA 84, SUZ 87, GUO 89, HAL 89, PAV 80, KWO 88] ou aplicar em sequência uma série de procedimentos que resultem no esqueleto [DAV 91 , NAC 84, ARC 85].

Os algoritmos de afinamento podem ser classificados em **sequenciais e paralelos**. Nos algoritmos sequenciais, os pixels são processados sequencialmente um de cada vez e a avaliação de cada pixel depende da avaliação do pixel anterior. O volume de processamento necessário para obter o esqueleto é muito menor do que nos algoritmos paralelos por que estes tem que avaliar todos os pixels da imagem a cada iteração. Portando em máquinas sequenciais os estes algoritmos são mais rápidos que os paralelos.

2.2.3 Algoritmos Paralelos

Um algoritmo é paralelo quando o processo de desativação de um pixel não interfere no processo de outros pixels da imagem, podendo um pixel ser avaliado ao mesmo tempo que seus vizinhos. A grande vantagem dos algoritmos paralelos é que cada região da imagem pode ser tratada de forma independente, o que é apropriado a sistemas de processamento paralelo.

O problema de conseguir algoritmos paralelos reside no fato de que para apagar um pixel e ao mesmo tempo manter a conectividade do esqueleto, é preciso conhecimento da vizinhança imediata do pixel.

No exemplo da figura 2, para manter a conectividade do esqueleto, **q** deve ser mantido se **p** for desativado e vice-versa. No caso de algoritmos paralelos, a avaliação de **p** não depende da avaliação de **q** e janelas de vizinhança 3x3 não fornecem informação suficiente para esta decisão.

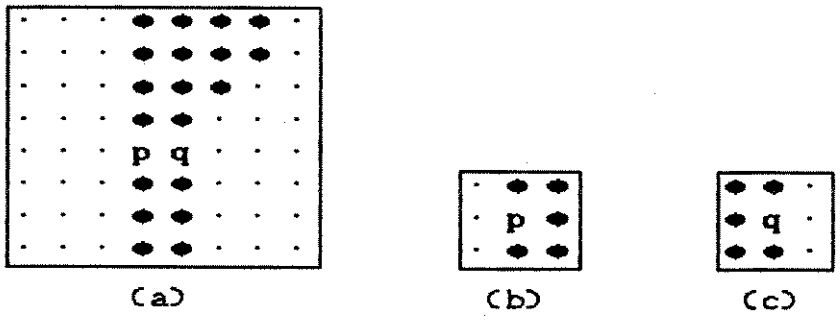


Fig.2 - (a) imagem, (b) e (c) região de 3x3 em torno de p e q respectivamente.

Para resolver esta situação, 3 soluções são encontradas na literatura:

1. Uso de janelas maiores. Alguns algoritmos usam vizinhanças de 3x4, 4x4 ou 5x5 pixels para tomar a decisão. Uma interessante solução proposta por Chin e Wan [CHI 87], usa um conjunto de 8 máscaras 3x3 e mais duas máscaras 4x1.

2. Divisão de cada iteração em passos (em geral 2 ou 4), onde cada passo considera um conjunto diferente de configurações de vizinhança. Um exemplo deste tipo de solução é o popular algoritmo de Zhang/Suen [ZHA 84]. Configurações simétricas como as de p e q, são tratadas em passos diferentes e a conectividade do esqueleto é garantida usando apenas suporte 3x3. Esta solução tem a desvantagem de aumentar o volume de processamento, porque cada passo (ou sub-iteração) de cada iteração exige o processamento de todos os pixels da imagem. Em geral, o tempo de processamento deste tipo de algoritmo aumenta com o aumento do número de passos.

3. Divisão da imagem em dois campos [GUO 89], de forma que um pixel que pertenca ao campo 1, só tenha entre seus vizinhos-4 pixels do campo 2 e vice-versa. As iterações processam alternadamente os pixels de apenas um dos campos. Guo e Hall [GUO 89], provam que este tipo de algoritmo é sempre mais rápido do que os algoritmos que usam passos.

Uma quarta solução usando o mapa de vizinhança da imagem binária é proposta neste trabalho e descrita a seguir.

2.2.4 O algoritmo proposto

A idéia básica é efetuar o afinamento pelo processamento do mapa de vizinhança da imagem. Cada ponto (x,y) no mapa, representa o número de vizinhos da região 3x3 em torno do pixel (x,y) da imagem. Portanto, uma janela de 3x3 no mapa de vizinhança armazena informação sobre uma área de 5x5 pixels. O acréscimo de informação dado a cada ponto no mapa de vizinhança, permite que o algoritmo resolva situações que seriam impossíveis de resolver com uma janela 3x3 em uma imagem binária. Um exemplo é mostrado na figura 3.

Considerando apenas a posição e número de vizinhos na vizinhança 3x3, **p** e **q** têm exatamente a mesma configuração e é impossível que uma regra baseada apenas nessas informações estabeleça qualquer diferença entre eles. No entanto se for considerado o valor dos vizinhos de **p** na vizinhança 3x3 no mapa de vizinhança, a diferença torna-se visível. No exemplo da figura 3, existe um ponto com valor 8 que é vizinho de **p**, garantindo que **p** esteja em uma região com no mínimo 3 pixels de espessura. Já a partir dos valores dos vizinhos de **q**, é possível deduzir (embora não seja necessário) que a região em torno de **q** tem 2 pixels de largura.

Figura 3 - (a) Imagem original, (b) mapa de vizinhança da imagem.

Um esqueleto ideal deveria ter apenas linhas com um pixel de espessura, mas como as imagens são discretas, para conseguir isto em algum momento o algoritmo precisa tomar decisões arbitrárias. Um exemplo típico é mostrado na figura 2. O pixel **p** ou o pixel **q** deve ser desativado e nenhum deles é o ponto médio da região. Qualquer critério usado para escolher um dos dois será arbitrário e nesse momento o esqueleto deixa de ser simétrico e isotrópico. Em geral, nos algoritmos com dois ou quatro passos, esta decisão é

implementada quando é decidido qual será o passo inicial. Embora o algoritmo possa definir uma ordem para os passos, esta será sempre circular e escolher o passo inicial é uma decisão arbitrária.

É importante notar que como o algoritmo proposto utiliza apenas operadores isotrópicos, não garante que o esqueleto tenha apenas linhas com um pixel de largura. Se isto for desejável, outro algoritmo de afinamento pode ser usado para retirar os pixels excedentes. Caso este segundo algoritmo seja iterativo, apenas uma iteração será suficiente. No sistema desenvolvido para o reconhecimento de símbolos musicais é usado como acabamento o algoritmo de 4 passos desenvolvido na UNICAMP pelo Prof. Dr. Clésio Luís Tozzi. Este algoritmo é descrito na seção 2.7.

2.2.5 Sensibilidade a ruídos no contorno

A sensibilidade a ruídos está relacionada com o conjunto de configurações de vizinhança que classificam os pixels como pontos terminais. Quanto maior o número de configurações neste conjunto, maior será o número de pontos terminais no esqueleto. Quanto mais precisamente os pontos terminais representarem as protuberâncias e pontas do objeto, mais o esqueleto estará sujeito a perturbações provocadas por ruído na imagem.

Em uma imagem de baixa resolução, cada pixel é percentualmente muito mais significativo que em uma imagem de alta resolução. Da mesma maneira, cada região do objeto tem importância para o esqueleto proporcional à relação entre a área do objeto e a área da região. Assim, pequenas protuberâncias no contorno do objeto podem ter importância maior ou menor conforme o caso. Por este motivo é desejável que exista uma maneira de ajustar a sensibilidade do algoritmo a pequena modificações no contorno, de modo a permitir uma filtragem de ruídos na imagem.

2.3 Definições e Notação

P_8	P_1	P_2
P_7	P	P_3
P_6	P_5	P_4

$$V_P = [P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8]$$

Figura 4 - Conjunto de vizinhos de um pixel

O conjunto V_p contém todos os vizinhos-8 do pixel **p**.

O mapa de vizinhança é uma matriz de inteiros com as mesmas dimensões que a imagem. Cada ponto no mapa armazena o número de vizinhos do pixel que representa se este for ativo, ou seja, para cada pixel **p**, o mapa armazena **N(p)**. Para pixels desativados o valor no mapa é zero (ver figura 3). O mapa de vizinhança é dado por **N(p)**:

$$N(p) = \begin{cases} \sum_{i=1}^{8} p_i & \text{se } p \neq 0, \quad p_i \in V_p \text{ ou} \\ 0 & \text{se } p = 0 \end{cases} \qquad (1)$$

p_{max} é o valor máximo encontrado nos vizinhos de **p** no mapa de vizinhança.

T (p) retorna o número de objetos 4-conectados em V_p. Também pode ser pensado como sendo o número de transições de fundo para objeto que ocorrem na vizinhança-8 de **p**. **T(p)** é dado por:

$$T(p) = \sum_{i=1}^{8} \begin{cases} 1 & \text{se } p_i \text{ é fundo e } p_{i+1} \text{ é objeto} \\ 0 & \text{em outro caso} \end{cases} \qquad (2)$$

onde $p_9 = p_1$

Quando **T(p) = 1**, então **p** não é o ponto de junção entre duas regiões distintas do objeto e a desativação de **p** não desconecta o esqueleto. Se **T(p) > 1**, é necessário processamento adicional para que a conectividade do esqueleto seja garantida. No caso da figura 5, existem duas regiões distintas do objeto conectadas a **p**. Se **p** for desativado não se garante que o esqueleto se mantenha conectado.

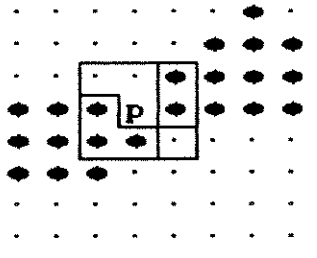
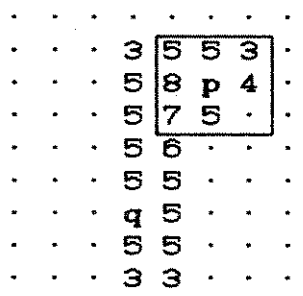


Fig 5 - Exemplo de configuração com T(p) = 2.

O conjunto ordenado H_p está definido apenas quando N(p) = 7. Ele contém todos pontos na vizinhança-8 de **p** no mapa de vizinhança. Em H_p os pontos estão ordenados de maneira que o primeiro elemento do conjunto seja o vizinho desativado de **p**. Os elementos seguintes podem ser encontrados percorrendo a vizinhança de **p** em um sentido arbitrado. Neste trabalho foi escolhido o sentido horário. O iésimo elemento de H_p será chamado de h_i.



$H_p = [\ 0, 5, 7, 8, 5, 5, 3, 4\ ]$

$h_1 = 0$

$h_8 = 4$

Fig. 6 - Exemplo de configuração do conjunto Hp.

2.4 O Algoritmo

Cada iteração testa cada pixel ativo com as condições abaixo. O pixel p é apagado se (I) ou (II) forem satisfeitas:

(I) $Min <= N(p) < 7$ (3)

e

$T(p) = 1$ (4)

e

$p_{max} > Max(\ N(p)+1,\ R\ )$ (5)

onde $1 <= Min <= 4$ e $2 <= R <= 6$.

(II) $N(p) = 7$ e (6)

$h_4 = h_5 = h_6 = 8$ e (7)

$(h_2, h_3) \in [\ (6,8),\ (5,7),\ (4,6),\ (4,5)\ ]$ (8)

ou

$(h_8, h_7) \in [\ (6,8),\ (5,7),\ (4,6),\ (4,5)\ ]$ (9)

onde (h_i, h_j) é um par ordenado.

Quando N(p) < 7, A condição (3) garante que o pixel testado não é interno ao objeto e não é ponto terminal do esqueleto. A condição (4) garante que p não é o ponto de junção entre duas ou mais regiões do objeto. Finalmente a condição (5) testa a espessura da região em torno de p.

É válido supor que um pixel p que está na borda de um objeto, tenha vizinhos mais internos ao objeto do que ele próprio. Significa que deve existir um vizinho v onde N(v) > N(p), ou seja, $p_{max} > N(p)$. Experimentalmente foi escolhida a condição de $p_{max} > N(p) + 1$ para preservar a conectividade do esqueleto.

Os parâmetros **Min** e **R**, permitem controlar a sensibilidade à ruído nas bordas aumentando ou diminuindo o número de configurações de vizinhança que permitem a desativação do pixel.

O parâmetro R é o fator limitante que impede que a erosão seja excessiva. Está relacionado com a espessura da região em torno do pixel em avaliação porque estabelece um valor mínimo para p_{max}.

O valor de **Min** especifica o número mínimo de vizinhos que um pixel deve ter para que possa ser desativado. Se N(p) < **Min**, p é considerado ponto terminal do esqueleto e deve ser mantido. Quanto maior o valor de **Min**, mais pontos terminais surgirão no esqueleto.

Aumentando o valor de **Min**, a sensibilidade do algoritmo aumenta, já que um número menor de configurações será testado para desativação do pixel. A probabilidade de um pixel ser apagado diminui e portanto o número de iterações pode mudar. Para **Min** = 4, o número de pontos terminais no esqueleto é maximo e para **Min** = 1 é mínimo. Valores típicos para **Min** são 3 e 4.

A condição (I) é suficiente para chegar ao esqueleto, porém a inclusão de apagamento do pixel na condição (II) diminui o número iterações do algoritmo. Na condição (II), quando N(p) = 7, a condição (7) seleciona apenas as configurações de vizinhança onde existem 3 pontos com valor 8 nas posições 4, 5 e 6 de **Hp**. Se a configuração satisfaz (6) e (7), então ela tem uma rotação ou inversão que combina com uma das duas máscaras apresentadas na figura 7.

Fig. 7 - Configurações que satisfazem as condições (6) e (7).

As condições (8) e (9) são simétricas, uma é a inversão da outra. Existem 11 configurações possíveis de vizinhança (desconsiderando as rotações) que satisfazem a condição (8). A figura 8 mostra todas as configurações que satisfazem as condições (6), (7) e (8).

A condição de término do algoritmo é que ocorra uma iteração onde nenhum pixel é desativado. No caso particular em que R = 6, o número de pixels com mais de 6 vizinhos pode ser controlado em cada iteração, de forma a evitar a última iteração. O esqueleto não deve conter pontos com 7 ou 8 vizinhos, caso em que pelo menos mais uma iteração é necessária. Por outro lado, para que um pixel seja desativado é necessário que tenha 7 ou 8 vizinhos. O mesmo raciocínio não é válido quando R < 6, porque podem existir pixels onde **pmax** > R que devem ser mantidos.

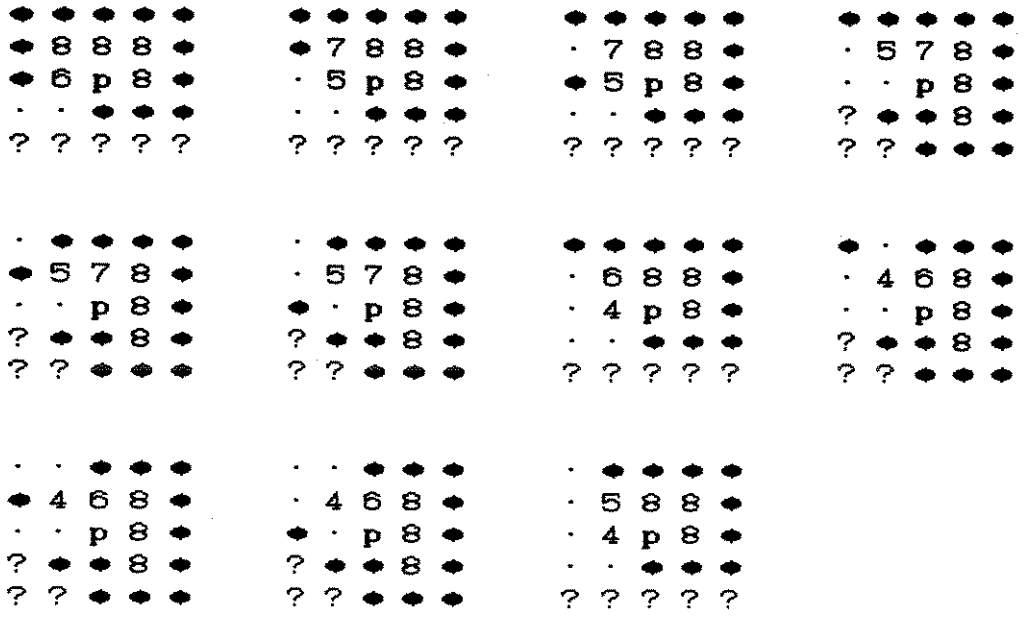


Fig. 8- Configurações de vizinhança que satisfazem (6), (7) e (8).

2.5. Implementação e Experimentos

Algumas implementações de algoritmos de afinamento utilizam o conceito de **look-up table** como forma de reduzir o tempo de processamento da imagem. A look-up table é uma tabela onde estão armazenados dados relativos a cada configuração de vizinhança possível. Em uma janela 3x3, existem 8 vizinhos e portanto 256 configurações diferentes de vizinhança. A posição de cada configuração na tabela é encontrada associando um bit a cada vizinho do pixel considerado e formando um número com os bits. Dados típicos que podem estar armazenados na look-up table são **N(p)** e **T(p)**, mas

conforme a implementação, pode ser viável armazenar outras informações relativas à vizinhança do pixel.

O uso da look-up table diminui o volume de processamento durante o afinamento, mas obriga que todos os vizinhos de cada pixel sejam percorridos para encontrar a posição na tabela. Por este motivo o método é raramente usado. No caso deste algoritmo, seu uso é justificável. Se cada ponto do mapa de vizinhança armazenar em vez de N(p), a posição na look-table, sempre que um pixel p for apagado são percorridos apenas os vizinhos ativos do pixel. Em cada visita, o bit do vizinho que representa o pixel é zerado. Desta maneira a posição na tabela está sempre atualizada e as funções N(p) e T(p) ficam disponíveis sem processamento adicional.

Para comparação com outros algoritmos, a look-up table não foi implementada. Na verdade o mapa de vizinhança já é uma tabela que armazena N(p) de todos os pixels, e sem a necessidade de visitar todos os vizinhos para encontrar a posição que deve ser consultada. Os vizinhos ativos do pixel p, devem ser visitados apenas quando p é desativado afim de manter o mapa de vizinhança atualizado. Para isso, todos vizinhos ativos de p, devem ter seu valor decrementado, já que todos perderam um vizinho.

A função T(p) foi implementada de maneira a retornar um valor booleano. Retorna TRUE quando T(p) = 1 e FALSE em outro caso. Desta maneira não é preciso visitar todos os vizinhos de p todas as vezes que a função é chamada.

A tabela 1 mostra o tempo medido e o número de iterações necessárias para afinar 5 imagens de dimensões diferentes com o algoritmo proposto e o algoritmo de Zhang/Suen.

A coluna do pré-processamento informa o tempo usado para criar o mapa de vizinhança inicial. O mapa é atualizado durante o afinamento. A coluna do afinamento mostra o tempo gasto para executar todas as iterações necessárias. A coluna Total, é a soma do tempo de pré-processamento e do tempo de afinamento. Para o acabamento foram usadas 2 sub-iterações com o algoritmo de Zhang/Suen sobre a imagem afinada. A coluna que mostra as iterações não inclui o acabamento. Todos os programas foram codificados em Turbo C e executados em um AT-386. Os tempos em segundos são a média de várias execuções sobre as imagens.

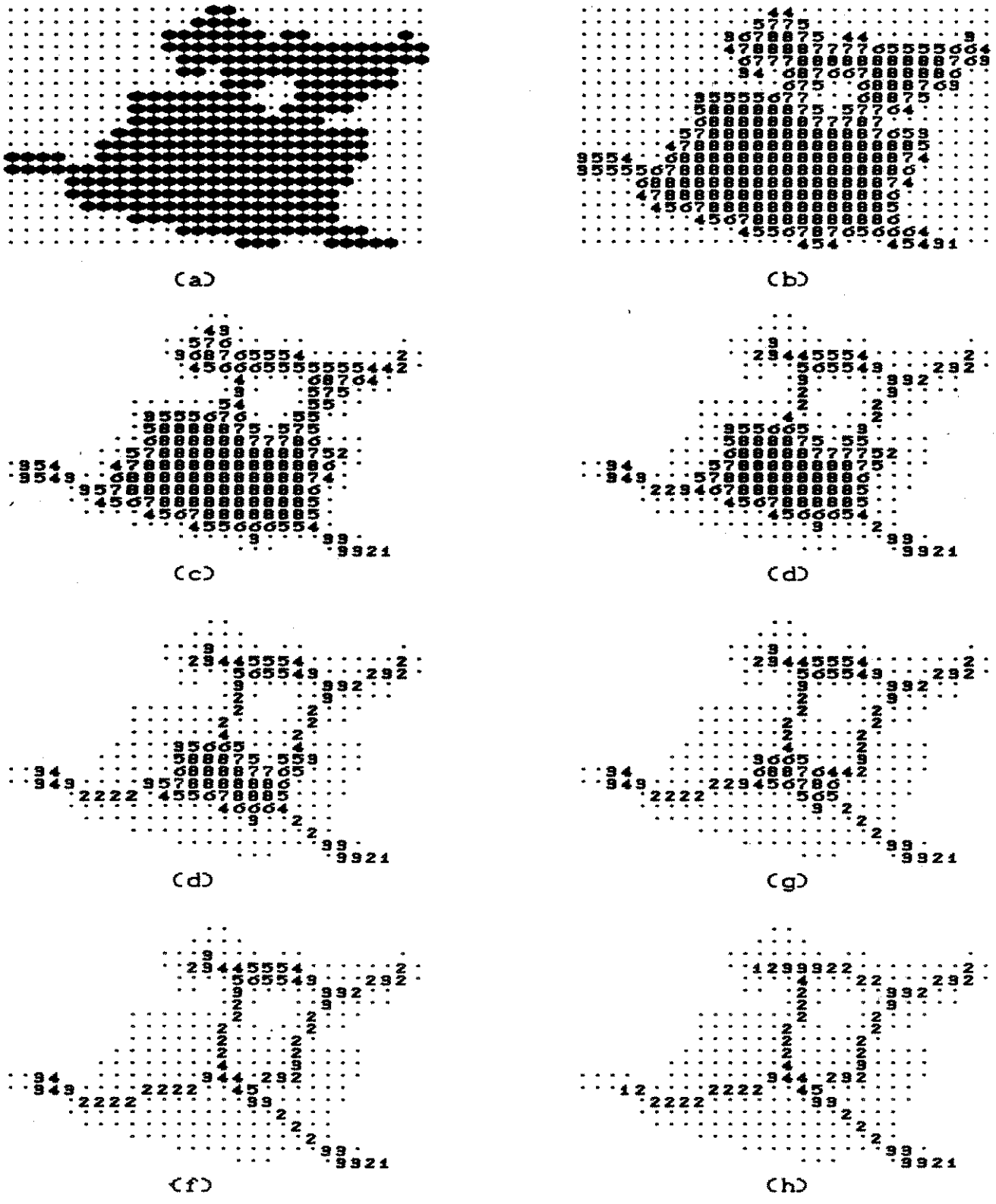


Fig 9- (a) imagem original, (b) mapa de vizinhança, (c) a (g) mapa depois de 1, 2, 3, 4, e 5 iterações respectivamente usando Min = 3 e R = 4. (h) após o acabamento com duas sub-iterações do algoritmo de Zhang/Suen.

A figura 9 mostra os resultados intermediários após cada iteração do algoritmo sobre a imagem 5 da tabela 1. Na figura 10 é apresentado o esqueleto final com acabamento do algoritmo de Zhang/Suen, de todas as combinações válidas dos parâmetros **Min** e **R**,

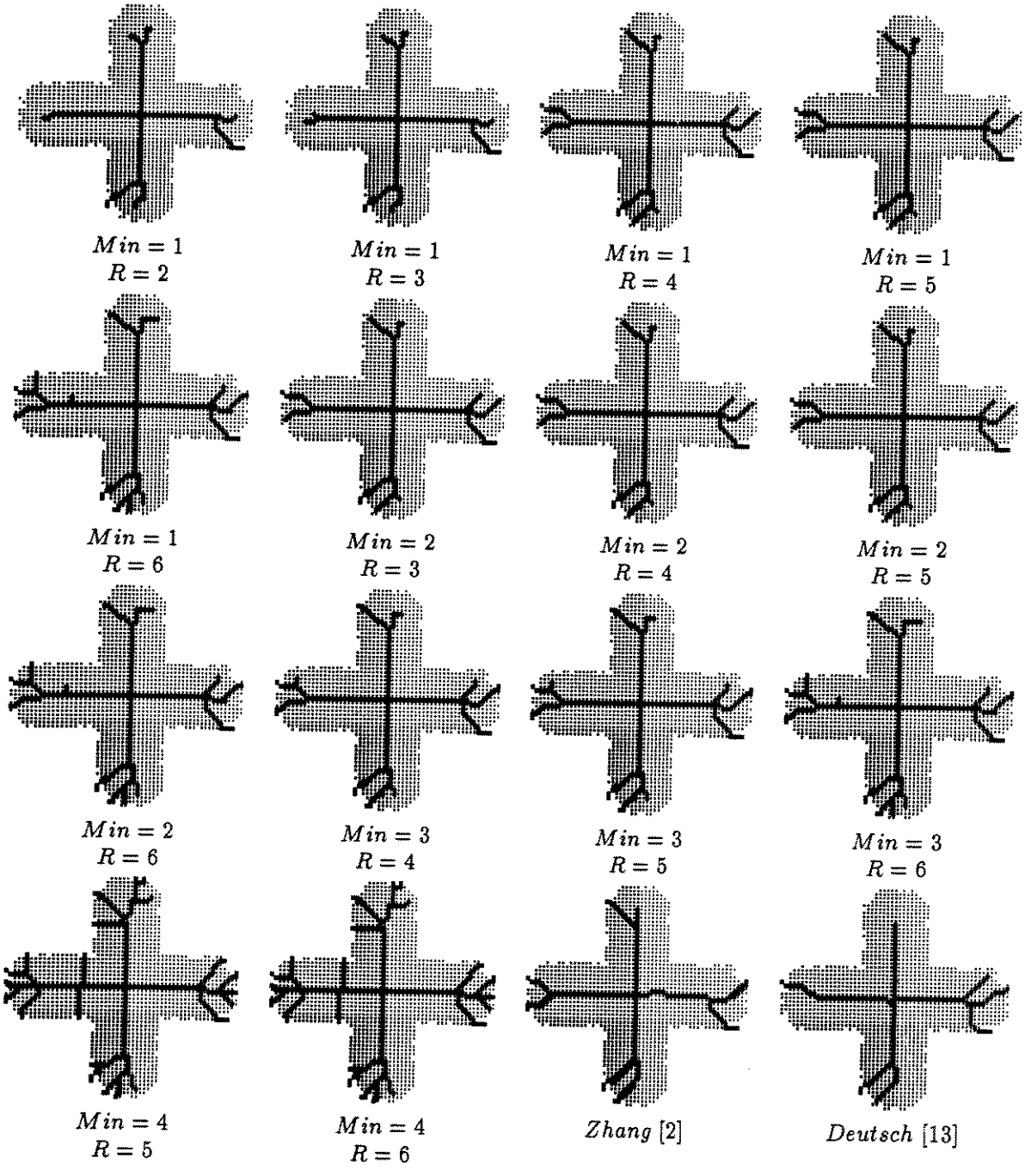


Fig 10 - Exemplo de afinamento da imagem 1 da tabela 1, com todas as combinações válidas dos parâmetros **Min** e **R**, e o resultado obtido com o algoritmo de Zhan/Suen [**ZHA 84**] e o algoritmo de Deutsch [**DEU 72**].

bem como o esqueleto gerado por outros algoritmos de afinamento. O número de pixels apagados em cada combinação dos parâmetros é mostrado na tabela 2. A imagem utilizada para o exemplo da figura 10 é a imagem 1 da tabela 1.

	Algoritmo Proposto — Tempo (s)					Algoritmo de Zhan/Suen	
Imagem	Pré-Pr	Afinam	Total	Acabam	Iter	Tempo	Sub-Iter
1	0.22	1.04	1.26	0.17	9	9.79	17
2	0.33	1.98	2.31	0.22	14	7.32	26
3	0.44	2.41	2.85	0.66	9	4.12	11
4	0.99	6.48	7.47	1.82	9	12.72	10
5	0.06	0.22	0.28	0.06	6	0.17	10
Total	2.04	12.13	14.21	2.93	47	28.06	75

Tabela 1 - Tempo de processamento e número de iterações com várias imagens.

R/Min	1	2	3	4
2	1344	-	-	-
3	1341	1318	-	-
4	1318	1318	1304	-
5	1318	1318	1304	1237
6	1296	1296	1292	1237

Tabela 2. Número de pixels desativados na imagem 1 da tabela 1, usando todas as combinações válidas dos parâmetros Min e R.

2.6. Conclusões

Este trabalho apresenta um novo algoritmo de afinamento que introduz o conceito de processamento de suporte 3x3 no mapa de vizinhança. Trabalha em um passo porque cada ponto no mapa fornece uma quantidade de informação maior do que um pixel em uma imagem binária poderia fornecer. Processar o mapa de vizinhança em uma janela 3x3 é equivalente a dispor de um suporte 5x5 na imagem binária. Isto permite que o algoritmo chegue a mais conclusões a respeito da região considerada tornando o processamento mais inteligente e eficiente.

O algoritmo proposto tem a vantagem de ser simples em conceito, embora sua implementação em sistemas paralelos seja mais complexa que os outros algoritmos porque são necessários mais canais de comunicação entre os processadores. É isotrópico, já que nenhuma relação leva em consideração a orientação das configurações de vizinhança. Possui parâmetros que permitem ajustar a sensibilidade do esqueleto à ruidos no contorno. Finalmente é rápido e produz esqueletos de boa qualidade.

2.7 O algoritmo de acabamento

Como citado, o esqueleto gerado pelo algoritmo proposto não garante linhas com um pixel de espessura. Na verdade podem ocorrer regiões com até 3 pixels de espessura, o que cria uma série de situações ambíguas durante a extração do chain-code e vetorização. Para contornar esses problemas, é usado um segundo algoritmo que retira os pixels excedentes do esqueleto. Os algoritmos de dois passos também não garantem linhas de um pixel, embora gerem esqueletos mais "finos" que o algoritmo proposto. Apenas com os algoritmos de 4 passos é possível esperar linhas realmente "finas". Este algoritmos afinam a imagem com mais cuidado gerando esqueletos de ótima qualidade porém com um custo computacional extremamente elevado. O uso combinado do algoritmo proposto com um algoritmo de 4 passos aproveita as vantagens dos dois processos. O "grosso" da imagem é erodido rapidamente pelo algoritmo de um passo, e a seguir o algoritmo de 4 passos se encarrega de retirar os pixels excedentes. Apenas uma iteração (com os 4 passos) é suficiente para garantir linhas com um pixel de espessura. O tempo de processamento total para a execução dos dois algoritmos ainda é significativamente menor que o tempo de processamento dos algoritmos de 2 passos.

O algoritmo descrito a seguir foi escolhido para o acabamento do esqueleto. Trabalha em 4 passos e foi desenvolvido pelo Prof. Dr. Clésio Luís Tozzi na UNICAMP.

2.7.1 Descrição do algoritmo de acabamento

Este algoritmo é paralelo e corroi os objetos da imagem iterativamente. Cada iteração é dividida em 4 passos e cada passo considera para desativação um subconjunto das configurações de vizinhança dos pixels nas bordas dos objetos. Isso também é verdade nos algoritmos de 2 passos, a diferença é que os algoritmos de 4 passos classificam as configurações em 4 subconjuntos, permitindo um controle mais preciso da erosão. Graças a esta precisão, este algoritmo admite que pixels com $T(p) > 1$ sejam considerados passíveis de erosão, o que não é comum nos algoritmos de 2 ou 1 passos.

Considerando a vizinhança-8 do pixel **p** na imagem binária encontramos 8 pixels conectados a **p**. Estes pixels serão nomeados de acordo com os pontos cardeais como ilustrado abaixo:

NW	N	NE
W	p	E
SW	S	SE

Os pixels são classificados de acordo com sua posição relativa à borda do objeto. Os que tem seu vizinho Norte zerado são considerados pixels da fronteira Norte. Os que tem o vizinho Sul zerado são pixels da fronteira Sul, e assim por diante. Um pixel pode fazer parte de mais de uma fronteira. Em cada passo são desativados apenas os pixels de uma das fronteiras.

Cada passo processa todos os pixels da imagem e desativa aqueles que satisfazem as condições do passo que estiver sendo executado.

Passo Norte:

$$N = 0 \qquad e$$
$$(NE \cdot \overline{E}) + (NW \cdot \overline{W}) = 0 \qquad e$$
$$\overline{S} \cdot (E + SE) \cdot (W + SW) = 0$$

Passo Sul:

$$S = 0 \qquad e$$
$$(SW \cdot \overline{W}) + (SE \cdot \overline{E}) = 0 \qquad e$$
$$\overline{N} \cdot (W + NW) \cdot (E + NE) = 0$$

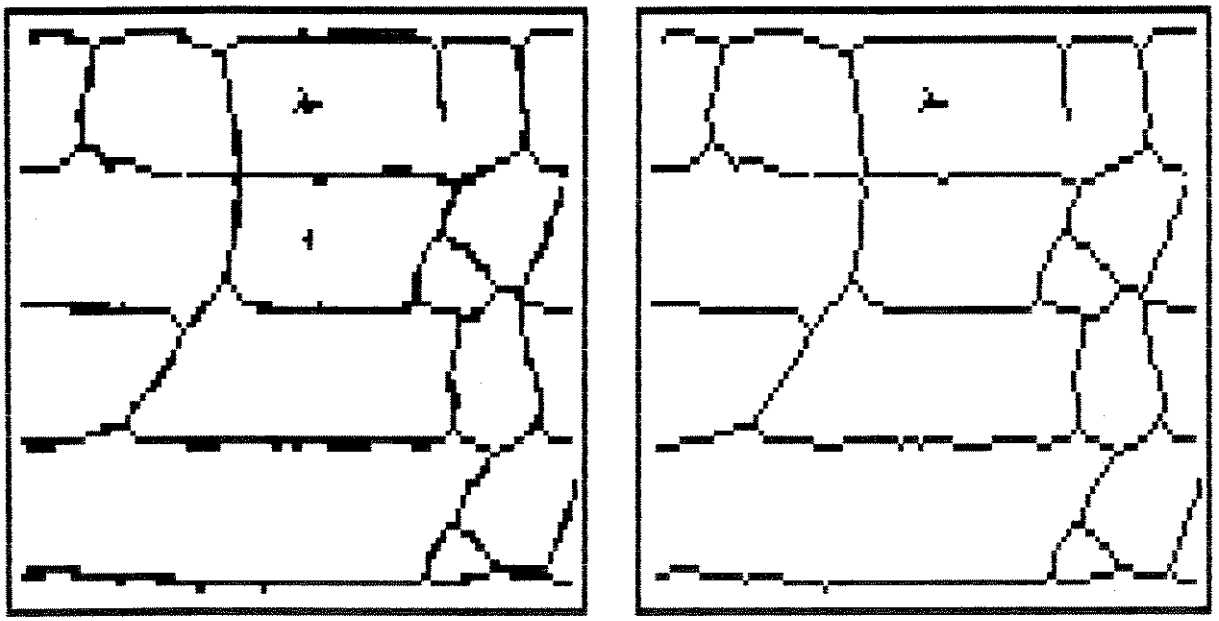
Passo Leste:

$$L = 0 \quad e$$
$$(SE \cdot \overline{S}) + (NE \cdot \overline{N}) = 0 \quad e$$
$$\overline{W} \cdot (S + SW) \cdot (N + NW) = 0$$

Passo Oeste:

$$W = 0 \quad e$$
$$(NW \cdot \overline{N}) + (SW \cdot \overline{S}) = 0 \quad e$$
$$\overline{E} \cdot (N + NE) \cdot (S + SE) = 0$$

A condição de término é que ocorra uma iteração em que nenhum pixel é desativado. A ordem sugerida pelo autor para a execução dos passos é a seguinte: Leste, Norte, Sul e Oeste.



a) b)

Fig 11 - Exemplo de uso do algoritmo de acabamento. a) esqueleto gerado pelo algoritmo de 1 passo proposto. b) após o processamento do algoritmo de 4 passos descrito.

3. Vetorização

Vetorizar é entendido aqui como o processo de aproximar as curvas do esqueleto por segmentos de retas. O objetivo principal da vetorização neste trabalho é obter os pontos críticos da imagem (pontos terminais, de cruzamento e inflexão), mas já que a imagem é vetorizada, o pentagrama também é localizado através dos vetores. Os pontos terminais e de cruzamento são facilmente identificados pela função **T(p)** definida na seção **2.3**. Os pontos de mudança de direção (inflexão) são encontrados pelo algoritmo proposto neste capitulo. O algoritmo desenvolvido trabalha sobre o chain-code das linhas, e este é extraído processando um mapa da função **T(p)**. É pré-requisito para o processo que o esqueleto não contenha regiões espessas para que não surjam situações ambíguas. A situação ideal é que todas as linhas tenham apenas um pixel de espessura.

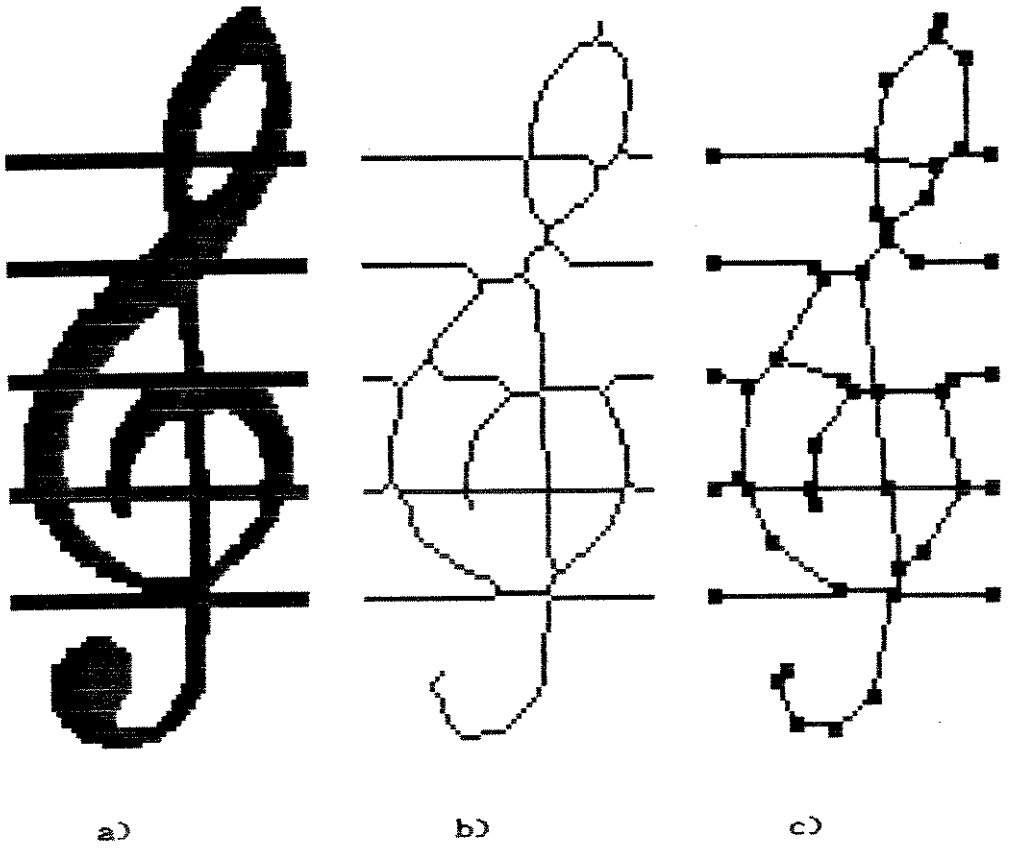


Fig 12 - a) Imagem original, b) Imagem afinada, c) imagem vetorizada.

3.1 Definições

- **T(p)** é o número de objetos 4-conectados a **p** (ver seção 2.3).
- **Nv(p)** é o número de vetores que partem do ponto **p**.
- **Pontos de cruzamento** são pontos onde as linhas do esqueleto se cruzam. Na imagem afinada os pontos de cruzamento tem **T(p)** > 2. Na imagem vetorizada são os pontos onde **Nv(P)** > 2.
- **Pontos terminais** são os pontos onde as linhas do esqueleto terminam. **T(p)** = 1 na imagem afinada ou **Nv(P)** = 1 na imagem vetorizada.
- **Pontos intermediários** são definidos apenas na imagem vetorizada. São aqueles em que **Nv(P)** = 2. Correspondem aos pontos de inflexão na imagem afinada.
- **Pontos críticos** são os pontos terminais e de cruzamento.
- **Membro** de esqueleto é uma sequência de segmentos de reta conectados que tem suas duas extremidades em pontos críticos que podem ser distintos ou não. Se o membro tem as duas extremidades no mesmo ponto crítico, é considerado um membro circular.

3.2 Extração do chain-code

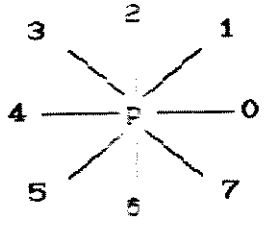
[CHA 84] define um contorno digital fechado como uma cadeia de pontos tais que se um ponto pertence ao contorno, ele tem exatamente dois pontos em sua vizinhança que também pertencem ao contorno. Nos contornos abertos esta condição é violada pela existência dos pontos terminais que tem um vizinho no contorno. O chain-code consiste em uma cadeia de pequenos vetores que podem assumir um número limitado de direções (8 neste trabalho) e são usados são normalmente usados para representar contornos e linhas. Os vetores são usados para representar o deslocamento de um ponto do contorno até o próximo.

As linhas do esqueleto tem suas extremidades em pontos terminais e se cruzam nos pontos de cruzamento. Tais pontos precisam ser identificados antes de iniciar o processo porque o chain-code será sempre extraído de um ponto crítico (terminal ou de cruzamento nesta etapa) até outro. Após a extração do trecho entre dois pontos, o chain-code é vetorizado e os pontos intermediários encontrados. Devido a complexidade de preservar a conectividade do chain-code de toda a imagem, a vetorização e a extração do chain-code são feitas simultaneamente. Mesmo assim foi feito um esforço para preservar a

conectividade dos membros e uma extensa estrutura de dados foi implementada nos programas com esse objetivo.

O algoritmo desenvolvido para a extração do chain-code, processa um mapa da função **T(p)**. Cada pixel **p** da imagem é representado no mapa por **T(p)**. Desta maneira os pontos críticos são facilmente identificados quando processados. A função **T(p)** neste caso indica quantos caminhos podem ser seguidos a partir de cada pixel do esqueleto.

As 8 direções do chain-code serão convencionadas conforme a figura abaixo. O algoritmo localiza os pontos terminais e de cruzamento da imagem e percorre as linhas entre cada par de pontos conectados gerando o chain-code do trecho percorrido. Cada trecho é vetorizado logo após a extração e se transforma em um membro.

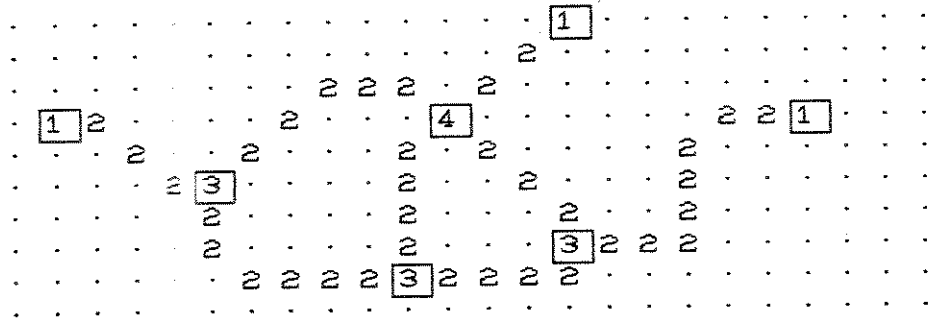


Direções convencionadas para o chain-code.

Depois de encontrado um ponto inicial, a vizinhança-8 do pixel é percorrida em um sentido arbitrado (sentido horário no sistema) em busca de um vizinho ativo, que satisfazendo as condições do algoritmo, será o próximo ponto no chain-code.

Os pontos no mapa são decrementados a medida que são percorridos. O decremento ocorre quando o algoritmo chega em um ponto e também quando sai dele. Os pontos críticos, que iniciam e terminam os trechos extraídos são decrementados apenas uma vez em cada trecho e os outros pontos 2 vezes. O processo termina quando todos os pontos do mapa estiverem zerados.

a)



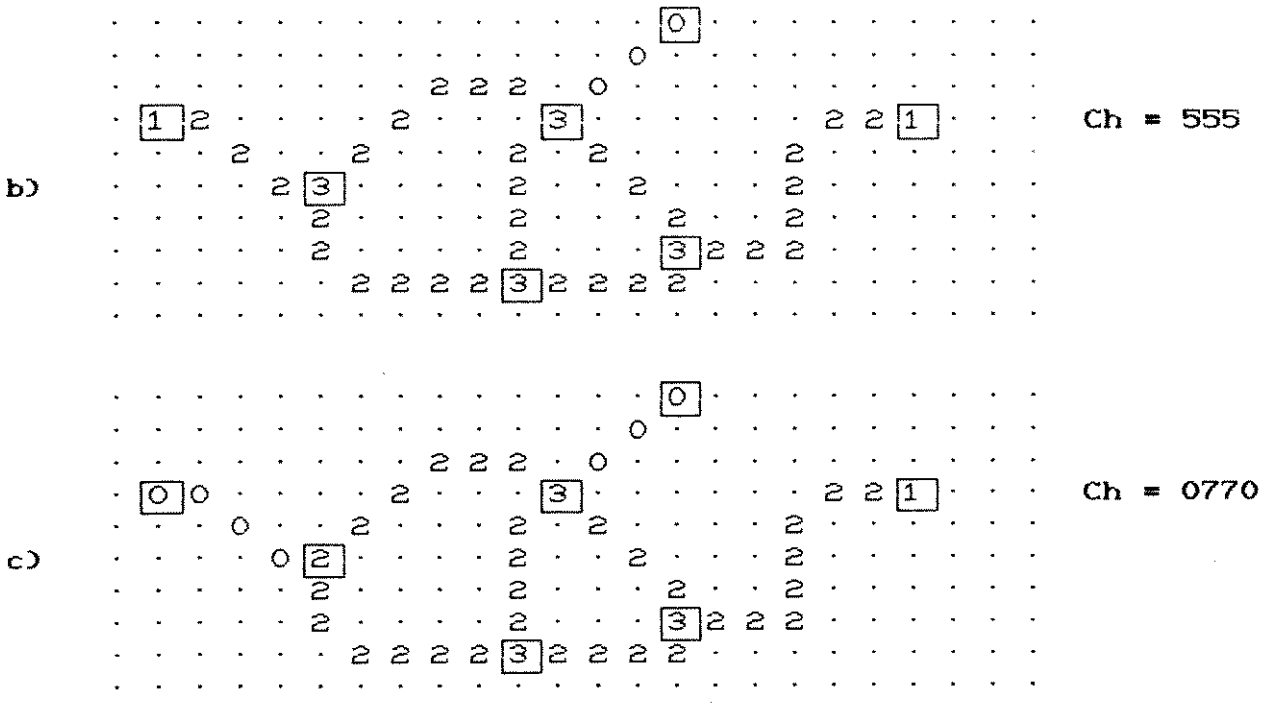


Fig 13 - a) mapa de T(p). Os pontos críticos estão em destaque. b) 1º trecho extraído, c) 2º trecho extraído. Ch é o chain-code obtido.

Para evitar que um ponto crítico que foi decrementado seja confundido com os pontos não-críticos (exemplo na fig 13c), uma lista de pontos críticos deve ser montada antes de iniciar o processo. Os pontos iniciais do chain-code podem então ser obtidos na lista. Pode ocorrer que um objeto circular não tenha pontos críticos. Para que estes objetos não sejam ignorados, depois de zerados todos os pontos da lista, é feita uma busca no mapa. Se existir algum ponto ativo terá o valor 2, pois caso contrário constaria na lista de pontos críticos. Este ponto é colocado na lista e o trecho que inicia (e certamente termina) é extraído. Este processo é repetido até que o mapa esteja completamente zerado.

O procedimento para extrair as direções do chain-code é apresentado abaixo.

Seja **D** uma direção válida para o chain-code com valor **n**, **i** um inteiro tal que $0 \leq i < 8$ e $p(x,y) = T(p)$ um ponto no mapa. Para cada valor de **n** é associado um deslocamento (x_n, y_n). Ch_i é a iésima direção do chain-code **Ch**. Nestes têrmos são definidas as seguintes operações:

(1) $D + i = (n + i) \bmod 8$

(2) $D - i = (n + 8 - i) \bmod 8$

(3) $-D = (n + 4) \bmod 8$

(4) $p + D = p + (x_n, y_n)$

Tab de deslocamento

n	(x_n, y_n)
0	(1, 0)
1	(1,-1)
2	(0,-1)
3	(-1,-1)
4	(-1, 0)
5	(-1, 1)
6	(0, 1)
7	(1, 1)

1) Montar uma lista L com todos os pontos críticos da imagem. Cada elemento da lista armazena a posição e o valor de um ponto crítico no mapa.

2) Encontrar um ponto p na lista cujo valor não seja zero. Este será o ponto inicial do trecho. Em seguida, uma direção inicial D é escolhida para o teste da primeira iteração do passo 3. A direção inicial é escolhida de modo que aponte para um pixel zerado (T(p+D) = 0). Se todos os pontos da lista estiverem zerados, passar para o passo 4.

3) i := 0

Repetir:

{

Se (p + D) = 0 então D := D - 1 *(sentido horário)*

senão

{

Se (D é ímpar) e (p + (D-1) ≠ 0)

então D := D - 1;

Ch_i := D

p(x,y) := p(x,y) - 1 (*) *(decrementar p(x,y) no mapa e na lista)*

p := p + D

p(x,y) := p(x,y) - 1 (*)

D := -D - 1

i := i + 1

}

}

até que p(x,y) pertença à lista L

Ch agora contém o chain-code entre dois pontos críticos e pode ser usado ou armazenado para uso futuro. No sistema desenvolvido, o chain-code é vetorizado e Ch é liberado para o próximo trecho. i indica quantas direções estão armazenadas em Ch.

Retornar ao passo 2.

4) Fazer uma busca no mapa. Se for encontrado algum ponto não zerado, este ponto é incluído na lista L e passa a ser considerado ponto crítico. Em seguida retornar ao passo 2. Se todos os pontos no mapa estão zerados, significa que todas as linhas do esqueleto já foram processadas e o processo termina.

3.3 O algoritmo proposto

Após a extração do chain-code, o problema é determinar em que posições do chain-code devem ser marcados os pontos que serão as extremidades dos vetores que conectam os pontos críticos. Para este processo foi criado um algoritmo que identifica os pontos de duas maneiras: através de retas de controle e da detecção de retorno do chain-code.

As retas de controle são concorrentes no ponto inicial do chain-code. Quando este atravessa uma das retas, o ponto de travessia é marcado e é gerado um vetor do ponto inicial até o ponto imediatamente anterior ao da travessia. O ponto de travessia passa a ser o início de um novo vetor e o processo recomeça. As retas são definidas por dois pontos de apoio colocados de forma a deixar o chain-code entre eles. Estes pontos, periodicamente são trazidos para perto do chain-code fazendo com que o ângulo entre as retas fique cada vez mais agudo a medida que o chain-code se afasta do ponto inicial.

Pode-se visualizar graficamente o algoritmo da seguinte maneira: cada direção do chain-code define um quadrado que contêm a posição da direção em sua borda e tem o centro no início do chain-code. Se ao longo do chain-code o quadrado diminui de tamanho significa que houve retorno. O chain-code deve também se manter entre as intersecções das retas de controle com a borda do quadrado. Poderia ser usado o círculo em vez de o quadrado. Pode-se imaginar que os resultados seriam mais exatos, mas o custo computacional seria muito mais elevado.

Para detectar o retorno do chain-code, são definidas 3 direções de retorno que dependem da direção inicial. Se o chain-code inicia por exemplo com a direção 0 (esquerda para a direita) a direção 4 (direita para a esquerda) é considerada uma direção de retorno e se for encontrada durante o processamento, o ponto anterior

será marcado como extremidade do vetor. Para cada direção inicial são associadas 3 direções de retorno.

O espaço do chain-code é o plano cartesiano com origem em seu ponto inicial. Cada direção do chain-code define um único quadrado com centro na origem e com uma de suas arestas ou vértices sobre a posição da direção. Os pontos de apoio são colocados sobre as arestas do quadrado de maneira a deixar o chain-code entre eles. A distância entre os pontos de apoio e o chain-code é o parâmetro que ajusta a sensibilidade do algoritmo.

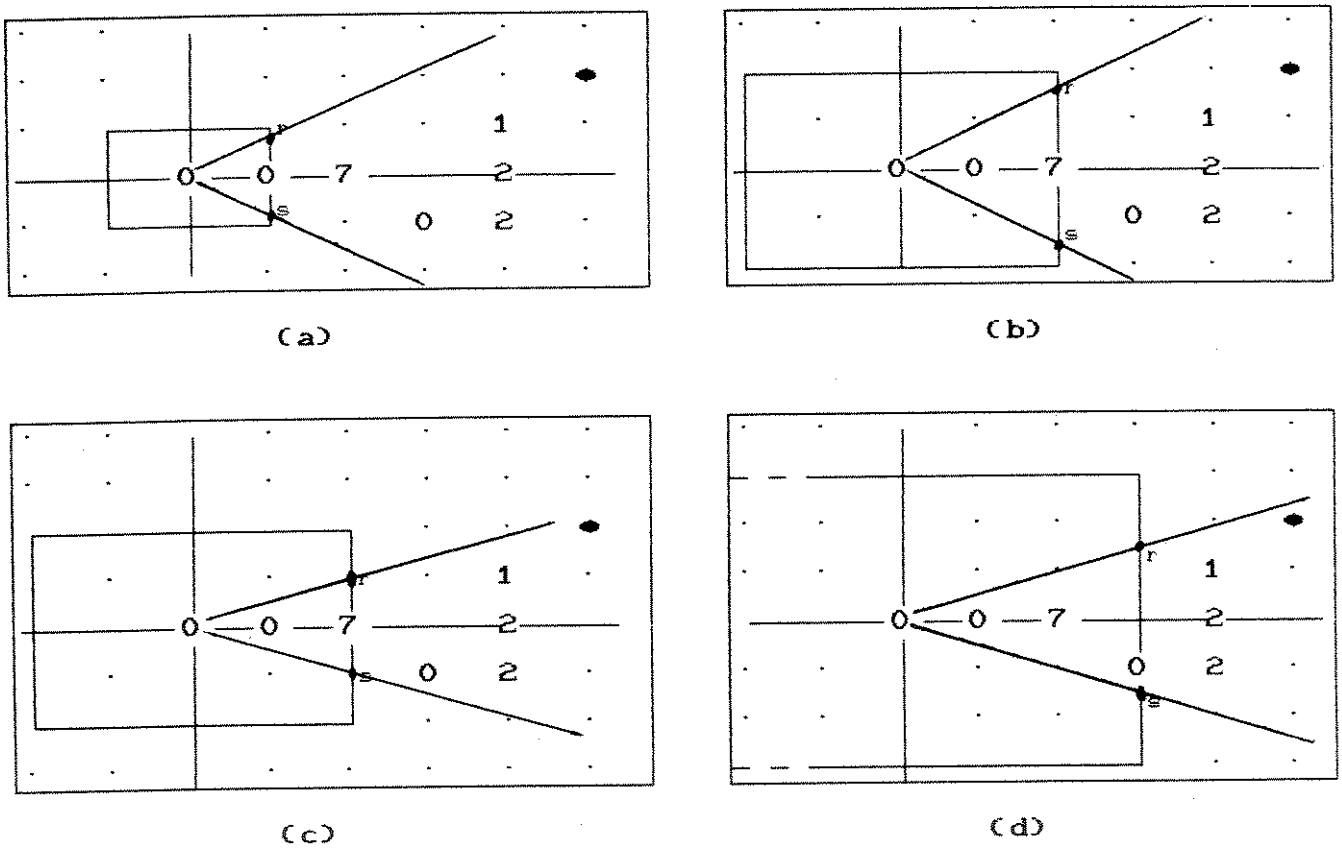


Fig 14 - a) Configuração inicial. Os pontos de apoio r e s são posicionados na borda do quadrado a uma distância dada do chain-code. b) Processamento da 2ª direção. As retas definidas pelos pontos de apoio são projetadas no novo quadrado definindo as novas posições dos pontos. c) Os pontos de apoio aproximam-se do chain-code fechando as retas de controle. d) A posição do chain-code é testada usando as novas posições dos pontos r e s.

Depois de definida a posição dos pontos de apoio e por consequência das retas de controle, a cada direção apontada pelo chain- code os pontos são deslocados sobre as retas de maneira que fiquem sobre as arestas do quadrado definido pelo chain-code no momento.

Em intervalos crescentes, os pontos de apoio são trazidos a uma distância ξ do chain-code. O valor de ξ determina a abertura das retas, e portanto a sensibilidade do algoritmo. A faixa entre 0.5 e 1.5 parece suficiente para a maior parte das aplicações. No caso deste trabalho, como não são esperadas retas muito longas e a resolução das imagens é sempre parecida, o valor 0.75 deu bons resultados.

Trazer os pontos de apoio para a vizinhança do chain-code significa fechar o ângulo entre as retas de controle. Isto é feito após um número determinado de iterações. Este número não é constante e deve crescer a medida que o chain-code se afasta do ponto inicial. Para este trabalho foram usadas as potências de 2 (1,2,4,8,16,...), mas se forem esperadas retas muito longas ou se a imagem for de resolução muito alta, estes valores podem ser inadequados.

Cada direção **D** do chain-code é um vetor que tem uma origem (x_i, y_i) e aponta para uma posição (x_f, y_f). O pontos de apoio **r** e **s** são posicionados inicialmente alinhados com (x_f, y_f) da primeira direção do chain-code.

A primeira direção do chain-code determina a posição inicial dos pontos de apoio **r** e **s**. Determina também as direções que serão consideradas de retorno. Para o chain-code que inicia com a direção **D**, são consideradas direções de retorno as direções **-D-1**, **-D** e **-D+1**.

A cada iteração **i** uma direção Ch_i do chain-code é processada executando os seguintes passos:

Seja $q_i(x_i, y_i)$ a origem do vetor definido pela direção Ch_i.

1) **Se Ch_i é a última direção do chain-code passar para o passo 7.**

2) **Se Ch_i é direção de retorno passar para o passo 7.**

3) **Deslocar os pontos de apoio até as arestas do quadrado definido por Ch_i.** Na prática isto é feito deslocando os pontos pelas retas que já definem até que uma das coordenadas do ponto (**x** ou **y** conforme a aresta do quadrado) coincida com a coordenada correspondente de q_i.

4) **Se q_i não está entre r e s passar para o passo 7.**

5) **Se i é potência de 2, posicionar r e s.** Os pontos de apoio são colocados na borda do quadrado à distância ξ de q_i, um de cada lado.

6) **Obter a próxima direção do chain-code.**
Passar para o passo 1.

7) **Trecho vetorizado.** Um vetor é definido para o trecho processado do chain-code. Se Ch_i é a última direção do chain-code, então q_i é extremidade do vetor e o processo termina. Se não, então q_{i-1} será a extremidade do vetor e q_i será a primeira direção para o próximo vetor.

3.4 Membros

A saída do processo de vetorização é a lista de membros que representam os objetos vetorizados. Cada membro é representado pela lista dos pontos extremos dos segmentos que o compõe. Stein e Medioni [STE 92] descrevem uma estrutura bastante parecida a que chamam de "super-segmento". Exemplos de membros são mostrados na figura 12. Para uma lista **Mb** temos:

$$\mathbf{Mb} = \{ m_i : i = 1, \ldots, Nmb \}$$

onde cada membro em **Mb** consiste em um conjunto ordenado de pontos:

$$m = \{ p_j = (x_j, y_j) : j = 1, \ldots, Npt \}$$

Dois pontos consecutivos definem um vetor $v_j = \overline{p_j\ p_{j+1}}$ cujo comprimento Tm_j é dado por:

$$dx_j = x_{j+1} - x_j$$

$$dy_j = y_{j+1} - y_j$$

$$Tm_j = \sqrt{dx_j^2 + dy_j^2}$$

O ângulo de inclinação α_j do vetor v_j pode ser calculado por:

$$\alpha_j = \begin{cases} \Pi/2 & \text{se } dx_j = 0 \text{ e } dy_j > 0 \\ 3\Pi/2 & \text{se } dx_j = 0 \text{ e } dy_j < 0 \\ \text{ArcTan}(dy_j / dx_j) & \text{se } dx_j \neq 0 \end{cases}$$

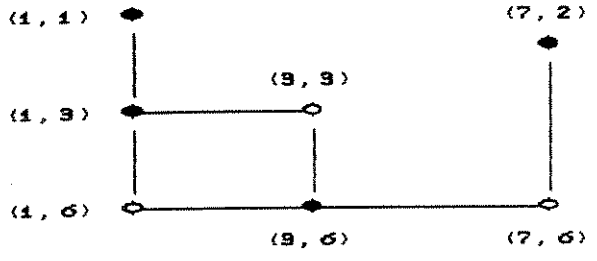
para $dx_j \neq 0$ ou $dy_j \neq 0$

A_i é a projeçao de α_i no primeiro quadrante do círculo trigonométrico.

$$A_i = \begin{cases} \alpha_i & \text{se} \quad 0 \le \alpha_i < \frac{\Pi}{2} \\ | \alpha_i - \Pi | & \text{se} \quad \frac{\Pi}{2} \le \alpha_i < \Pi \\ \alpha_i - \Pi & \text{se} \quad \Pi \le \alpha_i < \frac{3\Pi}{2} \\ | \alpha - 2\Pi | & \text{se} \quad \frac{3\Pi}{2} \le \alpha_i \le 2\Pi \end{cases}$$

O perímetro Pm_i do membro m_i é a soma dos tamanhos dos vetores que compõe m_i.

$$Pm_i = \sum_{j=1}^{Npt_i} Tm_j$$



Membro	Lista de pontos
0	(1,1) , (1,3)
1	(1,3) , (3,3) , (3,6)
2	(1,3) , (1,6) , (3,6)
3	(3,6) , (7,6) , (7,2)

Fig 15 - Exemplo de saída da vetorização de um objeto. As bolinhas cheias representam pontos de críticos e as vazadas pontos intermediários.

3.5 Filtro

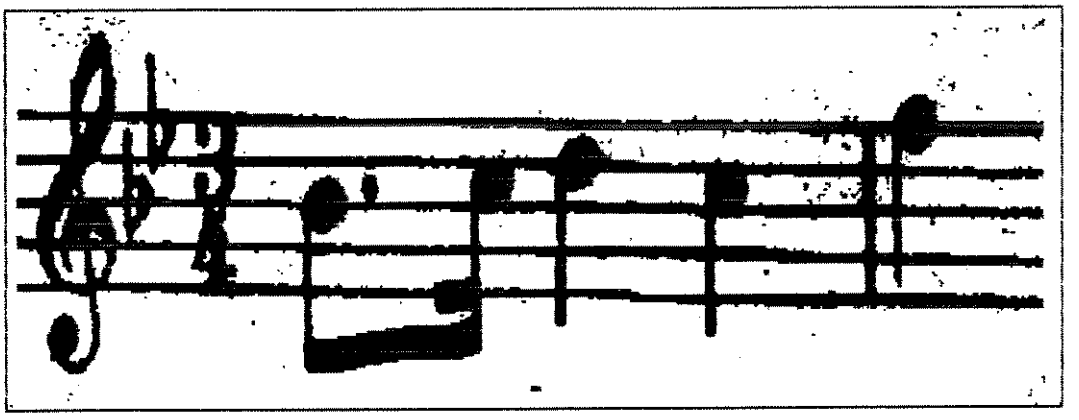
Durante o processo de aquisição, é comum o aparecimento de ruído em algumas regiões da imagem. Boa parte do ruído assume a forma de nuvens de pontos. Os pontos isolados desaparecem durante a vetorização, mas os pontos que estão conectados a outros pontos da nuvem geram uma nuvem de pequenos vetores. É vantajoso para o reconhecimento que tais vetores sejam retirados da lista de vetores porque não representam informação relevante. O filtro usado para este processo deleta todos os membros formados por um único vetor, cujo módulo é menor do que o valor mínimo permitido, e que não está conectado a outros membros. A lista de membros resultante será usada em várias etapas do processamento e será chamada de **Mb**.

Seja **tmp** o tamanho mínimo permitido. Um membro é retirado da lista se:

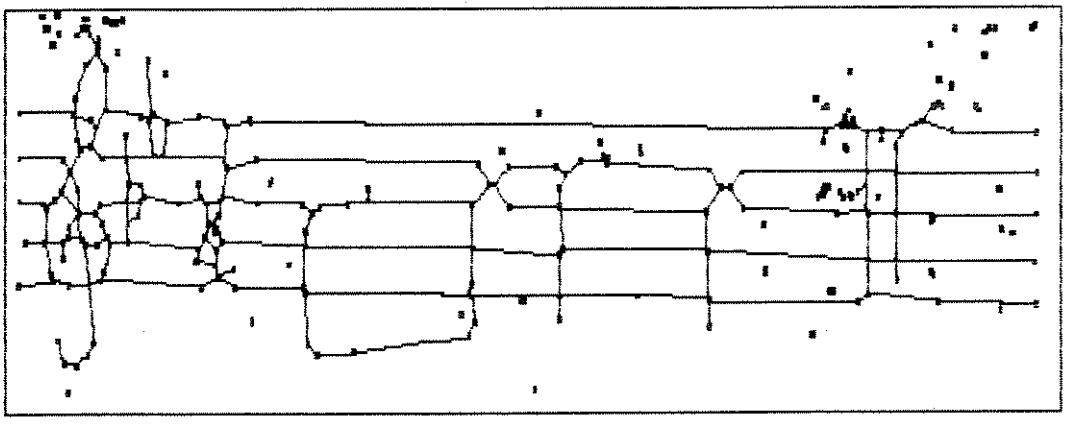
I) $Npt = 2$ e

II) $Nv(P_1) = 1$ e $Nv(P_2) = 1$ e

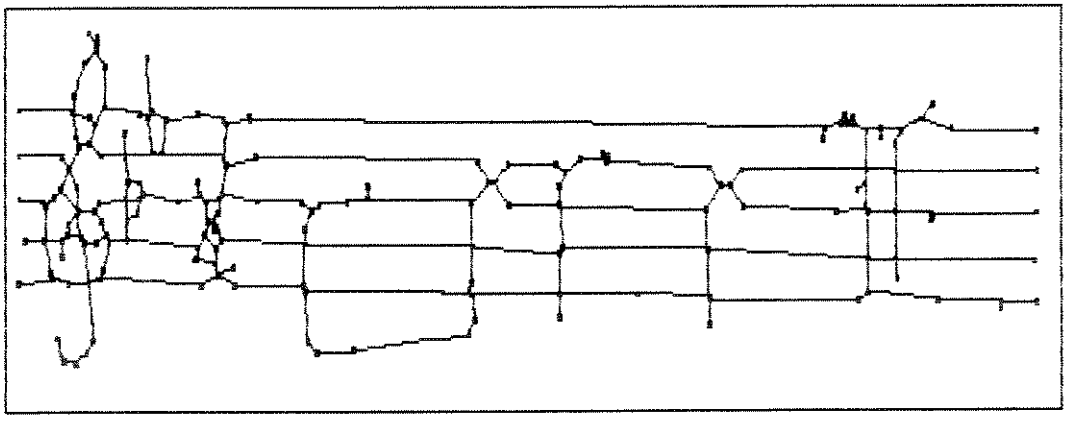
III) $(|x_2 - x_1| < tmp)$ e $(|y_2 - y_1| < tmp)$



a)



b)



c)

Fig 16 - a) Imagem binária com ruído. b) Imagem vetorizada. c) Após o processamento do filtro.

4. O pentagrama

O primeiro padrão que precisa ser reconhecido em um sistema de OMR é sempre o pentagrama. É em relação ao pentagrama que os símbolos ocupam posições em uma linha de texto. Dele depende a identificação da altura das notas e a distância entre suas linhas fornece a escala dos símbolos. Finalmente, sua extensão indica a extensão da linha de texto.

A grande extensão das linhas, favorece o uso de projeções para localizar o pentagrama. É comum o uso da projeção horizontal diretamente dos pixels da imagem binária. As linhas do pentagrama ficam bem evidentes na projeção. Neste trabalho optou-se por projetar a lista de membros da imagem. O uso dos vetores traz mais flexibilidade ao processo e permite que sejam projetados apenas os vetores com maior probabilidade de pertencer ao pentagrama tornando os resultados mais confiáveis.

Para localizar o pentagrama, é gerada uma lista dos membros horizontais da lista **Mb**. Estes membros são projetados horizontalmente em um histograma. A partir da projeção, a distância entre as linhas e a posição do pentagrama são estimadas. Idealmente as linhas do pentagrama devem ser horizontais, mas pequenas inclinações são toleradas pelo sistema.

4.1 Craqueamento dos membros

Nesta etapa os membros são fragmentados sempre que é detectado um ângulo muito acentuados entre dois vetores. O objetivo é fazer com que todos os vetores de um membro tenha inclinações parecidas. A classificação dos membros em horizontais ou verticais é baseada na inclinação média dos vetores que o compõe. Este processo confere mais precisão na classificação porque diminui a diferença entre a inclinação dos vetores do membro e sua inclinação média. A lista resultante deste processo será chamada de $\mathbf{Mb}_c$.

Seja φ a tolerância para o ângulo entre dois vetores.

Um membro **m** é quebrado no ponto $\mathbf{p}_{i+1}$ se:

$$| \alpha_i - \alpha_{i+1} | > \varphi$$

4.2 Filtro horizontal

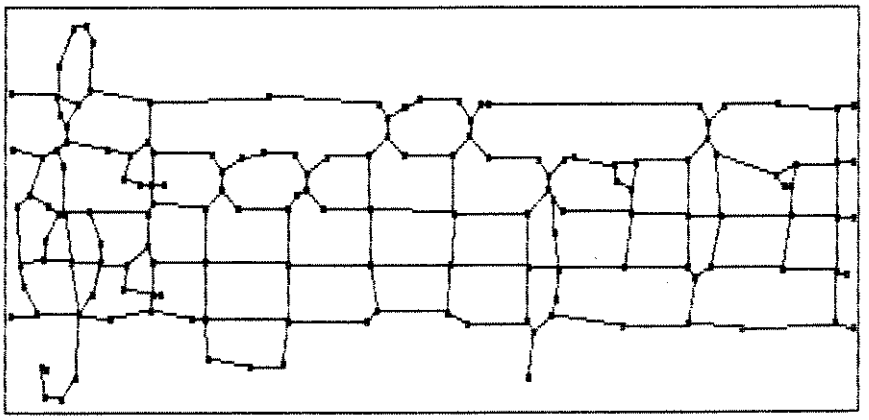
Um membro é considerado horizontal se a média das inclinações ponderada pelo tamanho de seus vetores for próxima de zero. Este filtro deixa passar apenas os membros aproximadamente horizontais, ou seja, aqueles cuja inclinação é próxima de zero.

Seja φ a tolerância permitida para a inclinação média do membro.

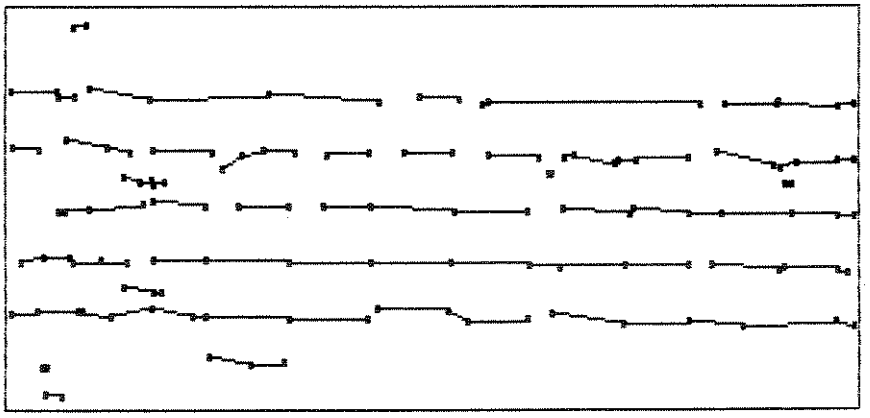
Am é o angulo médio do membro ponderado pelo tamanho dos segmentos.

$$Am = \frac{\sum_{i=0}^{Npt-1} A_i \cdot Tm_i}{\sum_{i=0}^{Npt-1} Tm_i}$$

O membro **m** pertence à lista Mb_h se $Am \leq \varphi$



a)



b)

Fig 17 - Exemplo do processamento do filtro horizontal.
a) Imagem original, b) Imagem filtrada.

4.3) Projeção horizontal

O pentagrama é localizado através da projeção horizontal da lista Mb_h. Toda projeção discreta pode ser encarada como um conjunto ordenado de acumuladores. Neste caso foi usado um acumulador para cada linha da imagem.

Cada linha tem uma área comprendida entre duas retas paralelas que atravessam horizontalmente a imagem. Os vetores atravessam essas retas. A idéia é acumular o comprimento do trecho do vetor que está entre duas retas vizinhas. Cada vetor da lista Mb_h influi no valor dos acumuladores associados às linhas que o vetor atravessa.

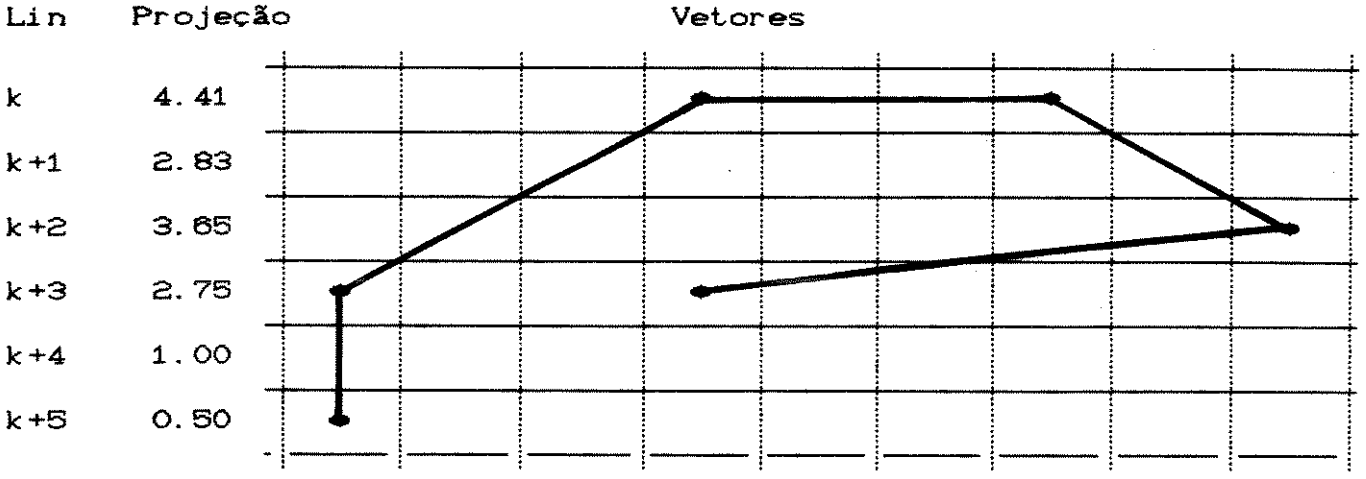


Fig 18 - Exemplo de projeção horizontal

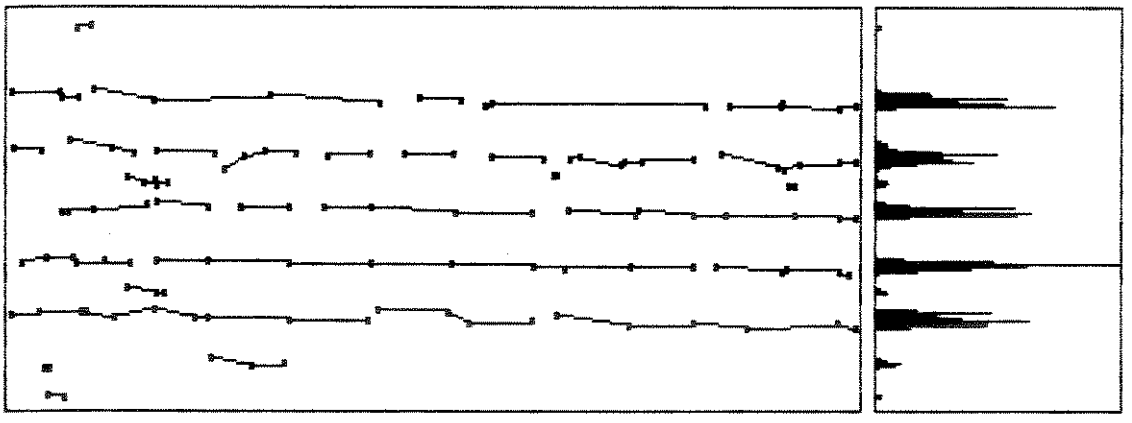


Fig 19 - Imagem filtrada e projeção horizontal.

Seja **H** conjunto ordenado de acumuladores que armazenam a projeção horizontal da imagem e h_k a projeção dos vetores na linha **k**. Inicialmente todos os acumuladores são zerados.

$$H = \{ h_k,\ h_k \in (y_i,\ y_{i+1}) \}$$

Para cada vetor v_i da lista Mb_h é repetido o seguinte algoritmo:

Se $(dy_i = 0)$ **então**
{
  $k = y_i = y_{i+1}$
  $h_k = h_k + Tm_i$
}
senão
{

$$Tt_i = \begin{cases} 1 & \text{se } dx_i = 0 \\ Tm_i \ / \ dy_i & \text{em outro caso} \end{cases}$$

Para $k = y_i$ **até** y_{i+1}
{

$$h_k = h_k + \begin{cases} Tt_i \ / \ 2 \text{ se } (k = y_i) \text{ ou } (k = y_{i+1}) \\ Tt_i \text{ em outro caso} \end{cases}$$

}
}

4.4) Distância entre as linhas

Devido à grande extensão das linhas do pentagrama, a projeção horizontal apresenta picos bem definidos nas posições em que elas ocorrem. Para determinar a distância entre as linhas do pentagrama é feito um histograma **G** de maneira que cada elemento g_k acumula o produto de todos os pares **h** que distam **k** entre si. O valor de **k** que apresentar pico neste histograma indica a distância mais provável entre as linhas do pentagrama.

Sejam:

DistL a estimativa da distância entre as linhas do pentagrama.

g_k o k-ésimo elemento do histograma **G**.

NLin o número de linhas na imagem e portanto o número de acumuladores na projeção **H**.

DistMin o valor mínimo aceito para **DistL**.

O valor de **DistL** é encontrado pelo seguinte algoritmo:

Passo 1: Zerar todos os elementos de **G**.

Passo 2: **Para i = 1 até (NLin - DistMin)**
{
Para j = (I + DistMin) até NLin
{
$g_{j-i} = g_{j-i} + (h_i \cdot h_j)$
}
}

Passo 3: Encontrar i onde g_i é máximo em G.
DistL = i

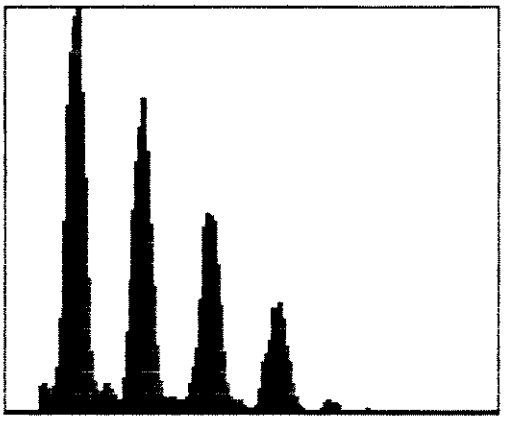


Fig 20 - Histograma da distância dos elementos da projeção horizontal apresentada na Fig 19.

4.5) Localização da primeira linha do pentagrama

Sendo já conhecido o valor de **DistL**, resta encontrar a posição dos 5 picos que representam as linhas do pentagrama na projeção **H**. O algoritmo abaixo armazena no histograma S a soma dos 5 elementos de **H** que distam **DistL** a partir da posição em teste.

Seja: **PosP** a coordenada **y** da primeira linha do pentagrama.
s_k o k-ésimo elemento do histograma S.

Passo 1: Zerar todos os elementos de S.

Passo 2: **Para i = 1 até (NLin - (DistL . 4))**
{

$$s_i = \sum_{j=0}^{4} h_{i+(DistL.j)}$$

}

Passo 3: Encontrar i onde s_i é máximo em S.
PosP = i

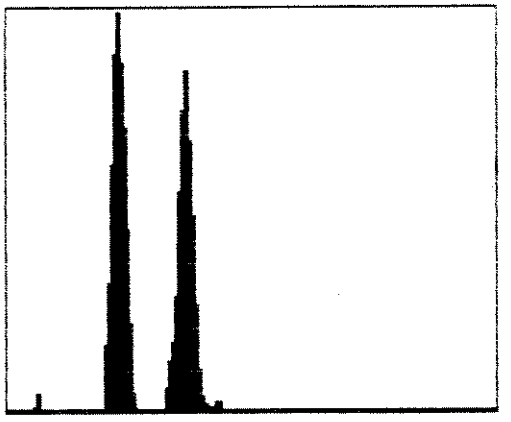


Fig 21 - Histograma S extraído a partir da projeção horizontal da Fig 16. O pico deste histograma indica a posição inicial do pentagrama.

5. Extração de características

O vetor de características usado neste trabalho para a classificação, é obtido através da análise da vizinhança dos pontos críticos. É definida uma região retangular em torno de cada ponto crítico que será o domínio das funções usadas como atributos dos pontos. As dimensões do retângulo usam como unidade a distância entre as linhas do pentagrama (ver seção **4.4**). O retângulo domínio foi dimensionado de maneira que a altura seja 3 vezes maior que a largura. Espera-se assim diminuir a influência de um símbolo em outro durante o cálculo dos atributos. Também por este motivo, as funções são inversamente proporcionais à distância do ponto em estudo. Características do objeto próximas ao ponto influem mais no valor do atributo que as mais distantes.

5.1 Agrupamento de membros

Em um esqueleto vetorizado pode ocorrer que dois pontos críticos estejam muito próximos e conectados por um pequeno vetor. É vantajoso para o reconhecimento que tais pontos sejam agrupados no seu ponto médio, porque isso diminui consideravelmente o número de configurações possíveis de posição dos pontos de um símbolo determinado.

O algoritmo abaixo procura na lista **Mb** os membros que conectam os pontos próximos. São membros em geral com um único vetor de pequeno tamanho. Cada ponto crítico pode ser extremidade de vários vetores e um ponto é mais ou menos significativo conforme o número de vetores que partem dele. Por esta razão, o ponto médio é ponderado pelo número de vetores que partem dos pontos. A figura 19 mostra o resultado do processamento deste filtro.

Os membros onde **Npt** = 2, são formados por um único vetor. Neste caso o perímetro do membro é igual ao tamanho do vetor.

Seja **Rmin** a distância mínima entre dois pontos.

$p_{ij}(x_{ij}, y_{ij})$ o j-ésimo ponto do membro m_i

Passo 1: Para cada membro m_i tal que $Npt_i = 2$ e $Pm_i < Rmin$ calcular:

$$mx = \frac{(x_{i1} \cdot Nv(p_{i1})) + (x_{i2} \cdot Nv(p_{i2}))}{Nv(p_{i1}) + Nv(p_{i2})}$$

$$my = \frac{(y_{i1} \cdot Nv(P_{i1})) + (y_{i2} \cdot Nv(P_{i2}))}{Nv(P_{i1}) + Nv(P_{i2})}$$

Passo 2: Atribuir os valores de mx e my a todos os pontos tais que $(x = x_{i1}$ e $y = y_{i1})$ ou $(x = x_{i2}$ e $y = y_{i2})$

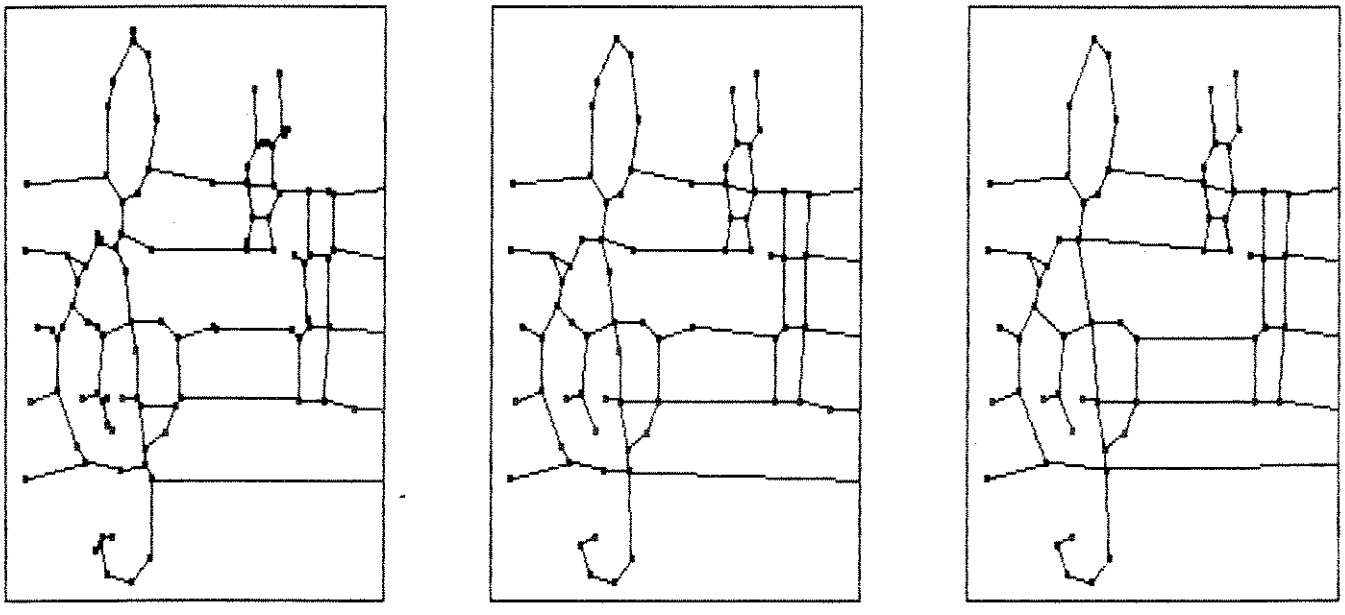
5.2) Pontos redundantes

Um ponto p é considerado redundante quando dele partem apenas dois vetores e a inclinação destes é parecida. Tais pontos não são significativos e podem confundir o reconhecimento. O algoritmo abaixo identifica e retira da lista os pontos redundantes.

Seja φ o ângulo mínimo entre dois vetores para que o ponto entre eles seja considerado significativo.

O ponto p_{ij+1} é retirado da lista se

$$| A_{ij} - A_{ij+1} | \leq \varphi$$



a) b) c)

Fig 22 - a) Imagem original, b) após o agrupamento dos pontos e c) pontos redundantes retirados.

5.3 Atributos dos pontos

São usadas 5 funções como atributos dos pontos críticos. Em todas elas o domínio é um retângulo em torno do ponto crítico. A primeira depende da posição dos pontos contidos no domínio, a segunda da quantidade de pontos, a terceira depende da posição dos vetores dentro do retângulo, a quarta da inclinação dos vetores e a última é a transformada de distância do ponto em estudo. As quatro primeiras são inversamente proporcionais à distância ao centro do retângulo.

Dados um ponto **p(x,y)** e um retângulo domínio **R** de dimensões **rx** e **ry**, com centro em **p**, define-se:

Npr o número de pontos críticos dentro de **R**.

$pr_k(x_k, y_k)$ o k-ésimo ponto dentro de **R**.

$DistR(pr_k)$ a distância do ponto pr_k ao centro de **R**.

$\Delta pr_k = pr_k - p$

Nvt o número de vetores com ponto médio em **R**.

v_k o k-ésimo vetor com ponto médio em **R**.

$DistV(v_k)$ a distância do ponto médio de v_k ao centro de **R**.

Δpv_k a diferença entre o ponto médio de v_k e **p**.

As funções **Cg(p)**, **Np(p)**, **Cv(p)** e **Im(p)** são definidas em **R**:

Cg(p) = (cx,cy) é o "centro de gravidade" dos pontos críticos no retângulo domínio. A "massa" de um ponto **q** é o número de vetores que partem dele **Nv(q)**, dividido pela distância ao centro do retângulo mais 1.

$$Cg(p) = \frac{\sum_{k=1}^{Npr} \frac{\Delta pr_k \cdot Nv(pr_k)}{1 + DistR(pr_k)}}{\sum_{k=1}^{Npr} \frac{Nv(pr_k)}{1 + DistR(pr_k)}}$$

Np(p) = n é proporcional a "massa" próxima ao centro de **R**.

$$Np(p) = \sum_{k=1}^{Npr} \frac{Nv(pr_k)}{1 + DistR(pr_k)}$$

Cv(p) = (vx, vy) é o centro de gravidade dos vetores cujo ponto médio está em **R**. Os vetores são representados pelos seus pontos médios. Nesta função a "massa" dos vetores é proporcional ao seu tamanho.

$$Cv(p) = \frac{\sum_{k=1}^{Nvt} \frac{\Delta pv_k \cdot Tm_k}{1 + DistV(v_k)}}{\sum_{k=1}^{Nvt} \frac{Tm_k}{1 + DistR(v_k)}}$$

Im(p) = a é a inclinação média dos vetores em **R**, ponderada pelo tamanho e distância do ponto médio do vetor ao centro de **R**.

$$Im(p) = \frac{\sum_{k=1}^{Nvt} \frac{A_k \cdot Tm_k}{1 + DistV(v_k)}}{\sum_{k=1}^{Nvt} \frac{Tm_k}{1 + DistR(v_k)}}$$

Td(p) = t é a transformada de distância do pixel **(x,y)** da imagem binária. Esta função não depende de **R** e é calculada sobre a imagem de entrada original sem qualquer processamento.

5.4 Vizinhança de vetores

A cada ponto P da lista **Mb**, é associado o conjunto de inclinações e tamanhos dos vetores que partem de P. Estas inclinações serão usadas em conjunto com os atributos para determinar o quanto um ponto do objeto é parecido ou distante de outro no banco de dados. O número de vetores com origem no ponto P é dado por **Nv(P)**.

Sejam:

av_{Pi} o ângulo de inclinação do iésimo vetor que parte do ponto P. O valor de av_i é tal que $0 <= av_i <= 2\Pi$

tv_{Pi} o tamanho do iésimo vetor que parte do ponto P.

5.5 Atributos dos símbolos

Depois que os candidatos a símbolos são construídos, para caracterizar os símbolos são usados os mesmos atributos que os pontos, mas o domínio das funções que os calculam passa a ser o retângulo envolvente dos candidatos. No caso dos pontos as funções são inversamente proporcionais à distância das características que estão sendo medidas ao ponto em estudo, que ocupa sempre o centro geométrico do retângulo domínio. Para os candidatos, as funções são inversamente proporcionais ao centro do retangulo envolvente.

Como é criado um ponto fictício no centro do retângulo, o atributo de espessura é não pode ser obtido através da transformada de distância. Uma solução é usar a média ponderada da espessura dos pontos crítico contidos no retângulo. O atributo de espessura do símbolo s é dado por:

$$Td_s(s) = \frac{\sum_{k=1}^{Npr} \frac{Td(pr_k)}{1 + DistR(pr_k)}}{\sum_{k=1}^{Npr} \frac{1}{1 + DistR(pr_k)}}$$

5.6) Normalização

Os atributos do ponto têm valores em escalas diferentes. Para poder calcular a distância entre pontos é necessário antes normalizar os valores. Por conveniência o sistema transforma os valores em números entre zero e 100. Para isso, várias imagens são analisadas afim de encontrar os valores máximo e mínimo típicos de cada atributo. De posse destes valores, basta aplicar uma regra de três para fazer a normalização.

Os valores máximo e mínimo podem ser encontrados por uma variedade de métodos. A curva de Gauss talvez seja o mais indicado, mas foram obtidos bons resultados apenas armazenando o maior e o menor valor encontrado para cada atributo.

5.7 Distância entre dois pontos

Pode-se pensar nos atributos do ponto como coordenadas de um espaço multidimensional. Cada configuração de atributos ocupa uma posição neste espaço. A distância entre dois pontos P_1 e P_2 deste espaço pode ser calculada geometricamente.

Seja:

$\aleph$ o espaço de 6 dimensões com coordenadas (cx, cy, n, vx, vy, a, t).

D(P, Q) a distância entre os pontos P e Q no espaço $\aleph$.

$$D(P,Q) = \sqrt{\Delta cx^2 + \Delta cy^2 + \Delta n^2 + \Delta vx^2 + \Delta vy^2 + \Delta a^2 + \Delta t^2}$$

5.8 Distância entre configurações de vetores

De cada ponto crítico P partem Nv(P) vetores. Para calcular a distância entre configurações de vetores, cada vetor é representado pela sua inclinação e tamanho (ver seção 5.4) no par ordenado (av_P, tv_P).

A distância dos vetores entre os pontos P e Q só será definida quando Nv(P) = Nv(Q). Cada vetor de P é associado ao vetor de Q que tem a inclinação mais próxima. O algoritmo abaixo ordena duas listas de vetores de maneira que cada vetor ocupe a mesma posição na lista que o seu vetor correspondente na outra lista.

```
Passo 1:  Para I = 1 até Nv(P)
          {
             d = | av_Pi - av_Qi |
             se (d > Π) então d = 2Π - d
             Para J = I+1 até Nv(Q)
             {
                ax = | av_Pi - av_Qj |
                se (ax > Π) então ax = 2Π - ax
                se (ax < d)
                {
                   d = ax
                   Trocar os valores de (av_Qi, tv_Qi) com (av_Qj, tv_Qj)
                }
             }
          }
```

Passo 2: Executar o passo 1 invertendo as posições de P e Q.

Depois de ordenadas as listas de vetores, a distância Dv(P,Q) entre os pontos P e Q é dada por:

$$Dv(P,Q) = \sqrt{\sum_{i=1}^{i=Nv(P)} (av_{Pi} - av_{Qi})^2 + (tv_{Pi} - tv_{Qi})^2}$$

5.9 Distância entre símbolos

Com exceção da transformada de distância, os atributos usados para caracterizar os pontos críticos também podem ser usados para caracterizar os símbolos. Neste caso o domínio dos atributos passa a ser o retângulo envolvente do símbolo que deve ser previamente conhecido. A normalização destes atributos pode ser feita aproveitando os valores maximos e mínimos obtidos para os pontos críticos e nesse caso os valores normalizados não ficarão apenas na faixa de zero a 100. O sistema funciona desta maneira e foram obtidos bons resultados. É claro que a situação ideal é que os atributos dos símbolos tenham valores máximos e mínimos próprios e isto poderá ser implantado no futuro.

A distância entre os atributos de dois símbolos é calculada da mesma maneira que nos pontos críticos.

Sejam: $(cx_s, cy_s, n_s, vx_s, vy_s, a_s)$ os atributos de um símbolo.

Ds(S,T) a distância entre os atributos dos símbolos S e T.

$$Ds(S,T) = \sqrt{\Delta cx_s^2 + \Delta cy_s^2 + \Delta n_s^2 + \Delta mx_s^2 + \Delta my_s^2 + \Delta im_s^2}$$

6. Banco de dados

Dado que o sistema usa uma abordagem estatística para a classificação dos objetos, um banco de dados é necessário para armazenar os atributos de cada classe.

O banco de dados do sistema é composto por um banco de pontos e um banco de símbolos. O banco de pontos armazena os atributos dos pontos críticos. Estes atributos são calculados dentro de um retângulo domínio em torno do ponto em estudo. Para o banco de símbolos os atributos são calculados de modo similar dentro do retângulo envolvente do símbolo. A idéia é usar o banco de pontos para construir candidatos a símbolos e o banco de símbolos para descartar os candidatos com menor probabilidade.

O banco é alimentado fornecendo ao sistema exemplos de símbolos para treinamento. Para cada imagem do treinamento deve ser feita (manualmente) uma lista com o nome e as coordenadas do retângulo que envolve cada símbolo na imagem. O sistema calcula os atributos dos pontos críticos e dos símbolos e os armazena no banco de dados. Para os pontos críticos é armazenado também o símbolo associado e sua posição em relação ao retângulo envolvente do símbolo.

6.1 Banco de pontos

Neste banco são armazenadas todos os dados dos pontos críticos em valores normalizados. Antes de acrescentar um ponto, é feita uma busca nos pontos do banco. Se for encontrado um ponto muito próximo e do mesmo símbolo, então é feita uma média dos dois pontos levando em consideração número de pontos que já contribuiram para a média. Se nenhum ponto no banco for considerado próximo do ponto em estudo, o ponto é acrescentado ao banco de dados. Cada registro contêm os seguintes campos:

1) **Atributos.** Descritos na seção 5.3.
2) **Lista de vetores.** Inclinação e tamanho de todos os vetores que partem do ponto. (Ver seção 5.4)
3) **Nome do símbolo associado.**
4) **Tamanho do retângulo envolvente do simbolo (Tx, Ty)**
5) **Posição do ponto dentro do retângulo envolvente.**

6.2 Banco de símbolos

Usando como domínio os retângulos envolventes fornecidos durante o treinamento, os atributos são calculados e armazenados no banco. Da mesma maneira que no banco de pontos, é feita uma busca no banco e caso seja encontrada uma entrada do mesmo símbolo e com atributos muito próximos, a média ponderada dos dois símbolos é armazenada. Se não for encontrada nenhuma entrada com atributos próximos, o símbolo é incluído no final do banco. Para cada símbolo podem haver várias entradas no banco. Os registros são formados pelos seguintes campos:

1) **Atributos.** Descritos na seção 5.5. (Ver também seção 5.9)
2) **Nome do símbolo.**
3) **Tamanho do retângulo envolvente.**

6.3 Observações

O cálculo da distância entre cada ponto crítico obtido na imagem e os pontos do banco de dados é a operação mais cara do sistema. Boa parte do tempo de processamento é gasto calculando distâncias entre os pontos. Por fugir ao escopo deste trabalho, não foi feito nenhum esforço para otimizar o uso do banco de dados. Para determinar que pontos do banco de dados são mais próximos de um ponto na imagem, o sistema calcula a distância do ponto crítico a todos os pontos no banco de dados selecionando os mais próximos. No entanto uma ordenação adequada dos dados em conjunto com um algoritmo de busca poderia reduzir drasticamente o número de chamadas a função de distância diminuindo consideravelmente o tempo de processamento do sistema.

7. Classificação

O reconhecimento de padrões usa principalmente duas grandes linhas de abordagem: classificação e reconhecimento sintático.

O reconhecimento sintático procura identificar um objeto a partir da relação entre suas partes usando principalmente a teoria de linguagens formais.

A classificação consiste em atribuir a cada objeto, uma das várias classes possíveis. Para cada objeto a ser classificado, um vetor de características deve ser extraído. Este vetor define um ponto em um espaço de atributos onde cada classe ocupa uma região. Desta maneira o vetor de características associa cada objeto a uma classe. Um dos problemas nesta abordagem é determinar o volume da região no espaço de atributos que cada classe ocupa. As regiões não têm necessariamente o mesmo volume ou formato. Podem ocorrer também regiões de intersecção que estão associadas a duas ou mais classes, assim como regiões vazias, ou seja, não associadas a nenhuma classe.

O volume e formato das regiões que as classes ocupam podem ser obtidos estatisticamente a partir de exemplos de classes previamente conhecidas. Os objetos que "caem" em regiões de intersecção ou de vácuo podem ser classificados por critérios de distância. O objeto seria então associado à classe mais próxima.

Neste trabalho a classificação ocorre em três estapas distintas. Em primeiro lugar cada ponto crítico é associados à lista dos pontos mais próximos no banco de dados. Esta lista é montada com base no vetor de características dos **pontos**. A partir da lista são contruídos candidatos a símbolos na segunda etapa. Finalmente os candidatos são analisados e aqueles com menor probabilidade são descartados. A análise é feita baseada no vetor de características dos **símbolos**.

7.1 Lista de pontos próximos

Nesta etapa de processamento, o objetivo é associar cada ponto crítico da imagem à lista dos **W** pontos mais próximos no banco de dados. A distância entre dois pontos é estimada pela média entre a distância euclidiana no espaço de atributos (secão **5.7**) e a distância entre vetores (seção 5.8). A lista é ordenada pela distância estimada. O sistema calcula a distância entre cada ponto crítico e todos os pontos no banco de dados e armazena os **W** pontos mais próximos. O valor da constante **W** é proporcional ao número de elementos no universo de símbolos. Para um universo de 25 símbolos o valor 10 deu bons resultados. Até o momento não foi feito nenhum estudo sobre a melhor maneira de calcular **W**, mas sabe-se que quando **W** cresce, aumenta o número de candidatos encontrados na etapa seguinte. Cada elemento na lista contém os seguintes campos: (ver seçao **6.1**)

1) **Símbolo associado**
2) **Dimensões do retângulo envolvente**
3) **Posição do ponto em relação ao início do retângulo envolvente**
4) **Distância ao ponto crítico associado**

7.2 Construção de candidatos

Cada ponto na lista, indica uma posição relativa dentro do retângulo envolvente. Nesta etapa o sistema varre as listas de todos os pontos críticos detectando agrupamentos de retângulos do mesmo símbolo e em posições parecidas. Se vários pontos dispersos em uma região da imagem estão associados ao mesmo símbolo e indicam retângulos parecidos é provável que o símbolo exista nessa região. Quanto mais pontos concordarem com o símbolo maior é a probabilidade de sua existência. O retângulo do candidato é a média dos retângulos indicados pelos pontos concordantes. Este processo cria muitos candidatos sobrepostos, mas também associa a cada candidato um valor numérico proporcional à probabilidade de existência do símbolo na imagem. Sobre este valor pode ser aplicado um treshold que elimine os menos prováveis. Este treshold no entanto não é suficiente para eliminar todos os candidatos incorretos, mas pode diminuir bastante o número de candidatos.

7.2.1 Algoritmo

Cada ponto na lista está associado a um símbolo (S) e aponta para um retângulo envolvente com início em (x,y). A lista é ordenada pela distância (d) ao ponto crítico associado. O sistema detecta concentrações de pontos na lista associados ao mesmo símbolo e com retângulos envolventes em posições parecidas. Um ponto da lista participa da construção de um único candidato. Para isso, os pontos recebem uma marca quando são processados. O algoritmo para a criação dos candidatos é descrito abaixo e sua estrutura de dados ilustrada pela figura 23. A saída desta etapa é uma lista de candidatos que contêm o nome do símbolo, seu retângulo envolvente e peso estimado.

Sejam:

pc_i o iésimo ponto crítico da imagem

Lp_i a lista dos W pontos do banco de dados mais próximos de pc_i

Lp_{ij} o j-ésimo ponto da lista Lp_i

d_{ij} a distância de Lp_{ij} ao ponto crítico pc_i

S_{ij} o símbolo associado ao ponto Lp_{ij}

R_{ij} o retângulo envolvente do símbolo S_{ij} com início em (x_{ij}, y_{ij})

DistL é a distância entre as linhas do pentagrama (seção **4.4**).

$R_c(x_c, y_c)$ é o retângulo estimado para um candidato a símbolo.

G_c é o peso do candidato R_c.

Passo 1: Escolher um ponto Lp_{ij} que não esteja marcado. Não há nenhum método para a escolha do primeiro ponto. O sistema começa com o primeiro elemento da lista **Lp** associada ao primeiro ponto crítico **pc**. O ponto Lp_{ij} recebe uma marca para que não seja processado novamente.

$$G_c = 100 / (1 + d_{ij})$$

Passo 2: Para todo Lp_{mn} não marcado tal que $m \neq i$:

Se ($S_{ij} = S_{mn}$) e
($|x_{ij} - x_{mn}| < DistL/3$) e
($|y_{ij} - y_{mn}| < DistL/3$)
então {
$G_{mn} = 100 / (1 + d_{mn})$
$x_c = [(x_c . G_c) + (x_{mn} . G_{mn})] / (G_c + G_{mn})$
$y_c = [(y_c . G_c) + (y_{mn} . G_{mn})] / (G_c + G_{mn})$
$G_c = G_c + G_{mn}$
O ponto Lp_{mn} é marcado.
}

Passo 3: (x_c, y_c) contém agora a posição do início do retângulo envolvente R_c do candidato construído. Armazenar R_c e voltar ao passo 1. O processo termina quando todos os pontos estiverem marcados.

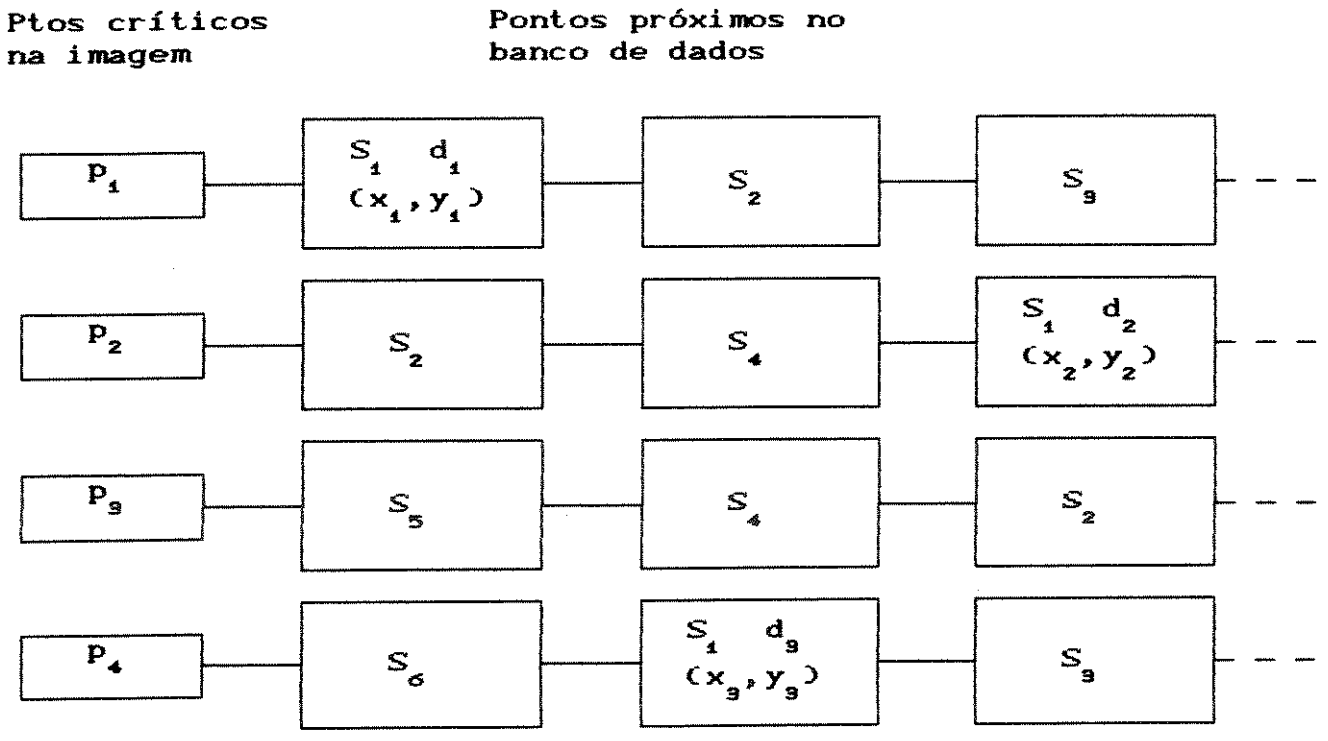


Fig 23 - P1, P2, P3 e P4 são pontos críticos da imagem, a cada um é associada a lista dos pontos mais próximos do banco de pontos.

7.3 Seleção dos candidatos

Na etapa anterior, é obtida uma lista de candidatos a símbolos. O número de candidatos na lista é ainda muito maior que o número de símbolos na imagem. O objetivo nesta etapa é selecionar apenas os candidatos mais prováveis.

O primeiro nível de seleção é feito aplicando um treshold sobre o peso que foi acumulado durante a construção do candidato. Candidatos com peso muito baixo são eliminados da lista. Como o número médio de pontos críticos varia muito de um símbolo para outro, o valor mínimo do peso para que o candidato não seja descartado deve variar de acordo com o símbolo.

O segundo nível de seleção atua sobre o retângulo envolvente do candidato. Para cada candidato é
calculado o vetor de características de seu retângulo envolvente. Este vetor é usado para estimar a distância entre o candidato e os símbolos armazenados no banco de símbolos. Nos casos de sobreposição de candidatos, esta distância pode indicar o candidato mais provável. Recomenda-se ponderar a distância por símbolo porque em símbolos diferentes a faixa de valores aceitáveis para a distância pode variar. Restam ainda os casos onde candidatos que se sobrepõe, tem distância parecida. Para estes casos são sugeridos mais dois níveis de seleção, onde seria considerada a grafia e a gramática da notação musical, que até este ponto não foram usadas na classificação dos símbolos.

O próximo nível de seleção de candidatos não foi implementado mas é recomendado para resolver ambiguidades que ocorrem quando dois candidatos a símbolos se sobrepõe. Neste nível é levada em conta a ortografia musical. Alguns símbolos como claves e barras de compasso não podem ocorrer em qualquer altura do pentagrama, o bequadro ocorre sempre imediatamente antes de uma nota, a nota cheia por sua vez tem que ter um cabo conectado a ela. Estas e outras regras simples de grafia aumentam a confiabilidade dos candidatos a símbolos que as satisfazem e permitem descartar outros.

É possível imaginar um terceiro nível de seleção que inclua conhecimento da gramática. Neste nivel seria feita uma verificação dos tempos entre as barras de compasso, a distribuição de acidentes e bequadros, as barras de repetição e outros testes.

8. Implementação e resultados

O sistema é composto por pequenos programas desenvolvidos em Turbo C. O hardware utilizado foi um scanner de mesa e um PC 386. Os programas comunicam-se sempre através de arquivos texto de maneira a facilitar o acompanhamento do processo. Este método tem a desvantagem de aumentar o tempo de processamemto devido a constantes acessos a disco, mas por outro lado permite controlar cada passo do processamento. Como não houve preocupação em otimizar os processos, o tempo de processamento pode ser reduzido consideravelmente. A saída do sistema é uma lista de candidatos a símbolos, com a posição de seus retângulos envolventes e o peso estimado.

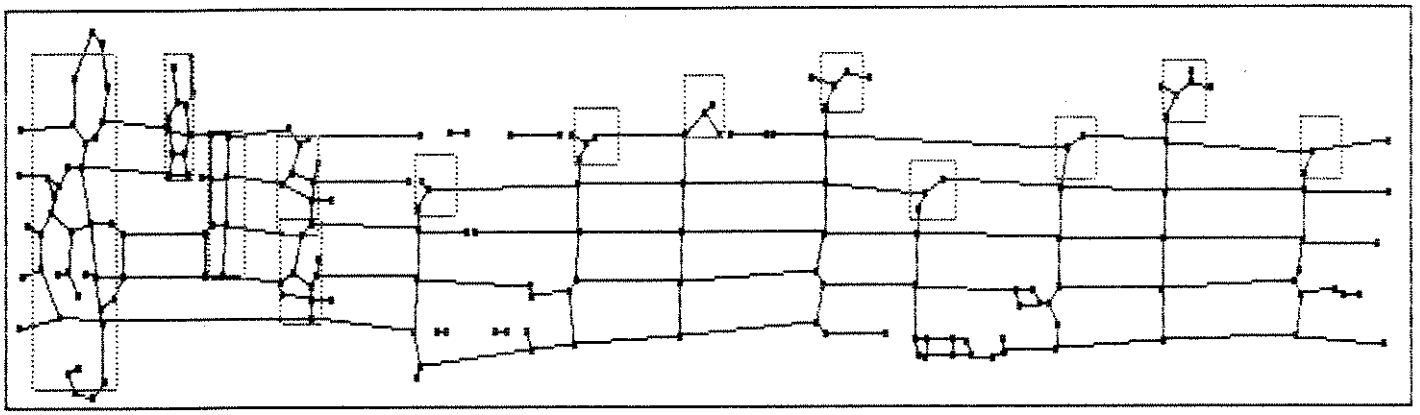
A figura 24 mostra os candidatos que foram classificados corretamente em uma imagem não usada no treinamento do banco de dados. A tabela 3 apresenta os valores obtidos para os candidatos apresentados na figura 24. A figura 25 e tabela 4, mostram os resultados para uma imagem treinada. Nos dois casos, todos os símbolos que existem na imagem foram delimitados e classificados corretamente por pelo menos um dos candidatos no conjunto de candidatos prováveis. As tabelas 3 e 4 listam os símbolos que estão presentes nas imagens e apresenta o peso (calculado sobre os pontos críticos), a distância do símbolo no retângulo envolvente estimado ao símbolo armazenado no banco de símbolos e a posição inicial do retângulo envolvente estimado.

Imagem 1 (não treinada): 165 candidatos			
Símbolo	Peso (pontos)	Distância (símbolo)	Posição inicial
ClaveSol	2935.1	27.3	(16, 16)
SustenidoLin	500.0	47.5	(94, 16)
SustenidoEsp	101.5	41.1	(119, 45)
Numero4	100.0	121.6	(160, 47)
Numero4	600.0	119.4	(162,119)
NtCheiaLin	301.5	41.4	(244, 54)
NtCheiaLin	200.0	53.3	(340, 37)
NtCheiaEsp	669.2	54.1	(405, 25)
NtCheiaLin	300.3	190.5	(489, 17)
NtCheiaLin	200.1	312.3	(544, 57)
NtCheiaLin	200.0	516.6	(632, 41)
NtCheiaLin	398.6	681.2	(696, 20)
NtCheiaLin	400.0	859.4	(779, 41)

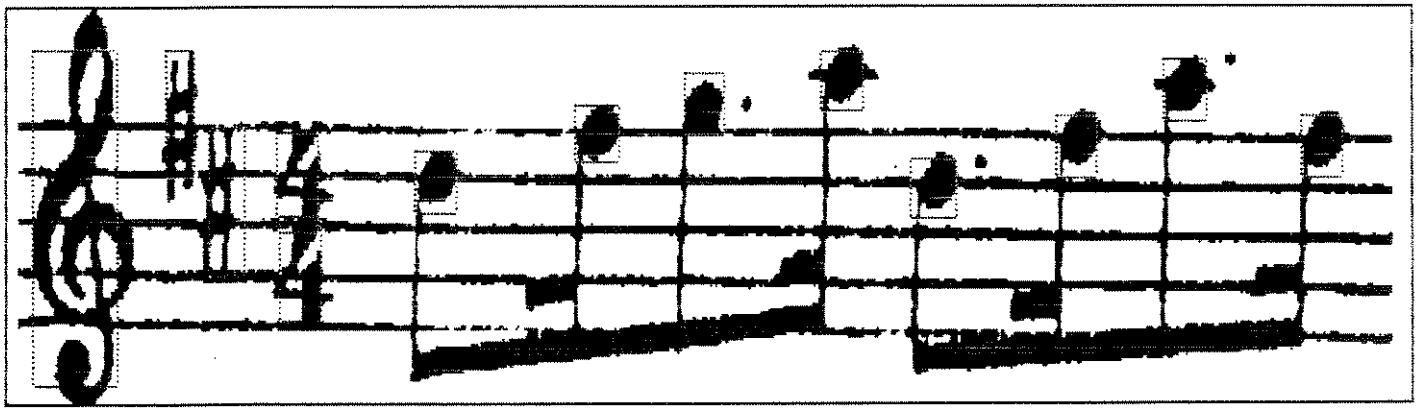
Tabela 3 - Peso, distância e posição inicial do retângulo estimados para Imagem 1, apresentada na figura 24.

Imagem 2 (treinada): 103 candidatos			
Símbolo	Peso (pontos)	Distância (símbolo)	Posição inicial
ClaveSol	5834.0	10.1	(4, 5)
Numero4	1200.1	36.0	(71, 33)
Numero4	1200.1	24.8	(70, 71)
NtCheiaEsp	1567.7	33.7	(120, 51)
NtCheiaEsp	1964.9	76.4	(171, 52)
NtCheiaEsp	2100.1	139.4	(220, 31)
NtCheiaEsp	1973.8	54.9	(269, 31)
NtCheiaEsp	2450.3	70.9	(317, 53)
PausaColcheia	600.0	18.8	(367, 55)
NtCheiaEsp	1900.0	27.7	(415, 33)
PausaColcheia	599.8	52.2	(465, 56)

Tabela 4 - Peso, distância e posição inicial do retângulo estimados para a imagem 2 (fig. 25). A imagem 2 foi usada para alimentar o banco de dados.

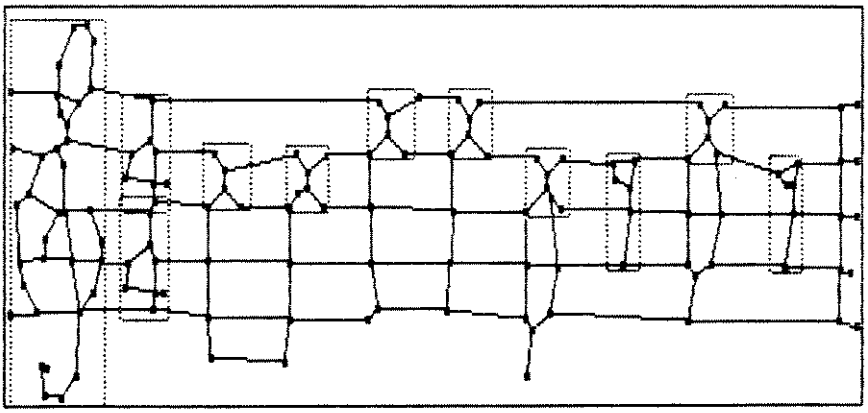


a)

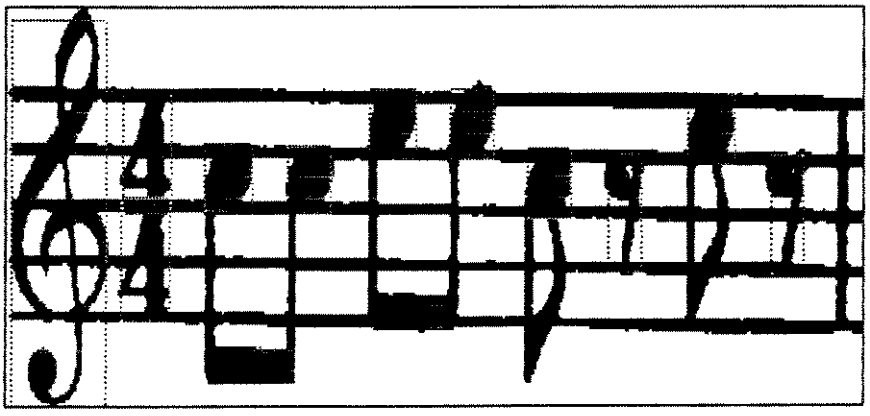


b)

Figura 24 - Imagem 1 e os retângulo dos candidatos estimados corretamente. a) Imagem vetorizada e b) Imagem binária original.



a)



b)

Figura 25 - Imagem 2 e retângulos corretos. a) Imagem vetorizada e b) Imagem binária original. Esta é uma das imagens usadas no treinamento.

9. Conclusões

Este trabalho propõe um caminho para um sistema de reconhecimento de símbolos musicais. A ênfase dada ao módulo de pré-processamento é justificada pela variedade de aplicações ligadas ao reconhecimento de padrões em que pode ser útil. Neste módulo são apresentados novos algoritmos para o afinamento, extração do chain-code e vetorização de imagens binárias. Estes algoritmos mostraram ser bastante rápidos e eficientes.

Para o reconhecimento o método desenvolvido mostrou ser mais eficiente na identificação de símbolos complexos, com muitas linhas e pontos de cruzamento. Já os símbolos mais simples podem confundir o sistema. Para resolver este problema é necessário um estudo mais profundo dos atributos utilizados e possivelmente a inclusão de mais dimensões no espaço de atributos. O uso de tresholds diferenciados para o peso e distância dos vários símbolos também é recomendado.

A lista de candidatos gerada é bastante extensa e embora o peso e a distância estimados sejam de grande ajuda durante a seleção dos candidatos, ocorrem casos em que candidatos incorretos apresentam peso maior e distância menor do que candidatos corretos. Também pode ocorrer que um candidato correto tenha peso muito baixo e distância alta, portanto o uso de tresholds sobre estes valores deve ser melhor estudado. Estes problemas podem ser parcialmente resolvidos com a inclusão de mais dois níveis de seleção de candidatos que levem em consideração a grafia e a gramática dos símbolos musicais. O sistema como se apresenta até o momento conta com muito pouco conhecimento de notação musical.

Alguns símbolos como ponto de aumento, staccato, pausa de semibreve, pausa de mínima e outros podem desaparecer completamente após o processo de afinamento e vetorização. Estes símbolos devem ser reconhecidos por outro método. As informações geradas por este sistema certamente facilitará o reconhecimento destes símbolos pelo método que for utilizado. As hastes e barras das notas que indicam sua duração, também não são tratadas pelo sistema. As informações da duração das notas devem ser obtidas em um nível mais alto aproveitando as informações fornecidas pelo sistema.

Todas as imagens utilizadas para treinamento e teste são de partituras com apenas uma voz (uma linha melódica), mas os resultados sugerem que o sistema pode ser ajustado para trabalhar com várias vozes e acordes.

Para concluir vale notar que o método de classificação proposto pode ser utilizado em outras aplicações fora do OMR. É especialmente indicado quando a delimitação dos símbolos é dificultada pela conexão entre eles. Uma possível aplicação pode ser o reconhecimento de símbolos em circuitos elétricos.

Referências

[ARC 85] C. Arcelli e G. Sanniti di Baja, "A Width-Independent Fast Tinning Algorithm", IEEE Trans. on Patter Analysis and Mach. Intell., **PAMI**-7, No.4, 1985, pp 463-474.

[BLO 92] D. Blostein e H.S. Baird, "A Critical Survey of Music Image Analisys", *Structured Document Image Analisys*, Ed Baird, Bunke, Yamamoto, 1992.

[CHA 84] B.B. Chaudhuri e M.K. Kundu, "Digital line segment coding: A new efficient contour coding scheme", IEE Proc., vol 131, No. 4, 1984

[CHI 87] R.T. Chin e H.K. Wan, "A One-Pass Thinning Algorithm and Its Parallel Implementation", Comput. Vision Graphics Image Process., vol 40, 1987, pp 30-40.

[DAV 81] E.R. Davies e P.N. Plummer, "Thinning Algorithms: A Critique And A New Methodology", Pattern Recognition, vol 14, 1981, pp 53-63.

[DEU 72] E.S. Deutsch, "Thinning algoritms on rectangular, hexagonal and triangular arrays", Comm. Ass. Comput. Mach., vol 15, 1972, pp 827-837.

[GUO 89] Z. Guo e R. W. Hall, "Parallel Thinning with Two-Subiteration Algorithms", Image Proces. and Comp. Vision, Vol 32, No 3, 1989.

[HAL 89] R. W. Hall, "Fast Parallel Thinning Algorithms: Parallel Speed and Connectivity Preservation", Comm. ACM, Vol 32, No. 1, 1989, p 124.

[KWO 88] P.C.K. Kwok, "A Thinning Algorithm by Contour Generation", Comm. ACM, vol 31, No. 11, 1988, pp 1314-1324.

[NAC 84] N.J. Naccache e R. Shinghal, "SPTA: A Proposed Algoritm for Thinning Binary Patterns", IEEE Trans. Vol SMC-14, No. 3, pp 409-418.

[OGO 90] L. O'Gorman, "k × k Thinning" , Comput. Vision Graphics Image Process., vol 51, 1990, pp 195-215.

[PAV 80] T. Pavlidis, "A Thinning Algoritm for Discrete Images", Comput.Graph. and Image Proces. Vol 13, pp 142-157

[PER 92] Francesco A. Perrotti e Roberto A. Lotufo, "Um Novo Algoritmo Paralelo de Afinamento". *Anais do SIBGRAPI V*, 1992, pp 285-293.

[ROA 88] J.W. Roach e J.E. Tatem, "Using Domain Knowledge in Low-Level Visual Processing to Interpret Handwritten Music: An Experiment", *Pattern Recognition*, vol 21, Nº. 1, 1988, pp 33-44.

[SIN 87] R.M.K. Sinha, "A Width-Independent Algoritm for Caracter Skeleton Estimation.", Comput. Vision Graphics Image Process., vol 40, 1987, pp 388-397.

[STE 92] F. Stein e G. Medioni, "Structural Indexing: Efficient 2-D Object Recognition", *IEEE Trans. Patt. Anal. Machine Intell*, vol 14, pp 1198-1204, 1992.

[SUZ 87] S. Suzuki e K. Abe, "Binary Picture Thinning by an Iterative Parallel Two-Subcicle Operation", Pattern Recog. Vol 20. No 3, 1987.

[ZHA 84] T.Y. Zhang e C.Y. Suen, "A Fast Paralel Algorithm for Thinning Digital Patterns.", Comm. ACM, vol 27, No. 3, 1984, pp 236-239.

Apêndice A

Noções básicas da notação musical

A-1 Pentagrama

Para que uma sequência de sons seja registrada graficamente é necessário que existam maneiras de representar a frequência dos sons (altura), sua duração no tempo e intensidade (volume). O sistema de escrita ocidental usa uma pauta de 5 linhas (pentagrama) que em conjunto com uma clave é capaz de representar a altura das notas. As notas são representadas por circunferências escritas sobre ou entre as linhas do pentagrama. Quanto mais aguda a nota, mais alta é escrita em relação ao pentagrama. Se a nota ultrapassar o pentagrama (acima ou abaixo), linhas suplementares são colocadas.

A-2 Claves

Como a amplitude dos instrumentos varia muito, se as notas tivessem posições fixas em relação ao pentagrama alguns intrumentos teriam suas partituras com todas as notas bem acima do pentagrama, enquanto outros com notas sempre abaixo. As linhas suplementares dificultam a leitura rápida da partitura e para resolver este problema é que existem as claves.

As claves indicam em que posições do pentagrama podem ser encontradas as notas. As tres claves, a de SOL, de FÁ e de DO, colocam uma das linhas do pentagrama em destaque. Esta linha é a posição da nota que a clave indica. A clave de SOL é normalmente escrita colocando a 2ª linha do pentagrama (de baixo para cima) em destaque. Significa que a nota na 2ª linha é SOL, o espaço acima é LÁ, na 3ª linha SI e assim por diante. A mesma clave pode ser colocada sobre linhas diferentes. Os instrumentos (ou vozes) com amplitude predominantemente grave usam a clave de FÁ para suas partituras, os instrumentos mais agudos usam a clave de SOL e a clave de DO é usada para instrumentos que produzem sons médios. Todo pentagrama que contém notas inicia obrigatoriamente com uma clave. Se a clave não for encontrada ou deduzida, não é possível saber a posição das notas no pentagrama.

A-3 Duração dos sons

A duração de tempo das notas é grafada acrescentando à circunferência que representa a nota primitivas que indicam a relação entre a duração da nota e a unidade de tempo da música. A duração em segundos da unidade de tempo varia com o *andamento* (velocidade) da música, mas sua relação com a duração das notas é constante. São 7 símbolos em uso atualmente, a saber: *semibreve*, *mínima*, *semínima*, *colcheia*, *semicolcheia*, *fusa* e *semifusa*. A duração da semibreve equivale à duração de duas mínimas, uma mínima dura tanto quanto duas semínimas e assim por diante sempre mantendo uma relação de dobro-metade. Há ainda um oitavo símbolo, a *breve*, mas raramente é usada, a não ser em originais muito antigos.

O ponto de aumento é usado para prolongar a duração de uma nota. É colocado à direita da nota e significa que a duração da nota deve ser aumentada na metade de seu valor. Uma colcheia com ponto de aumento dura tanto quanto uma colcheia e uma semicolcheia juntas. Se o ponto for duplo, a duração é aumentada na metade mais um quarto da duração da nota. Outra maneira de prolongar a duração da nota é através da *ligadura*. Um arco unindo duas notas vizinhas e da mesma altura significa que estas devem ser executadas como se fossem uma única nota com a duração da soma das duas. A ligadura também pode ser usada em notas de alturas diferentes, mas nesse caso tem outro significado. Existem ainda outros símbolos relacionados com a duração das notas (*staccatto* e *tercinas* por exemplo) mas a discussão completa do assunto foge ao escopo deste trabalho.

A-4 Barras de compasso

O *andamento* determina a velocidade da pulsação básica de uma música. Cada pulso é chamado de *tempo*. Os tempos são agrupados em *compassos* na partitura. Os compassos tem um número constante de tempos que é determinado pela *fórmula de compasso* colocada no início da música. Para separar um compasso de outro são usadas barras verticais que atravessam o pentagrama. Pelo fato de fornecer um elemento constante na música (o número de tempos por compasso) a identificação das barras de compasso oferece meios de verificar a exatidão do reconhecimento.

A-5 O volume

As informações a cerca do volume dos trechos de uma música são dadas através de abreviações das palavras italianas *forte*, *mezzo* e *piano*. São seis combinações para indicar o volume: *ff* - *fortissimo*, *f* - *forte*, *mf* - *mezzo forte*, *mp* - *mezzo piano*, *p* - *piano* e *pp* - *pianissimo*. Não há relação entre a intensidade indicada e algum valor estabelecido em decibéis. O quanto a intensidade de um trecho *ff* é maior que a de um trecho *mf* é bastante subjetiva e fica a critério do maestro ou instrumentista. Os sistemas de OMR raramente se preocupam com os símbolos de intensidade. Há ainda os símbolos de *diminuendo* e *crescendo*, representados por duas linhas em forma de cunha com o vértice para a direita e para a esquerda respectivamente. São usados para indicar variação gradual da intensidade.

Apêndice B

Resumo dos programas desenvolvidos.

B-1 Pré-processamento

É considerado pré-processamento o afinamento, vetorização e filtragem da imagem. Foram desenvolvidos tres programas para executar estes processos. Os resultados dos programas é apresentado na figura 12.

ESK.C - Implementa o algoritmo de afinamento descrito no capítulo 2. Usa como acabamento o algoritmo descrito na seção 2.7. O esqueleto é armazenado em formato PBM texto.

VET.C - Vetoriza o esqueleto gerado pelo programa ESK usando o algoritmo descrito no capítulo 3. A saída é a lista de membros do esqueleto vetorizado (ver seção 3.4).

REMOVMB.C - Retira da lista de membros os membros não conectados a outros membros e formados por um único vetor cujo módulo é menor ou igual ao valor passado como parâmetro. Este programa é usado para eliminar parte do ruído da imagem (seção 3.5).

B-2 Localização do pentagrama

Antes de localizar o pentagrama, os membros do esqueleto vetorizado são quebrados e os que não forem aproximadamente horizontais são descartados. Estes passos não são essenciais, mas melhoram o resultado.

QUEBRAMB.C - Fragmenta os membros de maneira que cada membro tenha apenas vetores com inclinações parecidas. A saída é uma lista temporária com os membros fragmentados (seção 4.1).

CLASMEMB.C - Seleciona apenas os membros cuja inclinação média esteja dentro da faixa passada nos parâmetros. Neste caso, próxima de zero (seção 4.2).

LOCPENTA.C - A partir dos membros selecionados, localiza o pentagrama e mede a distância mais frequente (em pixels) entre

suas linhas. Também mede a inclinação média do pentagrama. Estas informações são colocadas em um arquivo texto e ficam disponíveis para os programas seguintes (seções **4.3**, **4.4** e **4.5**).

B-3 Extração de características

Neste módulo é calculado o vetor de características de todos os pontos críticos da imagem.

AGRUPA.C - Agrupa os pontos críticos muito próximos e conectados em seu ponto médio. Este programa reduz consideravelmente o número de pontos na imagem. Embora o número de configurações possíveis entre os pontos seja praticamente infinito, este programa contribui para diminuir este número. Os pontos ficam mais "bem comportados" facilitando o reconhecimento (seção **5.1**).

TIRAPT.C - Retira da lista de membros os pontos considerados redundantes. Este programa também colabora para reduzir o número de configurações entre os pontos críticos (seção **5.2**).

PTMEDIO.C - Calcula e armazena o ponto médio, a inclinação e comprimento de todos os vetores da imagem (seção **5.3**).

PTJUNC.C - Obtem e armazena a lista de vetores que partem de cada ponto crítico da imagem (seção **5.4**).

REGIAO.C - Usando a saída dos dois programas anteriores e funções que atuam sobre a imagem binária, este programa calcula e armazena para os próximos módulos o vetor de características dos pontos críticos (seção **5.3**).

B-4 Treinamento

Este módulo alimenta o banco de dados do sistema. Exige que cada imagem fornecida seja acompanhada de uma lista das posições e nomes de todos os símbolos contidos nela.

PEGAMAX.C - Armazena os valores máximos e mínimos típicos dos atributos do vetor de características dos pontos críticos (seção **5.6**).

NORMALDR.C - Normaliza os valores dos atributos para uma escala entre zero e 100. Este programa também é usado no módulo de classificação (seção 5.6).

SEPARAS.C - Usando a lista de posições dos símbolos, cada ponto crítico da imagem é associado a um símbolo e uma posição em relação ao retângulo envolvente do símbolo. Estas informações são gravadas no banco de pontos. O programa também alimenta o banco de símbolos calculando o vetor de características dos símbolos fornecidos usando como domínio o retângulo envolvente correspondente (capítulo 6.).

B-5 Classificação

Aqui os candidatos a símbolos são construídos e selecionados. A saída do módulo (e também do sistema) é a uma lista de candidatos que contém o nome do símbolo, seu retângulo envolvente e um número que é proporcional à probabilidade de acerto do símbolo.

PTPROX.C - Obtêm a lista dos **W** pontos do banco de pontos mais próximos de cada ponto crítico da imagem. A lista é ordenada pela distância entre os pontos (seção 7.1).

CLASSPT.C - Constrói os candidatos com a lista de pontos próximos. Calcula também um peso proporcional ao número de pontos concordantes do candidato. Candidatos com peso muito baixo são descartados já neste nível (seção 7.2).

RETIRAS.C - Compara os candidatos com os símbolos armazenados no banco de símbolos e atribui um novo peso a cada candidato. Os candidatos com peso abaixo de um treshold fornecido nos parâmetros são descartados. A lista de candidatos resultante contem os pesos dos dois níveis (seção 7.3).