Universidade Federal de Pernambuco
Departamento de Eletrônica e Sistemas
Mestrado em Engenharia Elétrica

DISSERTAÇÃO DE MESTRADO

ESTRATÉGIAS PARA MELHORIA DO DESEMPENHO DE FERRAMENTAS COMERCIAIS DE RECONHECIMENTO ÓPTICO DE CARACTERES

Por: NEIDE FERREIRA ALVES

Orientador: Prof. Dr. RAFAEL DUEIRE LINS

Recife

2003

NEIDE FERREIRA ALVES

ESTRATÉGIAS PARA MELHORIA DO DESEMPENHO DE FERRAMENTAS COMERCIAIS DE RECONHECIMENTO ÓPTICO DE CARACTERES

Dissertação apresentada ao Departamento de Eletrônica e Sistemas da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica, sob a orientação do Prof. Dr. Rafael Dueire Lins.

Recife

2003

NEIDE FERREIRA ALVES

Estratégias para Melhoria do Desempenho de Ferramentas Comerciais de Reconhecimento Óptico de Caracteres

Aprovado em 08 de agosto de 2003

Dissertação apresentada como requisito parcial à obtenção do grau de mestre. Programa de Pós-Graduação em Engenharia Elétrica do Centro de Tecnologia e Geociências/Escola de Engenharia de Pernambuco, Universidade Federal de Pernambuco.

BANCA EXAMINADORA

Prof. Rafael Dueire Lins, PhD.
Orientador

Prof. Ascendino Flávio Dias e Silva, PhD.

Prof. Carlos Alexandre Barros de Mello, PhD.

AGRADECIMENTOS

À FUCAPI, por proporcionar a materialização de um sonho por tanto tempo acalentado, em particular ao Prof. Dr. Erno Ivan Paulinyi (in memorian).

À UFPE e UTAM, pela iniciativa em prol do desenvolvimento tecnológico da Região Amazônica.

Ao Prof. Dr. Rafael Dueire Lins, que sempre esteve presente na orientação deste trabalho, não poupando esforços na busca do seu aperfeiçoamento.

Aos professores, pelos ensinamentos e experiências.

Aos colaboradores da UTAM e UFPE, pelo afinco no desempenho de suas atividades.

Aos amigos e aos colegas de mestrado, pela torcida e incentivo nos momentos mais difíceis no decorrer deste estudo.

À minha família e em especial a Ariosto Júnior companheiro de todas as horas.

A todos que contribuíram direta e indiretamente para realização deste trabalho, minha eterna gratidão.

O verdadeiro vencedor é aquele que tem a coragem de calar o mundo em si e caminhar pela porta do coração.

(Mônica Buonfiglio)

RESUMO

Para avaliar a qualidade do desempenho de ferramentas comerciais de Reconhecimento Óptico de Caracteres (OCR) é necessário adquirir métricas para avaliar o quanto um texto transcrito está próximo do texto original, uma vez que quando uma imagem sofre alterações, por menores que sejam, estas influenciam nas transcrições dos OCR's.

Neste trabalho será apresentada uma nova métrica para avaliar transcrições de OCR's: através da aplicação de técnicas de filtragem (brilho, contraste, resolução, rotação, etc.) na imagem original, para que as mudanças mínimas gerem inúmeras imagens, as quais serão submetidas ao OCR e resultarão em textos distintos.

Um algoritmo foi desenvolvido para comparar os textos gerados, analisando desde a quantidade de linhas até a igualdade entre os caracteres. Através da análise de maior freqüência entre os caracteres, este algoritmo gera um novo arquivo-texto. Com o uso desta metodologia, o arquivo gerado ficou muito próximo do original com um índice de acerto maior que os arquivos transcritos sem o processo de filtragem.

ABSTRACT

To evaluate the performance of commercial Optical Character Recognition (OCR) tools it is necessary to acquire a metric for evaluating of how close a transcribed text is in comparison with the original text. This is important because even the minimum changes on the image may cause influences into the OCRs transcriptions.

In this work it will be presented a new metric to evaluate OCRs transcriptions through the application of filtering techniques (brightness, contrast, resolution, rotation, etc.) in the original image, so that the minimum changes generate innumerable images, which will be submitted to the OCR and will result in distinct texts.

An algorithm was developed to compare the generated texts, and analyze since the amount of lines until the equality among the characters. Through the analysis of bigger frequency around the characters, this algorithm will create a new archive text. Based on this methodology, one may conclude that the new archive text is more similar to the original archive than the one that has been digitalized with no treatment.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

INTRODUÇÃO

Este capítulo descreve a motivação e os objetivos desta Dissertação, bem como o contexto no qual está inserida. Apresenta-se, também, a idéia principal dos demais capítulos que constituem este trabalho.

Documentos em Papel *versus* Documentos Digitalizados

Atualmente, a maior parte das informações existentes nos diversos setores da atividade humana está impressa em papel. De acordo com o CENADEM (Centro Nacional de Desenvolvimento do Gerenciamento da Informação) [Cen03] nos últimos 50 anos, a humanidade gerou a mesma quantidade de informação que nos 5 mil anos anteriores.

Cada vez mais estamos gerando maiores quantidades de documentos em papel. Segundo a AIIM International (Association for Information and Image Management International) [AIIM03], EUA, a maior associação do mundo sobre gerenciamento da documentação, 95% das informações dos Estados Unidos estavam em papel em 1990. Em 2002, cerca de 92% das informações ainda estavam em papel. A cada dia 2,7 bilhões de novas folhas de papel são preenchidas e arquivadas. Essa avalanche de papel gera maiores problemas:

- Um executivo gasta em média quatro semanas por ano procurando documentos;
- Fazem-se, em média, 19 cópias de cada documento;
- Gasta-se US$ 250,00 para recriar cada documento perdido.

Segundo IDC-EUA [Cen03], estudos revelam que os escritórios criam cerca de 1 bilhão de páginas de papel por dia. Esse total é constituído de 600 milhões de páginas de relatórios de computador, 234 milhões de fotocópias e 24 milhões de documentos diversos. Isso somente nos Estados Unidos.

A história do papel [Cha03] está intimamente ligada à reprografia. De todo papel produzido no mundo, 90% é impresso. E foi como suporte reprográfico que o papel surgiu.

Desde os primórdios da humanidade, o homem vem desenhando as lembranças visuais de sua vida. Isto data de 30 mil anos atrás. Esses desenhos nas paredes das cavernas eram chamados de pictografias. Depois, se tornaram mais complexos, e vieram as ideografias e os cuneiformes persas até chegar nos hieróglifos egípcios por volta de 2.500 A.C.

Conclui-se que o desenvolvimento da inteligência humana veio acompanhado pelas representações gráficas, tornando-se cada vez mais complexas, chegando, desse modo, a representar idéias. A História registra o uso de diversos tipos de suportes para escrita, como tabletes de barro, tecidos de fibra vegetal, papiros, pergaminhos e, finalmente, o papel.

O papel possui vantagens óbvias, como portabilidade, facilidade de uso e baixo custo. Entretanto, o papel também possui várias desvantagens. Em determinado momento, entre três e cinco por cento dos arquivos de uma empresa são perdidos ou extraviados. Talvez a maior desvantagem do papel é que a busca por informações no computdor é normalmente mais fácil do que a impressa em papel

Obter a informação rapidamente ou prover "a informação correta, à pessoa correta, no tempo correto" é difícil e custoso com processos manuais em papel. Assim, enquanto o papel é a maior base de informação de um negócio, ele permanece fora dos sistemas de informação porque não é tão facilmente acessível tão pouco confiável.

Com o passar dos anos a quantidade de arquivos em papel torna-se cada vez maior: é incalculável a quantidade de papel existente para vários fins, desde o jornal que é produzido diariamente, às revistas semanais, mensais até mesmo os livros, documentos de empresas, materiais históricos (na maioria das vezes manuscritos), dinheiro para as transações comerciais do dia-a-dia, enfim, todos usam papel para registrar os fatos. O papel, por melhor que seja, tem um tempo de vida útil, além de ser frágil. A busca de informações armazenadas em papel normalmente é difícil devido a quantidade de espaço necessário para registrar os fatos. Como armazenar com segurança e ainda possibilitar a fácil acessibilidade das informações?

O problema do dinheiro praticamente foi solucionado com o uso dos cartões magnéticos: a dificuldade agora é somente cultural, pois a tecnologia já está disponível. A produção de jornais, revistas e livros também pode ser por meio eletrônico, pois muitas pessoas acessam a Internet para obter informações diariamente. É possível afirmar que a maioria das informações poderá ser acessada por meio eletrônico, porém o que fazer com os dados já produzidos? Como conservar por milhares de anos tanta informação? Pode-se digitalizar estes arquivos para obter imagens dos documentos, essas por sua vez também ocupam bastante espaço para armazenamento em meio magnético.

De acordo com [5, 31], uma imagem de texto em formato *Microsoft Windows Bitmap*®, mais conhecido como *BMP*, que ocupasse 300 Kilobytes para ser armazenada, no formato de texto (TXT), ocupa 3 Kilobytes, proporcionando uma redução de 99%. BMP é um formato de imagem nativo do sistema operacional Microsoft Windows®. A versão mais amplamente utilizada desse formato não apresenta nenhum tipo de compactação, o que traz uma demanda intensiva de espaço para armazenar as imagens.

A imagem de um documento de tamanho A4 digitalizado com resolução de 200 dpi[1] e comprimida a 10:1 requer aproximadamente 50 Kilobytes de armazenamento. Quinhentas páginas de texto requerem 1 Megabyte de armazenamento. Um Gigabyte acomoda 20 mil imagens.

Um arquivo de quatro gavetas, com 2.500 folhas de papel por gaveta, comporta, em média, 10 mil imagens de documento. Um CD-R mede 120mm de diâmetro e pode armazenar até 700 Megabytes de informação. Isso corresponde a 14 mil páginas de documentos.

Na Tabela 1 é feito um demonstrativo da durabilidade das mídias atuais, comparando as condições ambientais de temperatura e umidade relativa, assim como a durabilidade em anos.

A transposição de imagens para o formato de texto surge como uma solução para o problema de armazenamento de dados. O espaço ocupado por um arquivo de texto, como já exposto, é centenas de vezes menor que o ocupado por uma imagem, além de possibilitar a execução de mecanismos de busca por palavras-chave. Conforme [Mel02], uma transposição de imagem para texto não-automática é inaceitável devido aos custos envolvidos e às baixas velocidade e confiabilidade do processo. Um sistema automático deve reconhecer os caracteres presentes no documento, diferenciando-os de imagens ou outros dados que possam estar presentes e transpô-los para caracteres ASCII (*American Standard Code for Information Interchange* - padrão computacional de caracteres). Este processo é chamado de Reconhecimento Óptico de Caracteres (OCR - *Optical Character Recognition*). As dificuldades desse processo, basicamente estão em dois pontos:

1. A escolha do melhor método para realizar a transposição de imagem para texto;

[1] DPI (Dots Per Inch - Pontos Por Polegada) - Unidade de densidade de resolução usada para indicar a quantidade de pixels em uma polegada. Quanto maior a resolução de uma imagem melhor será a definição desta imagem.

2. A escolha dos melhores ajustes de parâmetros para digitalização já que ajustes inadequados de resolução, brilho, contraste, número de cores, etc., podem provocar uma atuação ineficiente dos algoritmos de reconhecimento, os quais são sensíveis às mudanças mínimas e podem requerer uma nova digitalização ou um processamento específico da imagem [Mel02].

Tabela 1. Durabilidade das Mídias

Nome da Mídia	Temperatura ºC	Umidade Relativa %	Durabilidade (Anos)
CD-ROM	40	80	2
	30	60	10
	20	40	50
	10	25	200
WORM	40	80	5
	30	60	20
	20	40	100
	10	25	200
CD-RW	40	80	2
	30	60	5
	20	40	30
	10	25	100
MAGNETO-ÓPTICO	40	80	2
	30	60	5
	20	40	30
	10	25	100
Microfilme com Qualidade Arquivística (Prata)	40	80	20
	30	60	50
	20	40	200
	10	25	500

Fonte: Cenadem [Cen03]

No próximo tópico está descrita uma proposta de desenvolvimento de um algoritmo capaz de amenizar as dificuldades existentes nas transcrições e na escolha de um texto muito próximo ao original.

Motivação e Objetivos da Dissertação

A digitalização de documentos e acervos em papel é geralmente executada através de scanners, que são aparelhos que capturam uma imagem e a convertem, num processo digital, para que possa ser armazenada em computador. O tópico *1.2.1 Aquisição de Imagem* do *Capítulo 1*, desta dissertação traz mais detalhes sobre os aspectros envolvidos na aquisição de imagens digitais.

Uma vez digitalizado o documento, os softwares de OCR são utilizados para a sua transcrição. A pesquisa descrita em [Mel02], mostrou que os OCR's são sensíveis às alterações mínimas na imagem. Desta forma qualquer mudança gera um arquivo com texto distinto. Os OCR's também são sensíveis à tonalidade do papel, textura, sensibilidade à rotação do documento, sombra e ruído, os quais são capturados e passados à imagem digitalizada.

Diante dos problemas citados o objetivo deste trabalho é capturar imagens de texto, passá-la em filtros para alterações mínimas de brilho, contraste, resolução, rotação, etc, buscando a melhoria das respostas dos OCR's.

O processo de transcrição no OCR criará um arquivo-texto para cada uma das imagens. Será desenvolvido um algoritmo para analisar cada arquivo-texto gerado e compará-los entre si. Como resultado da comparação será criado um novo arquivo, através da freqüência entre os caracteres. Com este novo arquivo pretende-se garantir uma maior eficiência na transposição de imagens para texto.

Organização desta Dissertação

Esta Dissertação é composta, além deste capítulo introdutório, de 5 (cinco) outros capítulos e as referências bibliográficas, conforme detalhamento feito a seguir:

Capítulo 1 - *Técnicas de Processamento de Imagens Digitais*: contém a revisão literária realizada para a pesquisa, assim como o conceito das técnicas para aquisição e extração de imagens de texto.

Capítulo 2 - *O Estado da Arte em OCR's*: descreve como os OCR's funcionam, os algoritmos e as técnicas usadas no reconhecimento de caracteres. Também apresenta a ferramenta de reconhecimento de caracteres *Omnipage*, a qual será utilizada nos experimentos de avaliação.

Capítulo 3 - *Métodos de Comparação entre Transcrições de Textos*: descreve a metodologia usada para avaliar ferramentas de OCR's, bem como os resultados obtidos com os experimentos.

Capítulo 4 - *Aplicação de Filtros para Melhoria de OCR's*: descreve a métrica e o algoritmo que foram criados para selecionar o texto com menor índice de erros dentre os vários oriundos de várias configurações, além dos resultados obtidos.

Finalizando com as *Conclusões*: esta seção apresenta uma análise crítica relativa aos resultados esperados e aos obtidos e também menciona as possibilidades de continuação deste trabalho.

1 TÉCNICAS DE PROCESSAMENTO DE IMAGENS DIGITAIS

Este capítulo trata dos conceitos e de algumas técnicas de manipulação de imagens digitais. A Figura 1, extraída de [GV94], mostra a relação entre imagens e dados assim como as áreas envolvidas nestes processos. Por serem áreas tão próximas é comum a confusão entre as aplicações. Um critério que pode ser utilizado para uma diferenciação baseia-se na natureza da entrada e saída de uma aplicação em cada área.

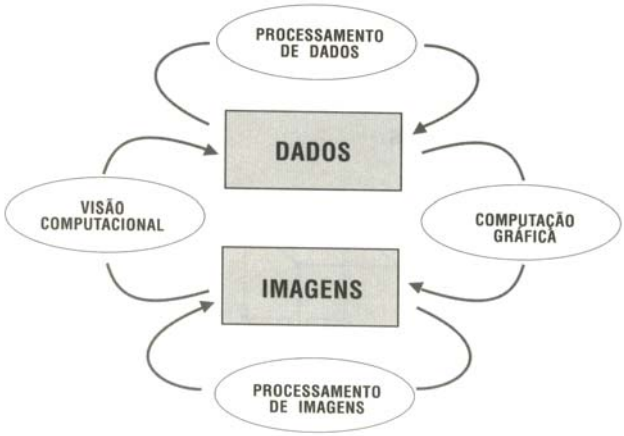


Figura 1. Relação entre Imagens, Dados e Áreas Correlatas

Na área de *Processamento de Dados*, o computador recebe dados, e após o processamento devolve, na saída, dados de mesma natureza.

Na *Computação Gráfica*, os dados de entrada são processados e o produto final é uma imagem que pode ser vista através de algum dispositivo de saída gráfica.

A área de *Visão Computacional* tem por finalidade obter, a partir de uma imagem (entrada), informações geométricas, topológicas ou físicas sobre o cenário que deu origem a essa imagem. As técnicas dessa área têm grande interesse na área de robótica, com o objetivo de prover o sentido da visão aos robôs.

Na área de *Processamento de Digital de Imagens*, o sistema admite uma imagem como entrada que, depois de processada, produz outra imagem na saída, ou seja, os dados de entrada e de saída do processo são imagens [GV94].

Na maioria das aplicações as quatro áreas acima atuam de um modo unificado e cooperativo. Desse moso, os dados resultantes de um sistema de processamento de imagens, ou de visão computacional, podem ser manipulados por alguma técnica de computação gráfica a fim de fornecer uma melhor informação qualitativa ao usuário.

Um dos objetivos de processamento digital de imagem é processar uma imagem, de modo que o resultado seja mais apropriado, para uma aplicação específica, do que a imagem original. Quando uma imagem é processada para interpretação visual, o observador é o árbitro final da qualidade do desempenho de um método particular. Quando o problema envolve processamento de imagens para percepção por máquina, a tarefa da avaliação é um pouco mais fácil. Entretanto, mesmo em situações em que um critério bem definido de performace possa ser imposto ao problema, o analista usualmente terá de realizar diversas tentativas antes de poder estabelecer uma abordagem adequada de processamento de imagens [GV94].

Segundo [FN99], de 1964 aos dias atuais, as aplicações de processamento de imagens permeiam quase todos os ramos da atividade humana.

A evolução da tecnologia de computação digital, bem como o desenvolvimento de novos algoritmos para lidar com sinais bidimensionais está permitindo uma gama de aplicações cada vez maior. Como resultado dessa evolução, a tecnologia de processamento digital de imagens vem ampliando seus domínios, que incluem as mais diversas áreas, como por exemplo: análise de recursos naturais e meteorologia por meio de imagens de satélites; transmissão digital de sinais de televisão ou fax; análise de imagens biomédicas, incluindo a contagem automática de células e exame de cromossomos; análise de imagens metalográficas e de fibras vegetais; obtenção de imagens médicas por ultra-som, radiação nuclear ou técnicas de tomografia computadorizada; aplicações em automação industrial envolvendo o uso de sensores visuais em robôs, etc.

Em Medicina, o uso de imagens no diagnóstico médico facilita a interpretação de imagens produzidas por equipamentos antigos bem como o desenvolvimento de novos equipamentos. Em Biologia, o processamento de imagens de microscópios, facilita as tarefas laboratoriais.

Nas áreas de Geografia, Sensoriamento Remoto, Geoprocessamento e Metereologia, o processamento e a interpretação de imagens capturadas por satélites auxiliam os seus

trabalhos. Além do uso de processamento de imagens nas áreas de Astronomia, Publicidade, Direito, Arqueologia e Segurança.

Para a transcrição de imagem para texto, precisamos de imagens com qualidade suficiente para extrair caractere a caractere com a maior fidelidade possível. Qualquer ruído ou rotação na captura da imagem pode prejudicar o processamento da imagem e acarretar em erros para todo o processo de reconhecimento.

1.1 Representação de Imagens Digitais

Uma imagem digital é uma matriz de números reais ou complexos representados por um número finito de bits [Jai89].

Matematicamente, uma imagem digital é uma função *f(x,y)* discretizada tanto em coordenadas espaciais quanto em brilho [GW92]. Uma imagem digital pode ser considerada como sendo uma matriz cujos índices de linhas e de coluna identificam um ponto na imagem e o correspondente valor do elemento da matriz identifica o nível de cinza naquele ponto. Os elementos dessa matriz digital são chamados de elementos da imagem, elementos da figura, "*pixels*" ou "*pels*", estes dois últimos, abreviações de "*picture elements*".

Os elementos da imagem consistem, essencialmente, das coordenadas dos pixels e da informação de cor de cada pixel. Esses dois elementos estão diretamente relacionados com a resolução espacial (número de linhas x número de colunas da matriz de pixels) e resolução de cor (número de bits utilizado para armazenar o vetor de cor de cada pixels) da imagem.

O número de componentes do pixel é a dimensão do espaço de cor utilizado. Dessa forma, cada pixel em uma imagem digital monocromática tem uma única componente.

O *gamute* de uma imagem digital é o conjunto das cores do espaço de cor quantizado da imagem. Uma imagem monocromática cujo gamute possui apenas duas cores é chamada de imagem de dois níveis, imagem bitmap ou imagem binária. Uma imagem monocromática cujo gamute possui mais de dois níveis é chamada de imagem meio-tom (*halftone*) ou imagem com escala de cinza (*grayscale*) [GV94].

Cada pixel tem como atributo principal sua cor. As cores formam um espaço de cromaticidade, um sub-espaço contínuo do R3. Chama-se palheta (ou paleta) de cores ao conjunto finito de valores discretos utilizado para representar, computacionalmente, o espaço de cromaticidade. A palheta de menor tamanho possui apenas uma cor e pode ser representada em apenas 1 *bit*, que pode ter o valor (0) zero para indicar o branco e o (1) um, para o preto.

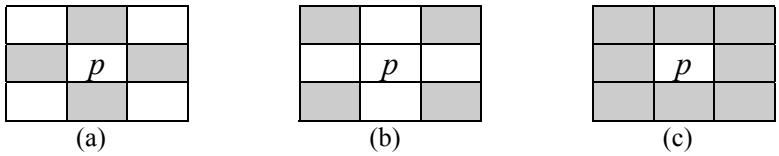
Atualmente, as imagens computacionais conhecidas como *true color* fazem uso de uma palheta com cerca de 16 milhões de cores e, desta forma, são necessários 24 bits para poder representar as cores possíveis de cada pixel [Mac01]. As características normalmente usadas para distinguir uma cor da outra são *brilho*, *matiz* e *saturação*:

- Brilho - incorpora a noção cromática da intensidade;
- Matiz - é um atributo associado com o comprimento de onda dominante em uma mistura de ondas de luz, deste modo o matiz representa a cor dominante;
- Saturação - refere-se à pureza relativa ou quantidade de luz branca misturada com o matiz.

O matiz e a saturação, quando tomados juntamente, são chamados *cromaticidade*, portanto uma cor pode ser caracterizada pelo seu brilho e cromaticidade.

O termo "imagem monocromática", ou simplesmente "imagem", refere-se à função bidimensional de intensidade da luz *f(x,y)*, onde *x* e *y* denotam as coordenadas espaciais e o valor de *f* em qualquer ponto (x, y) é proporcional ao brilho (ou níveis de cinza) da imagem naquele ponto. Às vezes se torna útil a visualização da função da imagem em perspectiva com um terceiro eixo representando o *brilho* [GW92].

Normalmente são utilizadas letras minúsculas para identificar um pixel em particular. Um pixel *p*, de coordenadas (x, y), tem 4 vizinhos horizontais e verticais, cujas coordenadas são (x+1, y), (x-1, y), (x, y+1), (x, y-1), como na Figura 2(a). Estes pixels formam a chamada "4-vizinhança" de *p*. Os quatros vizinhos diagonais de *p* são os pixels de coordenadas (x-1, y-1), (x-1, y+1), (x+1, y-1), (x+1, y+1), como na Figura 2(b). A junção dos 4 vizinhos horizontais e verticais mais os 4 vizinhos diagonais formam a chamada "8-vizinhança", na Figura 2(c), extraída de [FN99], esses vizinhos são ilustrados.



(a) (b) (c)

Figura 2. Conceitos de vizinhança

As operações lógicas e aritméticas orientadas à vizinhança utilizam o conceito de **convolução com máscaras** (ou janelas ou *templates*), essas operações são amplamente utilizadas tanto no processamento de imagens como na redução de ruído, afinamento, realce e detecção de características da imagem. Todas essas operações serão detalhadas nas próximas seções.

1.2 Etapas Fundamentais em Processamento de Imagem

Esta seção aborda as etapas fundamentais em processamento de imagem: aquisição, armazenamento, pré-processamento, segmentação, representação, reconhecimento e interpretação, base de reconhecimento e exibição. A Figura 3, extraída de [GW92], mostra a interação entre os elementos de um sistema de processamento de imagem.

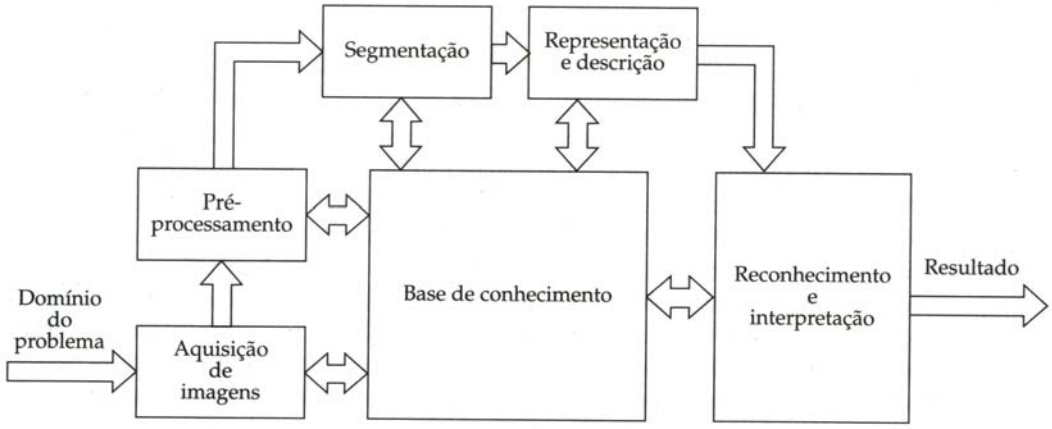


Figura 3. Etapas fundamentais em processamento de imagens

1.2.1 Aquisição de Imagem

O primeiro passo no processamento de uma imagem é a aquisição da mesma. Esse passo ainda é o mais delicado e dependente do usuário.

Dois elementos são necessários para a aquisição de imagens digitais. O primeiro é um **sensor**, dispositivo físico que seja sensível a uma banda do espectro de energia eletromagnética (como raios X, ultravioleta, espectro visível, ou banda infravermelha) e que produza um sinal elétrico de saída proporcional a um nível de energia percebida, ou seja, converterá a informação óptica em sinal elétrico. O sensor poderia ser uma câmera de TV monocromática ou colorida que produza uma imagem inteira do domínio do problema a cada 1/30s.

O segundo chamado **digitalizador** (*scanner*, câmera digital, filmadora com placa de vídeo) é um dispositivo para conversão da saída elétrica de um dispositivo de sensoriamento físico para a forma digital, isto é, converte o sinal elétrico analógico em informação digital, e pode ser representado através de bits 0s e 1s.

A aquisição de imagem também é conhecida como *frame grabber*. Diversos problemas de digitalização podem afetar o formato de um caractere, podendo, inclusive, perder pixels, deixando “buracos” os quais podem dificultar o reconhecimento.

A determinação da resolução utilizada na digitalização é essencial para um perfeito processo de reconhecimento. Um aumento na resolução implica no aumento do número de pixels na imagem e, conseqüentemente, no aumento na quantidade de informação necessária para processar e armazenar. Qualquer alteração necessária na resolução da imagem deve ser feita através de uma nova digitalização. Dessa forma, é essencial a escolha de um valor ótimo, levando em consideração o tamanho do arquivo resultante, a maneira como tais imagens serão visualizadas, a velocidade de digitalização, a velocidade de transmissão via rede, o custo dos dispositivos de digitalização, transmissão, armazenamento, visualização, etc. Observações também devem ser feitas quanto à dimensão da palheta de cores (número máximo de cores) escolhida para digitalização. Mesmo imagens com poucas cores (como documentos, onde, geralmente, as cores predominantes são as da tinta e do papel), sugerem uma digitalização em *true color* permitindo a captação de nuances de tonalidade nos documentos. Isso garante uma melhor qualidade na imagem final a ser processada.

1.2.2 Armazenamento

Após adquirir uma imagem será necessário armazená-la. Como, em altas resoluções as imagens, ocupam um número elevado de bytes, o armazenamento pode ser dividido em três categorias:

- Armazenamento de curta duração, enquanto a imagem é utilizada nas várias etapas do processamento;
- Armazenamento de massa, para operações de recuperação de imagens relativamente rápidas; e
- Arquivamento de imagens, para recuperação futura quando isto se fizer necessário.

No armazenamento de curta duração, parte da memória RAM do computador principal é utilizada. Outra opção é o uso de placas especializadas, chamadas *frame buffers*, as quais permitem um armazenamento maior e com acesso rápido.

Para o armazenamento em massa, o uso de discos magnéticos de algumas centenas de megabytes é necessário, ou ainda, discos magneto-ópticos agrupados em *jukeboxes*.

O arquivamento de imagens para recuperação futura e esporádica requer o uso de discos ópticos WORM (*Write-Once-Read-Many*), os quais estão substituindo as fitas magnéticas pela grande capacidade de armazenamento até mesmo em terabytes.

1.2.3 Pré-Processamento

Após a fase de aquisição e de armazenamento da imagem, inicia-se a fase de pré-processamento. A função chave no pré-processamento é melhorar a imagem de forma a aumentar as chances para o sucesso dos processos seguintes. Algoritmos eficientes de processamento de imagens devem ser aplicados antes do reconhecimento de caracteres, procurando corrigir defeitos na imagem ou extrair dados não só inúteis para o reconhecimento, mas também que podem dificultar o processo. O pré-processamento tipicamente envolve técnicas para o realce de contraste, remoção de ruído, isolamento de regiões cuja informação está concentrada e a identificação de possível erro, oriundo de rotação durante o processo de digitalização.

Todos esses aspectos serão abordados com mais detalhe no tópico sobre as técnicas usadas no pré-processamento.

1.2.4 Segmentação

Definida em termos gerais, a segmentação divide uma imagem de entrada em partes ou objetos constituintes, ou seja, particiona a imagem em regiões, onde cada região contém uma entidade completa e isolada. Em geral, a segmentação automática é uma das tarefas mais difíceis no processamento de imagem digital. Por um lado, um procedimento de segmentação robusto favorece substancialmente a solução bem sucedida de um problema de imageamento. Por outro lado, algoritmos de segmentação fracos quase sempre asseveram falha no processamento. No caso de reconhecimento de caracteres, o papel básico da segmentação é extrair caracteres individuais e palavras do fundo da imagem.

A saída do estágio de segmentação é constituída tipicamente por dados em forma de pixels, correspondendo tanto à fronteira de uma região como a todos os pontos dentro da mesma. Em ambos os casos, converter os dados para uma forma adequada ao processamento computacional é necessário.

1.2.5 Representação

Após a segmentação é necessário decidir se os dados devem ser representados como fronteiras ou como regiões completas. A representação por fronteira é adequada quando o interesse se concentra nas características da forma externa, tais como cantos ou pontos de inflexão. A representação por região é adequada quando o interesse se concentra em propriedades internas, tais como textura ou a forma do esqueleto. Em algumas aplicações,

entretanto, essas representações coexistem, como no caso das aplicações de reconhecimento de caracteres, que freqüentemente requerem algoritmos baseados na forma da borda, bem como também esqueletos e outras propriedades internas.

A escolha de uma representação é apenas parte da solução para transformar os dados iniciais numa forma adequada para o subseqüente processamento computacional. Um método para descrever os dados também deve ser especificado, de forma que as características de interesse sejam enfatizadas. O processo de descrição, também chamado *seleção de características*, procura extrair características que resultem em alguma informação quantitativa de interesse ou que sejam básicas para discriminação entre classes de objetos.

1.2.6 Reconhecimento e Interpretação

Reconhecimento é o processo que atribui um rótulo a um objeto, baseado na informação fornecida pelo seu descritor. A interpretação envolve a atribuição de significado a um conjunto de objetos reconhecidos. A identificação de um caractere *c*, por exemplo, requer a associação dos descritores para aquele caractere com rótulo *c*. A interpretação procura atribuir significado a um conjunto de entidades rotuladas, como uma cadeia de cinco números seguidos por um hífen mais três números, pode ser interpretada como um código de endereçamento postal.

1.2.7 Base de Conhecimento

O conhecimento sobre o domínio do problema está codificado em um sistema de processamento de imagens na forma de uma base de conhecimento. Esse conhecimento pode ser tão simples quanto o detalhamento de regiões de uma imagem, no qual sabe-se que a informação de interesse pode ser localizada, limitando assim a busca que precisa ser conduzida na procura por aquela informação. Por exemplo, se a etapa de reconhecimento e interpretação recebesse *7* (sete) caracteres ao invés de *8* (oito), como no caso específico do CEP, a base de conhecimento deveria ser capaz de realimentar a etapa de segmentação (provável responsável pela falha) para que esta procurasse segmentar novamente a subimagem “suspeita” (aquela de maior largura), buscando dividí-la em duas.

1.2.8 Exibição

Os monitores de vídeo e as impressoras são os elementos fundamentais para exibição de imagens. Os monitores são capazes de exibir imagens com resolução de pelo menos

640x480 pixels com 256 cores distintas. Os monitores mais recentes trabalham com a tecnologia de cristal líquido, a qual possui uma propriedade física que possibilita o seu uso em dispositivos de geração de imagens: quando não excitado eletricamente se apresenta opaco, caso ocorra o contrário suas moléculas se orientam de maneira a permitir a passagem de luz.

Existem diversas formas de reprodução de imagens em papel. A melhor, e mais cara, é a reprodução fotográfica, onde o número de gradações de cinza é função da densidade dos grânulos de prata no papel. Dispositivos periféricos de saída, especializados na produção de cópias da imagem em forma de fotografias, slides ou transparências, também estão se tornando cada vez mais usuais.

1.3 Técnicas usadas no Pré-Processamento de Imagem

Como foi citado anteriormente o pré-processamento de imagens procura corrigir os defeitos na imagem através de algoritmos eficientes.

Para melhorar o contraste e o contorno das imagens utiliza-se o algoritmo de equalização do histograma. Depois transforma-se a figura de tons de cinza para preto e branco com o auxílio de alguns filtros (técnica conhecida como binarização). Para melhorar a qualidade das imagens é necessário utilizar técnicas de realce de modo que a imagem resultante seja mais adequada que a imagem original. Algumas destas técnicas são filtragem ou suavização e realce de imagem. Outras são usadas para rotacionar imagem, extrair ruído, afinar caracteres, essas técnicas podem ser usadas diretamente em reconhecimento de caracteres.

As técnicas de melhoria da qualidade de imagens podem ser divididas em duas famílias: as de realce e as de restauração de imagens. Quando a melhoria é usada para combater um processo de degradação conhecido ou avaliado por métodos da teoria da filtragem, a palavra usada é restauração de imagens. A restauração difere de realce pelo fato de que a primeira procura obter a imagem “real” tendo, se possível, um conhecimento a priori da degradação.

1.3.1 Histograma

De acordo com [FN99], o histograma de uma imagem pode ser visto como a representação em percentual de pixels que apresentam um determinado nível de cinza. Estes valores são normalmente representados por um gráfico de barras que fornece para cada nível de cinza o número (ou percentual) de pixels correspondentes na imagem. Através da visualização do histograma de uma imagem obtém-se uma indicação de sua qualidade quanto ao nível de contraste e quanto ao seu brilho médio (se a imagem é predominantemente clara ou escura).

Cada elemento deste conjunto é calculado através da função discreta:

$$p_r(r_k) = \frac{n_k}{n}$$

onde:

$0 \leq r_k \leq 1$;

r_k = é o k-ésimo nível de cinza;

k = 0, 1, ..., L-1, onde L é o número de níveis de cinza da imagem digitalizada;

n = número total de pixels na imagem;

$p_r(r_k)$ = probabilidade do k-ésimo nível de cinza;

n_k = número de pixels cujo nível de cinza corresponde a k.

De acordo com [GW92], um gráfico dessa função para todos os valores de k fornece uma descrição global da aparência de uma imagem. Na Figura 4, segue um exemplo de histograma com os quatro tipos básicos de imagens.

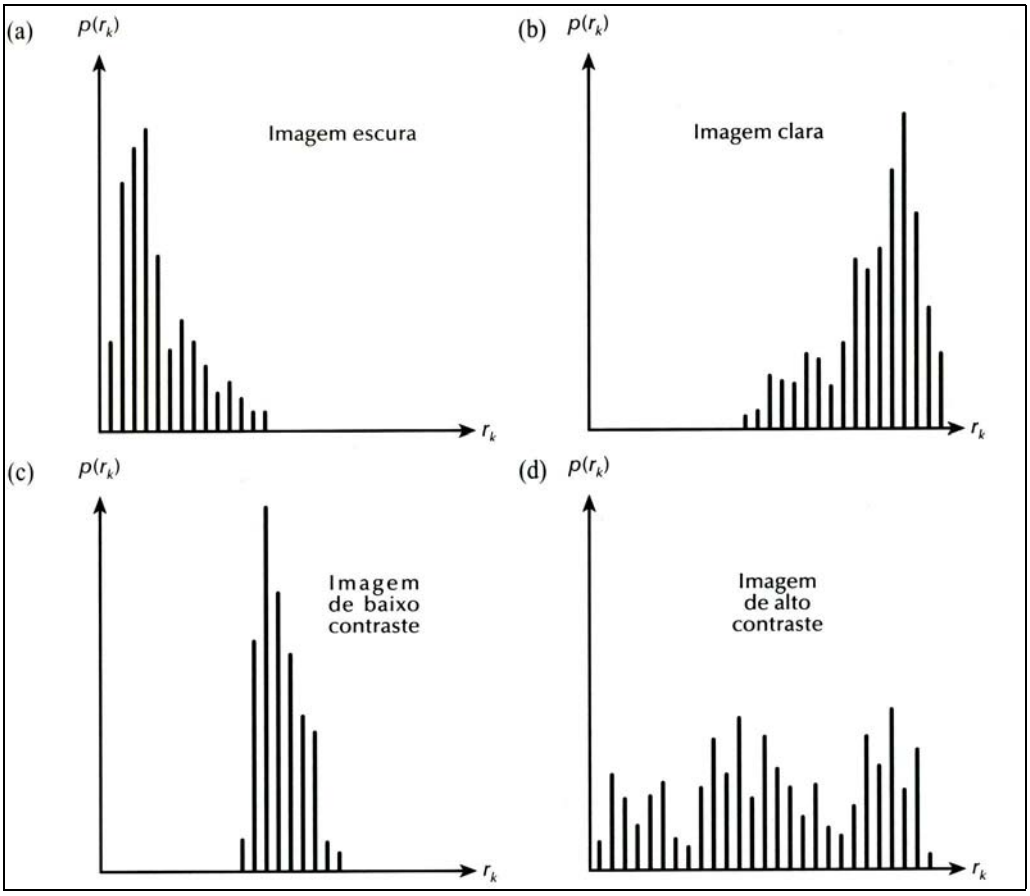


Figura 4. Histogramas correspondentes a quatro tipos básicos de imagens

- A Figura 4 (a) mostra que os níveis de cinza estão concentrados em direção à extremidade escura do intervalo de níveis de cinza, ou seja, esse histograma corresponde a uma imagem com características predominantemente escuras;
- Na Figura 4 (b) verifica-se o oposto da Figura 4(a), neste caso a imagem possui características de uma imagem clara;
- Um histograma muito estreito como na Figura 4(c) implica em uma imagem com baixo contraste;
- E um histograma com espalhamento significativo, chamado de bimodal, implica em um objeto que está em contraste com o fundo ou uma imagem com alto contraste.

Se uma imagem não está utilizando todos os níveis de cinza disponíveis, pode-se alterá-la, para melhorar o contraste. Para manipular o histograma de uma maneira consistente e significativa usa-se o processo de equalização.

Equalizar o histograma significa obter a máxima variância do histograma de uma imagem, obtendo assim uma imagem com o melhor contraste. O contraste é uma medida qualitativa e que está relacionada com a distribuição dos tons de cinza em uma imagem.

Como foi dito, um histograma bem distribuído, ou seja, uma imagem que utiliza quase toda a gama de tons de cinza, se apresenta com um alto contraste facilitando assim algum processamento futuro. A forma mais usual de se equalizar um histograma é utilizar a função de distribuição acumulada (cdf - *cumulative distribution function*) da distribuição de probabilidades originais, que é expressa por:

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} = \sum_{j=0}^{k} p_r(r_j)$$

onde:

$0 \leq r_k \leq 1$;

$k = 0, 1, ..., L-1$;

1.3.2 Filtragem

Qualquer imagem adquirida por meio óptico, eletro-óptico ou eletrônico está susceptível a ser degradada pela sensibilidade ao meio. A degradação pode ser na forma de ruído, turbulência na imagem, etc. Como exemplo, o ruído pode ser causado por falta de foco da câmera, enquanto a turbulência da imagem, do movimento do objeto relativo a câmera. A restauração da imagem é feita por filtragem, buscando minimizar os efeitos da degradação. A eficácia da restauração da imagem pelos filtros depende do grau de precisão e do conhecimento do processo de degradação, bem como dos critérios do projeto dos filtro [Jai89]..

Considerando que uma imagem digital seja representada por uma matriz, as operações unárias com imagens são chamadas de filtragem. O uso da filtragem em um sistema de processamento de imagem é de extrema importância nas diversas etapas do processo de síntese de imagens [GV94]. Na etapa de pós-processamento vários filtros são também usados com finalidades diversas:

- Transformada de uma imagem com a finalidade de obter imagens com diferentes resoluções de forma a compatibilizar a imagem com diversos dispositivos gráficos de saída;
- O uso de filtros para obter efeitos especiais, como o de uma imagem desfocada.

Nas técnicas de filtragem, o processamento de um nível de cinza de um pixel depende dos valores de nível de cinza desse pixel e de seus pixels vizinhos. Em geral, na vizinhança, os pixels mais próximos contribuem mais na definição do novo valor de nível de cinza do que os pixels mais afastados [Fac93].

As técnicas de filtragem e realce podem ser divididas em técnicas no domínio espacial e técnicas no domínio freqüencial. Nos próximos tópicos será abordado o funcionamento de cada um desses princípios.

a) Filtragem

As técnicas de filtragem no domínio espacial são aquelas que atuam diretamente sobre a matriz de pixels ou imagem digitalizada.

A base matemática das técnicas de filtragem no domínio da freqüência é o teorema da convolução, como citado no tópico sobre representação de imagens digitais. Seja g(x,y) a imagem formada pela convolução (denotada pelo símbolo *) da imagem f(x,y) com um operador linear h(x,y), ou seja, g(x,y) = f(x,y) * h(x,y).

De acordo com o teorema da convolução a seguinte relação no domínio da freqüência é verificada: G(u,v) = H(u,v) F(u,v), em que G, H e F são transformadas de Fourier de g, h e f, respectivamente [GW92]..

b) Filtros de Suavização ou Filtragem Passa-Baixas

Filtros de suavização são usados para borramento e redução de ruído. O borramento é utilizado em pré-processamento, tais como remoção de pequenos detalhes de uma imagem antes da extração de objetos (grandes), e conexão de pequenas descontinuidades em linhas e curvas. A redução de ruídos pode ser conseguida pelo borramento com filtro linear assim como por filtragem não linear. Se o objetivo for alcançar a redução de ruído em vez de borrar, uma abordagem alternativa consiste no uso de filtros por mediana. Isto é, o nível de cinza de cada pixel é substituído pela mediana dos níveis de cinza na vizinhança daquele pixel, ao invés da média. Esse método é particularmente efetivo quando o padrão de ruídos apresentar componentes fortes do tipo espingarda (“spike”), sendo que a característica a ser preservada é a agudeza das bordas. Dentre as técnicas mais conhecidas de suavização estão a filtragem pela média e o filtro da mediana [FN99].

- Filtro da média – a forma mais simples de implementar um filtro passa-baixas (em que todos os coeficientes são positivos) é construir uma máscara 3x3 com todos seus coeficientes iguais a 1, dividindo o resultado da convolução por um fator de normalização, neste caso igual a 9. Este filtro é denominado *Filtro da Média*;

- Filtro da mediana – uma das principais limitações do filtro da média, em situações onde o objetivo é remoção de ruídos em imagens, está na sua incapacidade de preservar bordas e detalhes finos da imagem. Para contorná-las, uma técnica alternativa é o filtro da mediana. Nesta técnica, o nível de cinza do pixel central da janela é substituído pela mediana dos pixels situados em sua vizinhança. A mediana m de um conjunto de n elementos é o valor tal que metade dos n elementos do conjunto situem-se abaixo de m e a outra metade acima de m. Quando n é impar, a mediana é o próprio elemento central do conjunto ordenado. Nos casos em que n é par, a mediana é calculada pela média aritmética dos dois elementos mais próximos do centro.

c) Realce

O principal objetivo das técnicas de realce é o de destacar detalhes finos na imagem. No domínio espacial serão abordados os métodos: filtro passa-altas, realce por diferenciação e ênfase em alta freqüência.

- Filtro passa-altas - atenuam ou eliminam os componentes de alta-freqüência enquanto deixam as freqüências baixas inalteradas. Os componentes de alta-freqüência caracterizam bordas e outros detalhes finos de uma imagem, de forma que o efeito resultante da filtragem passa-baixas é o borramento da imagem. Do mesmo modo, filtros passa-altas atenuam ou eliminam os componentes de baixa-freqüência, como esses componentes são responsáveis pelas características que variam lentamente em uma imagem, tais como o contraste total e a intensidade média, o efeito resultante da filtragem passa-altas é uma redução destas características, correspondendo a uma aparente intensificação bordas e outros detalhes finos. Filtragem passa-banda remove regiões selecionadas de freqüência entre altas e baixas freqüências. Esses filtros são usados para restauração de imagens e raramente são interessantes para realce de imagens. Uma imagem

filtrada por passa-altas pode ser computada como a diferença entre a imagem original e a versão filtrada passa-baixas daquela imagem [GW92]:

Passa-altas = Original - Passa-baixas

- Realce por diferenciação – sabendo-se que o cálculo da média dos pixels, em um trecho de imagem, produz como efeito a remoção de seus componentes de alta freqüência e que o conceito de média é análogo à operação de integração, é razoável esperar que a diferenciação produza o efeito oposto e, portanto, enfatize os componentes de alta freqüência presentes em uma imagem. O método mais usual de diferenciação em aplicações de processamento de imagens é o gradiente. Em termos contínuos, o gradiente de *f(x,y)* em um certo ponto *(x,y)* é definido como o vetor:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

A magnitude deste vetor é dada por:

$$\nabla f = mag(\nabla f) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

e é utilizada por várias técnicas de realce de imagens por diferenciação [FN99];

- Ênfase em alta freqüência ou Filtragem *high-boost* – a imagem original é multiplicada por um fator de amplificação *A* e subtraído da versão filtrada passa-baixas. Logo, quando A=1, o filtro se comporta de forma idêntica a um passa-altas. Nos casos em que A>1, parte da imagem original é adicionada ao resultado, restaurando parcialmente os componentes de baixa freqüência. O resultado é uma imagem que se parece com a original, com um grau relativo de realce das bordas, dependente do valor de *A*. O processo genérico de subtração de uma imagem borrada da imagem original é conhecido na literatura como *unsharp masking* [GW92].

Passa-altas = (*A*) (Original) - Passa-baixas

O Filtro high-boost pode ser implementado utilizando a máscara

$$1/9 * \begin{bmatrix} -1 & -1 & -1 \\ -1 & w & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

onde w = 9A - 1, com A ≥ 1.

1.3.3 Limiarização

O princípio da limiarização ou *threshold* consiste em separar as regiões de uma imagem quando esta apresenta duas classes (o fundo e o objeto) [FN99]. Devido ao fato da limiarização produzir uma imagem binária na saída, o processo também é denominado, muitas vezes, **binarização**. A forma mais simples de limiarização consiste na bipartição do histograma, convertendo os pixels cujo tom de cinza é maior ou igual a um certo valor de limiar (*T*) em brancos e os demais em pretos.

Matematicamente, a operação de limiarização pode ser descrita como uma técnica de processamento de imagens na qual uma imagem de entrada *f(x,y)* de N níveis de cinza produz na saída uma imagem *g(x,y)*, chamada de imagem limiarizada, cujo número de níveis de cinza é menor que *N*. Normalmente, *g(x,y)* apresenta 2 níveis de cinza, sendo:

$$g(x,y) = 1 \text{ se } f(x,y) \geq T$$
$$= 0 \text{ se } f(x,y) < T$$

onde os pixels rotulados com **1** (um) correspondem aos objetos e os pixels etiquetados com **0** (zero) correspondem ao fundo (*background*) e **T** é um valor de tom de cinza predefinido, ao qual denomina-se **limiar**.

A limiarização pode ser vista como uma operação que envolve um teste com relação a uma função *T* do tipo $T = T\,[x, y, p(x, y), f(x, y)]$, onde *f(x, y)* é o tom de cinza original no ponto *(x, y)* e *p(x, y)* indica alguma propriedade local deste ponto, por exemplo a média de seus vizinhos. Quando *T* depende apenas de *f(x, y)*, o limiar é chamado **global**; quando *T* depende de *f(x, y)* e de *p(x, y)*, o limiar é chamado **local**. Se, além disso, *T* depende das coordenadas espaciais de *(x, y)*, o limiar é chamado **dinâmico** ou **adaptativo**.

1.3.4 Rotação

Caso o documento tenha sido mal-posicionado sobre a mesa de digitalização, a imagem gerada poderá sofrer uma inclinação em seu eixo. Essa inclinação pode gerar falhas

na etapa final de reconhecimento. A rotação da imagem de um documento pode ser tratada principalmente pelo uso da *Transformada de Hough* [GW92]. Outros métodos, porém, também apresentam bons resultados, o sistema de detecção de grau de rotação é proposto baseado no cálculo do máximo de uma função de energia de um conjunto de projeções de caracteres.

Uma imagem pode ser rotacionada de um ângulo arbitrário, tanto no sentido horário quanto no anti-horário. Rotações com ângulos múltiplos de 90º são mais simples de implementar, pois consistem na cópia de pixels que estão organizados em linhas, reordenando-os em colunas na direção em que se deseja rotacionar a imagem. A rotação por ângulos quaisquer é uma tarefa mais complexa. Matematicamente, a rotação de cada ponto (X, Y) de uma imagem por um ângulo arbitrário *Ang*, mapeará este ponto na localidade de coordenadas (X`, Y`), onde X` e Y`são calculados pelas equações:

$$X` = X \cos(Ang) + Y \operatorname{sen}(Ang)$$

$$Y` = Y \cos(Ang) - X \operatorname{sen}(Ang)$$

1.3.5 Ruído

Define-se como ruído a presença de pixels na imagem que não correspondem a informação original (obtidos algumas vezes, através da digitalização) ou que não contém informação relevante ao processo em estudo. No caso da digitalização a partir de papéis, o ruído pode vir do próprio papel (tais como, rugosidade devido a dobras ou amassados, fungos, manchas de umidade, manchas de manuseio, respingos da tinta utilizada na escrita ou na impressão, etc.), não sendo acrescentado pelo scanner. Outros tipos de ruídos, no entanto, são devidos ao próprio processo de digitalização da imagem real, como a junção de caracteres [2, 18].

Existem muitos tipos de ruídos que podem estar presentes em imagens, como o ruído **sal e pimenta**, que normalmente ocorre devido a defeitos no sistema de geração da imagem. Esse ruído contém apenas dois valores na escala de cinza, os pixels ruidosos brancos são chamados sal, enquanto os pixels de ruído preto são chamados pimenta. A Figura 5, extraída de [Mel02], mostra um exemplo de uma imagem sob a presença de ruídos.

Figura 5. Ruído em imagens monocromáticas

1.3.6 Afinamento

A operação denominada afinamento ou esqueletização remove todos os pixels redundantes de uma imagem produzindo uma nova imagem simplificada com largura de um único pixel. O problema para os algoritmos de afinamento é o de determinar quais pixels são redundantes em uma imagem. Por outro lado, o processo de afinamento é muito melhor que o processo de erosão (operação morfológica, que será detalhada posteriormente), pois os pixels que devem ser removidos são primeiramente marcados e posteriormente removidos, em um segundo estágio. Este processo é repetido até que não existam mais pixels redundantes, até o ponto que os pixels remanescentes são aqueles que pertencem ao esqueleto do objeto. O esqueleto do objeto precisa permanecer intacto e deve respeitar as seguintes propriedades:

- As regiões afinadas precisam ter um pixel de largura;
- Os pixels que formam o esqueleto precisam permanecer próximos do centro da região de cruzamento de regiões;
- É necessário que os pixels do esqueleto formem o mesmo número de regiões que a imagem original apresentava.

Será descrito o método de esqueletização elaborado por Zhang-Suen [ZS84]. A idéia básica do método é decidir se um determinado pixel será eliminado olhando somente seus 8-vizinhos. Existem duas regras para decidir se o pixel deve ou não ser removido.

A primeira regra diz que o pixel somente pode ser apagado, se o número de conectividade do mesmo for igual a um. Isto significa que o pixel é conectado somente a uma única região. Se um pixel possuir o número de conectividade igual a dois então, duas regiões conectadas poderão separar e isto viola a terceira propriedade de esqueletização.

A segunda regra diz que o pixel somente pode ser apagado se este tiver mais de um e menos de **7** (sete) vizinhos. Esta regra assegura que os pixels resultantes foram retirados sucessivamente das bordas da região da imagem, e não de suas partes internas. Para afinar

uma região estas regras devem ser aplicadas para todos os pixels que pertencem à região, e os pixels que satisfazem às condições acima podem ser removidos. A seguir é descrito o algoritmo de Zhang-Suen.

a) Algoritmo de Afinamento de Zhang-Suen

O algoritmo é dividido em duas sub-iterações. Em uma sub-iteração, o pixel I(i,j) é eliminado (ou marcado para deleção) dependendo do valor verdade das seguintes condições:

1. O número de conectividade é 1

P9	***P2***	***P3***
P8	***P1***	***P4***
P7	***P6***	***P5***

   o número de conectividade é definido como sendo o número de transições de **branco para preto**, nos pixels que circundam o pixel central (iniciando em P2 e terminando em P9) que deve ser **exclusiva** uma.

2. Existem ao menos dois pixels vizinhos pretos e não mais do que seis;

P9	*P2*	*P3*
P8	***P1***	*P4*
P7	*P6*	*P5*

   o número de vizinhos refere-se também aos pixels na faixa P2, ..., P9 que não são fundo.

3. Ao menos um dos I(i,j+1), I(i-1,j) e I(i, j-1) são fundo da imagem (branco);

P9	***P2***	*P3*
P8	***P1***	***P4***
P7	*P6*	*P5*

   a condição se refere aos pixels P2, P4 e P8.

4. Ao menos um dos I(i-1,j), I(i+1,j) e I(i, j-1) são fundo da imagem (branco);

P9	***P2***	*P3*
P8	***P1***	*P4*
P7	***P6***	*P5*

   a condição se refere aos pixels P2, P6 e P8.

No final desta sub-iteração os pixels marcados são eliminados. A próxima sub-iteração é a mesma, exceto para os passos 3 e 4, logo:

3. Ao menos um dos I(i-1,j), I(i,j+1) e I(i+1, j) são fundo da imagem (branco);

P9	***P2***	*P3*
P8	***P1***	***P4***
P7	***P6***	*P5*

a condição se refere aos pixels P2, P4 e P6.

4. Ao menos um dos I(i,j+1), I(i+1,j) e I(i, j-1) é fundo da imagem (branco);

P9	*P2*	*P3*
P8	***P1***	***P4***
P7	***P6***	*P5*

a condição se refere aos pixels P4, P6 e P8.

E novamente, cada um dos pixels marcado é excluído. É importante ressaltar que em ambas as iterações, os pixels só devem ser eliminados no final da iteração. Se no final da segunda iteração não existem pixels para serem eliminados, então a esqueletização está completa e o programa pára.

1.3.7 Operações Morfológicas

"Operação morfológica" ou "morfologia matemática" é uma ferramenta para a extração de componentes de imagens que sejam úteis na representação e descrição da forma de uma região, como fronteiras, esqueletos e o fecho convexo [GW92].

A idéia base da morfologia é comparar os objetos que queremos analisar com um outro objeto de forma conhecida chamado elemento estruturante. A partir desse elemento estruturante, é possível testar e quantificar de que maneira o elemento estruturante "está ou não contido" na imagem. Cada elemento estruturante fornece uma aparência nova do objeto, donde surge a importância de sua escolha [Fac93]. A seguir são destacadas algumas operações morfológicas:

- **Dilatação** - expande uma imagem, ou seja, os efeitos da dilatação são engordar as partículas, diminuir as anseadas, preencher os pequenos buracos e permitir a conexão de grãos próximos;

- **Erosão** - reduz uma imagem, ou seja, aumenta os espaços entre os caracteres e conseqüentemente evita as sombras. O efeito da erosão é fazer desaparecer os elementos de tamanho inferior ao tamanho do elemento estruturante;
- **Poda** – são um complemento aos algoritmos de afinamento, uma vez que esses tendem a deixar componentes "parasitas" que precisam ser limpos através de pós-processamento. A grande diferença com o processo de afinamento reside no número de ciclos de supressão de pontos extremos, o ideal é determinar o número de ciclos;
- **Transformação *hit-or-miss*** - é uma ferramenta básica para a detecção de formas ou reconhecimento de padrões. A transformada *hit-or-miss* permite capturar as informações do interior e do exterior de uma imagem. Para isso, precisa de dois elementos estruturantes, um para testar o interior e o outro para testar o exterior da imagem. Através da definição de diferenças de conjuntos e a relação entre erosão e dilatação é possível determinar o quanto houve de acertos (*hit*) entre os elementos estruturantes e a imagem original.

Uso de Morfologia

A Figura 6, extraída de [GW92], será utilizada como exemplo do uso das técnicas de morfologia em processamento de imagens.

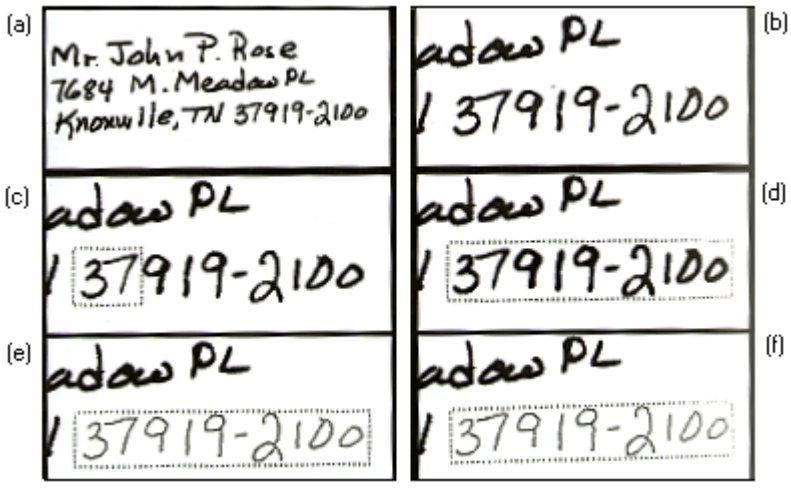


Figura 6. Uso das Técnicas Morfológicas

A Figura 6(a), mostra o pedaço de um endereço de um envelope depois da limiarização de uma imagem. Depois da localização do endereço, todos os componentes conexos na área que abrange esse endereço foram extraídos. Cada componente conexo foi envolvido pelo menor retângulo que o contenha completamente. Os retângulos e seus

conteúdos formam a base para extração da região contendo o código postal. A Figura 7(b) é uma aproximação da região contendo o código postal. Nota-se neste exemplo três problemas:

1. Os primeiros caracteres (**3** e **7**) estão emendados;
2. O primeiro **1** está quebrado ao meio;
3. A elipse formada na extremidade inferior do número **2** está quebrada.

Os caracteres emendados podem ser detectados de diversas maneiras uma vez que o código postal tenha sido localizado. Por exemplo, se a análise dos retângulos envolventes revelar menos de cinco caracteres (ou menos que nove no caso do código completo), a busca por caracteres emendados começa com a avaliação da largura relativa dos retângulos que envolvem os caracteres. Um retângulo mais largo do que o normal geralmente corresponde a dois ou mais caracteres emendados. Uma abordagem morfológica para a separação dos caracteres emendados consiste na **erosão** do conteúdo do retângulo até a separação das regiões parecidas com caracteres. A Figura 7(c) enfatiza esse resultado, mostrando a separação dos caracteres **3** e **7** após 5 (cinco) iterações de erosão.

O problema de caracteres quebrados pode ser normalmente tratado por **dilatação**. Na etapa de pré-processamento, suspeita-se que os caracteres estejam quebrados quando, por exemplo, os retângulos envolvendo alguns caracteres forem pequenos em relação ao tamanho esperado, ou quando dois ou mais retângulos formarem uma configuração inusitada, como no caso de estarem empilhados. Essa última condição revela a presença de um caractere quebrado no caso do **1** mostrado na Figura 7(b). Assumindo que a razão para tais quebras seja inconsistência na largura dos segmentos tende-se a esperar que outras quebras possam estar presentes no código de endereçamento sem resultar em componentes conexos separados (como a quebra no 2). Portanto, faz sentido a realização da dilatação em todos os caracteres, acompanhada de uma monitoração que verifique se os novos caracteres (ou os caracteres que estivessem separados antes) não sejam juntados pelo processo de dilatação. A Figura 7(d) mostra o resultado depois de três iterações sobre todos os caracteres. Nota-se que o vão no meio do **1** quebrado foi preenchido, bem como o vão do **2**.

Uma das principais abordagens para o reconhecimento estrutural de caracteres baseia-se na análise do esqueleto de cada caractere. A Figura 7(e) mostra os esqueletos dos caracteres do código da Figura 7(d), este **afinamento** pode ser obtido pelo Algoritmo de Zhang-Suen [ZS84], como descrito na seção 1.3.6, vale ressaltar que um dos problemas com esqueleto está na geração de ramificações parasitas. Se não forem tratadas apropriadamente, elas constituem uma grande fonte de erros em reconhecimento de padrões. Nesse exemplo,

uma pequena ramificação esta presente no canto do **7**, enquanto uma grande aparece o topo de um dos **0**'s. O resultado da realização de sete passos de **poda** elimina ambas as ramificações parasitas, como a Figura 7(f) mostra.

O número de iterações de poda é uma escolha tipicamente heurística[2]. Por exemplo, se a ramificação sobre o **0** estivesse abaixo à direita, o caractere seria um **Q**, significando que sua eliminação causaria um erro. Não existe nenhuma maneira absolutamente correta para se tratar esse problema além da utilização de conhecimento contextual. Nesse exemplo, sabe-se que o caractere devia ser um número devido ao fato de pertencer ao código postal. Em situações mais complexas, como endereços (de ruas), o uso do contexto envolve a comparação de código postal com nomes válidos para aquele código.

[2] Heurística - são regras que qualificam a possibilidade de uma pesquisa estar prosseguindo na direção correta.

2 O ESTADO DA ARTE EM OCR'S

Este capítulo faz um apanhado do uso das técnicas de processamento de imagens para o reconhecimento de caracteres. Além de descrever as características de uma ferramenta comercial de OCR.

É comum a perda de arquivos de texto e, conseqüentemente, a necessidade de redigitação. Nesse caso, é possível resolver, em parte, digitalizando o texto impresso e usando um software de reconhecimento óptico de caracteres (OCR).

OCR's são softwares capazes de extrair texto, caso estes existam, de imagens digitalizadas. Ele transforma páginas digitalizadas de fax, livros, revistas e jornais em textos que podem ser editados. Basicamente, dois tipos de componentes podem ser encontrados em documentos: texto e figuras. É importante proceder com essa identificação para que a ferramenta de OCR não perca tempo tentando processar áreas do documento compostas apenas por elementos gráficos (figuras, fotos, diagramas, etc.).

Vários grupos de estudos, no mundo todo, estão pesquisando OCR's, principalmente para o reconhecimento de texto manuscrito, desde o latim até o alfabeto siliric, como meio de identificação, ou mesmo reconhecimento de código postal para acelerar a classificação das correspondências.

Quatro passos devem ser seguidos na imagem antes do reconhecimento final dos caracteres da imagem de um documento:

1. Aquisição dos dados - feita no papel do documento através de um digitalizador (tal como um *scanner*);
2. Processamento da imagem - análise de rotação, redução de ruídos inerentes ao papel ou inseridos pelo processo de digitalização, detecção de junção de caracteres, segmentação da imagem, etc.;
3. Análise de características - após o processamento da imagem, características intermediárias devem ser encontradas para facilitar o posterior processo de reconhecimento, como, por exemplo, o tamanho das fontes do documento e análise de diagramação;

4. Análise e reconhecimento de texto e gráficos - nesta fase, o documento final é gerado com grupos de pixels transcritos para títulos, subtítulos, corpo de texto, etc. Os gráficos podem ser inseridos no documento final como imagem, caso o formato de saída, suporte tais elementos.

Este processo pode ser visualizado na Figura 7, extraída de [Mel02].

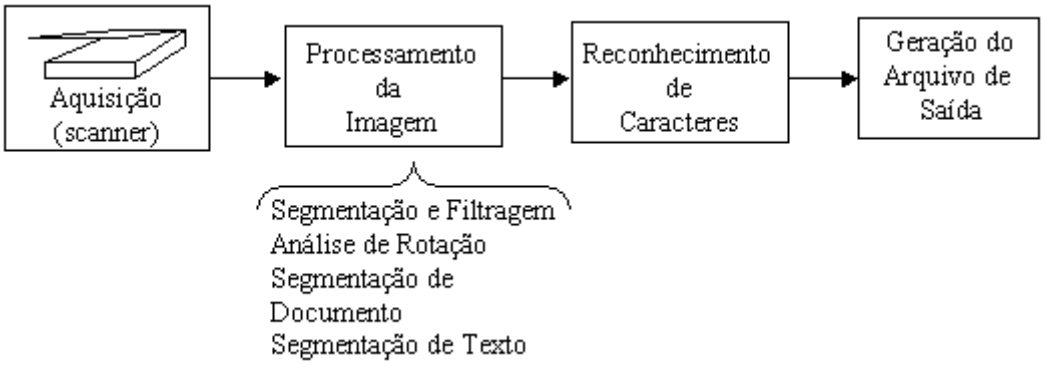


Figura 7. Esquema do processo completo para reconhecimento de caracteres

As etapas de aquisição e processamento foram descritas no capítulo 1, neste será detalhado o reconhecimento de caracteres.

Quando um *scanner* lê a imagem de um documento, ele converte os elementos escuros (texto e parte gráfica) da página, num mapa de bits (bitmap), uma matriz de pixels quadrados que podem estar ativos (pretos) ou inativos (brancos). Como os pixels são maiores que os detalhes da maior parte do texto, este processo degrada as extremidades mais finas dos caracteres, um problema que é responsável pela maior parte dos problemas para os sistemas de OCR's.

O programa de OCR lê o bitmap (imagem) gerado pelo *scanner* e pondera as áreas de pixels ativos e inativos da página, ou seja, mapeia o espaço em branco da página. Isso possibilita que o programa separe em blocos os parágrafos, colunas, títulos e partes gráficas. O espaço em branco entre as linhas de texto contidos num bloco define a base de cada linha, um detalhe essencial para o reconhecimento de caracteres no texto.

Na primeira etapa de conversão de imagens em texto, o programa tenta reconhecer cada caractere através de uma comparação pixel a pixel com o modelo de caracteres que o programa guarda na memória. Os modelos são compostos de conjuntos completos: números, letras, pontuação e caracteres especiais.

Como esta técnica obriga a uma correspondência muito próxima, os atributos do caractere, tais como negrito e itálico, devem ser idênticos para serem reconhecidos. Uma digitalização de má qualidade não consegue bons resultados neste aspecto. Os caracteres não reconhecidos passam por um processo mais minucioso e demorado conhecido como extração de recursos.

O programa calcula a altura das letras do texto e analisa cada combinação das linhas retas, curvas e áreas preenchidas de cada caractere, como no caso da letra **o** ou da **b**. Os programas OCR sabem, por exemplo, que o caractere com uma curva descendente abaixo da linha de base e uma área preenchida acima tem grande possibilidade de ser um **g** minúsculo. O programa faz sempre uma comparação com a tabela interna de que dispõe o que faz com que a velocidade de reconhecimento aumente.

Como estes dois processos acabam por não decifrar todos os caracteres, os programas de OCR podem usar outros métodos para reconhecer os símbolos remanescentes:

1. Marcando os caracteres não reconhecidos com um caractere especial como ~ ou **@** e desistem. Neste caso faz-se necessário o uso de um processador de texto para localizar tais caracteres especiais, corrigindo-os manualmente;
2. Mostrando um bitmap em zoom e pedir que seja pressionada a tecla correspondente ao caractere em questão, que deverá ser substituído pelo bitmap;
3. Solicitando um corretor ortográfico especial para procurar erros óbvios e localizar as possíveis alternativas para as palavras que contêm caracteres especiais não reconhecidos. Por exemplo, para os programas de OCR, o número **1** e a letra **l** são muito similares, da mesma forma que o **5** e o **S**, ou ainda **cl** e o **d**. Uma palavra como **aclimatar** poderia transformar-se em **adimatar**. O corretor ortográfico reconhece esses erros típicos do OCR e corrige-os.

2.1 Histórico

Os primeiros programas de OCR foram introduzidos em 1959 por Intelligent Machine Corporation [SS03]. Estes podiam ler apenas um tipo de fonte de tamanho fixo e foi utilizado para processar formulários de hipotecas pré-impressos, dentro do setor bancário. Posteriormente, máquinas com várias fontes foram desenvolvidas, estas podiam ler dez ou mais tipos de fontes usando um modelo padrão, no qual a imagem obtida do scanner era comparada com uma biblioteca de imagens armazenadas. A precisão era boa, com índices de acertos altos, desde que as fontes na biblioteca fossem cuidadosamente selecionadas.

Máquinas com várias fontes foram destinadas para serem usadas em escritórios onde havia bibliotecas de fontes de máquinas de escrever. Máquinas utilizadas pelas instituições tais como, agências governamentais e empresas de cartão de crédito, onde uma precisão excepcional era necessária, os quais utilizavam equipamentos de informática com fontes personalizadas para reduzir o máximo possível a confusão entre caracteres de aparências similares. Em 1966, um padrão de fontes Americano, chamado OCR-A e um padrão Europeu chamado OCR-B foram desenvolvidos.

a) 1970 a 1980

Kurzweil Computer Products [Kur03] introduziu um sistema em 1978 que poderia ser treinado para ler qualquer fonte. Após várias horas de treinamento o aprendizado era completado, a informação era armazenada em discos de modo a que o retrabalho não seria necessário.

A maioria dos sistemas de OCR's da década de 70 e meados da década de 80 eram baseados em padrões. Cada imagem de caractere era quebrada em um conjunto de características tais como linhas e curvas. Os padrões de um determinado caractere eram comparados às características extraídas. Este método de trabalhado, com imagens claras onde os caracteres eram degradados e páginas com ruídos, não era adequado.

b) 1980 a 1990

Ao final dos anos 80 foram introduzidos os sistemas que podiam ler páginas complexas contendo uma mistura de fontes sem utilizador um padrão. Ao invés de utilizar padrões o sistema utilizava redes neurais, que são algoritmos capazes de aprender através de exemplos. Neste mesmo capítulo será abordado com mais detalhe as redes neurais. Os criadores alimentaram o sistema com mais de 10.000 (dez mil) caracteres em inglês, extraídos de diversos tipos de material do mundo real, incluindo páginas degradadas. O computador fez suas próprias generalizações sobre os caracteres individualmente, pesquisando e selecionando os caracteres. O resultado foi um sistema de OCR suficientemente inteligente, capaz de manusear fontes que nunca havia encontrado antes e muito mais eficaz na hora de reconhecer documentos danificados.

Em 1993, com o Adaptive Recognition Technology™, a empresa Calera [SS03] incorporou o mais alto nível de inteligência no contexto das máquinas de redes neurais. Para

cada nova página, um modelo foi desenvolvido para pré-descrever as características da página reconhecida, incluindo informações tais como fonte, tamanho, tipo e nível de degradação. Esta página específica contém dados combinados com pré-compilados, o reconhecimento do OCR produziu, consideravelmente, uma alta precisão sobre documentos degradados como faxes.

Hoje em dia, são usados algoritmos especializados para os diversos campos de reconhecimento de caracteres, como estilo de fontes, utilizam informações de dicionários, identificam danos específicos dos documentos. Cada um pode, deste modo, indicar seu "voto" acerca da interpretação de caracteres.

2.2 Classificação de Padrão

De acordo com [AT95], Classificação de Padrão refere-se simplesmente ao processo onde um objeto desconhecido de uma imagem é identificado em um grupo particular dentro de um grupo de objetos possíveis.

O termo Padrão refere-se a uma coleção de descritores ou aspectos derivados de um desconhecido objeto. Uma Classe de Padrão é um grupo de padrões que compartilha alguma propriedade comum. Classificação ou Reconhecimento é o processo de associar corretamente um padrão desconhecido para as respectivas classes de padrões. O ideal seria que o processo de classificação fosse completamente automático e livre de erros, entretanto, na prática, nem sempre a tarefa de associar classificação dentro das regras acontece, em particular quando os objetos contêm medidas com grau de variedade entre objetos na mesma classe de objetos.

Em geral há três propriedades básicas para classificação de padrão:

1. Estatística – onde as regras de decisão são baseadas em teorias estatísticas;
2. Sintática – utiliza uma estrutura base dos seus padrões. Em alguns casos as leis de informações não têm inter-relação entre elas, então o aspecto de interconexão do aspecto de associação entre os processos de classificação. Para classificação é necessário quantificar e estruturar a informação e associar similaridades entre a estrutura de padrões;
3. Redes Neurais – usa uma arquitetura que pode ser treinada para uma associação correta de entradas de padrões. Na seção 2.4 serão abordados com mais detalhes as características de uma Rede Neural.

De acordo com [Fac93], dentre os assuntos de pesquisa em reconhecimento de padrões, pode-se citar:

- O reconhecimento de caracteres cujos recentes progressos nas pesquisas permitem o desenvolvimento de sistemas de leitura automática, já disponíveis no mercado. A dificuldade de uma tal pesquisa reside no grande número de tipos de letras usados e na necessidade de incluir o conhecimento sintático, semântico e pragmático do idioma em questão (árabe, japonês, latim);
- O reconhecimento de impressões digitais cujo interesse para o bom funcionamento da sociedade é bastante útil e pode ser em parte automatizado. A dificuldade do confronto de uma impressão digital latente com um banco de dados reside na fragilidade de um levantamento cuidadoso do fragmento da impressão e no tamanho e forma de organização do banco de dados que coloca restrições de desempenho no desenvolvimento de sistemas rápidos;
- O reconhecimento de assinaturas, cujo interesse para entidades bancárias e burocráticas faz-se necessário. Mas a facilidade que o ser humano tem de mudar a sua própria assinatura, de falsificar a assinatura de outra pessoa e/ou de conseguir reconhecê-la, inviabilizou até agora sistemas automatizados de identificação.

2.2.1 Métodos Estatísticos

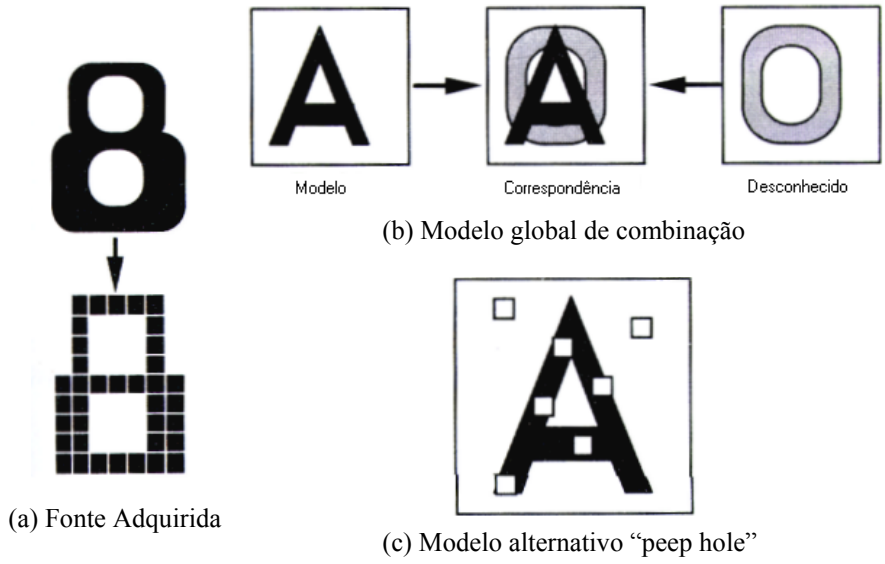
Considerações sobre probabilidade são importantes em reconhecimento de padrões, devido à aleatoriedade na qual as classes de padrões estão envolvidas. É possível derivar uma abordagem de classificação que seja ótimo no sentido que, na média, seu uso leve à menor probabilidade de erros de classificação.

De acordo com [AT95], em reconhecimento de caracteres três tipos de problemas podem ser identificados, dependendo do grau de complexidade:

- O sistema trabalha com uma fonte específica;
- O sistema de OCR tem de reconhecer caracteres impressos em qualquer tipo de fontes;
- O sistema deve reconhecer caracteres sem definição, isto é, caracteres manuscritos.

A Figura 8(a) ilustra um padrão típico de fontes de caracteres após a aquisição. A Figura 8(b) ilustra o uso de modelos globais para reconhecer padrões de caracteres. Algumas aplicações podem falhar porque inúmeros caracteres possuem áreas comuns, como as letras **O** e **Q**. Por último a Figura 8(c) mostra a variação da aproximação que é aplicável em cada caso onde um pequeno número de pixels na área do caractere é selecionado e usado para comparar com um conjunto de dados armazenados. Isto é provavelmente a comparação de um pequeno

conjunto de características de caracteres, onde o número de posições dos pixels tem que ser cuidadosamente selecionado desde que possa classificar todos os protótipos dos caracteres exclusivamente. A máscara "peep hole" é limitada para justamente simplificar o problema de classificação e pode ser sensível para ruído, para ocasionar mudanças nos valores esperados dos pixels.



(a) Fonte Adquirida

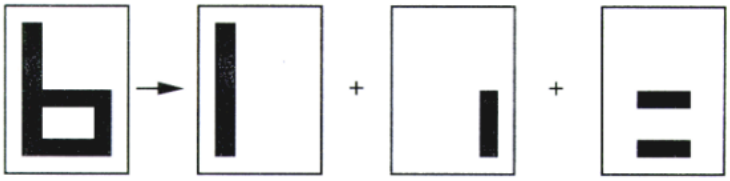
(b) Modelo global de combinação

(c) Modelo alternativo "peep hole"

Figura 8. Modelos de padrões em OCR

Uma técnica baseada na correlação de um grupo de propriedades individuais ou características descrevem classes de caracteres diferentes e analisam as características. A Figura 9(a) mostra o uso de análise por parte em que os caracteres são classificados de acordo com a estrutura de linhas horizontais e verticais, além disso a figura mostra que o número **6** é construído da combinação de dois traços verticais, um curto e outro mais longo, juntamente com dois traços verticais. Este padrão é um tanto primitivo, mas pode ser bem utilizado como padrão de caracteres numerais.

A Figura 8(b) mostra o uso de características geométricas de caracteres. Aspectos como fim de linha, canto, cruzamento, transição, etc. são utilizados para classificar os caracteres. A mesma figura mostra que o número **4** (quatro) contém quatro fins de linha, uma transição e um canto, enquanto que o número **6** (seis) contém um fim de linha, um cruzamento e uma curva circular.



(a) Análise de Traço

(b) Análise de Características Geométricas

Figura 9. Análise de Características

2.2.2 Métodos Sintáticos

A idéia por trás do reconhecimento sintático de padrões está na especificação de um conjunto de *primitivas* de padrões, um conjunto de regras (na forma de *gramática*) que governe suas interconexões e um *reconhecedor* (*autômato*) cuja estrutura é determinada pelo conjunto de regras na gramática. Pode-se considerar *gramática de cadeias e autômatos* e *gramática de árvores e seus autômatos*.

Gramática de cadeias

Classes de primitivas podem ser interpretadas como símbolos permitidos de um alfabeto de alguma gramática, sendo que uma gramática é um conjunto de regras de sintaxe que governam a geração de sentenças formadas a partir dos símbolos do alfabeto. O conjunto de sentenças geradas por uma gramática é chamado de sua linguagem. As sentenças são cadeias de símbolos, ou padrões e as linguagens correspondem a classes de padrões.

Autômatos como reconhecedores de padrões

Autômatos são modelos matemáticos capazes de reconhecer um padrão pertencente a uma linguagem gerada por uma gramática.

Gramática de árvores

Segue um formato similar da gramática de cadeia, porém com um formato de árvore, ou seja, as primitivas são nós terminais ou não-terminais. Uma primitiva possui seus descendentes diretos de um nó cujo rótulo seja um terminal na gramática.

Autômatos de árvores

Começa a varrer sua cadeia de entrada de símbolos, simultaneamente em cada nó da fronteira (as folhas tomadas na ordem da esquerda para direita) de uma árvore de entrada e prosseguir ao longo de caminhos paralelos em direção a raiz.

2.3 Redes Neurais Artificiais

2.3.1 Introdução

De acordo com [BCL00], Redes Neurais Artificiais são sistemas paralelos distribuídos compostos por unidades de processamento simples (nodos) que calculam determinadas funções matemáticas, normalmente não-lineares. Tais unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões, geralmente unidirecionais. Na maioria dos modelos estas conexões estão associadas a pesos, os quais armazenam o conhecimento representado no modelo e servem para ponderar a entrada recebida por cada neurônio da rede. O funcionamento destas redes é inspirado em uma estrutura física concedida pela natureza: o cérebro humano.

O sistema nervoso é formado por um conjunto extremamente complexo de células, os neurônios. Eles têm um papel essencial na determinação do funcionamento e comportamento do corpo humano e do raciocínio. Os neurônios são formados pelos **dendritos**, que são um conjunto de terminais de entrada, pelo corpo central, e pelos **axônios** que são longos terminais de saída. O ponto de contato entre a terminação axônica de um neurônio e o dendrito de outro é chamado de **sinapse**. É pelas sinapses que os nodos se unem funcionalmente, formando redes neurais. As sinapses funcionam como válvulas, e são capazes de controlar a transmissão de impulsos.

Os impulsos recebidos por um neurônio **A**, em um determinado momento, são processados, e atingindo um dado limiar de ação, o neurônio **A** dispara, produzindo uma substância neurotransmissora que flui do corpo celular para o axônio, que pode estar conectado a um dendrito de um outro neurônio **B**. O neurotransmissor pode diminuir ou aumentar a polaridade da membrana pós-sináptica, inibindo ou excitando a geração dos pulsos

no neurônio **B**. Este processo depende de vários fatores, como a geometria da sinapse e o tipo de neurotransmissor.

Em média, cada neurônio forma entre mil e dez mil sinapses. O cérebro humano possui cerca de 10^{11} neurônios, e o número de sinapses é de mais de 10^{14}, possibilitando a formação de redes muito complexa.

2.3.2 Histórico

O interesse em redes neurais [Hay94] data do início dos anos 40, como exemplificado pelo trabalho de McCulloch e Pitts (1943), Hebb (1949) e Rosemblatt (1958). Estas publicações introduziram o primeiro modelo de redes neurais simulando "máquinas", o modelo básico de rede de auto-organização e o modelo *perceptron* de aprendizado supervisionado, respectivamente.

Em 1958, Frank Rosemblatt [Ros58] demonstrou, com o seu novo modelo, o *perceptron*, que, se fossem acrescidas de sinapses ajustáveis, as redes neurais com nodos poderiam ser treinadas para classificar certos tipos de padrões. O *perceptron* simples possui três camadas: a primeira recebe as entradas do exterior e possui conexões fixas (retina); a segunda recebe impulsos da primeira através de conexões cuja eficiência de transmissão (peso) é ajustável e, por sua vez, envia saídas para a terceira camada (resposta). Em 1982, John Hopfield introduziu o algoritmo de treinamento *backpropagation*.

2.3.3 Características Gerais

Uma rede neural artificial é composta por várias unidades de processamento, cujo funcionamento é bastante simples. Essas unidades, geralmente são conectadas por canais de comunicação que estão associados a determinado peso. As unidades fazem operações apenas sobre seus dados locais, que são entradas recebidas pelas suas conexões. O comportamento inteligente de uma Rede Neural Artificial vem das interações entre as unidades de processamento da rede.

A operação de uma unidade de processamento, proposta por McCullock e Pitts em 1943, pode ser resumida da seguinte maneira:

- sinais são apresentados à entrada;
- cada sinal é multiplicado por um número, ou peso, que indica a sua influência na saída da unidade;
- é feita a soma ponderada dos sinais que produz um nível de atividade;

- se este nível de atividade exceder um certo limite (*threshold*) a unidade produz uma determinada resposta de saída.

Suponha que se têm:

- sinais de entrada X1, X2, ..., Xp, assumindo valores booleanos (0 ou 1);
- pesos w1, w2, ..., wp, assumindo valores reais;
- limitador (*threshold*) t;

Neste modelo, o nível de atividade a é dado por:

$$a = w1X1 + w2X2 + ... + wpXp$$

A saída y é dada por:

$y = 1$, se $a >= t$ ou

$y = 0$, se $a < t$.

A maioria dos modelos de redes neurais possui alguma regra de treinamento, onde os pesos de suas conexões são ajustados de acordo com os padrões apresentados dada uma configuração de peso inicial. Arquiteturas neurais são tipicamente organizadas em camadas, com unidades que podem estar conectadas às unidades da camada posterior, como na Figura 10 extraída de [Red03]. Usualmente as camadas são classificadas em três grupos:

- **Camada de Entrada** - onde os padrões são apresentados à rede;
- **Camadas Intermediárias ou Escondidas** - onde é feita a maior parte do processamento, através das conexões ponderadas, podem ser consideradas como extratoras de características;
- **Camada de Saída -** onde o resultado final é concluído e apresentado.

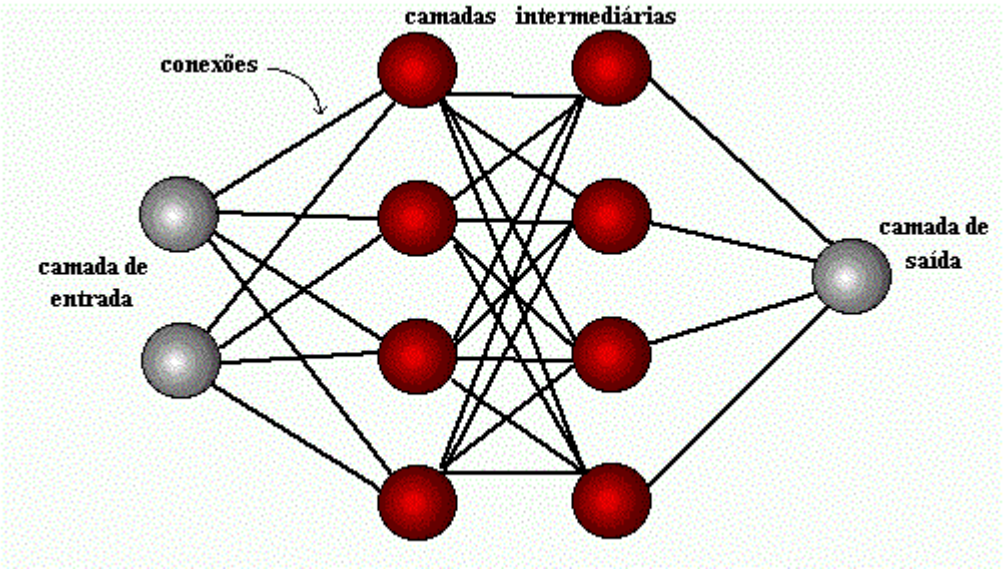


Figura 10. Camada Redes Neurais

Uma rede neural é especificada, principalmente, pela sua topologia, pelas características dos nodos e pelas regras de treinamento. A seguir, serão analisados os processos de aprendizado.

2.3.4 Processos de Aprendizado

A propriedade mais importante das redes neurais é a habilidade de aprender de seu ambiente e com isso melhorar seu desempenho. Isso é feito através de um processo iterativo de ajustes aplicado a seus pesos, o treinamento. O aprendizado ocorre quando a rede neural atinge uma solução generalizada para uma classe de problemas.

Denomina-se algoritmo de aprendizado a um conjunto de regras bem definidas para a solução de um problema de aprendizado. Existem muitos tipos de algoritmos de aprendizado específicos para determinados modelos de redes neurais, estes algoritmos diferem entre si principalmente pelo modo como os pesos são modificados.

Outro fator importante é a maneira pela qual uma rede neural se relaciona com o ambiente. Nesse contexto existem os seguintes paradigmas de aprendizado:

- **Aprendizado Supervisionado** - quando é utilizado um agente externo que indica à rede a resposta desejada para o padrão de entrada;
- **Aprendizado Não Supervisionado (auto-organização)** - quando não existe um agente externo indicando a resposta desejada para os padrões de entrada;
- **Reforço** - quando um crítico externo avalia a resposta fornecida pela rede.

Denomina-se ciclo uma apresentação de todos os **N** pares (entrada e saída) do conjunto de treinamento no processo de aprendizado. A correção dos pesos num ciclo pode ser executada de dois modos:

1. **Modo Padrão -** A correção dos pesos acontece a cada apresentação à rede de um exemplo do conjunto de treinamento. Cada correção de pesos baseia-se somente no erro do exemplo apresentado naquela iteração. Assim, em cada ciclo ocorrem **N** correções.
2. **Modo Batch -** Apenas uma correção é feita por ciclo. Todos os exemplos do conjunto de treinamento são apresentados à rede, seu erro médio é calculado e a partir deste erro fazem-se as correções dos pesos.

Treinamento Supervisionado

O treinamento supervisionado, do modelo de rede *Perceptron*, consiste em ajustar os *pesos* e os *thresholds* de suas unidades para que a classificação desejada seja obtida. Para a adaptação do *threshold* juntamente com os pesos pode-se considerá-lo como sendo o peso associado a uma conexão, cuja entrada é sempre igual à **-1** e adaptar o peso relativo a essa entrada.

Quando um padrão é inicialmente apresentado à rede, ela produz uma saída. Após medir a distância entre a resposta atual e a desejada, são realizados os ajustes apropriados nos pesos das conexões de modo a reduzir esta distância. Este procedimento é conhecido como **Regra Delta**. Deste modo, temos o seguinte esquema de treinamento.

1. Iniciar todas as conexões com pesos aleatórios;
2. Repita até que o erro **E** seja satisfatoriamente pequeno (**E** = e).
3. Para cada par de treinamento (X, d), faça:
    - Calcular a resposta obtida O;
4. Se o erro não for satisfatoriamente pequeno **E** > e, então:
    - Atualizar pesos: $W_{novo} := W_{anterior} + \eta\ \mathbf{E}\ X$

onde:

- O par de treinamento (X, d) corresponde ao padrão de entrada e a sua respectiva resposta desejada;
- O erro **E** é definido como: Resposta Desejada - Resposta Obtida (d - O);

- A taxa de aprendizado η é uma constante positiva, que corresponde à velocidade do aprendizado.

As respostas geradas pelas unidades são calculadas através de uma função de ativação. Existem vários tipos de funções de ativação, as mais comuns são: Hard Limiter, Threshold Logic e Sigmoidal.

Perceptron Multi-Camadas (MLP)

Quando Redes Neurais Artificiais de uma só camada são utilizadas os padrões de treinamento apresentados à entrada são mapeados diretamente em um conjunto de padrões de saída da rede, ou seja, não é possível a formação de uma representação interna. Neste caso, a codificação proveniente do mundo exterior deve ser suficiente para implementar esse mapeamento.

Tal restrição implica que padrões de entrada similares resultem em padrões de saída similares, o que leva o sistema à incapacidade de aprender importantes mapeamentos. Como resultado, padrões de entrada com estruturas similares, fornecidos do mundo externo, que levem a saídas diferentes não são possíveis de serem mapeados por redes sem representações internas, isto é, sem camadas intermediárias. Um exemplo clássico deste caso é a função ou-exclusivo (XOR).

Minsky e Papert analisaram matematicamente o Perceptron e demostraram que redes de uma camada não são capazes de solucionar problemas que não sejam linearmente separáveis. Como não acreditavam na possibilidade de se construir um método de treinamento para redes com mais de uma camada, eles concluíram que as redes neurais seriam sempre suscetíveis a essa limitação.

Contudo, o desenvolvimento do algoritmo de treinamento backpropagation, por Rumelhart, Hinton e Williams, em 1986, precedido por propostas semelhantes ocorridas nos anos 70 e 80, mostrou que é possível treinar eficientemente redes com camadas intermediárias, resultando no modelo de Redes Neurais Artificiais mais utilizado atualmente, as redes **Perceptron Multicamadas** ou MLP (MultiLayer Perceptron), treinadas com o **algoritmo backpropagation**.

Nessas redes, cada camada tem uma função específica. A camada de saída recebe os estímulos da camada intermediária e constrói o padrão que será a resposta. As camadas intermediárias funcionam como extratoras de características, seus pesos são uma codificação

de características apresentadas nos padrões de entrada e permitem que a rede crie sua própria representação, mais rica e complexa, do problema.

Se existirem as conexões certas entre as unidades de entrada e um conjunto suficientemente grande de unidades intermediárias, pode-se sempre encontrar a representação que irá produzir o mapeamento correto da entrada para a saída através das unidades intermediária.

Como provou Cybenko, a partir de extensões do Teorema de Kolmogoroff, são necessárias no máximo duas camadas intermediárias, com um número suficiente de unidades por camada, para se produzir quaisquer mapeamentos. Também foi provado que apenas uma camada intermediária é suficiente para aproximar qualquer função contínua.

Algoritmo de Treinamento Backpropagation

Durante o treinamento com o algoritmo backpropagation, a rede opera em uma sequência de dois passos. Primeiro, um padrão é apresentado à camada de entrada da rede. A atividade resultante flui através da rede, camada por camada, até que a resposta seja produzida pela camada de saída. No segundo passo, a saída obtida é comparada à saída desejada para esse padrão particular. Se esta não estiver correta, o erro é calculado. O erro é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas vão sendo modificados conforme o erro é retropropagado.

As redes que utilizam backpropagation trabalham com uma variação da regra delta, apropriada para redes multi-camadas: a regra delta generalizada. A regra delta padrão, essencialmente, implementa um gradiente descendente no quadrado da soma do erro para funções de ativação lineares. Redes sem camadas intermediárias podem resolver problemas onde a superfície de erro tem a forma de um parabolóide com apenas um mínimo. Entretanto, a superfície do erro pode não ser tão simples e suas derivadas mais difíceis de serem calculadas. Nestes casos devem ser utilizadas redes com camadas intermediárias. Ainda assim, as redes ficam sujeitas aos problemas de procedimentos "*hill-climbing*", ou seja, ao problema de mínimos locais.

A regra delta generalizada funciona quando são utilizadas na rede unidades com função de ativação semi-linear, que é uma função diferenciável e não decrescente. Note que a função *threshold* não se enquadra nesse requisito. Uma função de ativação amplamente utilizada, nestes casos, é a função sigmoidal.

A taxa de aprendizado é uma constante de proporcionalidade no intervalo [0,1], pois este procedimento de aprendizado requer apenas que a mudança no peso seja proporcional à η

(velocidade do aprendizado). Entretanto, o verdadeiro gradiente descendente requer que sejam tomados passos infinitesimais. Assim quanto maior for essa constante, maior será a mudança nos pesos, aumentando a velocidade do aprendizado, o que pode levar à uma oscilação do modelo na superfície de erro. O ideal seria utilizar a maior taxa de aprendizado possível que não levasse à uma oscilação, resultando em um aprendizado mais rápido.

O treinamento das redes MLP com backpropagation pode demandar muitos passos no conjunto de treinamento, resultando um tempo de treinamento consideravelmente longo. Se for encontrado um mínimo local, o erro para o conjunto de treinamento pára de diminuir e estaciona em um valor maior que o aceitável. Uma maneira de aumentar a taxa de aprendizado sem levar à oscilação é modificar a regra delta generalizada para incluir o termo *momentum*, uma constante que determina o efeito das mudanças passadas dos pesos na direção atual do movimento no espaço de pesos.

Desta forma, o termo *momentum* leva em consideração o efeito de mudanças anteriores de pesos na direção do movimento atual no espaço de pesos. O termo *momentum* torna-se útil em espaços de erro que contenham longas gargantas, com curvas acentuadas ou vales com descidas suaves.

Treinamento da rede MLP

O treinamento supervisionado da rede MLP utilizando backpropagation consiste em dois passos. No primeiro, um padrão é apresentado às unidades da camada de entrada e, a partir desta camada, as unidades calculam sua resposta que é produzida na camada de saída. O erro é calculado e, no segundo passo, este é propagado a partir da camada de saída até a camada de entrada. Os pesos das conexões das unidades das camadas internas vão sendo modificados, utilizando a regra delta generalizada. Com isso o erro vai sendo progressivamente diminuído.

2.3.5 Utilização

Depois que a rede estiver treinada e o erro estiver em um nível satisfatório, ela poderá ser utilizada como uma ferramenta para classificação de novos dados. Para isto, a rede deverá ser utilizada apenas no modo progressivo (*feed-forward*). Ou seja, novas entradas são apresentadas à camada de entrada, são processadas nas camadas intermediárias e os resultados são apresentados na camada de saída, como no treinamento, mas sem a retropropagação do erro. A saída apresentada é o modelo dos dados, na interpretação da rede.

2.3.6 Limitações

As redes neurais que utilizam backpropagation, assim como muitos outros tipos de redes neurais artificiais, podem ser vistas como "caixas pretas", na qual quase não se sabe porque a rede chega a um determinado resultado, uma vez que os modelos não apresentam justificativas para suas respostas. Neste sentido, muitas pesquisas vêm sendo realizadas visando a extração de conhecimento de redes neurais artificiais, e na criação de procedimentos explicativos, onde se tenta justificar o comportamento da rede em determinadas situações.

Uma outra limitação refere-se ao tempo de treinamento de redes neurais utilizando backpropagation, que tende a ser muito longo. Algumas vezes são necessários milhares de ciclos para se chegar à níveis de erros aceitáveis, principalmente se estiver sendo simulado em computadores seriais, pois a CPU deve calcular as funções para cada unidade e suas conexões separadamente, o que pode ser problemático em redes muito grandes, ou com grande quantidade de dados. Muitos estudos estão sendo realizados para implementação de redes neurais em computadores paralelos, além de construção de chips neurais como Intel 80170NX Eletronically Trainable ANN ou placas aceleradoras como BrainMaker Accelerator Board CNAPS.

É muito difícil definir a arquitetura ideal da rede de forma que ela seja tão grande quanto o necessário para conseguir obter as representações necessárias, ao mesmo tempo pequena o suficiente para se ter um treinamento mais rápido. Não existem regras claras para se definir quantas unidades devem existir nas camadas intermediárias, quantas camadas, ou como devem ser as conexões entre essas unidades. Para resolver este tipo de problema, Algoritmos Genéticos poderiam ser utilizados para encontrar automaticamente boas arquiteturas de redes neurais, eliminando muitas armadilhas associadas às abordagens de engenharia humana.

2.4 Reconhecimento de Caracteres Manuscritos

O reconhecimento de caracteres manuscritos utiliza todas as ferramentas de reconhecimento de caracteres datilografados, porém a complexidade é maior devido ao fato da grande variação de estilos e características individuais da escrita de cada pessoa. Os caracteres são classificados como cursivos e letra de forma.

Outro problema encontrado em textos manuscritos é quantidade de ruídos gerados em documentos escritos nos dois lados do papel, em [ML99b] e [Tan00] são propostas técnicas

de segmentação para separar o texto do fundo extraindo os ruídos provenientes da transposição da tinta de um lado para outro do papel. No artigo de Silva-Rodrigues-Thomé [SRT01] é proposto um método de projeção poligonal do contorno de cada caractere para o reconhecimento de caracteres manuscritos. O trabalho de [OSBS03] é específico para o reconhecimento de caracteres numéricos manuscritos.

Neste trabalho, a princípio, não será aplicada a métrica desenvolvida para caracteres manuscritos, porém nada impede o uso nestes caracteres, uma vez que os resultados dos OCR's são caracteres datilografados.

2.5 Ferramenta Comercial *Omnipage*

Omnipage é uma ferramenta de reconhecimento de caracteres desenvolvida pela ScanSoft [SS03], empresa americana fornecedora de soluções para imagem e voz e neste momento está na sua versão 12.

Esta nova versão é a primeira solução OCR para computadores pessoais que converte, automaticamente para o formato XML (Extensible Markup Language) grandes quantidades de documentos e arquivos digitalizados. Dentre as funcionalidades do Omnipage está à capacidade de reconhecer e gerar textos, ou documentos em pápeis para PDF, com saída em quatro opções: normal, normal com substitutos de imagem, imagem somente e imagem sobre o texto. Como na versão anterior ele diferencia texto de figura, desbloqueia e edita informações de arquivos PDF, arquivos da Web. Documentos recebidos por e-mail ou gravados como somente leitura podem ser convertidos em vários formatos (Word, Excel, RTF, WordPerfect, PDF e HTML) e mantém cabeçalhos, rodapés e numeração de página. O programa reconhece 114 idiomas e é compatível com mais de 100 tipos de *scanners*. Permite o processamento em lote que possibilita a conversão de várias imagens em texto ao mesmo tempo, além de usar o revisor gramatical.

A ferramenta define um novo padrão em tecnologia para captação de composições gráficas, além de editar, processar e compartilhar documentos. Com o software é possível converter dados que estejam em diferentes tipos de linguagem computacional, acabando com a necessidade de recriar "manualmente" documentos gráficos.

Além do Omnipage, a Scansoft desenvolveu o Omniform que está na versão 5 e foi especialmente concebido para automatizar o desenho, distribuição e preenchimento de formulários e facilitar a gestão da respectiva informação, permitindo a conversão de formulários em papel em formulários digitais. A versatilidade deste programa permite-lhe

implementar formulários utilizando formatos PDF, HTML, RTF ou o formato inteligente do Omniform. A informação pode ser exportada para bases de dados ODBC, como é o caso do MS-Access e Oracle.

Funcionamento

Com o auxílio de um *scanner*, o documento é digitalizado e o Omnipage faz o reconhecimento, um corretor é ativado, para verificar palavras desconhecidas e não identificadas, e sugere a correção. Finalmente, grava o arquivo final em vários formatos.

Palavras desconhecidas estão, normalmente, relacionadas a problemas de degradação da imagem que prejudicam a resolução. É o caso de transmissões de fax e cópias ou equívocos na identificação de algumas palavras, principalmente as acentuadas. O módulo **Despeckle** é responsável por tentar corrigir problemas como esses, para reduzir o tempo de revisão e aumentar sua produtividade.

Neste trabalho optamos por trabalhar com a ferramenta Omnipage devidos os resultados obtidos na pesquisa de [Ml99a]. No próximo capítulo é descrito os experimentos e os resultados realizados com a esta ferramenta.

3 MÉTODOS DE COMPARAÇÃO ENTRE TRANSCRIÇÕES DE TEXTOS

Para poder avaliar a qualidade do desempenho de OCR's é necessário adquirir métricas para medir o quanto um texto gerado está próximo de sua transcrição correta. Uma simples aproximação foi proposta na referência [ML99a], através de um estudo comparativo para analisar ferramentas comerciais de OCR. Neste capítulo é apresentada uma métrica proposta por Junker-Hoch-Dengel [JHD99] para analisar a qualidade de transcrições de OCR's; uma breve introdução da métrica de Levenshtein [Lev66] para comparar textos é fornecida; e uma nova métrica para avaliar transcrições será apresentada.

Após o estudo das métricas serão descritos os resultados obtidos com o uso da ferramenta Omnipage em textos transcritos em inglês e português, no formato *true color* e em escala de cinza.

3.1 Definição de Métrica

De acordo com [Lim03], a definição de métrica pode ser definida como segue:

Seja **M** um conjunto. Uma métrica em **M** é uma função **d**: **M** x **M** $\rightarrow \Re$ tal que para quaisquer **x**, **y**, **z** $\in$ **M** tenhamos:

1. d (x, y) $\geq$ 0 e d (x, y) = 0 se e só se x = y;
2. d (x, y) = d (y, x);
3. d (x, z) $\leq$ d (x, y) + d (y, z).

Qualquer função que satisfizer estas três propriedades servirá para "medir" a distância entre pontos de um conjunto e tais funções serão chamados métricas,

O par (M, d), onde **M** é o conjunto e **d** uma métrica em **M,** será chamado Espaço Métrico. Quando a métrica **d** for facilmente subentendida pode-se escrever apenas **M** para indicar o espaço métrico (M, d).

3.2 A Métrica de Junker-Hoch-Dengel

Na referência [JHD99], Junker, Hoch e Dengel analisaram a qualidade de textos transcritos com OCR através de métricas para avaliar "recall", "precision" e "accuracy", e essas são calculadas como segue:

$$\text{Recall} = \frac{\#(\text{character/word x correctly recognised})}{\#(\text{character/word x occurs})}$$

$$\text{Precision} = \frac{\#(\text{character/word x correctly recognised})}{\#(\text{character/word x recognised})}$$

$$\text{Accuracy} = \frac{\#(\text{character/word x correctly recognised})}{\#(\text{all characters/words})}$$

Esta métrica dependem do fato que *recall*, *precision* e *accuracy* são as maiores probabilidades estimadas de prováveis caracteres desconhecidos. Duas metas centrais foram propostas na avaliação experimental: para garantir um menor erro para a eficácia de um componente e para comparar dois componentes baseados em um conjunto de testes.

Segundo [Vuo98], um experimento é muito preciso quando ele consegue resultados cuja flutuação em torno de um valor médio é pequena. Um experimento é dito muito acurado quando sua discrepância em relação ao valor verdadeiro é muito pequena.

3.2 Distância de Levenshtein

A Distância de Levenshtein [Lev66] foi introduzida em 1966 no contexto da teoria de códigos. Esta distância é usada como base para o desenvolvimento de várias novas métricas para comparar textos nas aplicações de biologia computacional, mas especialmente algoritmos genéticos. A Distância de Levenshtein [Lev66] entre duas palavras é o menor número de caracteres que devem ser inseridos, apagados ou substituídos em uma das palavras para que a mesma fique igual a outra.

Por exemplo, a palavra "***Paranambouc***" originaria do tupi-guarani, a língua falada pela maior parte dos índios do Brasil, tornou-se "***Pernambuco***" em português. A distância de Levenshtein entre elas é:

Paranambouc =

Parnambouc - 1 exclusão “a”

Pernambouc - 1 substituição “a” for “e”

Pernambuc - 1 exclusão “o”

Pernambuco - 1 inserção “o”

Portanto, a distância de Levenshtein entre ***Paranambouc*** e ***Pernambuco*** é 4 (2 exclusões, 1 troca e 1 inserção). Observa-se que a distância de Levenshtein forma uma métrica, pois

$$\mathcal{L}(A, A) = 0,\ \mathcal{L}(A, B) = \mathcal{L}(B, A)$$

e a inequação triangular

$$\mathcal{L}(A, C) < \mathcal{L}(A, B) + \mathcal{L}(B, C)$$

são válidos, onde A, B e C são caracteres e $\mathcal{L}$ (X, Z) são a Distância de Levenshtein entre os caracteres X e Z.

3.3 Erros gerados pelos OCR’s

É importante observar que a comparação entre a resposta do OCR e o texto original não é uma tarefa simples. Nos experimentos realizados em [ML99a], com diferentes valores de brilho, por exemplo, diversos tipos de erros foram encontrados na saída do OCR. Nas imagens testadas foram detectados as seguintes classes de erros nos textos de saída:

- Substituição de um caractere por outro (**e** por **c**);
- Substituição de um caractere por mais de um (como em **m** por **r** e **n**);
- Substituição de mais de um caractere por apenas um (como em **r** e **n** por **m**);
- Perda de caracteres (supressão);
- Junção de palavras sem perda de caracteres (supressão de espaços em branco);
- Junção de palavras com perda de caracteres;
- Perda completa de linhas de texto;
- Inserção de caracteres;
- Inserção de ruído.

O mesmo estudo [ML99a], afirma que o primeiro item acima é o mais fácil de ser tratado. Este problema não causa nenhuma mudança no comprimento da palavra original,

apenas a substituição de um caractere diferente de espaço. A substituição de um caractere por outro é o erro mais comum, seguido pela inclusão de caracteres, geralmente, devido a ruídos no documento original.

O pior caso para detectar é a exclusão de uma ou mais linhas completas. O sistema deve decidir se houve mesmo uma exclusão ou se a linha foi reconhecida inteiramente errada. Embora esta seja a classe mais difícil de detectar é também a mais rara, sendo detectada apenas em altos valores de brilho.

A junção de duas palavras com ou sem perda de algum caractere é também um erro comum encontrado em altos valores de brilho. Entre essas duas classes, o erro mais freqüente é de duas palavras sem a perda de qualquer caractere.

Em altas resoluções, o tipo de erro mais comum é causado por um reconhecimento errado de algum caractere. Esse tipo de erro pode ser corrigido, se um dicionário for usado em conjunto com o OCR. Em baixas resoluções, todas as classes de erro estão presentes no texto de saída e é praticamente impossível utilizar apenas uma técnica para corrigí-los.

3.4 Uma nova métrica para OCR

A distância de Levenshtein foi usada como base para medir a qualidade dos textos gerados pelos OCR's. Na distância de Levenshtein é possível encontrar a quantidade de erros, sem indicar onde eles ocorrem. Neste trabalho além de encontrá-los, eles são classificados. Antes da comparação entre os textos estes passam por um processo de normalização que consiste em:

- Todos os espaços entre palavras são de, exatamente, um espaço em branco;
- Todas as linhas são ajustadas na margem esquerda;
- Não há linhas em branco.

A métrica original de Levenshtein não leva em consideração a onde os erros ocorrem. Por exemplo:

"To be or not to be, thaz´s the queztion."

"To be or not to be, that´s the querkion."

A primeira frase do famoso monólogo de Hamlet, de Shakespeare, tem a mesma distância de Levenshtein. Entretanto, o fato da segunda transcrição ter dois erros na mesma palavra ("querkion" em vez de "question"), reduz bastante a probabilidade de recuperação do erro. Então a distância de Levenshtein foi extendida com a taxonomia de classificar os erros e

sua posição. Deste modo, os erros foram classificados em erros de caracteres, de palavras e de linhas.

1. Os erros de Caractere foram classificados de acordo com:
   - Inclusão - caracteres não existentes são incluídos no texto;
   - Exclusão - caracteres são excluídos;
   - Troca Simples - 1 caractere é trocado por outro;
   - Troca Múltipla Incluinte - 1 caractere é trocado por 2 caracteres, como: **m** por **r** e **n**;
   - Troca Múltipla Excludente - 2 caracteres são trocados por um caractere, como: **r** e **n** são trocados por **m**;
   - Junção de Palavras com Perda - o caractere branco mais o próximo caractere são excluídos e as palavras são unidas com perda de um caractere;
   - Junção de Palavras sem Perda - o caractere branco é excluído e as palavras são unidas sem perda de caractere.
2. Os erros de Palavras foram classificados de acordo com:
   - Inclusão - caracteres não existentes são incluídos no texto;
   - Exclusão - caracteres são excluídos;
   - Troca - 1 caractere é trocado por outro.
3. Os erros de Linha são:
   - Inserção - uma linha inexistente no texto é inserida na transcrição;
   - Exclusão - uma linha existente no texto é excluída durante a transcrição.

Duas métricas são apresentadas para calcular o número de caracteres e palavras transcritos corretamente. Elas são calculadas como segue, respectivamente:

$$\text{Caractere\_OK} = 100 - \left(\frac{\text{Total\_ECaractere x } 100}{\text{Total\_Caractere}} \right)$$

$$\text{Palavra\_OK} = 100 - \left(\frac{\text{Total\_EPalavra x } 100}{\text{Total\_Palavra}} \right)$$

onde:

Total_ECaractere - número total de erros por caractere no texto transcrito;

Total_Caractere - número total de caracteres no texto original (correto);

Total_EPalavra - número total de palavras erradas no texto transcrito;

Total_Palavra - número total de palavras no texto original (correto).

Na próxima seção, a taxonomia e a distância apresentada são usadas como referências para comparar o desempenho de ferramentas comerciais de OCR.

3.5 Experimentos com Omnipage

Em [ML99a], como mencionado anteriormente, foi feito um estudo comparativo entre as ferramentas comerciais de OCR, onde constatou-se o poder do Omnipage. Neste trabalho, foi feita uma nova avaliação da ferramenta que está na 12ª versão, a mais recente do mercado. O objetivo é avaliar as melhorias na nova versão e a consistência entre os resultados obtidos anteriormente. A Tabela 2 mostra um comparativo entre as características das versões 8ª e 12ª.

Tabela 2. Principais características da ferramenta *Omnipage*

Características	Omnipage 8.0	Omnipage 12.0
Tipos de Imagens (cores)	Tons de Cinza	Preto e branco Tons de cinza Colorida
Imagens Rotacionadas	Sim	Sim
Dicionário	Sim	Sim
Mudanças no Dicionário	Sim	Sim
Resoluções	250 dpi	75-600 dpi
Arquivos de Entrada	Vários	Vários
Técnica de OCR	Redes Neurais	Redes Neurais
Diferentes Fontes	Não	Sim
Diferentes Diagramações	Sim	Sim

Para os novos experimentos, foi adotada a mesma estratégia de [ML99a], utilizando 5 (cinco) textos em inglês e 5 (cinco) textos em português de diversos assuntos, sendo todos extraídos da Internet, com as cores de fundo e de fonte variadas, fontes com tamanhos e tipos também variadas, impressos em impressora laser, marca Okipage 6W [Oki03] com 300 dpi de resolução. Os documentos, após serem capturados da Internet, foram copiados para o Microsoft Word 2000 para garantir as configurações das fontes. Foram impressos em papel branco A4 Chamex [Cha03] com 210x297mm de dimensão e 75g/m² de gramatura. Os textos têm em média 3.000 (três mil) caracteres e 500 (quinhentas) palavras.

Após a impressão, os textos foram digitalizados com o scanner Scanjet 4C da Hewlett-Packard, possuindo resolução máxima de 720 dpi. Para a digitalização, os valores de brilho e contraste foram configurados conforme o padrão do software HP Deskscan II, o qual acompanha o scanner. Os valores de brilho foram ajustados entre 118% e 119% e os valores de contraste entre 133% e 137%, estes valores representam uma distribuição uniforme no histograma entre brilho e contraste nos documentos capturados. Os documentos foram digitalizados em *true color*, da esquerda para a direita e de cima para baixo, conforme o procedimento padrão dos digitalizadores.

Para a geração de imagens com mudanças de rotação, resolução, brilho e contraste foi utilizado o software Paint Shop Pro 8.0, da empresa Jasc Software [Jasc03]. Primeiramente as imagens geradas pelo *scanner* eram BMP em *true color*. Neste estudo foram feitas duas simulações: primeiramente, trabalhou-se com imagens *true color* e, em outro momento, com imagens em tons de cinza (*grayscale*). De posse das imagens bases, as demais foram geradas com mudanças de rotação, resolução e brilho, obedecendo ao que segue:

- Rotação - os documentos foram rotacionados no sentido horário em 1º, 2º, 3º, 10º e 12º;
- Resolução - foram gerados documentos com valores de 75, 100, 150, 200, 250, 300, 400 e 500 dpi;
- Brilho - considerando que a imagem gerada pelo scanner era o ponto inicial o brilho foi ajustado para variar a cada 5% no intervalo de -255% até +255%.

A Figura 12 mostra um comparativo entre o arquivo original e os arquivos gerados após a mudança de rotação em 12º, resolução de 300 dpi, brilho para +100% e contraste para +100%, respectivamente.

A cada mudança de configuração nas imagens *true color* e tons de cinza era gerado um novo arquivo em BMP (sem compressão). Todos estes arquivos foram passados pelo Omnipage que gerava textos em preto e branco no formato TXT.

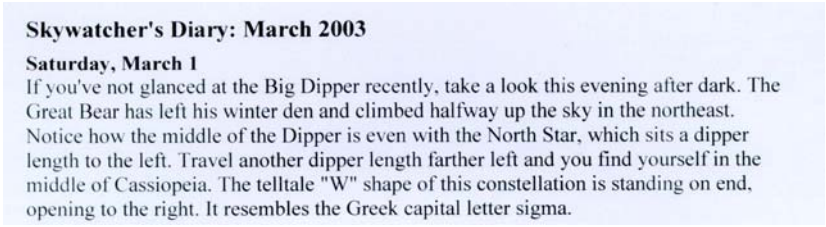
A comparação entre os arquivos gerados pela ferramenta de OCR e os arquivos corretos (originais) teve que ser realizada mediante algumas restrições. Devido à saída do OCR ser composta por ruídos (caracteres provenientes de sujeiras na imagem), um pré-processamento foi feito em todos os arquivos a fim de facilitar o processo de comparação automática entre os textos. Nesse pré-processamento para indicar linhas excluídas e linhas inseridas adotou-se os seguintes procedimentos:

1. Para a exclusão de linhas foi adotada, em uma linha gerada manualmente, a escrita do caractere menos (-);
2. Para a inclusão de linhas foi adotada a escrita do caractere mais (+) no início da linha, sem excluir os demais caracteres, os quais eram considerados caracteres incluídos.

Em alguns casos a linha foi duplicada, porém permanecendo na mesma linha, estes caracteres gerados a mais também foram considerados como incluídos. A Figura 11 mostra o uso dos símbolos (+ e -) em um dos arquivos testados. Para concluir o pré-processamento os textos foram normalizados, conforme os procedimentos descritos anteriormente.

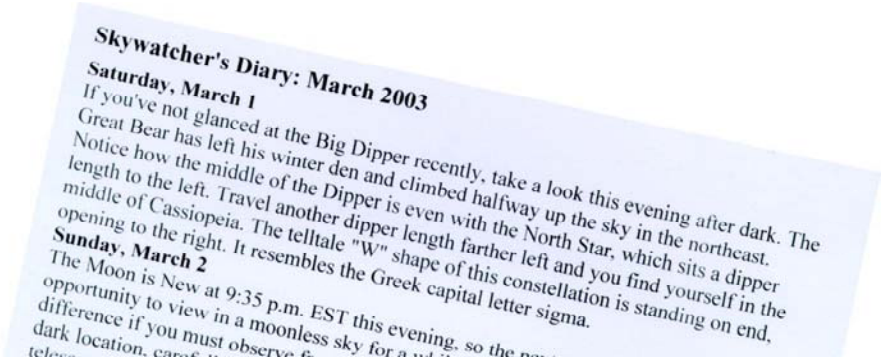
```
The FBI Fingerprint Image Compression
+ vestiqat
Standard
-
This page contains a summary of the Federal Bureau of Investigation's image coding
```

Figura 11. Exemplo de inclusão e exclusão de linhas



Skywatcher's Diary: March 2003

Saturday, March 1
If you've not glanced at the Big Dipper recently, take a look this evening after dark. The Great Bear has left his winter den and climbed halfway up the sky in the northeast. Notice how the middle of the Dipper is even with the North Star, which sits a dipper length to the left. Travel another dipper length farther left and you find yourself in the middle of Cassiopeia. The telltale "W" shape of this constellation is standing on end, opening to the right. It resembles the Greek capital letter sigma.



Skywatcher's Diary: March 2003

Saturday, March 1
If you've not glanced at the Big Dipper recently, take a look this evening after dark. The Great Bear has left his winter den and climbed halfway up the sky in the northeast. Notice how the middle of the Dipper is even with the North Star, which sits a dipper length to the left. Travel another dipper length farther left and you find yourself in the middle of Cassiopeia. The telltale "W" shape of this constellation is standing on end, opening to the right. It resembles the Greek capital letter sigma.

Sunday, March 2
The Moon is New at 9:35 p.m. EST this evening, so the
opportunity to view in a moonless sky for a
difference if you must observe f
dark location, care

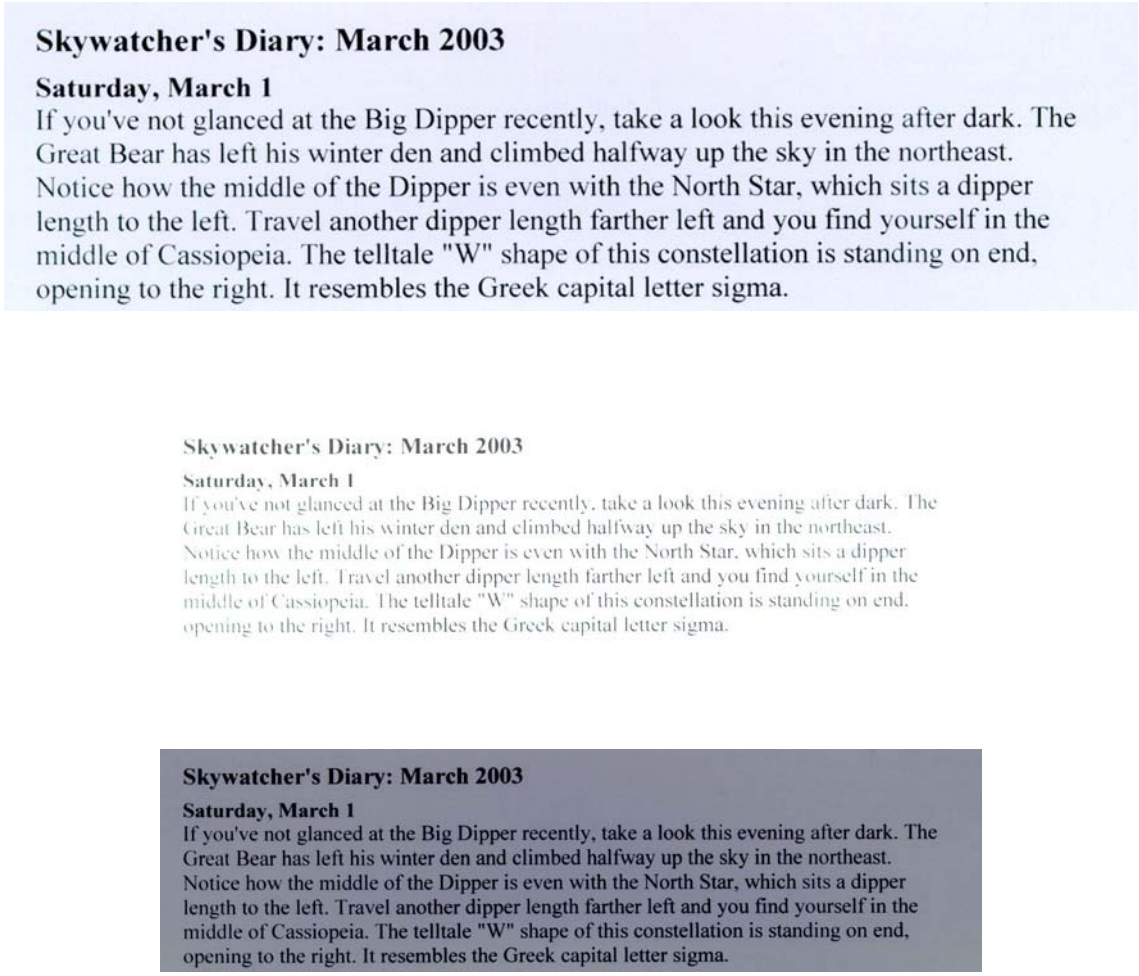


Figura 12. Trecho de arquivos original e gerados após rotação, resolução, brilho e contraste

A Figura 13, ilustra as etapas de geração de cada arquivo BMP e TXT, bem como o processo de comparação entre o arquivo correto e os demais arquivos gerados.

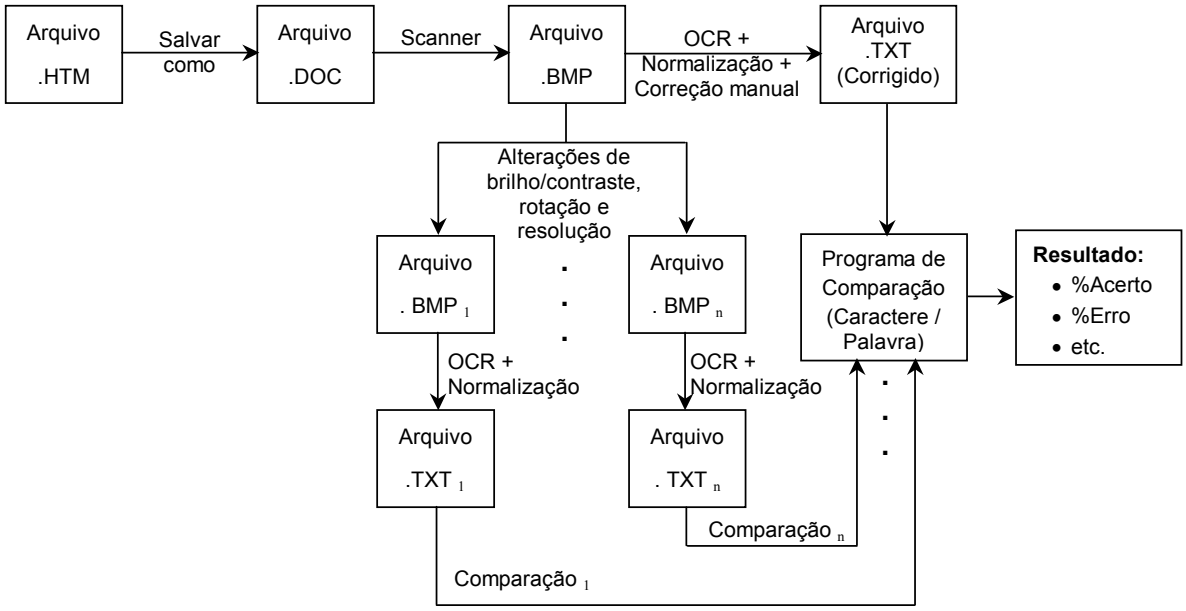


Figura 13. Etapas da geração de BMP para TXT

Para este estudo, além da avaliação por caractere foi considerada a avaliação por palavra. Tanto nos experimentos por caractere quanto nos experimentos por palavras foram testadas a sensibilidade dos documentos quanto à resolução, rotação e brilho. Nos próximos tópicos estão descritos os resultados obtidos com os experimentos.

3.5.1 Sensibilidade à Resolução

Nos experimentos de resolução, em relação ao estudo feito em [ML99a], foram testados mais 3 (três) resoluções: 300 dpi, 400 dpi e 500 dpi. Nos testes observou-se que quanto maior a resolução, maior a quantidade de detalhes e, conseqüentemente, maior a presença de ruídos, os quais geram caracteres fora do texto e até colunas extras. Em alguns arquivos foram gerados caracteres fora do texto os quais criaram uma ou duas colunas de textos além da margem do documento original, mas com o uso da técnica de normalização os caracteres ficaram todos alinhados à esquerda e foram classificados como erros de inclusão.

Percebe-se que para as resoluções de 75 e 100 dpi os resultados foram melhores para escala de cinza, nas demais resoluções a superioridade é visível para true color. Observa-se também que a resolução 300 dpi obteve um melhor resultado. Vale destacar que a ferramenta Omnipage aconselha o uso desta resolução indicando como a de melhor resultado. Os índices de acertos quanto à sensibilidade à resolução, para as imagens em tons de cinza e *true color*

são mostrados nas Tabelas 3 e 4 para comparação entre caractere e palavra, respectivamente, usando a nova métrica descrita na seção anterior.

Tabela 3. Percentual de Acertos quanto à Sensibilidade à Resolução por Caractere

DPI	Escala de Cinza (%)	True Color (%)
75	75,59	72,95
100	84,94	83,72
150	87,86	89,18
200	92,62	94,41
250	86,74	94,13
300	92,80	95,10
400	88,18	91,57
500	80,11	90,99

Tabela 4. Percentual de Acertos quanto à Sensibilidade à Resolução por Palavra

DPI	Escala de Cinza (%)	True Color (%)
75	67,14	59,57
100	78,55	78,08
150	84,24	83,87
200	90,47	85,00
250	84,83	90,95
300	90,22	90,98
400	82,29	89,88
500	78,65	87,31

Os resultados por palavra ficaram abaixo dos acertos por caractere devido ao fato de que os erros ocorrem em palavras distintas, então o número de erros por caractere e palavra é igual, porém o universo (total de caractere e total de palavras) de comparação é bem distinto, pois há um número menor de palavras.

3.5.2 Sensibilidade à Rotação

Quanto à rotação, os testes realizados em [ML99a] não são claros quanto ao índice de acertos para 10 e 12 graus, porém é afirmado que para 12 graus o nível de acerto registrou uma diferença de 1% em relação aos graus anteriores. Neste trabalho foram testados os graus 10 e 12 para comprovar que a diferença é mínima em relação as demais rotações de 1, 2 e 3 graus. Percebe-se que o índice de acerto ficou equivalente para as 5 (cinco) comparações, porém o índice de acerto para 12 graus foi superior a todos os demais graus de rotação, registrando 87,26% e 88,68% para cinza e *true color*, respectivamente, estes resultados estão

descritos na Tabela 5. Na comparação entre caractere e palavra, 12 graus continuou apresentando o melhor resultado, como pode ser observado na Tabela 6.

Tabela 5. Acertos quanto à Sensibilidade à Rotação por Caractere

DPI	Escala de Cinza (%)	True Color (%)
1 grau	86,15	88,38
2 graus	86,80	87,47
3 graus	85,89	87,38
10 graus	83,85	86,75
12 graus	87,26	88,68

Tabela 6. Acertos quanto à Sensibilidade à Rotação por Palavra

DPI	Escala de Cinza (%)	True Color (%)
1 grau	81,68	81,88
2 graus	75,39	75,74
3 graus	81,54	82,07
10 graus	74,66	74,08
12 graus	82,43	83,05

3.5.3 Sensibilidade ao Brilho

Quanto ao brilho, foi observado que a partir de +230%, a imagem ficava totalmente branca, da mesma forma que a partir de -205% ou -215%, dependendo do documento, a imagem ficava totalmente escura, sem caracteres, gerando deste modo arquivos de textos em branco. Os resultados no intervalo de -180% até +40% são próximos de 100% de acerto como pode ser observado na Figura 14 e na Figura 15, as quais mostram os gráficos com a variação quanto ao brilho por caractere e palavra para todos os textos, em escala de cinza e *true color*. O melhor valor para brilho, como descrito na Tabela 7, para documentos em tons de cinza foi 175% para caractere e 185% para palavra; em *true color* os melhores valores foram de 170% e 185% para caractere e palavra, respectivamente.

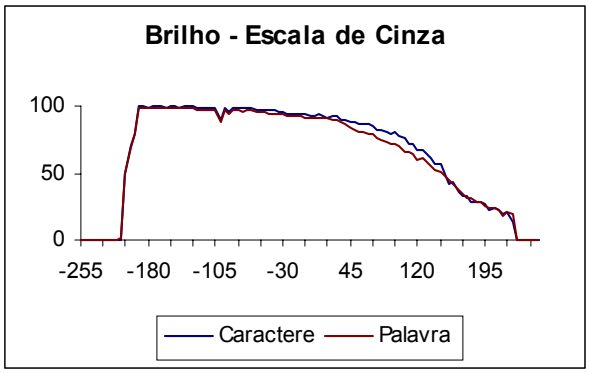


Figura 14. Gráfico de brilho para escala de cinza

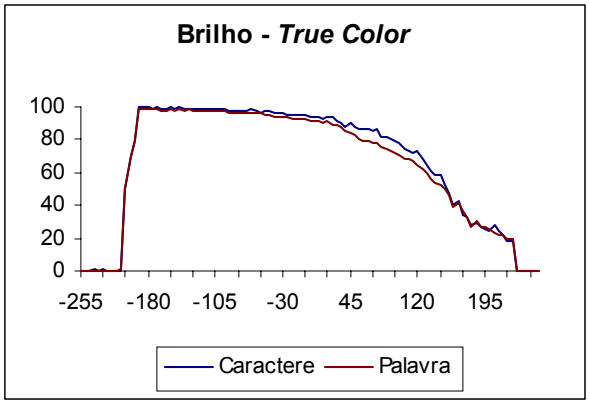


Figura 15. Gráfico de brilho para *true color*

A Figura 16, extraída de [ML99a], mostra o gráfico obtido com os experimentos para brilho do Omnipage 8, com este é possível ver que a tendência dos resultados não mudou muito, pois, ambos com valores extremos de brilho, não reconhecem os textos. Nos demais pontos do gráfico, o reconhecimento fica quase que constante, porém apresentando melhores resultados quando o documento tem menor valor de brilho ou maior contraste.

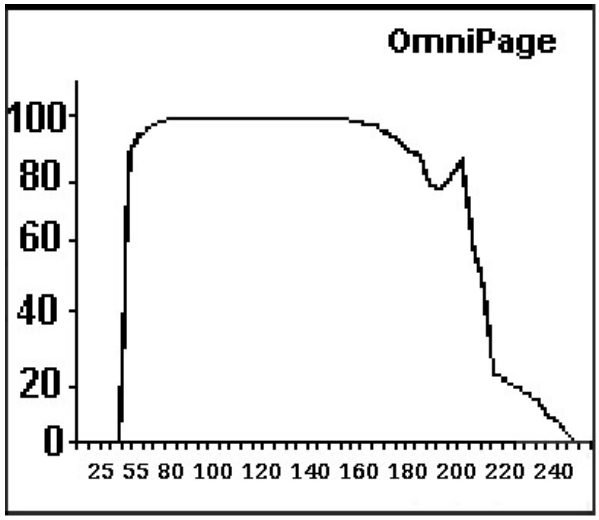


Figura 16. Gráfico de brilho Omnipage 8

Tabela 7. Valor de brilho que produziu melhores respostas do OCR

Software	Melhor valor de Brilho	
	Caractere	Palavra
Escala de Cinza	-175	-185
True Color	-170	-185

3.5.4 Classificação Geral dos Erros

Na comparação por caractere nos arquivos em tons de cinza, dentre os erros gerados, 42,03% foram detectados nos arquivos em língua portuguesa e 57,97% em língua inglesa. Nos arquivos em *true color* este percentual permaneceu estável, sendo 44,24% em língua portuguesa e 55,76% em língua inglesa. As Figuras 16 e 17 apresentam estes resultados graficamente.

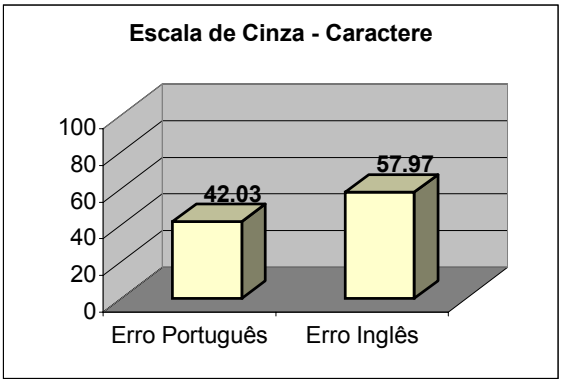


Figura 17. Erros por caractere em escala de cinza distribuídos por tipo de arquivo

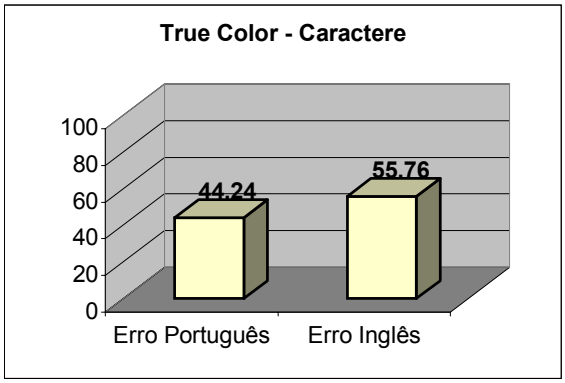


Figura 18. Erros por caractere em *true color* distribuídos por tipo de arquivo

Na comparação por palavra nos arquivos em tons de cinza, dentre os erros gerados, 42,73% foram detectados nos arquivos em língua portuguesa e 57,27% em língua inglesa. Nos arquivos em *true color* este percentual permaneceu estável, sendo 44,26% em língua portuguesa e 55,74% em língua inglesa. As Figuras 18 e 19 apresentam estes resultados graficamente.

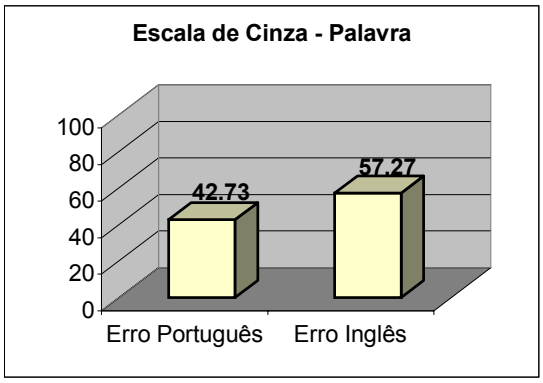


Figura 19. Gráfico dos erros por palavra em escala de cinza distribuídos por tipo de arquivo

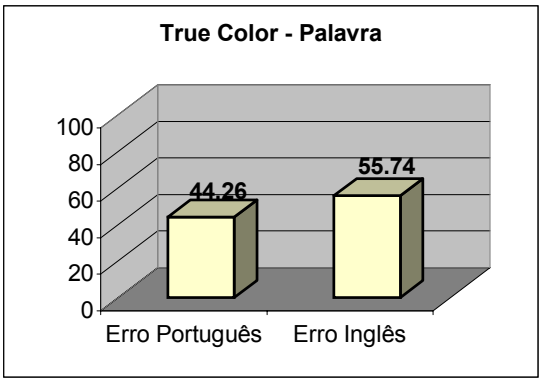


Figura 20. Gráfico dos erros por palavra em *true color* distribuídos por tipo de arquivo

De posse dos resultados, dos erros por arquivo, percebemos que os erros, tanto de caracteres quanto de palavra, em escala de cinza ou *true color*, são constantes. Os erros gerados por caractere foram classificados e distribuídos em:

- Inclusão;
- Exclusão;
- Troca;
- Junção.

Quanto a classificação por tipo de erro destacam-se os erros de troca os quais são quase a metade dos erros com 46,91% para tons de cinza e 46,94% para *true color*. A Figura 21 e a Figura 22 demonstram estes resultados em forma de gráfico.

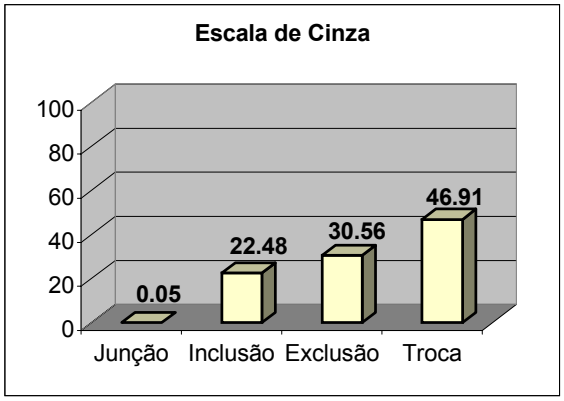


Figura 21. Gráfico dos tipos de erros em escala de cinza

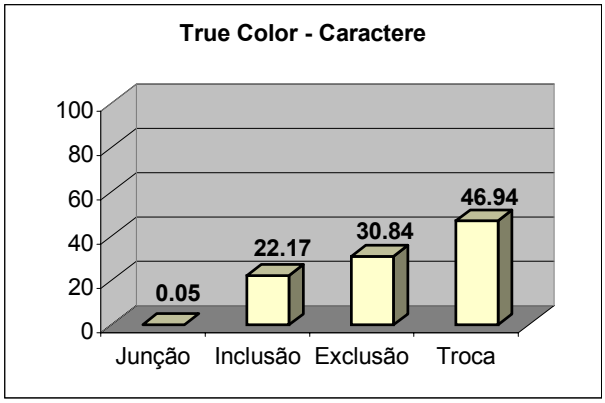


Figura 22. Gráfico dos tipos de erros em *true color*

Dentre os 3.856 erros de troca 1.098 ocorreram entre os 12 maiores erros, o que equivale a 28,48% dos erros de trocas. Analisando a Tabela 8 é possível observar que as maiores troca foram de acentuação gráfica. Daí conclui-se que a acentuação da língua portuguesa contribuiu com 17,85% para os erros de troca de caracteres.

Tabela 8. Erros de Troca mais Relevantes

Caractere	Qtde. de Erros	Erros (%)
c - v	51	1,32%
a - â	52	1,35%
e - o	52	1,35%
rn - m	52	1,35%
u - ú	52	1,35%
m - rn	54	1,40%
i - í	58	1,50%
- -	96	2,49%
e - é	102	2,65%
t - c	105	2,72%
a - ã	156	4,05%
a - á	268	6,95%
Outras Trocas	2.758	71,52%
Total de Erros	3.856	100,00%

Na Tabela 9 é mostrado o índice de erros e acertos de acordo com o tamanho da palavra, tanto em escala de cinza quanto em *true color*. É possível observar que não há uma relação direta entre erros gerados e tamanhos da palavra, pois o maior índice de erro está na palavra com 3 (três) caracteres e o menor na palavra de 1 (um) caractere, porém as palavras que foram mais aceitas foram as de 2 (dois) caracteres, tanto para escala de cinza quanto para *true color*.

Tabela 9. Média de acertos e erros por tamanho da palavra

Palavra	Escala de Cinza		*True Color*	
	Erros (%)	Acertos (%)	Erros (%)	Acertos (%)
1 Caractere	4,60	6,96	4,75	6,96
2 Caracteres	12,86	16,84	12,94	16,84
3 Caracteres	14,69	15,70	14,74	15,68
4 Caracteres	12,92	11,63	12,82	11,67
5 Caracteres	10,14	9,37	10,09	9,39
6 Caracteres	9,71	7,90	9,74	7,89
7 Caracteres	9,50	8,73	9,52	8,73
8 Caracteres	7,43	7,27	7,41	7,27
9 Caracteres	5,56	5,38	5,59	5,35
>9 Caracteres	12,59	10,22	12,40	10,22
Total	100,00	100,00	100,00	100,00

Foi verificado também que erros aparecem com maior freqüência em algumas palavras. Há casos em que, em uma mesma palavra, ocorrem erros diversos como de inclusão, exclusão e troca.

De posse de todos os resultados, foi possível observar que os arquivos em Língua Inglesa apresentaram um índice maior de erros do que aqueles em Língua Portuguesa. Acredita-se que este fato ocorre porque o software de reconhecimento seja mais preciso ao analisar arquivos que possuam características inerentes às da Língua Portuguesa, como a pontuação.

4 MELHORIA DE FERRAMENTAS DE OCR ATRAVÉS DE FILTROS

Quando imagens sofrem alterações através do aumento ou diminuição do brilho e do contraste, por exemplo, na realidade, como foi visto no capítulo 1, estas imagens estão sendo filtradas. Cada mudança, por menor que seja, no ajuste dos filtros gera uma nova imagem e conseqüentemente, esta, se passadas por um OCR, gera um novo arquivo-texto. Baseado nestas condições, este capítulo descreve o processo de criação de um arquivo-texto através da seleção de caracteres entre vários arquivos que sofreram mudanças mínimas na sua imagem.

4.1 Descrição do Experimento

De posse dos resultados dos testes de desempenho de ferramentas comerciais de OCR, descritos no capítulo 3, foi constatado que o melhor valor para brilho, em escala de cinza, era de -175%, considerando o intervalo -255% a +255%. Para os documentos em *true color* o intervalo de melhor desempenho foi de 170%. Quanto à resolução o melhor valor foi de 200 dpi para escala de cinza e 300 dpi para *true color*.

Com base nestes resultados foram utilizados os mesmos grupos de documentos do experimento anterior, além de mais 10 arquivos, sendo 5 textos em inglês e 5 em português. O processo de seleção e digitalização destes documentos é semelhante ao processo descrito anteriormente, porém a impressão foi em impressora jato de tinta, marca Canon S100 [Can03] com 300 dpi de resolução. Os textos também têm em média 3.000 (três mil) caracteres e 500 (quinhentas) palavras.

Para a digitalização os valores de brilho e contraste foram configurados conforme o padrão do software HP Deskscan II, o qual acompanha o scanner. Os valores de brilho foram ajustados entre 118% e 119% e os valores de contraste entre 133% e 137%. Os documentos foram digitalizados em *true color*, da esquerda para a direita e de cima para baixo.

Nestes novos documentos, o brilho foi o único fator que sofreu alterações, não havendo mudanças de rotação e resolução. Os documentos também foram normalizados

De posse dos arquivos-texto foram selecionados apenas 5 arquivos, para cada conjunto de texto, sendo 1 gerado diretamente do *scanner* e 4 com os valores de brilho variando entre os valores considerados ideais, ou seja, em torno de -170%.

Na fase de comparação e montagem do novo arquivo, primeiramente comparam-se as linhas e os arquivos que apresentarem a maior freqüência, quanto à quantidade de linhas, serão os selecionados para próxima fase. É importante ressaltar que as comparações feitas nos arquivos levam em consideração a medida da moda e não a maior quantidade. No caso das linhas, o arquivo que apresenta a maior quantidade de linhas não é necessariamente o mais correto, pois as linhas podem ter sido geradas por ruído. Por exemplo, se dentre os 5 arquivos selecionados para filtragem tem-se um com 70 linhas, um outro com 55 e três arquivos com 60, neste caso será considerado apenas os que apresentarem a maior freqüência, ou seja, 60 linhas, os demais arquivos serão desconsiderados.

Na segunda fase, os arquivos selecionados na fase anterior, são comparados caractere a caractere e, através de voto majoritário ponderado, o caractere que apresentar a maior freqüência será o escolhido e gravado no novo arquivo. Caso a freqüência seja igual a 50% do número de arquivos mais 1, este caractere é gravado, caso contrário, o caractere do arquivo gerado através do *scanner*, sem sofrer mudanças por filtros, será aceito como correto e gravado.

Uma das dificuldades na transcrição são os erros gerados por pontuação: é comum a troca do caractere ponto (**.**) pelo caractere vírgula (**,**) e vice-versa. O algoritmo desenvolvido, e chamado de **Algoritmo de Comparação de Texto**, faz uma análise para avaliar se ponto (**.**) e vírgula (**,**) estão sendo empregados corretamente.

Outro caso que acontece na pontuação é a troca de caracteres "normais" pelos caracteres "especiais" *e* comercial (**&**) e exclamação (**!**). O uso destas técnica de correção de pontuação está representado na Figura 23, como sendo um dicionário que, juntamente com o "voto majoritário ponderado", formam o novo texto. Porém, o dicionário só irá interferir caso o caractere resultante seja diferente do caractere do documento gerado diretamente do scanner, pois este documento tem peso maior na "votação". A Tabela 10 mostra as combinações de caracteres que são analisadas e as ações adotadas para correção da pontuação.

Tabela 10. Técnicas de Dicionário

Combinação	Ação	Resultado
vírgula + espaço + caractere maiúsculo	trocar vírgula por ponto	ponto + espaço + caractere maiúsculo
ponto + espaço + caractere minúsculo	trocar ponto por vírgula	vírgula + espaço + caractere minúsculo
caractere **&**	trocar por outro caractere (caso exista)	novo caractere

Complementando a tabela anterior, se aparecem os caracteres **&** e **!** como maior freqüência e existir, em pelo menos um arquivo, outro caractere este será considerado o correto. Após todas estas considerações um novo texto é gerado. A Figura 23, apresenta um esquema da transposição de imagem para texto, da solução proposta.

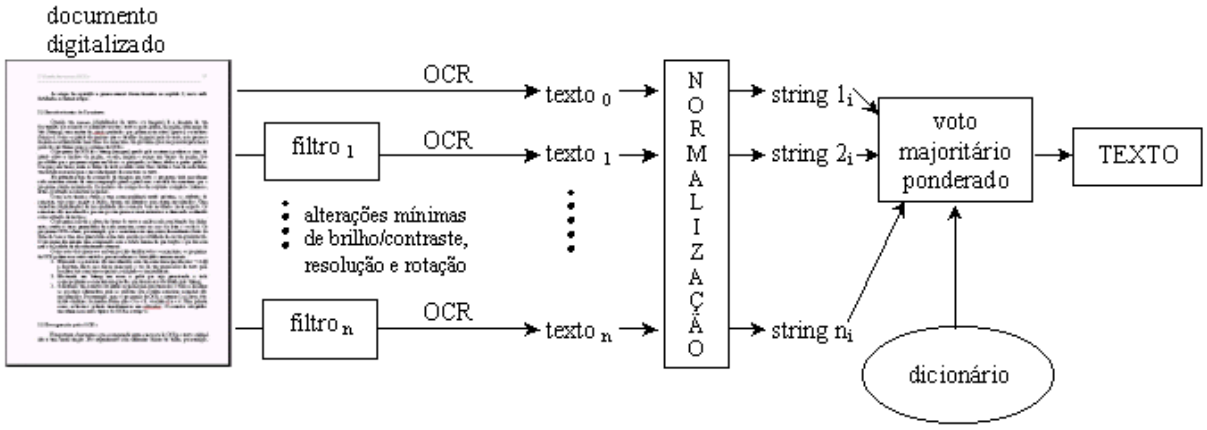


Figura 23. Esquema da solução proposta para Transposição de Imagem para Texto

O algoritmo utilizado no processo de filtragem dos textos está descrito a seguir. Este é composto de duas funções, uma para compara linhas e outra para comparar os caracteres. Em ambas monta-se uma matriz de comparação, esta é semelhante ao algoritmo da distância de Levenshtein, porém avalia a maior medida da moda. O Anexo B contém a listagem da implementação deste algoritmo.

Algoritmo de Comparação de Texto

1. Abre os arquivos;
2. Conta as linhas;
3. Conta os caracteres;
4. Chama a **Função Compara Linha**;
5. Monta as linhas de caracteres;
6. Chama a **Função Compara Caractere**;
7. Se freqüência de caracteres diferente do texto do scanner
   Então Analisa os caracteres vírgula (**,**), ponto (**.**), *e* comercial (**&**) e exclamação (**!**);
8. Escreve o caractere selecionado no novo arquivo;
9. Fim.

Função Compara Linha

1. Monta uma matriz para comparar quantos tamanhos de linhas são iguais;
2. Se existir um total de freqüência entre um arquivo maior que a média de arquivos
   Então Acha a Freqüência do número de Linhas;
   Senão Ordena as freqüências e pega a maior.

Função Compara Caractere

1. Monta uma matriz para comparar quantos tamanhos de caracteres são iguais;
2. Se existir um total de freqüência entre os arquivos maior que a média de arquivos
   Então Acha a Freqüência de Caracteres;
   Senão Ordena as freqüências e pegar a maior.

4.2 Resultados do Experimento

Para avaliar o percentual de acertos dos novos arquivos gerados, comparou-se, através das métricas propostas neste trabalho, o arquivo original com o novo arquivo, os resultados alcançados, em *true color* estão na Tabela 11 e em escala de cinza na Tabela 12, nestas os valores estão dispostos em quatro colunas, a saber:

- Direto - indica o percentual de acertos do arquivo passado diretamente do scanner que não sofreu qualquer alteração por filtros;
- Melhor - traz o melhor resultado alcançado nos experimentos descritos no capítulo 3. O resultado tanto pode ser de brilho quanto de resolução;
- Filtro - mostra o resultado alcançado através da técnica de filtragem proposta neste trabalho;
- Variação da Qualidade - mostra em quanto variou o percentual de acerto entre os valores de procedência do scanner e os obtidos através dos filtros.

Tabela 11. Resultados com o uso de Filtros em *True Color*

Arquivo	***True Color* (%)**			
	Direto (1)	**Melhor (2)**	**Filtro (3)**	**Variação da Qualidade (3-1)**
1. Inglês	99,67	100,00	100,00	0,33
2. Inglês	94,33	99,80	99,47	5,17
3. Inglês	95,60	100,00	99,88	4,29
4. Inglês	85,35	99,56	89,60	4,74
5. Inglês	94,57	99,90	99,87	5,31
6. Inglês	100,00	100,00	100,00	-
7. Inglês	99,78	99,89	99,89	0,11
8. Inglês	99,97	100,00	99,94	-0,03
9. Inglês	99,90	99,93	99,90	-
10. Inglês	100,00	100,00	100,00	-
1. Português	99,53	99,93	99,77	0,24
2. Português	91,31	99,66	99,63	8,35
3. Português	82,64	99,92	99,18	16,68
4. Português	99,22	99,79	99,74	0,52
5. Português	99,83	99,91	99,74	-0,09
6. Português	99,94	100,00	99,96	0,02
7. Português	99,90	99,97	99,90	-
8. Português	99,70	99,88	99,79	0,09
9. Português	99,71	99,91	99,83	0,12
10. Português	99,63	99,83	99,80	0,17

Tabela 12. Resultados com o uso de Filtros em Escala de Cinza

Arquivo	Escala de Cinza (%)			
	Direto (1)	**Melhor (2)**	**Filtro (3)**	**Variação da qualidade (3-1)**
1. Inglês	99,67	99,82	100,00	0,33
2. Inglês	86,83	89,22	99,47	12,71
3. Inglês	92,72	95,43	99,91	7,20
4. Inglês	81,43	84,72	90,86	10,38
5. Inglês	94,54	98,53	99,87	5,34
6. Inglês	100,00	100,00	100,00	-
7. Inglês	99,89	99,89	99,89	-
8. Inglês	99,97	99,97	99,94	-0,03
9. Inglês	99,90	99,93	99,90	-
10. Inglês	100,00	100,00	100,00	-
1. Português	99,55	99,55	99,72	0,17
2. Português	92,14	96,31	99,63	7,52
3. Português	85,98	90,80	99,18	13,31
4. Português	99,22	99,79	99,79	0,57
5. Português	99,83	99,83	99,74	-0,09
6. Português	99,94	100,00	99,96	0,02
7. Português	99,90	99,97	99,90	-
8. Português	99,70	99,88	99,79	0,09
9. Português	99,71	99,91	99,83	0,12
10. Português	99,60	99,80	99,77	0,17

Analisando as Tabelas 11 e 12 percebe-se que os primeiros arquivos que vão de 1 a 5 tem um índice de acertos, através dos filtros, melhor que os arquivos entre 6 e 10 tanto para língua inglesa quanto para língua portuguesa. Acredita-se que a diferença provenha da qualidade de impressão dos textos gerados. Na impressora jato de tinta, os textos estavam mais legíveis. Quando os arquivos possuem poucos erros é mais difícil conseguir uma melhora, porém quando se têm vários arquivos com uma qualidade baixa é mais provável alcançar um resultado melhor como pode ser observado no arquivo 3 em *Português*, onde a taxa de erro estava em quase 15% e com o uso dos filtros o índice melhorou em 13,31%.

As Figuras 23 e 24 mostram graficamente de variação de acertos entre os textos extraídos diretamente do scanner e os textos gerados com o uso de filtros. Nesses gráficos estão todos os 20 arquivos testados, em *true color* e escala de cinza.

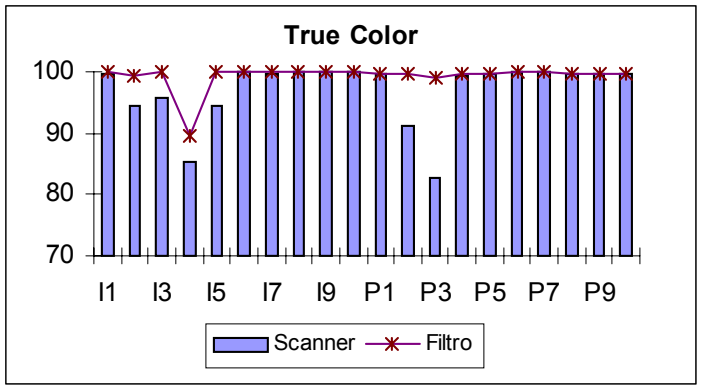


Figura 24. Gráfico de Resultado de Filtro para Arquivos em *True Color*

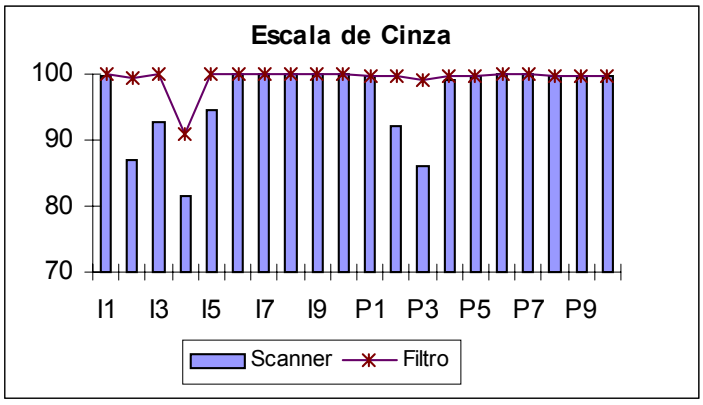


Figura 25. Gráfico de Resultado de Filtros para Arquivos em Escala de Cinza

Também foram feitos vários testes no intervalo de brilho, quando o valor do mesmo está próximo do valor do scanner, a melhora é pouco significativa, porém acontecem menos erros de troca de caracteres, pois a variação é pequena. Entretanto, quando se utiliza o intervalo onde ocorrem os melhores valores é possível que haja uma grande quantidade de erros, pois a comparação é feita entre a origem e os demais valores que estão muito distante no histograma. Nos experimentos, foi observado que as imagens com alto valor de brilho produzem mais erros de troca de caracteres, justamente pela claridade do documento, já as imagens escuras produzem inserção de caracteres, pois os ruídos ficam mais visíveis e qualquer ponto escuro pode vir a tornar-se um caractere.

Após a conclusão dos experimentos foi constatado que um número pequeno de erros (0,01%), mas frequente nos vários textos transcritos do mesmo arquivo, dificilmente levará a um reconhecimento de 100%, porém quando são gerados muitos erros (20% a 30%), mas disjuntos, é muito provável, através da análise dos vários textos gerados, encontrar a transcrição ideal com cerca de 100% de acertos.

As Figuras 23 e 24 mostram graficamente os resultados obtidos com a técnica de filtragem para *true color* e escala de cinza, respectivamente, para todos os arquivos testados. Comparando-se os resultados percebe-se uma eficiência quando se usa *true color*, porém esta eficiência é muito pequena frente ao custo de se trabalhar com imagens coloridas, principalmente o custo de armazenamento das imagens.

Caso as imagens sejam descartadas após a digitalização, pode-se usar uma melhor resolução na captura, mas se, além do texto, deseja-se armazenar a imagem e não se tenha espaço suficiente, é preferível trabalhar com figuras em escala de cinza, pois apresentam a melhor relação custo *versus* beneficio.

A Tabela 13 mostra um pequeno trecho do texto em português e em *true color* que melhorou 16,68%. Na coluna arquivo está relacionado os nomes dos arquivos e na coluna textos estão o texto gerado de acordo com cada arquivo. Na última linha demonstram os erros encontrados pelo arquivo de comparação quando comparou o *corrigido* com o texto vindo diretamente do *scanner*, neste pequeno trecho foram gerados 3 erros, ambos de troca de caracteres.

Tabela 13. Textos Gerados

Arquivo	**Textos**
Corrigido	Eles colecionam tudo que surge no estilo das chamadas -não sem motivo- "aloha shirts", ou
Scanner	Lies colecionani tudo que surge no estilo das chamadas -não sem motivo- "~tloha shirts", ou
Brilho -160	Eles colecionam tudo que surge no estilo das chamadas -não sem motivo- "aloha shirts", ou
Brilho -165	Eles colecionam tudo que surge no estilo das chamadas -não sem motivo- "aloha shirts", ou
Brilho -170	Eles colecionam tudo que surge no estilo das chamadas -não sem motivo- "aloha shirts", ou
Brilho -175	Eles colecionam tudo que surge no estilo das chamadas -não sem motivo- "aloha shirts", ou
Filtro	Eles colecionam tudo que surge no estilo das chamadas -não sem motivo- "aloha shirts", ou

Erros encontrados pelo programa de comparação de caracteres:

```
Troca 2x          : El - LI  Linha: 15   ColunaO: 0        ColunaG: 0
Troca 12          : m - ni   Linha: 15   ColunaO: 14       ColunaG: 14
Troca 12          : a - ~t   Linha: 15   ColunaO: 72       ColunaG: 73
```

CONCLUSÕES E TRABALHOS FUTUROS

Em plena era da informação, inúmeros documentos são gerados e arquivados diariamente. Mesmo com a possibilidade de uso de tecnologia digital, muitos destes documentos são gerados apenas em papel, não sendo digitalizados. Uma solução seria digitalizá-los e armazená-los na forma de arquivo-texto, devido a menor utilização de espaço em disco do que os arquivos em formato de imagem.

O processo de transcrição de documentos em papel para documentos digitalizados envolve várias etapas de processamento de imagem como:

- aquisição;
- pré-processamento;
- segmentação;
- representação; e
- reconhecimento óptico dos caracteres.

Porém, mesmo que nas fases descritas acima haja um alto grau de qualidade, não há como avaliar o quanto o texto transcrito está próximo do documento original. Como solução pode-se utilizar métricas ou metodologias para avaliar estas transcrições.

A meta das ferramentas para o Reconhecimento Óptico de Caracteres é o reconhecimento automático de caracteres em qualquer tipo de língua e símbolo, de maneira rápida e correta. Embora as ferramentas comerciais tenham auferido grande sucesso nessa direção, a solução geral do problema, que ofereça uma transcrição satisfatória, ainda é distante.

Nesta dissertação apresentou-se uma nova métrica, baseada na Distância de Levenshtein, para avaliar os erros gerados por caracteres, palavra e linhas de texto. Os documentos analisados são recentes e foram impressos em impressora laser. De acordo com a pesquisa descrita em [Mel02] documentos históricos, datilografados ou manuscritos, apresentam uma taxa de acerto bastante reduzida em relação aos documentos aqui estudados.

A classificação de erro proposta foi usada para analisar os aspectos da versão corrente da ferramenta Omnipage. O resultado dos experimetos mostrou que a melhor resolução para digitalização neste OCR é de 300 dpi, tanto para imagens em *true color* quanto em escala de cinza. Omnipage mostrou pouca sensibilidade para rotação do documento, porém para a rotação de 12 graus, o índice de acerto foi superior aos demais graus de rotação.

Quanto ao brilho, ao elevar em torno de 10% do valor padrão utilizado pelo *scanner* houve uma melhora para tons de cinza. Para *true color*, este valor melhorou quando o aumento foi de apenas 5% deste valor padrão. Apesar da dificuldade de reconhecimento em função da acentuação gráfica, os textos em Língua Portuguesa apresentaram um índice de acerto maior que os textos em Língua Inglesa. Acredita-se que este fato ocorre porque o software de reconhecimento seja mais preciso ao analisar arquivos que possuam características inerentes às da Língua Portuguesa.

Observando os resultados obtidos, nota-se que as transcrições melhoraram consideralvemente fazendo com que o objetivo do trabalho fosse alcançado. Taís resultados poderão ser adicionados às ferramentas comerciais de OCR's para facilitar e aumentar o índice de acertos das transcrições. A metodologia proposta serve para avaliar tanto caracteres datilografados quanto manuscritos.

Entre as contribuições desta dissertação destacam-se:

- Avaliação dos pontos de ótimos da ferramenta comercial Omnipage versão 12 para brilho, contraste, rotação e resolução;
- Criação de uma métrica para avaliar transcrições de OCR's, partindo-se da normalização dos textos até às comparações por caractere e por palavra;
- Uso de imagens obtidas através de filtros para alteração de brilho e contraste, gerando um texto, através de voto majoritário ponderado, com uma transcrição melhor que da imagem oriunda do *scanner*.

Como trabalho futuro pode-se indicar o desenvolvimento de um sistema de transcrição múltipla, utilizando mais de uma ferramenta de OCR e analisando suas diversas respostas. Também é possível a criação de uma interface gráfica que trabalhe com o SDK (Software Development Kit) do Omnipage, permitindo assim uma iteração mais rápida entre o sistema de filtros aqui descrito e o Omnipage. A interface pode facilitar o manuseio por usuários leigos quanto ao processo de digitalização de imagens e posterior transcrição.

REFERÊNCIAS BIBLIOGRÁFICAS

[AIIM03] AIIM Internacional (Association for Information and Image Management International). http://www.aiim.org Visitado em 16/06/2003.

[AT95] Awcock, G. J.; Thomas, R. **Applied Image Processing**. The Macmillan Press LTD, 1995.

[BCL00] Braga, A. de P.; Carvalho, A. C. P. L. F; Ludemir, T. B. **Redes Neurais Artificiais: Teoria e aplicações**. Rio de Janeiro: LTC, 2000.

[Can03] Canon: http://www.canonlatinamerica.com Visitado em 23/07/2003.

[Cen03] Cenadem (Centro Nacional de Desenvolvimento do Gerenciamento da Informação): http://www.cenadem.com.br Visitado em 06/05/2003.

[Cha03] International Paper: http://www.chamex.com.br Visitado em 03/04/2003.

[Fac93] Facon, J. **Processamento e Análise de Imagens**. VI Escola Brasileiro-Argentina de Informática, 1993.

[FN99] Marques Filho, O.; Vieira Neto, H. **Processamento Digital de Imagens**. Rio de Janeiro: Brasport, 1999.

[Fur02] Furasté, P. A. **Normas Técnicas para o Trabalho Científico**. 11ª ed. Porto Alegre: 2002.

[GV94] Gomes, J.; Velho, R. **Computação Gráfica: Imagem**. Rio de Janeiro: IMPA/SBM, 1994.

[GW92] Gonzalez, R. C.; Woods, R. E. **Digital Image Processing**. Addison Wesley Publishing Company, 1992.

[Hay94] Haykin, S. **Neural Networks - A Comprehensive Foundation**. Prentice Hall, 1994.

[Jai89] Jain, A. K. **Fundamentals of Digital Image Processing**. Prentice Hall, 1989.

[Jasc03] Jasc Software: http://www.jasc.com Visitado em 01/04/2003.

[JHD99] Junker, M.; Hoch, R.; Dengel, A. **On the Evaluation of Document Analysis Components by Recall, Precision, and Accuracy**. Proceedings of 5th International Conference on Documet Analysis and Recognition. India, pp. 713-716, 1999.

[Kur03] Kurzweil Technologies. http://www.kurzweiltech.com/founders.html Visitado em 25/06/2003.

[Lev66] Levenshtein, V. I. **Binary codes capable of correcting deletions, insertions and reversals**. *Soviet Phys. Dokl.*, páginas 707-710, Fevereiro de 1966.

[Lim03] Lima, E. L. **Espaços Métricos**. 3ª ed. Estado: IMPA, 2003.

[Mac01] Machado, D. S. A. **Aspectos do Desempenho de Algoritmos de Compressão de Imagens**. Recife: UFPE, 2001. Dissertação (Mestrado em Ciência da Computação), Centro de Informática, Universidade Federal de Pernambuco.

[Mel02] Mello, C. A. B. **Aspectos Computacionais do Processamento de Imagens de Documentos Históricos**. Recife: UFPE, 2002. Tese (Doutorado em Ciência da Computação), Centro de Informática, Universidade Federal de Pernambuco.

[ML99a] Mello, C. A. B.; Lins, R. D. **A Comparative Study on Commercial OCR Tools**. Proceedings of Vision Interface'99. Québec, Canadá, Maio, 1999.

[ML99b] Mello, C. A. B.; Lins, R. D. **Segmentação de Imagens de Documentos Históricos**. XVII Simpósio Brasileiro de Telecomunicações. Vila Velha-ES, Brasil, Setembro, 1999.

[Net98] França Neto, L. R. **Um Ambiente para Processamento de Grandes Acervos de Imagens**. Recife: UFPE, 1998. Dissertação (Mestrado em Ciência da

Computação), Depto. de Informática, Universidade Federal de Pernambuco.

[OK97] O'Gorman, L.; Kasturi, R. **Executive Briefing: Document Image Analysis**. California: IEEE Computer Society, 1997.

[Oki03] Oki Data: http://www.okidata.com/port/html/nf/Home.html Visitado em 03/04/2003.

[OSBS03] Oliveira, L. S.; Sabourin, R.; Bortolozzi, F.; Suen C. Y. **Impacts of Verification on a Numeral String Recognition System**. Pattern Recognition Letters, Vol.24, pp1023-1031. 2003.

[Pra91] Pratt, W. K. **Digital Image Processing**. 2ª ed. Wiley Interscience, 1991.

[Red03] Redes Neurais: http://www.icms.sc.usp.br/~andre/neural1.html Visitado em 16/06/2003.

[Ros58] Rosenblatt, F. **The perceptron: A probabilistic model for information storage and organizacion in the brain**. Psychol. Rev., 65:386-408,1958.

[SRT01] Silva, E.; Rodrigues, R. J.; Thomé, A. C. G. **Extração de Características para o Reconhecimento de Letras Manuscritas**. 5ª SBAI, Canela-RS. Nov-2001.

[SS03] ScanSoft. URL: http://www.scansoft.com Visitado em 17/03/2003.

[Tan00] Tan, C. L.; et al. **Removal of Interfering Strokes in Double-Seided Document Images**. Workshop on the Applications of Computer Vision 2000 (WACV00). California, pp.16-21, 2000.

[Vuo98] Vuolo, J. H. **Fundamentos da Teoria de Erros**. São Paulo: Edgard Blucher, 1998.

[ZS84] Zhang, T. Y.; Suen, C. Y. **A fast parallel algorithm for thinning digital patterns, Communications of the ACM**. v.27 n.3, p.236-239, Março 1984.

ANEXO A - IMAGENS DOS ARQUIVOS USADOS NOS EXPERIMENTOS

Welcome to OCR!

- OCR Qualifications are nationally recognised and represent the most respected names in education.
- OCR Qualifications provide enhancement and enrichment to learners of all ages and abilities.
- OCR Qualifications are created and provided through effective partnerships with over 13,000 educational establishments.
- OCR Qualifications reflect the requirements of business, industry and higher education, as well as creating increased levels of self-esteem.

Breadth of coverage
Through approximately 13,000 different educational establishments around the UK, OCR recognises the achievement of those people who know that qualifications can enhance their life, whether they be young people taking their GCSE's or older people returning to learning after many years' absence.

OCR is responsible to the Government as one of the UK's leading awarding bodies providing qualifications to students at school, college, in work or through part-time learning programmes.

Heritage
OCR was formed in 1998 from the University of Cambridge Local Examination Syndicate and RSA, which means that OCR offers an extremely broad range of qualifications, with the integrity, quality and assurances that you would expect from an organisation with such a respected heritage. Over 600 staff work for OCR on sites in Cambridge, Coventry, Birmingham and in six Regional Offices. Add to this the thousands of examiners, verifiers and consultants that mark, assess and write qualifications, and it will be recognised that we are one of the major organisations in the country.

National Qualifications Framework and OCR Headway
OCR Qualifications generally fall into two categories; those that are approved through the National Qualifications Framework, and those that are created through our OCR Headway Consultancy Service.
Those approved to the National Qualifications Framework attract funding and are available through nationally-recognised education centres. Those that are created through OCR Headway are developed specifically to meet the needs of particular employers or services.

The full range of the qualifications approved to the National Qualifications Framework can be viewed on this Website, together with the range of support services and materials available. We hope you find the information you require quickly and that it helps you in your teaching or studying.

Integrity of Qualification
It is very important to OCR that our qualifications accurately reflect the level of work undertaken by students. OCR does not compromise on standards, but we do provide a very high level of support and guidance to all those educational establishments running our qualifications. Our integrity is high, but we are also recognised as being extremely friendly and helpful.

Trusted Administration
OCR has a programme of continuous improvement in the management of our qualifications. We invest heavily in the training of our staff and the establishment of effective procedures. We became IiP accredited in 1999, and ISO accredited in 2000. Where problems have been identified we have worked hard to make sure they have been overcome, and that they do not re-occur. We work in partnership with our Centres to ensure between us the examination process is managed as efficiently as possible.

Figura 26. Inglês 1

The FBI Fingerprint Image Compression Standard

This page contains a summary of the Federal Bureau of Investigation's image coding standard for digitized fingerprints, developed and maintained by the FBI, Los Alamos National Lab, and the National Institute for Standards and Technology. The standard is a discrete wavelet transform-based algorithm referred to as Wavelet/Scalar Quantization (WSQ).

Summary and Examples:

The goal of this project is to design and implement a national standard for coding and compression of digitized fingerprint images. The FBI is digitizing the nation's fingerprint database at 500 dots per inch with 8 bits of grayscale resolution. At this rate, a single fingerprint card turns into about 10 MB of data!

Here's a sample fingerprint image measuring 768 x 768 pixels (= 589,824 bytes):

``Big deal," I hear you saying, ``I've got a gigabyte disk on my computer!"

Yes, but the FBI has been collecting fingerprint cards since 1924, and because (like most of us) they find it hard to throw things out, over the past 70 years their collection has grown to over 200 million cards occupying an acre of filing cabinets in the J. Edgar Hoover building back in Washington. (No; they don't have the entire U.S. population in their files, just lots of ``repeat customers.") This includes some 29 million records they examine each time they're asked to ``round up the usual suspects."

Your gigabyte drive starts to look pretty puny when faced with 2,000 terabytes' worth of images. And to make matters worse, fingerprint data continues to accumulate at a rate of 30,000-50,000 new cards PER DAY, which makes for a serious traffic jam on the Information Superhighway. (Go ahead; compute the time required to send a 10 MB card over the 9600 baud modem they're still using in Mayberry RFD. Hint: with a 20% communications overhead reducing the data rate to 7680 bits/second, it'll take just under 3 hours!)

``Okay," you say, ``so they need to use data compression. Better use a lossless method to preserve every pixel perfectly." Unfortunately, in practice lossless methods haven't done better than 2:1 on fingerprints. The FBI needs more than an order of magnitude reduction in bits to make a serious dent in this database. That implies lossy compression, which means we're going to have to tolerate some distortion in the compressed images.

Figura 27. Inglês 2

Roswell Case Summary

Is it true a UFO with aliens on board crashed and was recovered by the United States military?

New investigations, however,are uncovering startlingevidence indicating that a UFO may have crashed in the New Mexico desert in July 1947. A rancher named Mac Brazel discovered strange metal strewn across a wide area of range land he tended. Because of the material's unusual characteristics, Brazel took pieces of the debris to the authorities in Roswell, New Mexico. Intrigued by the debris, Colonel Blanchard, commanding officer at Roswell Army Air Field, ordered two intelligence officers to investigate. These two men were Major Jesse Marcel and Captain Sheridan Cavitt. Upon their report, Colonel Blanchard quietly ordered that the ranch area be cordoned off. Soldiers removed the debris, sending it to Army headquarters in Fort Worth, Texas.

At first the Army command at Roswell issued a press release announcing it had recovered a "flying disk," as UFOs were then called. This press release was retracted, and further press coverage restricted. At a press conference in Fort Worth, the Army explained that the intelligence officer and others at Roswell had misidentified the debris, which was, in fact, the remains of a downed balloon with a metallic radar reflector attached, and not a UFO. Public interest faded, and the Roswell event became a part of UFO folklore, with most ufologists accepting the official government version of the story.

It was not until the late 1970s, with Jesse Marcel's decision to comment publicly on the strange material and other aspects of the Roswell event, that the UFO crash story was revived. Since that time, new evidence indicates the weather balloon explanation was part of an elaborate government coverup, and in fact, the original report of a recovered flying disk was probably true. Investigations into the UFO crash story continue with the goal of pressuring the United States government to end the coverup and to reveal to the American public what actually crashed on the New Mexican desert that night in July 1947.

Rumors about the existence of secret alien bases have persisted for a long time. These bases are said to be located in various places, such as the moon, under the ocean, or in a tropical rain forest. A few extremists in ufology claim that aliens have already contacted government officials. There are even stories of alien installations within United States military bases. Some people have gone so far as to claim that they have worked on secret UFO projects for the government and seen UFOs at military installations. For the most part, individuals who have claimed knowledge of secret alien bases or secret UFO projects have proven to be unreliable witnesses; therefore, their statements must be considered cautiously until there is reliable and independent verification of their claims.

Whether the United States possesses alien technology is also open to question. If the Roswell UFO crash story is true, then the government has had some alien technology since 1947. Whether the United States has been able to understand and use that technology is another question. One researcher theorizes that the boomerang-shaped UFO seen in the mid-1980s around the Hudson Valley in New York was an American super-secret stealth aircraft based on UFO technology. Further investigation will be needed to prove this hypothesis.

Figura 28. Inglês 3

Featured resources

Special discount to IBM developerWorks Live!
IBM developerWorks Live! is IBM's premier technical event that debuted in 2002, and promises to again bring technical teams the year's most comprehensive and informative experience. If you or your team attend only one conference in 2003, make it IBM developerWorks Live!, 9-12 April in New Orleans. Faculty and students can attend for only US$795 (full conference price is US$1695). Just register with the promotional code "SPD1" to receive this special academic discount.

IBM developerWorks Live! will be filled with 350+ hours of technical exchanges, hands-on labs and birds-of-a-feather sessions, covering a dozen different product or technology areas. You can meet with product developers and experts on an informal one-on-one basis, participate in certification testing, visit the Solution Center with hundreds of demonstrations, and much more!

Eclipse Innovation Awards announced
Congratulations to the award recipients of the Eclipse Innovation Awards! Over US$1.2 million in awards were granted to 49 proposals.

e-business on demand
Technology has always accelerated the pace of change. New technologies enable new ways of doing business as markets shift, customer expectations evolve, and business models are redefined. Each major shift presents an opportunity. Institutions that understand and prepare for these changes can gain the advantage over competitors and lead their industries. IBM is defining and leading the next wave of change: e-business on demand. Read more.

Open technologies
Visit our open technologies resource center, where you'll find background information and resources to get a clear picture of today's software industry challenges and trends; access application development solutions with WebSphere and Eclipse; and take advantage of other computer science teaching resources such as developerWorks.

Resume database for students
If your students have been certified on IBM Business Partners in the U.S. for possible employment opportunities. Check it out!

Figura 29. Inglês 4

Skywatcher's Diary: March 2003

Saturday, March 1
If you've not glanced at the Big Dipper recently, take a look this evening after dark. The Great Bear has left his winter den and climbed halfway up the sky in the northeast. Notice how the middle of the Dipper is even with the North Star, which sits a dipper length to the left. Travel another dipper length farther left and you find yourself in the middle of Cassiopeia. The telltale "W" shape of this constellation is standing on end, opening to the right. It resembles the Greek capital letter sigma.
Sunday, March 2
The Moon is New at 9:35 p.m. EST this evening, so the next several nights offer the last opportunity to view in a moonless sky for a while. Of course, moonlight makes little difference if you must observe from a well-lit city environment. Assuming you can find a dark location, carefully examine the Orion Nebula with unaided eye, binoculars, and telescope. Look for subtle twists and knots of light and dark within the immense dust cloud out of which stars are forming. If you are unfamiliar with this object, first find Orion's Belt, the three equally spaced stars halfway up in the south in early evening. Look directly under the belt, about one "belt length" below, for a "fuzzy star." Zero in with binoculars
Monday, March 3
This evening you have a rare chance to catch a very young Moon, less than 24 hours old (past New). The hairline crescent sits 1 1/2 degrees (3 moon diameters) above the horizon between west and west-southwest 25 minutes after sunset. Start looking 10 minutes earlier. Use binoculars to slowly scan the area. Tomorrow night the Moon is much easier because the crescent is wider and the Moon sets in a darker sky. Look 45 minutes to an hour after sunset then.
Tuesday, March 4
As you gaze at the thin crescent tonight, imagine Muslims in the Middle East intently scanning the western horizon looking for that same Moon eight hours earlier. Their purpose was to establish the beginning of the New Year under the Islamic Hijrah calendar. That calendar is strictly lunar based, marking months by the first sighting of the crescent Moon after sunset. The Moon goes through its cycle of phases 12 times in only 354 days, about 11 days shorter than the familiar 365-1/4-day Gregorian calendar. The Muslim New Year, therefore, drifts earlier each year by 11 days.
Wednesday, March 5
Today the planet Mars passes between two of the more famous nebulas, or giant gas clouds. The brighter of the two is familiar to amateur astronomers as the Lagoon Nebula, named for its distinctive dark dust lane. It appears less than a degree to the lower right of Mars tomorrow morning. The Trifid Nebula sits to Mars' upper right less than a degree away. This nebula is often remembered because of the vivid blue and pink colors that appear in observatory photographs. Those hues are generally not visible when peering through the telescope. To verify Mars' position, go out when the sky is still dark, at least 1 1/2 hours before sunrise. Although binoculars will reveal the Lagoon, the Trifid requires a modest telescope.

Figura 30. Inglês 5

· Tecnologia aeroespacial

Começa a corrida ao Prêmio X, para naves espaciais de baixo custo

Em um aeroporto localizado em uma região remota do deserto do Mojave, Califórnia, no mês passado, uma pequena empresa do setor aerospacial exibiu uma espaçonave e um veículo de lançamento que vinha desenvolvendo em segredo há mais de dois anos. O aparelho provou ser o mais avançado dentre os que estão competindo pela conquista do alardeado Prêmio X, de US$ 10 milhões, para a primeira tripulação civil que chegar ao espaço. Mas se exageros na mídia movessem foguetes, o prêmio decerto já teria sido ganho, a essa altura. Embora os partidários dos vôos espaciais civis estejam absolutamente confiantes em que o prêmio será vencido por alguém em prazo de alguns meses, pode ser que a verdade não seja tão rósea.

Até mesmo uma análise apressada demonstra que muitos das dúzias de concorrentes têm pouco mais que projetos básicos como prova de seus esforços, e até mesmo as equipes que construíram alguma coisa teriam dificuldades para conseguir que seus aparelhos passassem por programas completos de testes até 2005, o prazo máximo oferecido pelo regulamento do prêmio, quanto mais que chegassem ao espaço.

E o fato de que as perspectivas de ganhar dinheiro com vôos espaciais civis tenham sido severamente prejudicadas pela perda do ônibus espacial Columbia e sua tripulação de sete astronautas tampouco ajuda muito. A verdadeira recompensa do Prêmio X será a chance de dominar o potencial mercado de turismo espacial, avaliado por alguns em US$ 1 bilhão ao ano. A competição nasceu da imaginação de Peter Diamandis, um empresário de St. Louis, no Missouri, que reconhece o papel que concursos e competições desempenharam no desenvolvimento de muitas tecnologias essenciais -entre as quais a aviação. O prêmio de US$ 25 mil oferecido pelo empresário de hotelaria norte-americano Raymond Orteig, por exemplo, para o primeiro vôo sem escalas entre Nova York e Paris inspirou nove tentativas, e terminou conquistado por Charles Lindbergh, em 1927. "Talvez o Prêmio X possa ter efeito semelhante quanto ao espaço", afirma Diamandis.

O Prêmio X será concedido à primeira equipe que lance uma tripulação de três pessoas ao espaço, em um vôo de pelo menos 100 quilômetros de altitude, e a traga à Terra em segurança. O feito precisa, a seguir, ser repetido dentro de duas semanas, e com a mesma espaçonave. E os 24 grupos que se inscreveram no concurso até agora tem prazo apenas até o dia 1° de janeiro de 2005 para fazê-lo.

À primeira vista, um vôo sub-orbital pode parecer tarefa bastante simples. Afinal, os engenheiros sabem o que é preciso para obtê-los há algumas décadas. Nos anos 60, a Nasa, agência espacial americana, executou dezenas deles com um aparelho experimental conhecido como X-15. Levado ao céu por um bombardeiro B-52, o X-15 era lançado a uma altitude de 13 mil metros, e a uma velocidade de 800 km/h. O aparelho era acionado por um motor-foguete que queimava todo o seu suprimento de combustível, uma mistura de amônia e oxigênio líquido, em cerca de 80 segundos. O X-15 podia atingir velocidades de Mach 6,7 em vôo horizontal, ou trocar essa velocidade por força ascensional, atingindo altitudes de até 100 quilômetros, com períodos de até cinco minutos de ausência de gravidade, para depois pousar em vôo planado.

O plano revelado no deserto pela Scaled Composites, de Mojave, apresenta grandes semelhanças com essa idéia. Um aparelho de vôo de altitude elevada conhecido como White Knight transportará um pequeno avião-foguete, apelidado SpaceShipOne, até uma altitude de 13 mil metros. Isso leva o foguete para além dos quilômetros de atmosfera mais densa, de modo que ele precise de menos combustível para seu vôo no espaço.

O projetista responsável, Burt Rutan, que dirige a Scaled Composites, está experimentando com diversas tecnologias novas em seus aparelhos. O motor-foguete do SpaceShipOne, por exemplo, depende de óxido nitroso líquido passando por um cilindro oco de borracha. O líquido é um poderoso oxidante que se combina com a borracha para queimar de maneira feroz, gerando empuxo. O sistema foi projetado para combinar a segurança de um motor-foguete de combustível líquido, que pode ser desligado rapidamente com o fechamento de uma válvula, e a simplicidade de um propulsor de combustível sólido, que requer poucas peças móveis. Esses motores híbridos não são capazes de se equiparar aos motores de oxigênio líquido e hidrogênio utilizados pela Nasa em termos de empuxo puro, mas isso não seria necessário, para vôos sub-orbitais.

Um dos problemas com os foguetes híbridos é que ninguém os usou no espaço até agora, diz Frank Macklin, engenheiro chefe da Spacéev, de Poway, Califórnia, uma das duas empresas que participam de uma concorrência para a construção dos motores-foguete especificados por Rutan. Embora mais simples do que muitos foguetes, os propulsores apresentam seus problemas. A maneira pela qual o óxido nitroso corre pela borracha pode gerar ondas de choque que causariam empuxo instável. A Spacéev e sua rival, a Environmental Aerosciences, de Miami, Flórida, realizaram ambas combustões de 30 segundos em seus motores-foguete, mas para um vôo sub-orbital o tempo de queima necessário seria de no mínimo 65 segundos.

Embora o aspecto aerodinâmico do SpaceShipOne tenha sido extensamente modelado em computador, Rutan ainda não executou nenhum teste em túnel de vento. Em lugar disso, ele espera validar o projeto em testes de vôo nos quais ele seria transportado pelo White Knight. Enquanto isso, os potenciais astronautas devem ganhar experiência em vôo subsônico com o White Knight, que terá controles dispostos de maneira semelhante à planejada para o avião-foguete.

Figura 31. Português 1

Alcântara

Acordo Brasil - EUA aprovado com todas as alterações pedidas pelos brasileiros. A despeito do jogo de cena do governo federal, os deputados em Brasília aprovaram uma versão modificada do Acordo de Salvaguardas Tecnológicas Brasil / Estados Unidos para uso da Base Espacial de Alcântara, como pediu a sociedade civil.
Embora ainda haja um ou outro ponto passível de questionamento, foi uma enorme vitória para o Brasil e uma derrota estratégica para o serviço de segurança dos Estados Unidos, que pretendia usar a base quase como uma embaixada. Uma fonte em Brasília disse que as alterações "irritaram enormemente os serviços de inteligência americanos" e que o governo dos Estados Unidos fará pressão para que as alterações não sejam mantidas.
A pressão da sociedade, por meio de cartas, e-mail, telefonemas e artigos na imprensa ajudaram os resultados, até aqui. É importante destacar que a base do governo também votou pelas modificações no acordo, mostrando que o discurso do Planalto junto aos Estados Unidos mudou rapidamente quando a pressão interna começou a classificar o caso Alcântara como ameaça a soberania nacional. As eleições estão próximas.
Naturalmente é importante que os americanos, e franceses, ingleses, alemães, japoneses, chineses, russos, usem a base. É para lançar foguetes que ela foi construída. Mas sob as nossas regras, não as deles. Os lançamentos espaciais brasileiros têm sido sabotados por muito tempo, de modo secreto. O acordo, como estava, era uma sabotagem aberta.
O Ministro de Ciência e Tecnologia divulgou uma nota a respeito e fez de conta que a aprovação foi prova de que estava tudo certo. Ele, politicamente, mostrou estar tudo sob controle, mas admitiu: "Com relação ao mérito das ressalvas, há evidentes divergências entre a Comissão e o Governo Federal, as quais continuarão sendo discutidas nas próximas instâncias no âmbito do Congresso." Naturalmente, ainda falta muito para que o acordo seja realmente aprovado, com estas ressalvas, ou reprovado. Pelo menos até a metade do ano 2002, nenhum lançador americano será levado para Alcântara. Mas em negociações deste tamanho, pressa é a maior das inimigas.
O que incomodou muito a setores do Governo dos Estados Unidos foi a reação de jornalistas e da oposição contra as cláusulas leoninas do contrato. Agora, caso seja aprovada pela Comissão de Ciência e Tecnologia e Constituição e Justiça, além do plenário, com as modificações incluídas no Relatório de Waldir Pires que "salvaguardam a soberania nacional", o novo texto terá de ser renegociado pelo Itamaraty com os Estados Unidos. Não é preciso dizer que isso não agradou em nada a Agência de Segurança Nacional (NSA), que tem planos de usar Alcântara para lançamento de um novo grupo de satélites espiões.
Agora, com o novo texto aprovado, e endossado pela Comissão, não haverá mais nenhuma restrição à aplicação dos recursos obtidos com o aluguel dos serviços de lançamento na base. Na versão anterior o Brasil era proibido de usar o dinheiro em seu próprio Programa Espacial. Os americanos também não poderão "autorizar" os brasileiros a circular na área de lançamento da base; essa função será monitorada pelo Brasil e pelos Estados Unidos em conjunto.

Figura 32. Português 2

Havaí usa traje casual todo dia da semana

No arquipélago mais isolado do mundo, a mais de 4.000 km do continente mais próximo, a natureza peculiar, a riqueza da cultura e o espírito relaxado do Havaí encantam escritores, poetas e exploradores há mais de 200 anos. Ali, quase esquecidos no meio do Pacífico, as rosas não têm espinhos, vulcões cospem fogo sem trégua e ondas gigantes se erguem do mar.

Alheios aos caprichos da natureza, os habitantes saúdam uns aos outros com seu simpático "aloha", palavra que, além de "bem-vindo", "oi" e "tchau", "paz" e "amor", significa todo o espírito das ilhas ao evocar as bênçãos e as dádivas divinas.

Mesmo na cosmopolita capital, Honolulu, na ilha de Oahu, somente marinheiros de primeira viagem saem às ruas com ternos ou camisas formais.

No ritmo do arquipélago, todo dia da semana é casual, e as vestes largas e confortáveis, em tecidos decorados com frutas tropicais, coqueiros e dançarinas de hula, são objeto de desejo do mais yuppie dos havaianos.

Eles colecionam tudo que surge no estilo das chamadas -não sem motivo- "aloha shirts", ou camisas "aloha".

Ao primeiro sinal do fim do expediente, todo mundo corre para a praia. Crianças, velhos, jovens, homens e mulheres se aglomeram na areia com a prancha debaixo do braço, deixando bem claro que o surfe é realmente o esporte nacional.

Iniciantes titubeiam meio sem jeito em águas quase paradas, e ídolos mundiais disputam espaço entre as ondas que chegam a 20 m de altura no inverno.

No interior das ilhas, trilhas isoladas percorrem enormes gargantas cavadas pela ação do vento, das chuvas e de gigantescos deslizamentos de terra que tornearam os contornos desse arquipélago vulcânico. Rios emoldurados por bananeiras e samambaias gigantes são alimentados por centenas de cachoeiras que brotam do alto das montanhas.

Resorts luxuosos exibem seus restaurantes finos e campos de golfe bem ao lado das plantações de abacaxi, das enormes fazendas de gado ou dos campos de taro, um tubérculo que, semelhante ao inhame, compõe a base da alimentação local.

Nas estradas, automóveis caindo aos pedaços, lotados de surfistas em busca da fama, dividem espaço com os esportivos conversíveis dos turistas.

Tal diversidade faz do Havaí um paraíso inesgotável. Nas ondas de Oahu, nos vulcões da ilha do Havaí, nas praias de Maui ou nos cânions de Kauai, o inesperado é a regra, e cada canto inexplorado, o início de uma nova aventura.

Figura 33. Português 3

Floresta absorve carbono, diz estudo

CLAUDIO ANGELO
Editor-assistente de Ciência da **Folha de S.Paulo**

Para quem ainda consegue ser otimista em relação ao clima do planeta, aí vai uma boa notícia: nas últimas duas décadas, as mudanças climáticas alimentaram o crescimento da floresta amazônica, que absorveu milhões de toneladas de carbono da atmosfera. A conclusão é de um estudo publicado hoje por um grupo dos EUA.

Analisando imagens de satélite produzidas entre 1982 e 1999, o grupo liderado por Ramakrishna Nemani, da Universidade de Montana, descobriu que a produtividade primária cresceu 6% no planeta inteiro nesse período. E a maior contribuição vem justamente da Amazônia: 42%.

"Produtividade primária" é um nome complicado para definir o crescimento das plantas. Com mais luz, água e calor, elas fazem mais fotossíntese, absorvendo carbono (na forma de gás carbônico, ou CO_2) e fixando-o na forma de folhas, caule e raízes.

Um aumento de 6% da produtividade primária do planeta significa que as plantas retiraram da atmosfera nada menos que 3,4 bilhões de toneladas de carbono. Só a Amazônia teria sido responsável pela absorção de 1,4 bilhão.

"O estudo mostra uma tendência, da qual já desconfiávamos, de que a Amazônia é um grande limpador da atmosfera", afirmou o biólogo Paulo Moutinho, do Ipam (Instituto de Pesquisa Ambiental da Amazônia). Ela estaria absorvendo CO_2 lançado no ar por atividades humanas, como a queima de petróleo e derivados.

As respostas da floresta ao aquecimento global e a quantidade de carbono que ela "sequestra" têm gerado debates acalorados entre os cientistas. Até recentemente, achava-se que o efeito estufa (a retenção do calor na atmosfera da Terra por uma capa de gases, como o CO_2) fosse estimular o crescimento das plantas só nas florestas frias e temperadas do hemisfério Norte. Nas quentes florestas tropicais, o aumento de temperatura não faria diferença.

Figura 34. Português 4

A Turquia na encruzilhada

A Turquia dos dias de hoje é apenas uma pequena sombra do outrora todo-poderoso Império Otomano dos séculos XVI e XVII, ocasião em que o estandarte do Crescente, comandado pelos sultões de Istambul, metia medo em toda a Europa. No presente, devastada pela recessão econômica e pressionada pelos Estados Unidos, ela se encontra na embaraçosa situação de ter que ceder o seu território para que dali a grande potência consuma a agressão final ao seu vizinho, o Iraque.

O Império dos Sultões

Disseram que o sultão Solimão, o Magnífico (reinou entre 1520-1566), temia mais as preces do papa do que os canhões dos cristãos. E tinha razão, pois foi Pio V quem, aliando-se a Felipe II da Espanha e a outras cidades italianas, depois de muitos anos de recuos no Mediterrâneo, mobilizou-os para uma cruzada contra os turcos. Cinco anos depois da morte do famoso sultão, a marinha ocidental comandada por João d´Áustria, um filho ilegítimo de Carlos V, cercou no dia 5 de outubro de 1571 a frota turca do almirante Ali Paxá em Lepanto, no golfo de Corinto, impondo-lhe uma memorável derrota, capturando uma boa parte das suas galeras. Batalha em que o grande Miguel de Cervantes perdeu a mão.

Até aquela época o mundo sultânico era impressionante. O Império Otomano, alargado pelas campanhas de Solimão, o Califa do Islã, sempre atrás de “novos inimigos , de novos súditos”, partia dos campos da Hungria, no meio da Europa, estendendo-se até o sul da península arábica, uns 7 mil quilômetros abaixo. Dimensão essa que permitiu a ele, o *Kanuni* (o fazer das leis) , como chamavam-no seus vassalos, graças aos tributos e saques, aproveitando-se da genialidade do arquiteto Sinan, lançar-se na construção de inúmeras mesquitas, tal como a Süleymaniye Camii, iniciada em 1550, que até hoje domina o perfil dos altos de Istambul, a antiga Constantinopla.

A decadência otomana

Dizem também que foi os dinares de ouro dele, circulando nos principados certos, quem garantiu a vitória do Protestantismo na metade Norte da Europa. A divisão da cristandade fora um claro objetivo da política externa dos turcos. Talvez fosse por isso que o papa Pio V, um líder da Contra-Reforma, resolvesse puni-los lançando a esquadra do bastardo imperial contra eles. Seja como for, a morte de Solimão, o Magnifico, em 1566, seguida do fiasco de Lepanto, em 1571, seriam as balizas que assinalaram a decadência dos otomanos. Colaborou ainda mais para tanto, uma fieira de sultões meio doidos que, com seu comportamento bizarro e má administração, debilitaram ainda mais a imagem do poder (o próprio sucessor de Solimão, o seu filho Selim I era chamado de Selim, “O Bêbado”)

A sensação do alívio que isso trouxe aos europeus é percebida na Marcha Turca, um rondó composto Mozart em 1790, no qual a outrora sonoridade marcial dos ferozes

Figura 35. Português 5

ANEXO B - CÓDIGO FONTE DOS PROGRAMAS

Apresentam-se a seguir o código fonte dos Programas Filtro, Normalização, Comparação de Caracteres e Comparação de Palavras. Os códigos foram implementados em linguagem C usando DevC++, que é um compilador C/C++ (ANSI).

1 - Programa Filtro

```
/* Programa Filtro */

/* Programa Filtro */

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <ctype.h>

#define TAM 5

FILE *arqFilename;
FILE *arqFilenameNew;
FILE *arqTexto[TAM];
FILE *arqTextoNew;

int contaCar(FILE *arq); //Funcao p/contar o total de caracteres por linha
int contaLin(FILE *arq); //Funcao p/contar o total de linha do arquivo
int Ordena(int lista[TAM]); //Bolha
int comparaLin(int totLin[TAM]);
char comparaCar(char car[TAM]), carGer;

int TAM2;
bool pulo;

main()
{
   int totCar[TAM], totLin[TAM];
   int i=0, j=0, k, p, freqLin;
   float mediaCar=0, mediaPal=0, mediaLin=0;
   char a, ant, ant2;
   char vetCar[TAM], NomeArquivo[60], NomeArquivo2[60];
   char LinhaGer[5][2000];
   bool fim;

   printf("**********************\n");
   printf("*** Programa Filtro ***\n");getche();

   if (TAM%2)
     TAM2=(TAM/2)+1;
   else
     TAM2=(TAM/2);


   arqFilename    = fopen ("arqFilenameF.txt","r");
   arqFilenameNew = fopen ("arqFilenameNew.txt","r");
   if ((!arqFilename) != (!arqFilenameNew))
   {
     printf("Erro na abertura dos Arquivos Auxiliares."); getche();
     exit(0);
   }

   for(k=0;k<10;k++) //PRINCIPAL - 10 ARQUIVOS
   {
     for(i=0;i<TAM;i++)
     {
       fscanf(arqFilename, "%s", NomeArquivo); //ler o conteudo do arquivo
       arqTexto[i] = fopen (NomeArquivo,"r");

       if (!arqTexto[i])
       {
```

```
      printf ("Erro na abertura do Arquivo Texto."); getche();
      exit(0);
    }
}

fscanf(arqFilenameNew, "%s", NomeArquivo2); //Le o nome dos novos arquivos
arqTextoNew = fopen (NomeArquivo2,"w");     //Cria o novo arquivo

printf("%s\n",NomeArquivo2);

if (!arqTextoNew)
{
   printf("Erro na abertura do Novo Arquivo %s",arqTextoNew); getche();
   exit(0);
}

for(i=0;i<TAM;i++)
{
   totCar[i] = contaCar(arqTexto[i]);     //Chama a Funcao Conta Caracteres
   totLin[i] = contaLin(arqTexto[i]);     //Chama a Funcao Conta Linhas
}

for(i=0;i<TAM;i++)
   fseek (arqTexto[i], 0L, SEEK_SET);     //Posiciona os arquivos no topo

freqLin = comparaLin(totLin);              //Chama a Funcao COMPARA LINHA

for(i=0;i<TAM;i++)
   fseek (arqTexto[i], 0L, SEEK_SET);     //Posiciona os arquivos no topo


for(j=1; j<=freqLin; j++)   //SECUNDARIO - LINHA-NOVOS ARQUIVOS
{
   a=' '; ant=' '; ant2=' ';
   for(i=0; i<TAM; i++)
   {
      if (totLin[i]==freqLin)
      {
         p=0;
         do      //Cria cada linha do arquivo Gerado
         {
            carGer = getc(arqTexto[i]);
            LinhaGer[i][p] = carGer;
            p++;
         }while(carGer!=10 && carGer!=13 && carGer!='\0' && carGer!=EOF);
      }
   }

   p=0;
   do
   {
      for(i=0; i<TAM; i++)
         vetCar[i]=LinhaGer[i][p]; //Monta o vetor de caracteres

      vetCar[i]='\0';
      p++;

      pulo=false;
      ant2=ant;
      ant=a;

      a = comparaCar(vetCar); //Chama a Funcao COMPARA CARACTERE

      if (a==vetCar[0])//Media e Origem / Origem
      {
         fprintf(arqTextoNew,"%c",a);  //Prioriza a Origem
      }
      //Excecoes: diferente da origem=Media
      else if (ant2=='.' && ant==',' && a==' ')
      {//Tratamento especifico p/Ing1
         fseek  (arqTextoNew, -2, SEEK_CUR);
         fprintf(arqTextoNew,",%c",a);
      }
      else if (a=='&')
      {//Tratamento p/o caractere especial (&)
         for(i=0; i<TAM; i++)
```

```
                if (vetCar[i]!='&')
                {
                  a = vetCar[i];  //Recebe o 1o caractere <> de &
                  fprintf(arqTextoNew,"%c",a);
                  break;
                }
            }
            else if (a=='!')
            {//Tratamento p/o caractere especial (!)
              for(i=0; i<TAM; i++)
                if (vetCar[i]!='!')
                {
                  a = vetCar[i];  //Recebe o 1o caractere <> de !
                  fprintf(arqTextoNew,"%c",a);
                  break;
                }
            }
            else if (ant2=='.' && ant==' ' && toupper(a)==a)
            {
              fprintf(arqTextoNew,"%c",a);
            }
            else if (ant2=='.' && ant==' ' && toupper(a)!=a)
            {
              fseek  (arqTextoNew, -2, SEEK_CUR);
              fprintf(arqTextoNew,",%c%c",ant,a);
            }
            else if (ant2==',' && ant==' ' && toupper(a)==a)
            {
              fseek  (arqTextoNew, -2, SEEK_CUR);
              fprintf(arqTextoNew,".%c%c",ant,a);
            }
            else if (ant2==',' && ant==' ' && toupper(a)!=a)
            {
              fprintf(arqTextoNew,"%c",a);
            }
            else
            {
              fprintf(arqTextoNew,"%c",a);
            }

          } while(a!=10 && a!=13 && a!='\0' && a!=EOF);
        }//For (fim arquivo)

        for(i=0;i<TAM;i++)
          fclose(arqTexto[i]);

        fclose(arqTextoNew);
      }//For (fim lista de arquivo)

      printf("\n*** Fim do Programa ***\n");getche();

      return(0);
}


/* F U N C O E S */
int contaCar(FILE *arq)   //Conta Caractere
{
  int totCaract=0;
  char caract;

  fseek (arq, 0L, SEEK_SET);
  caract = getc(arq);               // Le o primeiro caracter
  while (!feof(arq))    // Enquanto não se chegar no final do arquivo
  {
     totCaract++;
     caract = getc(arq);
  }
  return(totCaract);
}

int contaLin(FILE *arq) //Conta Linha
{
  int totLinha=0;
  char a;
```

```
   fseek (arq, 0L, SEEK_SET);
   a = getc(arq);                    /* Le o primeiro caracter */
   while (!feof(arq))    /* Enquanto não se chegar no final do arquivo */
   {
      if (a==13 || a==10) totLinha++;
      a = getc(arq);
      if (a==EOF) totLinha++;
   }
   return(totLinha);
}

int comparaLin(int totLin[TAM])
{
   int i, j, freq=0, tot[TAM];
   bool pulo=false;

   for(i=0;i<TAM;i++) tot[i]=0;

   for(i=0;i<TAM;i++)
   {
      for(j=0;j<TAM;j++)
         if (totLin[i]==totLin[j]) tot[i]++;

      if (tot[i] >= TAM2)
      {
         freq = totLin[i];
         pulo=true;
         break;
      }
   }

   if (!pulo)
      freq = totLin[Ordena(tot)]; //Devolve o maior;

   return(freq);
}

char comparaCar(char vetComp[TAM])
{
   int i, j, tot[TAM];
   char carac;

   for(i=0; i<TAM; i++) tot[i]=0;

   for(i=0; i<TAM; i++)
   {
      for(j=0; j<TAM; j++)
         if ( (vetComp[i]!='\0') && (vetComp[i]==vetComp[j]) ) tot[i]++;

      if (tot[i] >= TAM2)
      {
         carac = vetComp[i]; //Recebe o caractere de > frequencia
         pulo=true;
         break;
      }
   }

   if (!pulo) carac=vetComp[0]; //Recebe o caractere do arqOrigem

   return(carac);
}

int Ordena(int lista[TAM]) //Bolha
{
   int i, aux;
   bool troca=true;

   while(troca)
   {
      troca = false;

      for(i=0; i<TAM; i++)
      {
         if (lista[i] > lista[i+1])
         {
            aux = lista[i+1];
```

```
            lista[i+1] = lista[i];
            lista[i] = aux;
            troca = true;
          }
        }
      }

    return(lista[TAM-1]);
}
```

2 - Programa Normaliza

```
/* Programa Normaliza Arquivos */

#include <stdio.h>
#include <conio.h>
#include <string.h>

FILE *arqOri;                   //Arquivo Origem
FILE *arqNor;                   //Arquivo Normalizado
FILE *arqFilename;      //Arquivo c/os nomes dos arquivos de Origem e Destino a serem testados

int contaLin(char tipo); //Funcao p/contar o total de linhas por arquivo

int main()
{
   int k, totLinO, totCarLinO, lin, numArq=1;
   char caractOri, LinhaOri[2000];
   char NomeArquivo[20], NomeArquivoOld[20];
   bool fim;


   arqFilename = fopen ("ArqFilename.txt","r");
   if (!arqFilename)
   {
     printf("Erro na abertura do Arquivo."); getche();
     exit(0);
   }


   /*================================*/
   /*  L O O P   P R I N C I P A L  */

   printf("*** Programa Normaliza Arquivos ***\n");
   printf("***   Inicio da Normalizacao   ***\n\n\n");getche();

   while (!feof(arqFilename))   //Passa por todos os arquivos que estao no ArqFilename
   {
      fscanf(arqFilename, "%s", NomeArquivoOld); //Le o nome do arquivo Origem
      arqOri = fopen (NomeArquivoOld,"r");
      if (!arqOri)
      {
        printf ("Erro na abertura do Arquivo."); getche();
        exit(0);
      }

      fseek (arqOri, 0L, SEEK_SET);
      totLinO = contaLin('O');     //Chama a Funcao Conta Caracteres
      fseek (arqOri, 0L, SEEK_SET);

      lin = 1;
      printf("Arquivo %d : %s", numArq, NomeArquivoOld);

      NomeArquivo[0]='n'; //Alteracao do nome do novo arquivo normalizado
      NomeArquivo[1]='\0';
      strcat(NomeArquivo,NomeArquivoOld);
      arqNor = fopen (NomeArquivo,"w"); //Cria o arquivo p/normalizacao
      if (!arqNor)
      {
        printf ("Erro na abertura do Arquivo Normalizado."); getche();
        exit(0);
      }

      printf(" - %s\n", NomeArquivo);
```

```
      /*================================*/
      /* L O O P   S E C U N D A R I O */

      while (lin<=totLinO)  //Passa por todas as linhas do arquivo Origem e cria o normalizado
      {
         fim = false;
         k = 0;

         while (!fim)      //Cria cada linha do arquivo Origem
         {
            caractOri = getc(arqOri);
            if (caractOri==10 || caractOri==13 || caractOri=='\0' || caractOri==EOF) fim=true;
            else
            {
               LinhaOri[k] = caractOri;
               k++;
            }
         }
         LinhaOri[k] = caractOri;
         totCarLinO = k;
         k = 0;

         while (k<=totCarLinO)   //Passa por todos os caracteres da Origem
         {
            if ( (totCarLinO==1) && (LinhaOri[k]=='+' || LinhaOri[k]=='-') )
            {
               fim = true;
               k++;
            }
            else if (totCarLinO==0 || LinhaOri[k]==EOF) fim = true;
            else if ( (LinhaOri[k]==10 || LinhaOri[k]==13 || LinhaOri[k]=='\0') && (totCarLinO!=1) )
            {
               putc(LinhaOri[k],arqNor);
               fim = true;

            }
            else if (  !( (LinhaOri[k-1]==' ') && (LinhaOri[k]==' ')) && !( LinhaOri[k]==' ' && k==0 )) putc(LinhaOri[k],arqNor);

            k++;
         }
         lin++;

      }  //Fim While Linha
      numArq++;

   } //Fim While Arquivo

   printf("\n\n***    Fim  da  Normalizacao ***"); getche();

   fclose(arqOri);
   fclose(arqNor);
   fclose(arqFilename);
   return(0);
}


/* F U N C O E S */
int contaLin(char tipo) //Conta Linha
{
   int totLinha=0;
   char a;

   if (tipo=='O')
   {
      a = getc(arqOri);                          /* Le o primeiro caracter */
      while (!feof(arqOri))   /* Enquanto não se chegar no final do arquivo */
      {
         if (a==13 || a==10) totLinha++;
         a = getc(arqOri);
         if (a==EOF) totLinha++;
      }
   }
   return(totLinha);
}
```

3 - Compara Caracteres

```
/* Programa Compara Caracteres */

#include <stdio.h>
#include <conio.h>

FILE *arqOri;                          //Arquivo Origem
FILE *arqGer;          //Arquivo Gerado
FILE *arqFilename;      //Arquivo c/os nomes dos arquivos de Origem e Destino a serem testados
FILE *arqCarPercAcerto; //Arquivo com os valores e percentuais de Acertos por Caractere
FILE *arqCarPercErro;   //Arquivo com os valores e percentuais de Erros por Caractere
FILE *arqCarDescErro;   //Arquivo com a descricao dos Erros por Caractere
FILE *arqCarBranco;     //Arquivo com a descricao dos Erros por Caractere

int contaCar(FILE *arq); //Funcao p/contar o total de caracteres por linha
int contaPal(FILE *arq); //Funcao p/contar o total de palavras por linha
int contaLin(FILE *arq); //Funcao p/contar o total de linha do arquivo

main()
{
   unsigned int erroTotal, erroTotalG=0;
   int totCarO, totCarG, totPalO, totPalG, totLinO, totLinG;
   int totCarLinO, totCarLinG, totBranco;
   int erroIncG=0, erroExcG=0, erroIncLinG=0, erroExcLinG=0, erroOutroG=0;
   int erroTrocaG=0, erroTroca_1por2G=0, erroTroca_2por1G=0;
   int erroJuncaoPal_SemPerdaG=0, erroJuncaoPal_ComPerdaG=0;
   int erroInc, erroExc, erroIncLin, erroExcLin, erroOutro;
   int erroTroca, erroTroca_1por2, erroTroca_2por1;
   int erroJuncaoPal_SemPerda, erroJuncaoPal_ComPerda;
   int totAcertoPal, tamW, tamZ, numArq, totArqOri, totArqGer;
   int i, j, k, w, z, n, lin;

   float pErroTotalG=0, pErroIncG=0, pErroExcG=0, pErroIncLinG=0, pErroExcLinG=0, pErroOutroG=0;
   float pErroTrocaG=0, pErroTroca_1por2G=0, pErroTroca_2por1G=0;
   float pErroJuncaoPal_SemPerdaG=0, pErroJuncaoPal_ComPerdaG=0;
   float pErroTotal, pErroInc, pErroExc, pErroIncLin, pErroExcLin, pErroOutro;
   float pErroTroca, pErroTroca_1por2, pErroTroca_2por1;
   float pErroJuncaoPal_SemPerda, pErroJuncaoPal_ComPerda;
   float pAcertoTotal;

   char caractOri, caractGer;
   char LinhaOri[2000], LinhaGer[2000];
   char palavraOri[50], palavraGer[50];
   char NomeArquivo[50], resp;

   bool fim, pulo, pulo2, passou;

   printf("*** Programa Compara Caracteres ***\n");
   printf("Tipo Normal(1) ou Filtro(2) ?\n");scanf("%c",&resp);
   printf("Quantos Arquivos de Origem ?\n");scanf("%d",&totArqOri);

   if (resp=='2') //Filtro
   {
     arqFilename      = fopen ("ArqFilenameFN.txt","r");
     arqCarPercAcerto = fopen ("ArqFCarPercAcerto.txt","w");
     arqCarPercErro   = fopen ("ArqFCarPercErro.txt","w");
     arqCarDescErro   = fopen ("ArqFCarDescErro.txt","w");
     arqCarBranco     = fopen ("ArqFCarBranco.txt","w");
   }
   else          //OCR Comercial
   {
     arqFilename      = fopen ("ArqFilenameNorm.txt","r");
     arqCarPercAcerto = fopen ("ArqCarPercAcerto.txt","w");
     arqCarPercErro   = fopen ("ArqCarPercErro.txt","w");
     arqCarDescErro   = fopen ("ArqCarDescErro.txt","w");
     arqCarBranco     = fopen ("ArqCarBranco.txt","w");
   }

   if ( (!arqFilename) != (!arqCarPercAcerto) != (!arqCarPercErro) != (!arqCarDescErro) )
   {
     printf("Erro na abertura dos Arquivos Auxiliares."); getche();
     exit(0);
   }
```

```
totArqGer = (contaLin(arqFilename) / totArqOri); //Chama Conta Linha p/calcular total de arquivos por grupo
fseek (arqFilename, 0L, SEEK_SET);


/*================================*/
/* L O O P  P R I N C I P A L */
for(n=1; n<=totArqOri; n++) //Grupos de Arquivos
{
  fscanf(arqFilename, "%s", NomeArquivo); //Le o nome do arquivo Origem
  arqOri = fopen (NomeArquivo,"r");

  if (!arqOri)
  {
    printf ("Erro na abertura do Arquivo Origem."); getche();
    exit(0);
  }

  fprintf(arqCarPercErro,  "*** Arquivo Origem: %s ***\n\n",NomeArquivo);
  fprintf(arqCarDescErro,  "*** Arquivo Origem: %s ***\n",NomeArquivo);
  fprintf(arqCarBranco,    "*** Arquivo Origem: %s ***\n\n",NomeArquivo);
  fprintf(arqCarPercAcerto,"\tOrigem: %s\n",NomeArquivo);

  totCarO = contaCar(arqOri);     //Chama a Funcao Conta Caracteres
  fprintf(arqCarPercErro,"Total de caracteres Origem: %d\n", totCarO);
  totPalO = contaPal(arqOri);     //Chama a Funcao Conta Palavras
  fprintf(arqCarPercErro,"Total de Palavras Origem  : %d\n", totPalO);
  totLinO = contaLin(arqOri);     //Chama a Funcao Conta Linhas
  fprintf(arqCarPercErro,"Total de Linhas Origem    : %d\n", totLinO);


  /*================================*/
  /* L O O P  S E C U N D A R I O */
  for(numArq=1; numArq<totArqGer; numArq++)   //Passa por todos os arquivos que estao no ArqFilename
  {
    fscanf(arqFilename, "%s", NomeArquivo); //Le o nome do arquivo Gerado
    arqGer = fopen (NomeArquivo,"r");

    if (!arqGer)
    {
      printf ("Erro na abertura do Arquivo Gerado."); getche();
      exit(0);
    }

    printf("\nArqGer %d: %s", numArq, NomeArquivo); //Imprime na tela os arquivos comparados

    fprintf(arqCarPercErro,  "\nArquivo Gerado: %s\n",NomeArquivo);
    fprintf(arqCarDescErro,  "\nArquivo Gerado: %s\n",NomeArquivo);
    fprintf(arqCarBranco,    "\nArquivo Gerado: %s\n",NomeArquivo);

    totCarG = contaCar(arqGer);
    fprintf(arqCarPercErro,"Total de Caracteres Gerado: %d\n", totCarG);
    totPalG = contaPal(arqGer);
    fprintf(arqCarPercErro,"Total de Palavras Gerado  : %d\n", totPalG);
    totLinG = contaLin(arqGer);
    fprintf(arqCarPercErro,"Total de Linhas Gerado    : %d\n\n", totLinG);

    fseek (arqOri, 0L, SEEK_SET); //Posiciona os arquivos no topo
    fseek (arqGer, 0L, SEEK_SET);

    lin = 1;
    erroTotal=0; erroInc=0; erroExc=0; erroIncLin=0; erroExcLin=0; erroOutro=0;
    erroTroca=0; erroTroca_1por2=0; erroTroca_2por1=0;
    erroJuncaoPal_SemPerda=0; erroJuncaoPal_ComPerda=0;
    pErroTotal=0; pErroInc=0; pErroExc=0; pErroIncLin=0; pErroExcLin=0; pErroOutro=0;
    pErroTroca=0; pErroTroca_1por2=0; pErroTroca_2por1=0;
    pErroJuncaoPal_SemPerda=0; pErroJuncaoPal_ComPerda=0;
    pAcertoTotal=0;

    if (totLinG==0)     //Os erros sao todas as linhas nao geradas
    {                   //e nao entra no proximo LOOP
      erroExcLin=totLinO;
      erroTotal=totLinO;
    }
```

```
/*================================*/
/* L O O P   T E R C I A R I O */

while ( (lin<=totLinO) && (totLinG>0) )  //Passa por todas as linhas do arquivo Origem
{
   pulo = false;
   pulo2 = false;
   fim = false;
   k=0;

   while (!fim)      //Cria cada linha do arquivo Gerado
   {
      caractGer = getc(arqGer);
      if (caractGer==13 || caractGer==10 || caractGer=='\0' || caractGer==EOF) fim = true;
      else
      {
        LinhaGer[k] = caractGer;
        k++;
      }
   }
   totCarLinG = k;

   if ( (LinhaGer[0]=='+') )
   {
     fprintf(arqCarDescErro,"Inclusao Linha     : %c - %d", LinhaGer[0],totCarLinG);
     fprintf(arqCarDescErro,"\t\tLinha: %d\n", lin);
     erroIncLin++;
     erroTotal++;
     erroInc   = erroInc   + totCarLinG;
     erroTotal = erroTotal + totCarLinG;
     pulo=true;        //Nao entra no proximo loop
     pulo2=true;       //Forca Origem a permanecer na mesma linha
   }
   else if ( (totCarLinG==1) && (LinhaGer[k-1]=='-') )
   {
     fprintf(arqCarDescErro,"Exclusao Linha     : %c", LinhaGer[k-1]);
     fprintf(arqCarDescErro,"\tLinha: %d\n", lin);
     erroExcLin++;
     erroTotal++;
     pulo=true;        //Nao entra no proximo loop
   }

   fim = false;
   k=0;

   while ((!fim) && (!pulo2))      //Cria cada linha do arquivo Origem
   {
      caractOri = getc(arqOri);
      if (caractOri==13 || caractOri==10 || caractOri=='\0' || caractOri==EOF) fim = true;
      else
      {
        LinhaOri[k] = caractOri;
        k++;
      }
   }
   totCarLinO = k;

   if ((pulo) && (LinhaGer[k-1]=='-'))     //Trata exclusao de linha
   {
     erroExc   = erroExc   + totCarLinO;
     erroTotal = erroTotal + totCarLinO;
   }

   if (!pulo)
   {
     i=0;
     j=0;
   }

   /*==================================*/
   /* L O O P   Q U A T E R N A R I O */
   while( (i<totCarLinO) && (!pulo) )  //Passa por todos os caracteres da Origem
   {
      if (LinhaOri[i]!=LinhaGer[j]) //TRATAMENTO DOS ERROS
      {
        if ( (LinhaOri[i]==' ') && (i+1>=totCarLinO) && (totCarLinO>=totCarLinG) )
```

```
            {//03-Origem tem um caractere branco no final da linha, o qual nao eh considerado erro
              fprintf(arqCarBranco,"Caractere branco   : ");
              fprintf(arqCarBranco,"\t\tLinha: %d \tColunaO: %d \tColunaG: %d\n", lin,i,j);
            }
            else if ( (LinhaOri[i]==' ') && (i+1==totCarLinO) && (totCarLinO<totCarLinG) )
            {//04-Compara o ultimo caractere Origem e verifica q Gerado tem varios caracteres
              fprintf(arqCarDescErro,"Inclusao Caracteres: %d",totCarLinG-i);
              fprintf(arqCarDescErro,"\t\tLinha: %d \tColunaO: %d \tColunaG: %d\n", lin,i,j);
              erroInc = erroInc + totCarLinG - i;
              erroTotal = erroTotal + totCarLinG - i;
            }
            else if (LinhaOri[i+1]==LinhaGer[j+1] && LinhaOri[i+2]==LinhaGer[j+2])
            {//05-Troca simples de caracteres
              fprintf(arqCarDescErro,"Troca            : %c - %c", LinhaOri[i],LinhaGer[j]);
              fprintf(arqCarDescErro,"\tLinha: %d \tColunaO: %d \tColunaG: %d\n", lin,i,j);
              erroTroca++;                //a b c
              erroTotal++;                //d b c
            }
            else if ( (LinhaOri[i+1]!=LinhaGer[j+1]) && (LinhaOri[i+2]==LinhaGer[j+2]) )
            {//06-Troca 2 caracteres seguidos
              fprintf(arqCarDescErro,"Troca 2x         : %c%c - %c%c", LinhaOri[i],LinhaOri[i+1],LinhaGer[j],LinhaGer[j+1]);
              fprintf(arqCarDescErro,"\tLinha: %d \tColunaO: %d \tColunaG: %d\n", lin,i,j);
              erroTroca=erroTroca+2;       //a b c
              erroTotal=erroTotal+2;       //d e c
              i++;
              j++;
            }
            else if ( (LinhaOri[i]!=LinhaGer[j+1]) && (LinhaOri[i+1]!=LinhaGer[j+1]) && (LinhaOri[i+1]==LinhaGer[j+2]) &&
(LinhaOri[i+2]==LinhaGer[j+3]) )
            {//07-Troca 1 caractere por 2. Exemplo: coMpara = coRNpara
              fprintf(arqCarDescErro,"Troca 12         : %c - %c%c", LinhaOri[i],LinhaGer[j],LinhaGer[j+1]);
              fprintf(arqCarDescErro,"\tLinha: %d \tColunaO: %d \tColunaG: %d\n", lin,i,j);
              erroTroca_1por2++;           //a b c
              erroTotal++;                //d e b c
              j++;
            }
            else if ( (LinhaOri[i]!=LinhaGer[j+1]) && (LinhaOri[i+1]!=LinhaGer[j]) && (LinhaOri[i+2]==LinhaGer[j+1]) &&
(LinhaOri[i+3]==LinhaGer[j+2]) )
            {//08-Troca 2 caracteres por 1. Exemplo: coRNpara = coMpara
              fprintf(arqCarDescErro,"Troca 21          : %c%c - %c", LinhaOri[i],LinhaOri[i+1],LinhaGer[j]);
              fprintf(arqCarDescErro,"\tLinha: %d \tColunaO: %d \tColunaG: %d\n", lin,i,j);
              erroTroca_2por1++;           //a b c d
              erroTotal++;                //g c d
              i++;
            }
            else if ( (LinhaOri[i]==' ') && (LinhaOri[i+1]==LinhaGer[j]) )
            {//09-Juncao de Palavra Sem Perda
              fprintf(arqCarDescErro,"Juncao Pal S/Perda : %c%c - %c", LinhaOri[i],LinhaOri[i+1],LinhaGer[j]);
              fprintf(arqCarDescErro,"\tLinha: %d \tColunaO: %d \tColunaG: %d\n", lin,i,j);
              erroJuncaoPal_SemPerda++;     //_ a
              erroTotal++;                //a
              i++;
            }
            else if ( (LinhaOri[i]==' ') && (LinhaOri[i+1]!=LinhaGer[j]) && (LinhaOri[i+2]==LinhaGer[j]) )
            {//10-Juncao de Palavra Com Perda
              fprintf(arqCarDescErro,"Juncao Pal C/Perda : %c%c - %c", LinhaOri[i],LinhaOri[i+1],LinhaGer[j]);
              fprintf(arqCarDescErro,"\tLinha: %d \tColunaO: %d \tColunaG: %d\n", lin,i,j);
              erroJuncaoPal_ComPerda++;     //_ a b
              erroTotal++;                //b
              i=i+2;
            }
            else if ( (LinhaGer[j]!=' ') && (LinhaGer[j+1]==' ') && (LinhaOri[i]==LinhaGer[j+2]) )
            {//11-Inclusao 2x (branco + ruido)
              fprintf(arqCarDescErro,"Inclusao 2x       : %c - %c%c%c", LinhaOri[i],LinhaGer[j],LinhaGer[j+1],LinhaGer[j+2]);
              fprintf(arqCarDescErro,"\t\tLinha: %d \tColunaO: %d \tColunaG: %d\n", lin,i,j);
              erroInc=erroInc+2;           //c
              erroTotal=erroTotal+2;       //a _ c
              j=j+2;
            }
            else if ( (LinhaOri[i+1]==LinhaGer[j]) && (LinhaOri[i+2]==LinhaGer[j+1]) )
            {//12-Exclusao de caractere
              fprintf(arqCarDescErro,"Exclusao          : %c", LinhaOri[i]);
              fprintf(arqCarDescErro,"\t\tLinha: %d \tColunaO: %d\n", lin,i);
              erroExc++;                  //a b c
              erroTotal++;                //b c
              i++;
```

```
            }
            else if (LinhaOri[i]==LinhaGer[j+1])
            {//13-Inclusao de caractere
              fprintf(arqCarDescErro,"Inclusao          : %c - %c%c", LinhaOri[i],LinhaGer[j],LinhaGer[j+1]);
              fprintf(arqCarDescErro,"\tLinha: %d \tColunaO: %d \tColunaG: %d\n", lin,i,j);
              erroInc++;                    //a
              erroTotal++;                   //x a
              j++;
            }
            else if ( (LinhaOri[i+1]!=LinhaGer[j]) && (LinhaOri[i+2]==LinhaGer[j]) && (LinhaOri[i+3]==LinhaGer[j+1]) )
            {//14-Exclusao de caractere
              fprintf(arqCarDescErro,"Exclusao 2x       : %c%c", LinhaOri[i], LinhaOri[i+1]);
              fprintf(arqCarDescErro,"\t\tLinha: %d \tColunaO: %d\n", lin,i);
              erroExc=erroExc+2;            //a b c d
              erroTotal=erroTotal+2;        //c d
              i=i+2;
            }
            else if (LinhaOri[i]!=LinhaGer[j+1] && LinhaOri[i+1]!=LinhaGer[j] )
            {//15-Exclusao, Inclusao e Troca
              tamZ=i;                       //a b
              tamW=j;                       //d e
              totAcertoPal=0;
              for(z=0;z<50;z++) palavraOri[z]=' ';
              for(z=0;z<50;z++) palavraGer[z]=' ';
              z=0;
              w=0;

              while (LinhaOri[i]!=' ' && LinhaOri[i]!=10 && LinhaOri[i]!=13 && LinhaOri[i]!='\0')
              {//Cria a palavra Origem
                 palavraOri[z]=LinhaOri[i];
                 i++;
                 z++;
              }
              palavraOri[z]='\0';

              while (LinhaGer[j]!=' ' && LinhaGer[j]!=10 && LinhaGer[j]!=13 && LinhaGer[j]!='\0')
              {//Cria a palavra Gerada
                 palavraGer[w]=LinhaGer[j];
                 j++;
                 w++;
              }
              palavraGer[w]='\0';

              do
              {
                 if (palavraOri[z]==palavraGer[w]) totAcertoPal++;
                 z--;
                 w--;
                 passou=true;
              }while(z>=0 && w>=0);

              if(passou) totAcertoPal--;

              tamZ=i-tamZ;
              tamW=j-tamW;

              if (tamZ > tamW)  //Exclusao - Troca
              {
                erroExc=erroExc + (tamZ-tamW);
                erroTotal=erroTotal + (tamZ-tamW);
                erroTroca=erroTroca + (tamZ - totAcertoPal - (tamZ - tamW));
                erroTotal=erroTotal + (tamZ - totAcertoPal - (tamZ - tamW));
                fprintf(arqCarDescErro,"Exclusao e Troca   : %d e %d %s - %s\n", (tamZ-tamW), (tamZ - totAcertoPal - (tamZ -
tamW)),palavraOri, palavraGer);
                fprintf(arqCarDescErro,"\tLinha: %d\n", lin);
              }
              if (tamZ < tamW)  //Inclusao - Troca
              {
                erroInc=erroInc + (tamW-tamZ);
                erroTotal=erroTotal + (tamW-tamZ);
                erroTroca=erroTroca + (tamW - totAcertoPal - (tamW - tamZ));
                erroTotal=erroTotal + (tamW - totAcertoPal - (tamW - tamZ));
                fprintf(arqCarDescErro,"Inclusao e Troca   : %d e %d %s - %s\n", (tamW-tamZ), (tamW - totAcertoPal - (tamW
- tamZ)),palavraOri, palavraGer);
                fprintf(arqCarDescErro,"\tLinha: %d\n", lin);
              }
```

```
                if (tamZ==tamW)   //Troca
                {
                  erroTroca=erroTroca + tamZ;
                  erroTotal=erroTotal + tamZ;
                  fprintf(arqCarDescErro,"Troca             : %d %s - %s",tamZ,palavraOri, palavraGer);
                  fprintf(arqCarDescErro,"\tLinha: %d\n", lin);
                }
              }//fim if-15
              else if (LinhaOri[i+1]==LinhaGer[j] && LinhaOri[i+1]==LinhaGer[j+1])
              {//16-Troca
                fprintf(arqCarDescErro,"Troca             : %c%c", LinhaOri[i], LinhaGer[j]);
                fprintf(arqCarDescErro,"\t\tLinha: %d \tColunaO: %d\n", lin,i);
                erroTroca++;            //d a
                erroTotal++;            //a a
              }
              else
              {//17-Outros Casos
                fprintf(arqCarDescErro,"Troca Outros      : %c%c - %c%c", LinhaOri[i],LinhaOri[i+1],LinhaGer[j],LinhaGer[j+1]);
                fprintf(arqCarDescErro,"\tLinha: %d # \tColunaO: %d # \tColunaG: %d\n", lin,i,j);
                erroTroca++;
                erroTotal++;
              }
            } //Fim IF Tratamento de Erro
            i++;
            j++;

          } //Fim While Quaternario - Erros
          lin++;

        }//Fim While Terciario - Linha

        //CALCULO POR ARQUIVO
        if (erroTotal>0)
        {
          pErroInc = float(erroInc)/float(erroTotal)*100;
          pErroExc = float(erroExc)/float(erroTotal)*100;
          pErroTroca = float(erroTroca)/float(erroTotal)*100;
          pErroTroca_1por2 = float(erroTroca_1por2)/float(erroTotal)*100;
          pErroTroca_2por1 = float(erroTroca_2por1)/float(erroTotal)*100;
          pErroJuncaoPal_SemPerda = float(erroJuncaoPal_SemPerda)/float(erroTotal)*100;
          pErroJuncaoPal_ComPerda = float(erroJuncaoPal_ComPerda)/float(erroTotal)*100;
          pErroIncLin = float(erroIncLin)/float(erroTotal)*100;
          pErroExcLin = float(erroExcLin)/float(erroTotal)*100;
          pErroOutro = float(erroOutro)/float(erroTotal)*100;
          pErroTotal = float(erroTotal)/float(erroTotal)*100;
        }

        if (erroTotal>totCarO) pAcertoTotal=0;
        else if (totCarG>0) pAcertoTotal = 100 - ( (float(erroTotal)*100) / float(totCarO) ) ;

        fprintf(arqCarPercErro,  "Erros de Inclusao        : %d  %5.2f \n",erroInc, pErroInc);
        fprintf(arqCarPercErro,  "Erros de Exclusao        : %d  %5.2f \n",erroExc, pErroExc);
        fprintf(arqCarPercErro,  "Erros de Troca           : %d  %5.2f \n",erroTroca, pErroTroca);
        fprintf(arqCarPercErro,  "Erros de Troca 1 por 2   : %d  %5.2f \n",erroTroca_1por2, pErroTroca_1por2);
        fprintf(arqCarPercErro,  "Erros de Troca 2 por 1   : %d  %5.2f \n",erroTroca_2por1, pErroTroca_2por1);
        fprintf(arqCarPercErro,        "Erros de Juncao Sem Perda : %d        %5.2f    \n",erroJuncaoPal_SemPerda,
pErroJuncaoPal_SemPerda);
        fprintf(arqCarPercErro,        "Erros de Juncao Com Perda : %d        %5.2f    \n",erroJuncaoPal_ComPerda,
pErroJuncaoPal_ComPerda);
        fprintf(arqCarPercErro,  "Erros de Inclusao Linha  : %d  %5.2f \n",erroIncLin, pErroIncLin);
        fprintf(arqCarPercErro,  "Erros de Exclusao Linha  : %d  %5.2f \n",erroExcLin, pErroExcLin);
        fprintf(arqCarPercErro,  "Outros Erros             : %d  %5.2f \n",erroOutro, pErroOutro);
        fprintf(arqCarPercErro,  "Total de Erros           : %d  %5.2f \n",erroTotal, pErroTotal);
        fprintf(arqCarPercErro,  "*************************************************************\n");
        fprintf(arqCarPercAcerto,"%5.2f \tGerado: %s\n", pAcertoTotal, NomeArquivo);

        erroIncG = erroIncG + erroInc;
        erroExcG = erroExcG + erroExc;
        erroIncLinG = erroIncLinG + erroIncLin;
        erroExcLinG = erroExcLinG + erroExcLin;
        erroOutroG = erroOutroG + erroOutro;
        erroTrocaG = erroTrocaG + erroTroca;
        erroTroca_1por2G = erroTroca_1por2G + erroTroca_1por2;
        erroTroca_2por1G = erroTroca_2por1G + erroTroca_2por1;
        erroJuncaoPal_SemPerdaG = erroJuncaoPal_SemPerdaG + erroJuncaoPal_SemPerda;
        erroJuncaoPal_ComPerdaG = erroJuncaoPal_ComPerdaG + erroJuncaoPal_ComPerda;
```

```
        erroTotalG = erroTotalG + erroTotal;

      }//Fim FOR Secundario - Arquivo
   }//Fim FOR Primario - Grupo de Arquivos

   //CALCULO GERAL DE TODOS OS ARQUIVOS
   if (erroTotalG>0)
   {
      pErroIncG = float(erroIncG)/float(erroTotalG)*100;
      pErroExcG = float(erroExcG)/float(erroTotalG)*100;
      pErroTrocaG = float(erroTrocaG)/float(erroTotalG)*100;
      pErroTroca_1por2G = float(erroTroca_1por2G)/float(erroTotalG)*100;
      pErroTroca_2por1G = float(erroTroca_2por1G)/float(erroTotalG)*100;
      pErroJuncaoPal_SemPerdaG = float(erroJuncaoPal_SemPerdaG)/float(erroTotalG)*100;
      pErroJuncaoPal_ComPerdaG = float(erroJuncaoPal_ComPerdaG)/float(erroTotalG)*100;
      pErroIncLinG = float(erroIncLinG)/float(erroTotalG)*100;
      pErroExcLinG = float(erroExcLinG)/float(erroTotalG)*100;
      pErroOutroG = float(erroOutroG)/float(erroTotalG)*100;
      pErroTotalG = float(erroTotalG)/float(erroTotalG)*100;
   }

   fprintf(arqCarPercErro,  "Erros Geral de Inclusao        : %d  %5.2f \n",erroIncG, pErroIncG);
   fprintf(arqCarPercErro,  "Erros Geral de Exclusao        : %d  %5.2f \n",erroExcG, pErroExcG);
   fprintf(arqCarPercErro,  "Erros Geral de Troca           : %d  %5.2f \n",erroTrocaG, pErroTrocaG);
   fprintf(arqCarPercErro,  "Erros Geral de Troca 1 por 2   : %d  %5.2f \n",erroTroca_1por2G, pErroTroca_1por2G);
   fprintf(arqCarPercErro,  "Erros Geral de Troca 2 por 1   : %d  %5.2f \n",erroTroca_2por1G, pErroTroca_2por1G);
   fprintf(arqCarPercErro,      "Erros Geral de Juncao Sem Perda : %d      %5.2f \n",erroJuncaoPal_SemPerdaG,
pErroJuncaoPal_SemPerdaG);
   fprintf(arqCarPercErro,      "Erros Geral de Juncao Com Perda : %d      %5.2f \n",erroJuncaoPal_ComPerdaG,
pErroJuncaoPal_ComPerdaG);
   fprintf(arqCarPercErro,  "Erros Geral de Inclusao Linha  : %d  %5.2f \n",erroIncLinG, pErroIncLinG);
   fprintf(arqCarPercErro,  "Erros Geral de Exclusao Linha  : %d  %5.2f \n",erroExcLinG, pErroExcLinG);
   fprintf(arqCarPercErro,  "Outros Erros Gerais            : %d  %5.2f \n",erroOutroG, pErroOutroG);
   fprintf(arqCarPercErro,  "Total Geral de Erros           : %d  %5.2f \n",erroTotalG, pErroTotalG);

   printf("\n\n*** Fim  das  Comparacoes ***"); getche();

   fclose(arqOri);
   fclose(arqGer);
   fclose(arqFilename);
   fclose(arqCarPercAcerto);
   fclose(arqCarPercErro);
   fclose(arqCarDescErro);
   fclose(arqCarBranco);
   return(0);
}


/* F U N C O E S */
int contaCar(FILE *arq)   //Conta Caractere
{
  int totCaract=0;
  char caract;

  fseek (arq, 0L, SEEK_SET);
  caract = getc(arq);              /* Le o primeiro caracter Origem */

  while (!feof(arq))
  {
     totCaract++;
     caract = getc(arq);
  }
  return(totCaract);
}

int contaPal(FILE *arq)   //Conta Palavra
{
  int totPalavra=0;
  char a, ant;

  fseek (arq, 0L, SEEK_SET);
  a = getc(arq);                   /* Le o primeiro caracter Origem */
  ant = a;

  while (!feof(arq))
  {
```

```
      if ( (a==' ' || a==13 || a==10 || a=='\0') && ant!=' ' ) totPalavra++;
      ant = a;
      a = getc(arq);
      if (a==EOF) totPalavra++;
   }
   return(totPalavra);
}

int contaLin(FILE *arq) //Conta Linha
{
   int totLinha=0;
   char a;

   fseek (arq, 0L, SEEK_SET);
   a = getc(arq);                    // Le o primeiro caracter

   while (!feof(arq))
   {
      if (a==13 || a==10) totLinha++;
      a = getc(arq);
      if (a==EOF) totLinha++;
   }
   return(totLinha);
}
```

4 - Programa Compara Palavras

```
/* Programa Compara Palavras */

#include <stdio.h>
#include <conio.h>
#include <string.h>

FILE *arqOri;                          //Arquivo Origem
FILE *arqGer;          //Arquivo Gerado
FILE *arqFilenameNorm;  //Arquivo c/os nomes dos arquivos de Origem e Destino a serem testados
FILE *arqPalPercAcerto; //Arquivo com os valores e percentuais de Acertos por Palavra
FILE *arqPalPercErro;   //Arquivo com os valores e percentuais de Erros por Palavra
FILE *arqPalDescErro;   //Arquivo com a descricao dos Erros por Palavra
FILE *arqPalTamanho;    //Arquivo com a classiticacao do tamanho erro

int contaCar(FILE *arq); //Funcao p/contar o total de caracteres por linha
int contaPal(FILE *arq); //Funcao p/contar o total de palavras por linha
int contaLin(FILE *arq); //Funcao p/contar o total de linha do arquivo

int main()
{
   int totCarO, totCarG, totPalO, totPalG, totLinO, totLinG;
   int totCarLinO, totCarLinG;
   int erroIncLin, erroExcLin, erroPal, erroIncPal, erroExcPal, erroTotal;
   int erroIncLinG=0, erroExcLinG=0, erroPalG=0, erroIncPalG=0, erroExcPalG=0, erroTotalG=0;

   int palErro1, palErro2, palErro3, palErro4, palErro5;
   int palErro6, palErro7, palErro8, palErro9, palErroM9;
   int palErro1G=0, palErro2G=0, palErro3G=0, palErro4G=0, palErro5G=0;
   int palErro6G=0, palErro7G=0, palErro8G=0, palErro9G=0, palErroM9G=0, palErroG=0;

   int palCerta1, palCerta2, palCerta3, palCerta4, palCerta5;
   int palCerta6, palCerta7, palCerta8, palCerta9, palCertaM9, palCerta;
   int palCerta1G=0, palCerta2G=0, palCerta3G=0, palCerta4G=0, palCerta5G=0;
   int palCerta6G=0, palCerta7G=0, palCerta8G=0, palCerta9G=0, palCertaM9G=0, palCertaG=0;
   int i, j, t, k, m, lin, numArq=1, tamPal;

   float pErroIncLin, pErroExcLin, pErroPal, pErroIncPal, pErroExcPal;
   float pErroIncLinG=0, pErroExcLinG=0, pErroPalG=0, pErroIncPalG=0, pErroExcPalG=0;
   float pErroTotal, pAcertoTotal, pErroTotalG=0;

   char caractOri, caractGer;
   char LinhaOri[2000], LinhaGer[2000];
   char vPalOri[50], vPalGer[50], vPalAntOri[50], vPalAntGer[50];
   char NomeArquivo[20], resp;

   bool fim, pulo, pulo2, incPal=false, excPal=false;
```

```
printf("*** Programa Compara Palavras ***\n");
printf("Tipo Normal(1) ou Filtro(2) ?\n");
resp=getche();

if (resp=='2')                    //Filtro
{
  arqFilenameNorm  = fopen ("ArqFilenameFN.txt","r");
  arqPalPercAcerto = fopen ("ArqFPalPercAcerto.txt","w");
  arqPalPercErro   = fopen ("ArqFPalPercErro.txt","w");
  arqPalDescErro   = fopen ("ArqFPalDescErro.txt","w");
  arqPalTamanho    = fopen ("ArqFPalTamanho.txt","w");
}
else                              //Comparacao
{
  arqFilenameNorm  = fopen ("ArqFilenameNorm.txt","r");
  arqPalPercAcerto = fopen ("ArqPalPercAcerto.txt","w");
  arqPalPercErro   = fopen ("ArqPalPercErro.txt","w");
  arqPalDescErro   = fopen ("ArqPalDescErro.txt","w");
  arqPalTamanho    = fopen ("ArqPalTamanho.txt","w");
}

if ( (!arqFilenameNorm) != (!arqPalPercAcerto) != (!arqPalPercErro) != (!arqPalDescErro) != (!arqPalTamanho))
{
  printf("Erro na abertura dos Arquivos Auxiliares."); getche();
  exit(0);
}

fscanf(arqFilenameNorm, "%s", NomeArquivo); //Le o nome do arquivo Origem
arqOri = fopen (NomeArquivo,"r");

if (!arqOri)
{
  printf ("Erro na abertura do Arquivo Origem."); getche();
  exit(0);
}

fprintf(arqPalPercErro,  "*** Arquivo Origem: %s ***\n\n",NomeArquivo);
fprintf(arqPalDescErro,  "*** Arquivo Origem: %s ***\n",NomeArquivo);
fprintf(arqPalTamanho,   "*** Arquivo Origem: %s ***\n\n",NomeArquivo);
fprintf(arqPalPercAcerto,"*** Arquivo Origem: %s ***\n\n",NomeArquivo);
fprintf(arqPalPercAcerto,"Arquivo Gerado  -  Acertos\n");

totCarO = contaCar(arqOri);     //Chama a Funcao Conta Caracteres
fprintf(arqPalPercErro,"Total de caracteres Origem: %d\n", totCarO);

totPalO = contaPal(arqOri);     //Chama a Funcao Conta Palavras
fprintf(arqPalPercErro,"Total de Palavras Origem  : %d\n", totPalO);

totLinO = contaLin(arqOri);     //Chama a Funcao Conta Linhas
fprintf(arqPalPercErro,"Total de Linhas Origem    : %d\n", totLinO);

printf("***  Inicio das Comparacoes   ***\n");getche();


/*==============================*/
/* L O O P   P R I N C I P A L  */

while (!feof(arqFilenameNorm))   //Passa por todos os arquivos que estao no ArqFilename
{
  fscanf(arqFilenameNorm, "%s", NomeArquivo); //Le o nome do arquivo Gerado
  arqGer = fopen (NomeArquivo,"r");

  if (!arqGer)
  {
    printf ("Erro na abertura do Arquivo Gerado."); getche();
    exit(0);
  }

  printf("\nArquivo %d : %s", numArq, NomeArquivo); //Imprime na tela os arquivos comparados

  fprintf(arqPalPercErro,  "\nArquivo Gerado: %s\n",NomeArquivo);
  fprintf(arqPalDescErro,  "\nArquivo Gerado: %s\n",NomeArquivo);
  fprintf(arqPalTamanho,   "\nArquivo Gerado: %s\n",NomeArquivo);
  fprintf(arqPalPercAcerto,"\n%s - ",NomeArquivo);

  totCarG = contaCar(arqGer);
```

```
fprintf(arqPalPercErro,"Total de Caracteres Gerado: %d\n", totCarG);

totPalG = contaPal(arqGer);
fprintf(arqPalPercErro,"Total de Palavras Gerado  : %d\n", totPalG);

totLinG = contaLin(arqGer);
fprintf(arqPalPercErro,"Total de Linhas Gerado    : %d\n\n", totLinG);

fseek (arqOri, 0L, SEEK_SET); //Posiciona os arquivos no topo
fseek (arqGer, 0L, SEEK_SET);

lin = 1;
erroIncLin=0; erroExcLin=0; erroPal=0; erroIncPal=0; erroExcPal=0; erroTotal=0;
palErro1=0, palErro2=0, palErro3=0, palErro4=0, palErro5=0;
palErro6=0, palErro7=0, palErro8=0, palErro9=0, palErroM9=0;
palCerta1=0; palCerta2=0; palCerta3=0; palCerta4=0; palCerta5=0;
palCerta6=0; palCerta7=0; palCerta8=0; palCerta9=0; palCertaM9=0; palCerta=0;
pErroTotal=0; pAcertoTotal=0;

if (totLinG==0)      //Os erros sao todas as linhas nao geradas
{                    //e nao entra no proximo LOOP
  erroExcLin=totLinO;
  erroTotal=totLinO;
}


/*==================================*/
/*  L O O P   S E C U N D A R I O  */

while ( (lin<=totLinO) && (totLinG>0) )  //Passa por todas as linhas do arquivo Origem
{
   pulo = false;
   pulo2 = false;
   fim = false;
   k=0;

   while (!fim)      //Cria cada linha do arquivo Gerado
   {
      caractGer = getc(arqGer);
      if (caractGer==13 || caractGer==10 || caractGer=='\0' || caractGer==EOF) fim = true;
      else
      {
        LinhaGer[k] = caractGer;
        k++;
      }
   }
   totCarLinG = k;

   if ( (totCarLinG==1) && (LinhaGer[i-1]=='+') )
   {
     fprintf(arqPalDescErro,"Inclusao Linha     : %c", LinhaGer[i-1]);
     fprintf(arqPalDescErro,"\t\tLinha: %d\n", lin);
     erroIncLin++;
     erroTotal++;
     pulo=true;        //Nao entra no proximo loop
     pulo2=true;       //Forca Origem a permanecer na mesma linha
   }
   else if ( (totCarLinG==1) && (LinhaGer[i-1]=='-') )
   {
     fprintf(arqPalDescErro,"Exclusao Linha     : %c", LinhaGer[i-1]);
     fprintf(arqPalDescErro,"\t\tLinha: %d\n", lin);

     if (!incPal) //Conta o numero de palavras excluidas
     {
       k=0;
       while( LinhaOri[k]!=' ' && LinhaOri[k]!=10 && LinhaOri[k]!=13 && LinhaOri[k]!='\0' )
       {
          erroExcPal++;
          erroTotal++;
          k++;
       }
     }

     erroExcLin++;
     pulo=true;       //Nao entra no proximo loop
   }
```

```
fim = false;
k=0;

while ((!fim) && (!pulo2))      //Cria cada linha do arquivo Origem
{
  caractOri = getc(arqOri);
  if (caractOri==13 || caractOri==10 || caractOri=='\0' || caractOri==EOF) fim = true;
  else
  {
    LinhaOri[k] = caractOri;
    k++;
  }
}
totCarLinO = k;

if (!pulo)
{
  i=0;
  j=0;
}


/*================================*/
/* L O O P   T E R C I A R I O  */
while( (i<totCarLinO) && (!pulo) ) //Passa por todos os caracteres da Origem
{
   if (!incPal) //Caso ocorra inclusao nao gera a palavra novamente (Origem)
   {
     k=0;
     while(i<totCarLinO && LinhaOri[i]!=' ' && LinhaOri[i]!=13 && LinhaOri[i]!=10 && LinhaOri[i]!='\0' )
     {
        vPalOri[k]=LinhaOri[i];
        i++;
        k++;
     }
     vPalOri[k]='\0';
   }

   if (!excPal) //Caso ocorra exclusao nao gera a palavra novamente (Gerado)
   {
     m=0;
     while(j<totCarLinG && LinhaGer[j]!=' ' && LinhaGer[j]!=13 && LinhaGer[j]!=10 && LinhaGer[j]!='\0')
     {
        vPalGer[m]=LinhaGer[j];
        j++;
        m++;
     }
     vPalGer[m]='\0';
   }

   if (k>0) //Caso exista a palavra Origem
   {
     if (strcmp(vPalOri,vPalGer)) //Tratamento dos Erros
     {
       if (!strcmp(vPalAntOri,vPalGer))  //Caso Gerado = Anterior Origem (Erro de Inclusao)
       {
         fprintf(arqPalDescErro,"Inclusao de Palavra\n");
         erroIncPal++;
         erroPal--; //Subtrai o erro anterior, pois o erro era de inclusao
         incPal=true;
         excPal=false;
       }
       else if (!strcmp(vPalOri,vPalAntGer))  //Caso Origem = Anterior Gerado (Erro de Exclusao)
       {
         fprintf(arqPalDescErro,"Exclusao de Palavra\n");
         erroExcPal++;
         erroPal--; //Subtrai o erro anterior, pois o erro era de exclusao
         excPal=true;
         incPal=false;
       }
       else //Erro de Troca
       {
         tamPal=strlen(vPalOri);
         if      (tamPal==1) {fprintf(arqPalTamanho,"1  Caractere : "); palErro1++;}
         else if (tamPal==2) {fprintf(arqPalTamanho,"2  Caracteres: "); palErro2++;}
```

```
            else if (tamPal==3) {fprintf(arqPalTamanho,"3  Caracteres: "); palErro3++;}
            else if (tamPal==4) {fprintf(arqPalTamanho,"4  Caracteres: "); palErro4++;}
            else if (tamPal==5) {fprintf(arqPalTamanho,"5  Caracteres: "); palErro5++;}
            else if (tamPal==6) {fprintf(arqPalTamanho,"6  Caracteres: "); palErro6++;}
            else if (tamPal==7) {fprintf(arqPalTamanho,"7  Caracteres: "); palErro7++;}
            else if (tamPal==8) {fprintf(arqPalTamanho,"8  Caracteres: "); palErro8++;}
            else if (tamPal==9) {fprintf(arqPalTamanho,"9  Caracteres: "); palErro9++;}
            else                {fprintf(arqPalTamanho,">9 Caracteres: "); palErroM9++;}
            fprintf(arqPalTamanho,"%s - %s\n",vPalOri,vPalGer);

            fprintf(arqPalDescErro,"Palavra Errada: ");
            fprintf(arqPalDescErro,"%s - %s", vPalOri, vPalGer);
            fprintf(arqPalDescErro,"\tLinha: %d \tColunaO: %d \tColunaG: %d\n", lin,i,j);
            erroPal++;
            erroTotal++;
            incPal=false;
            excPal=false;
          }
        } //Fim IF
        else //Classificacao das Palavras Corretas
        {
          palCerta++; //conta as palavras corretas
          tamPal=strlen(vPalOri);
          if      (tamPal==1) palCerta1++;
          else if (tamPal==2) palCerta2++;
          else if (tamPal==3) palCerta3++;
          else if (tamPal==4) palCerta4++;
          else if (tamPal==5) palCerta5++;
          else if (tamPal==6) palCerta6++;
          else if (tamPal==7) palCerta7++;
          else if (tamPal==8) palCerta8++;
          else if (tamPal==9) palCerta9++;
          else                palCertaM9++;

          incPal=false;
          excPal=false;
        }
      }

      if (!incPal)
      {
        for(t=0;t<50;t++) vPalAntOri[t]=' '; //Inicializa a varaivel
        vPalAntOri=vPalOri;
        for(t=0;t<50;t++) vPalOri[t]=' ';
        i++;
      }
      if (!excPal)
      {
        for(t=0;t<50;t++) vPalAntGer[t]=' '; //Inicializa a varaivel
        vPalAntGer=vPalGer;
        for(t=0;t<50;t++) vPalGer[t]=' ';
        j++;
      }
    } //Fim While Caractere

    if (!pulo) lin++;

}  //Fim While Linha

if (erroTotal>0)
{
   pErroIncLin = float(erroIncLin)/float(erroTotal)*100;
   pErroExcLin = float(erroExcLin)/float(erroTotal)*100;
   pErroPal    = float(erroPal)/float(erroTotal)*100;
   pErroIncPal = float(erroIncPal)/float(erroTotal)*100;
   pErroExcPal = float(erroExcPal)/float(erroTotal)*100;
   pErroTotal  = float(erroTotal)/float(erroTotal)*100;
}

if (erroTotal>=totPalO) pAcertoTotal=0;
else if (totPalG>0) pAcertoTotal = 100 - ( (float(erroTotal)*100) / float(totPalO) ) ;

fprintf(arqPalPercErro,  "Erros de Inclusao Linha   : %d  %5.2f \n",erroIncLin, pErroIncLin);
fprintf(arqPalPercErro,  "Erros de Exclusao Linha   : %d  %5.2f \n",erroExcLin, pErroExcLin);
fprintf(arqPalPercErro,  "Erros de Palavra          : %d  %5.2f \n",erroPal, pErroPal);
fprintf(arqPalPercErro,  "Erros de Inclusao Palavra : %d  %5.2f \n",erroIncPal, pErroIncPal);
```

```
    fprintf(arqPalPercErro,  "Erros de Exclusao Palavra : %d  %5.2f \n",erroExcPal, pErroExcPal);
    fprintf(arqPalPercErro,  "Total de Erros            : %d  %5.2f \n",erroTotal, pErroTotal);
    fprintf(arqPalPercErro,  "*************************************************************\n");
    fprintf(arqPalPercAcerto,"%5.2f", pAcertoTotal);

    //Calculo por Arquivo
    erroIncLinG = erroIncLinG + erroIncLin;
    erroExcLinG = erroExcLinG + erroExcLin;
    erroPalG    = erroPalG + erroPal;
    erroIncPalG = erroIncPalG + erroIncPal;
    erroExcPalG = erroExcPalG + erroExcPal;
    erroTotalG  = erroTotalG + erroTotal;

    palErro1G = palErro1G + palErro1;
    palErro2G = palErro2G + palErro2;
    palErro3G = palErro3G + palErro3;
    palErro4G = palErro4G + palErro4;
    palErro5G = palErro5G + palErro5;
    palErro6G = palErro6G + palErro6;
    palErro7G = palErro7G + palErro7;
    palErro8G = palErro8G + palErro8;
    palErro9G = palErro9G + palErro9;
    palErroM9G = palErroM9G + palErroM9;
    palErroG = palErroG + (erroIncLin + erroPal + erroIncPal + erroExcPal);

    palCerta1G = palCerta1G + palCerta1;
    palCerta2G = palCerta2G + palCerta2;
    palCerta3G = palCerta3G + palCerta3;
    palCerta4G = palCerta4G + palCerta4;
    palCerta5G = palCerta5G + palCerta5;
    palCerta6G = palCerta6G + palCerta6;
    palCerta7G = palCerta7G + palCerta7;
    palCerta8G = palCerta8G + palCerta8;
    palCerta9G = palCerta9G + palCerta9;
    palCertaM9G = palCertaM9G + palCertaM9;
    palCertaG = palCertaG + palCerta;

    numArq++;
} //Fim While Arquivo

//Calculo Geral de todos os arquivos
if (erroTotalG>0)
{
   pErroIncLinG = float(erroIncLinG)/float(erroTotalG)*100;
   pErroExcLinG = float(erroExcLinG)/float(erroTotalG)*100;
   pErroPalG    = float(erroPalG)/float(erroTotalG)*100;
   pErroIncPalG = float(erroIncPalG)/float(erroTotalG)*100;
   pErroExcPalG = float(erroExcPalG)/float(erroTotalG)*100;
   pErroTotalG  = float(erroTotalG)/float(erroTotalG)*100;
}

fprintf(arqPalPercErro,"Erros Gerais de Inclusao Linha  : %d  %5.2f \n",erroIncLinG, pErroIncLinG);
fprintf(arqPalPercErro,"Erros Gerais de Exclusao Linha  : %d  %5.2f \n",erroExcLinG, pErroExcLinG);
fprintf(arqPalPercErro,"Erros Gerais de Palavra         : %d  %5.2f \n",erroPalG, pErroPalG);
fprintf(arqPalPercErro,"Erros Gerais de Inclusao Palavra: %d  %5.2f \n",erroIncPalG, pErroIncPalG);
fprintf(arqPalPercErro,"Erros Gerais de Exclusao Palavra: %d  %5.2f \n",erroExcPalG, pErroExcPalG);
fprintf(arqPalPercErro,"Total Gerais de Erros           : %d  %5.2f \n",erroTotalG, pErroTotalG);

fprintf(arqPalTamanho,"\nPalavra Certa");
fprintf(arqPalTamanho,"\n1  Caractere : %d",palCerta1G);
fprintf(arqPalTamanho,"\n2  Caracteres: %d",palCerta2G);
fprintf(arqPalTamanho,"\n3  Caracteres: %d",palCerta3G);
fprintf(arqPalTamanho,"\n4  Caracteres: %d",palCerta4G);
fprintf(arqPalTamanho,"\n5  Caracteres: %d",palCerta5G);
fprintf(arqPalTamanho,"\n6  Caracteres: %d",palCerta6G);
fprintf(arqPalTamanho,"\n7  Caracteres: %d",palCerta7G);
fprintf(arqPalTamanho,"\n8  Caracteres: %d",palCerta8G);
fprintf(arqPalTamanho,"\n9  Caracteres: %d",palCerta9G);
fprintf(arqPalTamanho,"\n>9 Caracteres: %d",palCertaM9G);
fprintf(arqPalTamanho,"\nTotal Geral  : %d",palCertaG);

fprintf(arqPalTamanho,"\n\nPalavra Errada");
fprintf(arqPalTamanho,"\n1  Caractere : %d",palErro1G);
fprintf(arqPalTamanho,"\n2  Caracteres: %d",palErro2G);
fprintf(arqPalTamanho,"\n3  Caracteres: %d",palErro3G);
fprintf(arqPalTamanho,"\n4  Caracteres: %d",palErro4G);
```

```
   fprintf(arqPalTamanho,"\n5  Caracteres: %d",palErro5G);
   fprintf(arqPalTamanho,"\n6  Caracteres: %d",palErro6G);
   fprintf(arqPalTamanho,"\n7  Caracteres: %d",palErro7G);
   fprintf(arqPalTamanho,"\n8  Caracteres: %d",palErro8G);
   fprintf(arqPalTamanho,"\n9  Caracteres: %d",palErro9G);
   fprintf(arqPalTamanho,"\n>9 Caracteres: %d",palErroM9G);
   fprintf(arqPalTamanho,"\nTotal Geral  : %d",palErroG);


   printf("\n\n*** Fim  das  Comparacoes ***"); getche();

   fclose(arqOri);
   fclose(arqGer);
   fclose(arqFilenameNorm);
   fclose(arqPalPercAcerto);
   fclose(arqPalPercErro);
   fclose(arqPalDescErro);
   fclose(arqPalTamanho);
   return(0);
}


/* F U N C O E S */
int contaCar(FILE *arq)   //Conta Caractere
{
   int totCaract=0;
   char caract;

   fseek (arq, 0L, SEEK_SET);
   caract = getc(arq);              /* Le o primeiro caracter Origem */
   while (!feof(arq))      /* Enquanto não se chegar no final do arquivo */
   {
      totCaract++;
      caract = getc(arq);
   }
   return(totCaract);
}

int contaPal(FILE *arq)   //Conta Palavra
{
   int totPalavra=0;
   char a, ant;

   fseek (arq, 0L, SEEK_SET);
   a = getc(arq);                        /* Le o primeiro caracter Origem */
   ant = a;
   while (!feof(arq))      /* Enquanto não se chegar no final do arquivo */
   {
      if ( (a==' ' || a==13 || a==10 || a=='\0') && ant!=' ' ) totPalavra++;
      ant = a;
      a = getc(arq);
      if (a==EOF) totPalavra++;
   }
   return(totPalavra);
}


int contaLin(FILE *arq) //Conta Linha
{
   int totLinha=0;
   char a;

   fseek (arq, 0L, SEEK_SET);
   a = getc(arq);                     /* Le o primeiro caracter */
   while (!feof(arq))   /* Enquanto não se chegar no final do arquivo */
   {
      if (a==13 || a==10) totLinha++;
      a = getc(arq);
      if (a==EOF) totLinha++;
   }
   return(totLinha);
}
```