UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

BRUNO TENÓRIO ÁVILA

ALGORITMOS E ARQUITETURAS PARA PROCESSAMENTO DE DOCUMENTOS DIGITALIZADOS MONOCROMÁTICOS

Recife, dezembro de 2006.

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ALGORITMOS E ARQUITETURAS PARA PROCESSAMENTO DE DOCUMENTOS DIGITALIZADOS MONOCROMÁTICOS

por

BRUNO TENÓRIO ÁVILA

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco como parte dos requisitos para a obtenção do grau de Mestre em Engenharia Elétrica.

ORIENTADOR: RAFAEL DUEIRE LINS, PhD

Recife, dezembro de 2006.

A958a **Ávila, Bruno Tenório**

Algoritmos e arquiteturas para processamento de documentos digitalizados monocromáticos / Bruno Tenório Ávila. – Recife: O Autor, 2006.

xv, 104 f., il. (algumas color.), gráfs., tabs.

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia Elétrica, 2006.

Inclui referências e apêndice.

1. Engenharia Elétrica. 2. Documentos Digitalizados Monocromáticos. 3. Processamento Digital de Documentos. 4. Processamento Digital de Imagens. I. Título.

621.3 CDD (22.ed.) **BCTG/2007-023**

Universidade Federal de Pernambuco

Pós-Graduação em Engenharia Elétrica

PARECER DA COMISSÃO EXAMINADORA DE DEFESA DE TESE DE MESTRADO ACADÊMICO DE

BRUNO TENÓRIO ÁVILA

TÍTULO

"ALGORÍTMOS E ARQUITETURAS PARA PROCESSAMENTO DE DOCUMENTOS DIGITALIZADOS MONOCROMÁTICOS"

A comissão examinadora composta pelos professores: RAFAEL DUEIRE LINS, DES/UFPE, HÉLIO MAGALHÃES DE OLIVEIRA, DES/UFPE e FLÁVIO BORTOLOZZI, PUCPR, sob a presidência do primeiro, consideram o candidato **BRUNO TENÓRIO ÁVILA** **APROVADO**.

Recife, 01 de dezembro de 2006.

JOAQUIM FERREIRA MARTINS FILHO
Coordenador do PPGEE

RAFAEL DUEIRE LINS
Orientador e Membro Titular Interno

FLÁVIO BORTOLOZZI
Membro Titular Externo

HÉLIO MAGALHÃES DE OLIVEIRA
Membro Titular Interno

AGRADECIMENTOS

Expresso aqui meus agradecimentos às pessoas que ajudaram no desenvolvimento desta dissertação. Em especial agradeço:

- à minha família, que sempre me apoiou incondicionalmente em todos os momentos;
- ao professor Rafael Dueire Lins pela aceitação como orientado no mestrado, por me mostrar o valor do trabalho científico, pela motivação, pela disponibilidade e, principalmente, por ter acreditado em mim no desenvolvimento deste trabalho;
- à banca examinadora composta pelos professores Rafael Dueire Lins, Flávio Bortolozzi e Hélio Magalhães de Oliveira pelas contribuições no desenvolvimento desta dissertação através de suas sugestões;
- aos amigos da pós-graduação que compartilharam as dificuldades nas disciplinas e no desenvolvimento desta dissertação. Em especial: André Ricardson, Andrei Formiga, João Marcelo e Márcio Lima.

Bruno Tenório Ávila

Universidade Federal de Pernambuco
01 de dezembro de 2006

Resumo da Dissertação apresentada à UFPE como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica.

ALGORITMOS E ARQUITETURAS PARA PROCESSAMENTO DE DOCUMENTOS DIGITALIZADOS MONOCROMÁTICOS

BRUNO TENÓRIO ÁVILA

Dezembro/2006

Orientador: Rafael Dueire Lins, PhD.

Área de Concentração: Telecomunicações.

Palavras-chave: documentos digitalizados monocromáticos, remoção de borda preta, detecção e correção de orientação e enviesamento, processamento de documentos, processamento de imagens.

Número de Páginas: 104.

RESUMO: Em projetos de digitalização de alto volume de documentos, a imagem dos documentos digitalizados pode ser gerada com vários defeitos acarretando dificuldades desde a sua leitura até sua transcrição automática. Além disso, o alto volume de documentos a serem processados gera a necessidade de automatização do processo de digitalização e à procura por melhores soluções para esses problemas.

Esta dissertação aborda os seguintes problemas encontrados nos documentos digitalizados monocromáticos: detecção e remoção de borda preta e; detecção e correção da orientação e enviesamento. Um novo filtro para remoção de borda preta foi desenvolvido, testado em 21 mil documentos e comparado com várias ferramentas comerciais. Um novo algoritmo de detecção de orientação e enviesamento com capacidade de estimar a rotação em qualquer ângulo com uma precisão de 0,1° foi desenvolvido e comparado com outro método da literatura. Um segundo algoritmo de detecção de enviesamento foi proposto com o objetivo de aumentar o desempenho do processamento. Observaram-se três problemas na correção de rotação do algoritmo clássico: pontos brancos, contorno acidentado e desconexão de parte dos objetos. Um algoritmo de correção de rotação foi proposto com capacidade de corrigir os três problemas. Um método quantitativo para medir a degradação

dos algoritmos de correção de rotação foi introduzido e utilizado para comparar o novo algoritmo com outros da literatura.

Além dos filtros, duas arquiteturas para processamento de documentos digitalizados monocromáticos foram estudadas: seqüencial e *cluster*. Um ambiente visual, intitulado *BigBatch*, foi construído com ambas as arquiteturas. A arquitetura em *grid* foi apenas especulada.

Abstract of the Dissertation presented to the UFPE as part of the necessary requirements for the title of Master in Electrical Engineering.

ALGORITHMS AND ARCHITECTURE FOR MONOCHROMATIC DOCUMENT IMAGE PROCESSING

BRUNO TENÓRIO ÁVILA

December/2006

Supervisor: Rafael Dueire Lins, PhD.

Concentration Area: Telecommunications.

Keywords: monochromatic document image, black border removal, skew and orientation detection and correction, document processing, image processing.

Number of Pages: 104.

ABSTRACT: In high demand digitalization projects, the document image can be digitalized with several defects leading to difficulties since its reading to its automatic transcription. Besides that, the high volume of documents to be processed creates the necessity to the automatization of the digitalization process and the search for better solutions to these problems.

This dissertation approaches the following problems found at the monochromatic document images: black border detection and removal and; skew and orientation detection and correction. A new filter for black border removal was developed, tested against 21 thousand documents and compared to several commercial tools. A new algorithm for skew and orientation detection with the capacity to estimate the rotation in any angle with a precision of 0,1° was developed and compared to other method in literature. A second algorithm for skew detection was proposed with the goal to increase the processing performance. It was observed three problems in the classical rotation correction problem: white pixels, uneven edges and disconnection of the object parts. A rotation correction algorithm was proposed with the capacity to correct all three problems. A quantitative method to measure the degradation of the rotation correction algorithm was introduced and used to compare the proposed algorithm against the classical and others in literature.

Besides the filters, two architectures for monochromatic document image processing were studied: sequential and cluster. A visual environment, entitle BigBatch, was built with both architectures. The architecture grid was only speculated.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE EQUAÇÕES

LISTA DE PSEUDOCÓDIGOS

1. INTRODUÇÃO

O papel é utilizado pelo homem há vários séculos e tornou-se o principal meio de armazenamento e publicação de conhecimento e informações dispostas em forma de idéias, relatos, estórias, mapas, desenhos e obras de artes que são transmitidos de geração em geração através de livros, jornais, revistas, artigos, teses, documentos, etc.

Apesar da importância do papel para o homem, ele apresenta uma série de desvantagens na sua utilização. O papel se desgasta com o tempo tornando-se amarelado e frágil. Caso não seja conservado em um ambiente adequado, pode sofrer a ação de fungos e insetos. Além disso, ele pode ser rasgado, amassado, dobrado, molhado, perdido, queimado, falsificado, etc.

Outra desvantagem é o custo relacionado ao papel. Mesmo que o preço unitário do papel seja extremamente pequeno, o gasto com materiais relacionados à sua utilização e manuseio, como por exemplo, cartucho de impressoras, lápis, pastas, clipes e pranchetas, cópias, etc., agregam um alto valor ao seu custo. A enorme quantidade de papéis, hoje existente nos mais diversos ambientes, ocupa um grande espaço físico. Em empresas públicas e privadas, tornou-se comum o uso de estantes e salas de arquivos para armazenar e organizar os documentos mais importantes, como por exemplo, contratos, notas fiscais, relatórios, manuais, etc. Esta necessidade também agrega um valor ainda maior ao custo do papel. As empresas maiores, com grandes quantidades de papéis, terceirizam o serviço de armazenamento e organização dos documentos que são guardados em galpões.

Tabela 1 - *Produção mundial e dos EUA de papel, em milhões de toneladas.*

Consumo do papel	**Produção Mundial**		**Produção dos EUA**	
	1997	**2001**	**1999**	**2001**
Papel impresso e escrito	90,0	94,8	23,6	21,6
Papel de jornais e revistas	36,0	37,8	6,4	5,7

Com o advento dos computadores, havia-se suposto que o papel se tornaria obsoleto. Então, surgiu o conceito de *Paperless Office*, segundo o qual as empresas não utilizariam papel e tudo seria feito no computador. Contudo, este conceito ainda não se concretizou. Segundo pesquisa realizada a cada dois anos pela Universidade da Califórnia [60], o consumo de papel no mundo entre 1997 e 2001 aumentou (**Tabela *1***), apesar da capacidade de armazenamento digital também ter aumentado. Por sua vez, a pesquisa

mostra que nos Estados Unidos ocorreu diminuição no consumo de papel entre 1999 e 2001 devido ao fato de ser um país moderno e informatizado.

Tabela 2 - *Produção e consumo de papel no Brasil: ambas estatísticas aumentam.*

Brasil	**2005**	**2006 (Projeção)**	**2007 (Previsão)**
Produção (milhões de toneladas)	8,6	8,8	9,0
Consumo per capita (kg/habitante)	39,5	41,1	42,3

Segundo dados da BRACELPA [57], a produção e consumo de papel no Brasil aumentou entre 2005 e 2006. A previsão para 2007 também é o aumento na produção e consumo de papel. Na realidade, o papel provavelmente nunca será extinto ou demorará muito tempo para ocorrer. A tendência é que o uso do papel seja feito de forma mais inteligente.

Vários fatores dificultaram a conversão do documento físico em digital: (1) o custo do *hardware*; (2) a popularização dos computadores; (3) a criação de tecnologias correlatas; (4) a segurança dos dados e; (5) leis para regularização. Entretanto, ocorreram muitos progressos nos últimos anos que permitiram inverter a tendência. O custo do *hardware* tem decaído mais rápido que o previsto pela Lei de Moore – a cada dois anos o poder de processamento aumenta – o que possibilitou a ampla difusão da computação, além de ter contribuído para o surgimento de novos algoritmos de processamento de imagens, sistemas de gerenciamento de documentos, *scanners* de alta produção, redes de banda larga, *hardwares* mais confiáveis, novas técnicas de segurança de dados, como certificado digital e criptografia de 128 bits. Além disso, novas leis foram aprovadas que regularizam e dão valor jurídico aos documentos digitalizados [63].

Os progressos tecnológicos recentes viabilizaram a implantação de soluções de GED (Gerenciamento Eletrônico de Documentos) e é um dos segmentos de serviços que mais cresce no Brasil. Segundo uma pesquisa realizada no final de 2002 pelo CENADEM [58], projetou-se que o mercado de GED no Brasil crescerá 51,5% ao ano no período 2003/2004.

A solução para a eliminação ou redução da utilização de papel de documentos é a digitalização. Em geral, os documentos físicos são dados desestruturados, ou seja, não possuem dados que os caracterize ou identifique-os. A digitalização é um método que toma como entrada documentos físicos e gera imagens de cada um deles. Para identificar cada imagem e associá-la ao documento físico, um conjunto mínimo de dados é criado, tornando, desta forma, o documento físico em um dado semi-estruturado. Em outras palavras,

digitalização é um processo de transformação dos dados desestruturados em dados semi-estruturados. As fases do processo de digitalização (**Figura *1***) são basicamente:

- **triagem**: processo manual de separação e preparação dos documentos físicos para a captura;
- **captura**: o documento físico (papel) é capturado através de um dispositivo eletrônico, como por exemplo, scanners e câmeras digitais;
- **indexação**: criação de um conjunto mínimo de dados de forma que possibilite a identificação da imagem gerada e a sua associação com o documento físico;
- **processamento da imagem**: vários filtros de tratamento da imagem capturada são aplicados com o objetivo de melhorar a sua qualidade visual, diminuir o tamanho de armazenamento ou executar a transcrição automática do documento (OCR - *Optical Character Recognition*);
- **controle de qualidade**: processo manual assistido pelo computador para avaliar cada uma das imagens em relação à qualidade visual e aos resultados da indexação. Caso a imagem seja rejeitada, volta para a fase de captura.

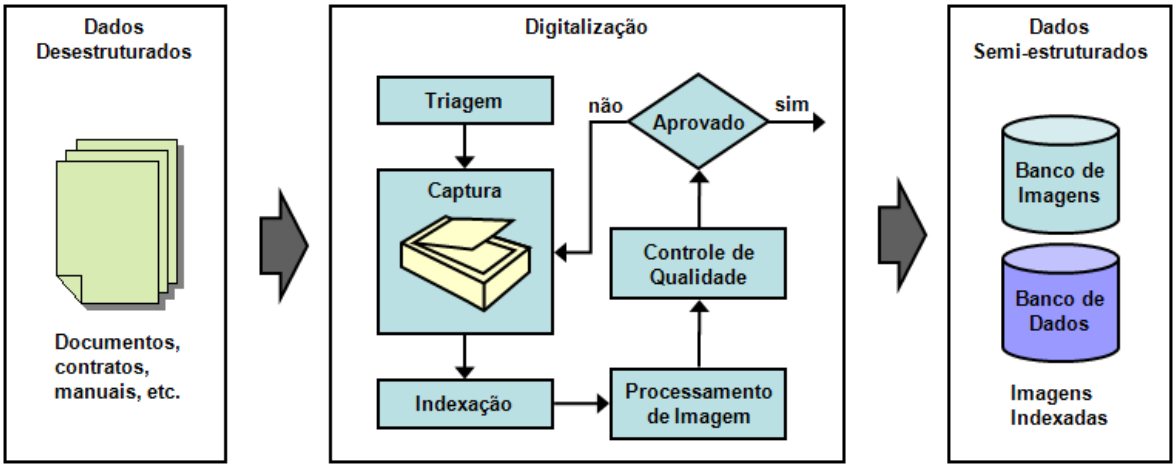
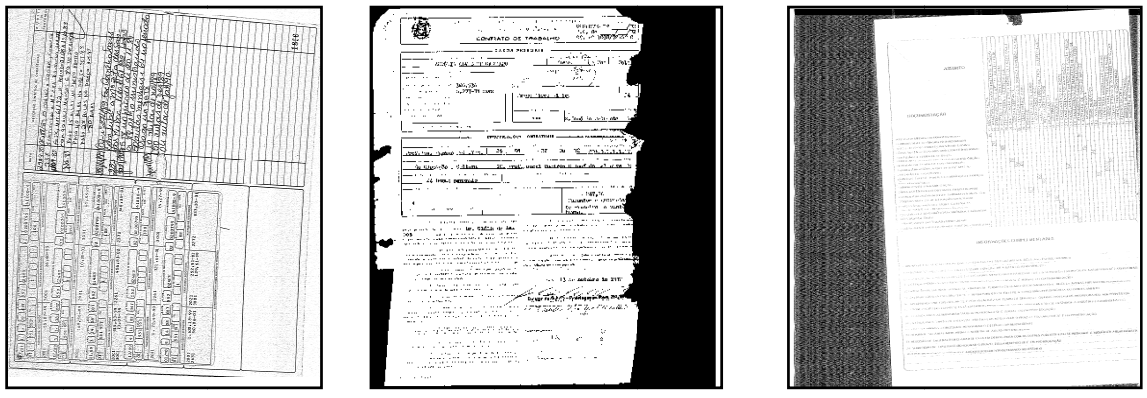


Figura 1 - *Processo básico de digitalização: transformação de dados desestruturados em semi-estruturados.*

A *triagem* é uma fase caracterizada por ser, em sua maioria, manual e é responsável em preparar e organizar os documentos físicos para a fase de captura. Em projetos de digitalização de alto volume de documentos, esta fase é assistida pelo computador para a geração de códigos de barra com dados referentes à sua identificação. Os códigos de barra impressos serão incluídos entre os papéis de forma a organizá-los e serão, posteriormente, detectados e seus dados reconhecidos e utilizados na fase de indexação.

A *captura* é a fase em que o papel sofre a digitalização, ou seja, é convertido em uma imagem e armazenado em uma mídia. Em projetos de digitalização de alto volume de documentos, um scanner de alta produção pode receber uma grande carga de papéis de diferentes formatos, tamanhos, estados de conservação e cor. Freqüentemente, o documento é capturado com falhas, como por exemplo, borda preta, ruídos, rotacionado e distorcido (**Figura 2**). Esses defeitos podem ocorrer por vários motivos: (1) o documento físico encontra-se em um estado de conservação ruim; (2) o *scanner* pode apresentar poeira; (3) devido aos diferentes tamanhos de papel, a entrada de papel do scanner deve se ajustar ao papel de maior dimensão, logo, a imagem de um documento menor pode ser rotacionado (**Figura 2**a) e, além disso, pode apresentar bordas pretas nas áreas externas à página e nas partes rasgadas (**Figura 2**b) e; (4) devido à baixa qualidade dos dispositivos de captura (**Figura 2**).

A *indexação* é uma fase importante do processo de digitalização. Ela consiste na criação de um conjunto mínimo de dados de forma que possibilite a identificação da imagem gerada e a sua associação com o documento físico. Em projetos de digitalização de alto volume de documentos, esta fase já foi iniciada na fase de triagem com a criação dos dados e geração dos códigos de barra. O trabalho restante é feito pelo computador através do reconhecimento dos códigos de barra e da organização das imagens em diretórios. O conjunto de dados dos documentos pode ser significativamente melhorado com a inclusão de palavras-chaves extraídas dos conteúdos dos documentos. A busca das palavras-chaves também pode ser realizada pelo computador através de técnicas de OCR [2][39].



(a) **(b)** **(c)**

Figura 2 - *Exemplos de documentos capturados com falhas: (a) documento rotacionado e com bastante ruído originado do papel; (b) documento com borda preta e rasgões no papel; (c) documento com ruído originado por falha do scanner.*

A fase de *processamento de imagem* consiste de três subfases: (1) tratamento de imagem; (2) transcrição do documento e; (3) compressão de imagem. A subfase de tratamento de imagem consiste em aplicar filtros com objetivo de melhorar a qualidade visual da imagem; alguns filtros podem funcionar como pré-processamento para melhorar a compressão e o reconhecimento do conteúdo. A subfase de transcrição do documento refere-se ao reconhecimento óptico de caracteres (OCR), ou seja, os caracteres representados em pixels na imagem são convertidos para caracteres ASCII. A subfase compressão de imagem consiste em aplicar técnicas de compressão específicas para o tipo de documento digitalizado. Em um projeto de digitalização de alto volume de documentos, toda a fase de processamento de imagem deve ser automatizada.

A fase de *controle de qualidade* caracteriza-se por ser a mais lenta do processo, em conseqüência da necessidade de inspecionar cuidadosamente os resultados das indexações e de processamentos de todas as imagens manualmente auxiliadas pelo computador. As tarefas desta fase incluem: verificar e corrigir a ordem das páginas e os resultados do OCR; avaliar a qualidade visual da imagem final e aplicar filtros específicos para tratamento da imagem, caso detectado algum erro. As imagens com falhas que não puderam ser corrigidas manualmente serão rejeitadas e reenviadas à fase de captura.

A medição da qualidade do documento digitalizado, assim como, a sua medição automática, ainda é um problema em aberto na área de engenharia de documentos e que devem ser motivo de estudos futuros. Caso o projeto de digitalização tenha que transcrever o conteúdo do documento digitalizado, a medição da taxa de acerto do OCR pode ser utilizada apenas para aprovar ou rejeitar parte dos documentos. Todavia, não é adequado para medir a qualidade do documento digitalizado.

A principal motivação para o estudo desta dissertação é a necessidade de automatização, principalmente, da fase de processamento de imagens devido ao grande volume de documentos digitalizados.

Um *scanner* de alta produção [61] tem capacidade de digitalizar 50 folhas por minuto, frente e verso, quando os papéis estão com orientação retrato. Em oito horas de trabalho por dia, 48.000 imagens brutas foram geradas necessitando de indexação, tratamento, transcrição, compressão e controle de qualidade. De forma a não acumular as imagens brutas, todo este processamento deve ser feito de um dia para o outro em, mais ou menos, 12 horas acarretando em uma taxa de processamento de uma imagem por segundo, em média. Sabe-se que apenas a transcrição do documento (capítulo 6) necessita, no estado da arte, de até um minuto para o reconhecimento. Mesmo que não haja necessidade de transcrição, a velocidade de processamento para as outras fases ainda é muito pequeno.

No entanto, muitas empresas de prestação de serviço de digitalização de documentos, inclusive brasileiras, realizam este trabalho manualmente. Isto acarreta no aumento do custo do projeto de digitalização e, assim como, gera atraso no cronograma e imprecisão na qualidade final do documento. O trabalho desta dissertação servirá para suprimir as necessidades dessas empresas, principalmente para o mercado local.

1.1. Objetivo

Esta dissertação tem o objetivo de estudar e aprimorar os algoritmos utilizados na subfase de tratamento de imagens do processo de digitalização de documentos: **remoção de borda preta, detecção e correção da rotação**. Além dos algoritmos, este estudo pesquisou sobre arquiteturas para a fase de processamento de imagens: seqüencial, *cluster* e *grid*. Apenas documentos digitalizados monocromáticos foram explorados nesta dissertação.

A pesquisa envolve o detalhamento dos problemas, o estudo do estado da arte, o desenvolvimento de novos métodos e a realização de testes comparativos entre as técnicas existentes na literatura.

1.2. Critérios e Limitações

A motivação deste estudo provém da necessidade de automatização do processamento dos documentos digitalizados. A capacidade de tratar a maior quantidade de tipos de documentos digitalizados sem comprometer a qualidade final e sem a intervenção do usuário torna possível a automação. Desta forma, deu-se prioridade à qualidade final dos documentos e à robustez dos algoritmos sobre a velocidade de processamento.

Outro motivo para a escolha desta ordem de critérios é o fato do documento só precisar ser processado uma vez, enquanto que a sua visualização e utilização depois de tratado ocorrerão várias vezes.

Vale a pena ressaltar a necessidade de discutir os critérios antes do desenvolvimento dos algoritmos uma vez que afetam a sua construção. De forma a viabilizar a automatização, os algoritmos devem ser construídos sem a utilização de parâmetros, ou se, pelo menos, os parâmetros puderem ser calculados automaticamente. Torna-se óbvio que o conhecimento prévio da caracterização dos documentos propiciará o desenvolvimento de algoritmos mais rápidos. Os critérios específicos de cada um dos filtros estudados serão descritos em seus respectivos capítulos.

1.3. Visão Geral desta Dissertação

Esta dissertação está organizada em seis capítulos, além desta introdução e das referências bibliográficas. O capítulo 2 descreve os principais algoritmos em processamento de imagens da literatura que foram utilizados neste estudo. Os capítulos 3, 4 e 5 abordam os problemas da remoção das bordas pretas, detecção e correção da rotação, respectivamente. O capítulo 6 descreve diversas arquiteturas para processamento de imagens. Finalmente, o capítulo 7 conclui este estudo com considerações finais e propostas de trabalhos futuros.

2. PROCESSAMENTO DE DOCUMENTOS DIGITALIZADOS

Este capítulo descreve as principais técnicas de processamento de imagens voltadas especificamente para documentos digitalizados monocromáticos e que foram utilizadas neste trabalho. Os algoritmos foram classificados em pré-processamento e pós-processamento, no sentido de serem executados antes e depois dos filtros propostos nesta dissertação, respectivamente.

2.1. Algoritmos de Pré-processamento

Os métodos apresentados nesta seção são: binarização, nomeação de componentes, remoção de ruído e diminuição de resolução.

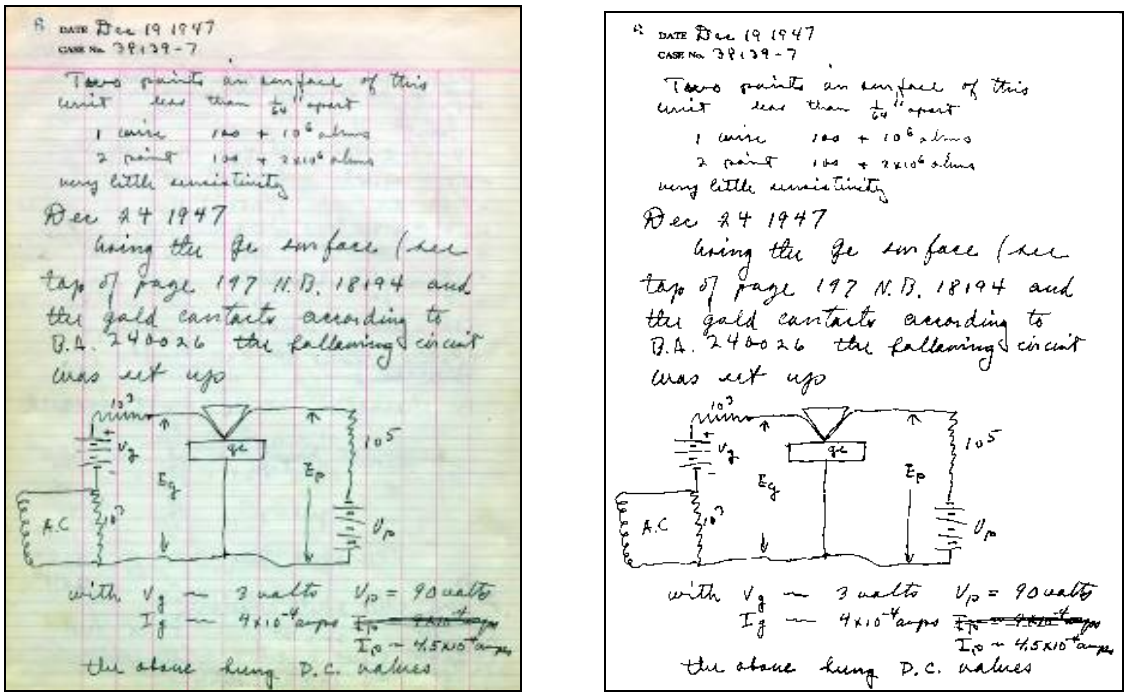
2.1.1. Binarização

A idéia da binarização (*Thresholding*) é reduzir a palheta de cores das imagens (coloridas ou em tons de cinza) para apenas duas cores, demarcando os pixels das regiões de interesse em preto e os pixels das regiões de fundo em branco (**Figura *3***).

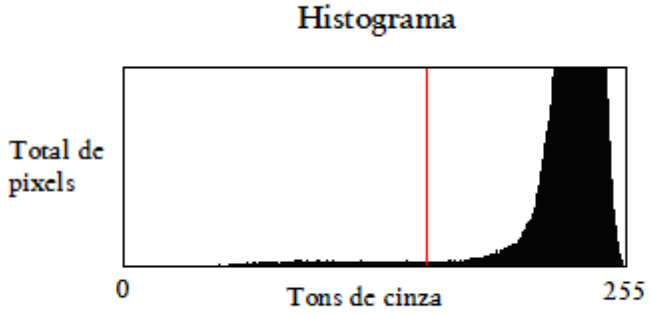
Binarização é um problema clássico em processamento de imagens e existem dezenas de técnicas diferentes. O recente artigo [45] publicado no *Journal of Electronic Imaging* apresenta 40 dos principais algoritmos de *thresholding* descritos na literatura. A questão principal relacionada ao problema é como escolher o critério e os parâmetros de separação (*Threshold*). Os algoritmos de binarização estão divididos em seis categorias:

- **Métodos baseados no formato do histograma**: os picos, vales e curvaturas dos histogramas suavizados são analisados;
- **Métodos baseados em agrupamento**: as amostras de tons de cinza são agrupadas em plano de fundo e primeiro plano (objetos);
- **Métodos baseados em entropia**: usam a entropia do plano de fundo e do primeiro plano ou a entropia entre a imagem original e a binarizada;
- **Métodos baseados nos atributos dos objetos**: procuram uma métrica de similaridade entre as imagens em tons de cinza e as binarizadas;
- **Métodos espaciais**: utilizam a distribuição da probabilidade de primeira ordem e correlação entre os pixels;

- **Métodos locais:** adaptam o valor do *threshold* para cada pixel em cada região da imagem.



(a) (b)



(c)

Figura 3 - *Exemplo de binarização de um documento manuscrito colorido: (a) imagem original; (b) imagem após a binarização; (c) histograma da imagem em escala de cinza com threshold global de 150 (linha em vermelho).*

Em projetos de digitalização de alto volume, a maioria dos documentos, principalmente de empresas, são digitalizados em duas cores. O principal motivo é que tais documentos não possuem valor iconográfico e o custo de armazenamento de uma grande quantidade de documentos em tons de cinza ou colorido seria alto. O custo de armazenamento de documentos monocromáticos são ordens de grandeza menores em

relação ao mesmo documento colorido [13]. Além disso, duas cores são suficientes para manter as principais informações dos documentos. Em geral, os scanners de alta produção já possuem implementados, em hardware, algoritmos de binarização.

Nesta dissertação, vamos considerar que os documentos a serem processados são monocromáticos, ou seja, já foram binarizados, estando o estudo detalhado de tais algoritmos fora do seu escopo.

2.1.2. Nomeação de Componentes

A nomeação de componentes, mais conhecido na literatura como *Component Labelling*, é uma das ferramentas mais importantes e clássicas em processamento de imagens, principalmente monocromáticas. Esta técnica é utilizada na maioria dos filtros propostos nesta tese, além de ser usado em outros métodos na literatura. A principal função do método é identificar unicamente conjuntos de pixels conectados entre si de uma imagem (**Figura *4***).

Os pixels podem estar conectados de duas formas: 4×4, leva em conta apenas os pixels vizinhos na direção vertical e horizontal (**Figura *4***b); 8×8, todos os vizinhos são utilizados, inclusive na diagonal (**Figura *4***c). Em geral, o uso de componentes conectados 8×8 é mais comum, devido aos pixels dos caracteres, após a binarização, também estarem dispostos nas diagonais (**Figura *4***a).

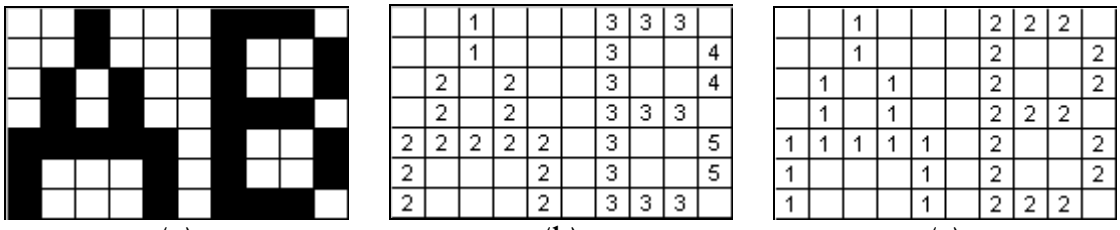


Figura 4 - *Exemplo de nomeação de componentes: (a) imagem original; (b) nomeação de componentes 4×4; (c) nomeação de componentes 8×8.*

Existem vários métodos de nomeação de componentes [19][22][39][44][47]. O método mais conhecido é o de *dois passos* [47]. Na primeira fase, todos os pixels são percorridos da esquerda para direita e de cima para baixo. Para cada pixel preto, os pixels vizinhos nomeados anteriormente em cima e na esquerda são verificados. Se nenhum deles for preto, então um novo valor será associado ao pixel atual. Caso contrário, se pelo menos um deles for preto, então o mesmo valor do pixel vizinho é associado ao pixel atual. Se mais

de um pixel vizinho for preto, então todos os valores associados a eles são colocados em uma classe de equivalência. Ao final dessa fase, o número de componentes é igual ao número de classes equivalentes mais o número de componentes fora da classe de equivalência. Na segunda fase, todos os valores associados em cada classe de equivalência são unidos para formarem um novo valor e então, são associados novamente aos pixels nomeados na primeira fase.

Processar imagens monocromáticas é mais simples em relação às imagens coloridas pelo fato de se poder identificar partes da imagem como componentes ou objetos. A importância da nomeação de componentes reside no fato de ser o primeiro passo para construir uma representação abstrata da imagem. Cada componente pode representar diferentes tipos de elementos textuais como letra, sinal de pontuação, acento, etc. Os componentes podem ser agrupados para formar outro tipo de elemento, como, por exemplo, uma linha de texto formato pelo conjunto de componentes representando letras, que podem ser reagrupados novamente e formar uma coluna e assim por diante. O nível de abstração desejado depende apenas da aplicação. Esta abordagem é chamada na literatura como *top-down*.

2.1.3. Remoção de Ruído

Todo processo de captação de dados, tanto em imagens quanto em sinais, sofre a ação de ruídos. Vários fatores podem acarretar no aparecimento de ruído nos dados: (1) falha no dispositivo de captura; (2) ação de fatores externos, como por exemplo, poeira ou interferência; (3) o próprio dispositivo de captura pode ter defeitos.

Em qualquer aplicação, os ruídos podem afetar os resultados dos processamentos e, desta forma, devem ser removidos. Além disso, outras conseqüências da presença de ruídos em imagens são: (1) imagem de pior qualidade e; (2) aumento do tamanho de armazenamento e transmissão via rede.

Uma conhecida técnica de remoção de ruído é o filtro da vizinhança (*Salt-and-pepper*). Para cada conjunto de pixels pretos conectados entre si de dimensões $k \times k$ na imagem binária, verifica-se os $4(k+1)$ pixels vizinhos. Se não houver nenhum pixel vizinho preto, o conjunto de pixels é considerado ruído e então, marcado de branco. Vale ressaltar que os pixels brancos no fundo preto também podem ser considerados ruídos e vice-versa.

Uma outra técnica chamada de *k-fill* é sugerida por O'Gorman [39]. O método é uma extensão do filtro da vizinhança. Para cada conjunto de pixels pretos conectados entre si de dimensões $(k-2) \times (k-2)$ na imagem binária, calcula-se: n, o número de pixels pretos na vizinhança; c, o número de carreiras de pixels pretos conectados na vizinhança e; r, o

número de pixels pretos nos cantos. A condição para preencher os pontos interiores de branco é:

$$(c = 1)\ and\ (((n > 3k - 4)\ or\ (n = 3k - 4))\ and\ (r = 2)) \qquad (1)$$

O filtro *k-fill* foi desenvolvido especificamente para textos e gráficos com a finalidade de remover ruído enquanto mantém a legibilidade (**Figura *5***). Outra finalidade é a de suavizar os caracteres na medida em que remove o ruído de pixels brancos dos caracteres em preto.

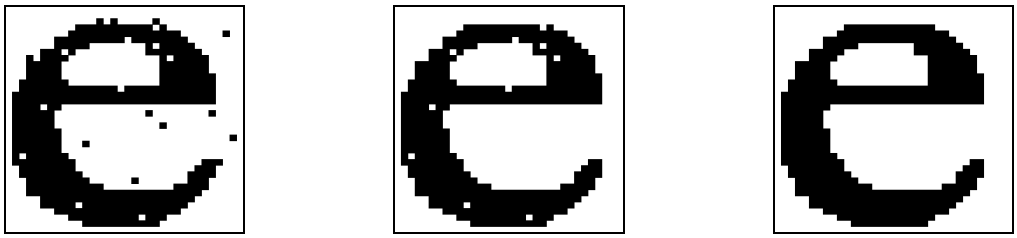


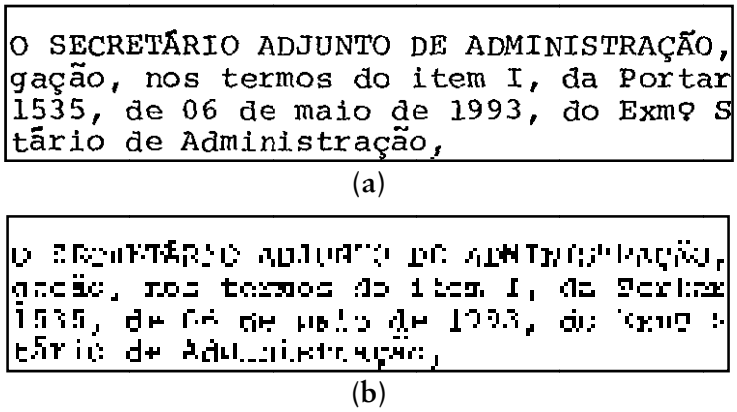
Figura 5 - *Exemplo de remoção de ruído utilizando o filtro k-fill com parâmetro k = 3: (a) imagem original; (b) remoção dos pixels espúrios pretos em fundo branco; (c) remoção dos pixels brancos em fundo preto.*

Os filtros citados acima de remoção de ruído são lentos, devido à necessidade de realizar verificações em todos os pixels da imagem. Uma otimização pode ser obtida para os algoritmos que utilizam a nomeação de componentes. Após a formação dos blocos, os componentes de dimensões menores à *kxk* são classificados como ruído e podem ser removidos ou apenas desconsiderados, desta maneira é efetuada a filtragem de ruído simultaneamente à nomeação de componentes.

2.1.4. Diminuição de Resolução

A digitalização de documentos burocráticos e sem valor iconográfico, se efetuada com resolução de 200 dpi, guarda todos os elementos essenciais do documento oferecendo um bom fator qualidade/espaço de armazenamento [36] e transmissão via rede de computadores [35]. Em caso de transcrição automática via OCR, estudos [2] mostram que a resolução de 200 dpi oferece uma boa taxa de acertos, embora a maior taxa de acertos tenha sido detectada em 300 dpi de resolução. Uma folha A4 digitalizada em 200 dpi tem cerca de quatro milhões de pixels; se digitalizada com 300 dpi, tem cerca de nove milhões de

pixels. A idéia da diminuição da resolução (*Subsampling* ou *Downsampling*) da imagem é reduzir o número de pixels para aumentar a velocidade de processamento.



(a)

(b)

Figura 6 - *Exemplo de diminuição de resolução de imagem: (a) imagem original com 200 dpi; (b) imagem reduzida para 50 dpi.*

A redução da resolução acarreta no desaparecimento de componentes pequenos do documento, como por exemplo, ruídos, sinais de pontuação, sinal da letra *i*, e sobram apenas componentes maiores, como letras (**Figura 6**) ou figuras. Dependendo da redução e do algoritmo escolhido, os caracteres também podem desaparecer comprometendo a precisão dos algoritmos de processamento.

Uma técnica para diminuir a resolução de uma imagem é a redução por *threshold*, proposto por Bloomberg [11]. O método de redução por *threshold* é implementado como uma cascata de operações de redução 2x, em que cada cascata com *threshold* de um. Para cada redução 2x, a imagem é separada em regiões de 2x2 pixels e o resultado do pixel final da nova imagem é preto, se pelo menos um dos quatro pixels for preto; caso contrário, o pixel final é branco.

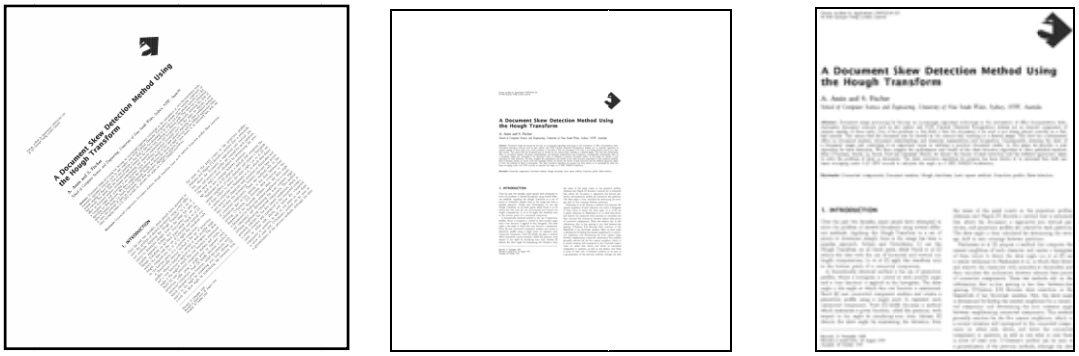
2.2. Algoritmos de Pós-processamento

Os métodos apresentados nesta seção são: remoção de margem branca e suavização de caracteres.

2.2.1. Remoção de Borda Branca

A remoção de borda branca é um procedimento simples e pode ser integrado ao algoritmo de rotação. A borda branca pode ser original do documento, criada pela remoção da borda preta ou pela rotação da imagem. Este filtro remove uma borda branca

proporcionando uma diminuição no custo de armazenamento e uma melhora na qualidade da imagem final.



(a) (b) (c)

Figura 7 - *Exemplo de remoção de margem branca: (a) imagem original; (b) imagem rotacionada; (c) imagem com as margens brancas removidas.*

No exemplo acima (**Figura** *7*), um documento digitalizado com enviesamento de 45° (**Figura** *7*a) é rotacionado. A imagem resultante (**Figura** *7*b) sofreu um aumento significativo das margens brancas. Ao aplicar o filtro, a margem branca da imagem é removida (**Figura** *7*c).

2.2.2. Suavização de Caracteres

A suavização de caracteres tem o objetivo de tornar a borda dos elementos do documento mais uniforme e menos acidentada. É comum o surgimento de pixels nos contornos das letras tornando-os não-uniformes durante a fase captura dos documentos. Este filtro melhora a qualidade visual do documento e acarreta em uma diminuição significativa do custo de armazenamento da imagem [27].

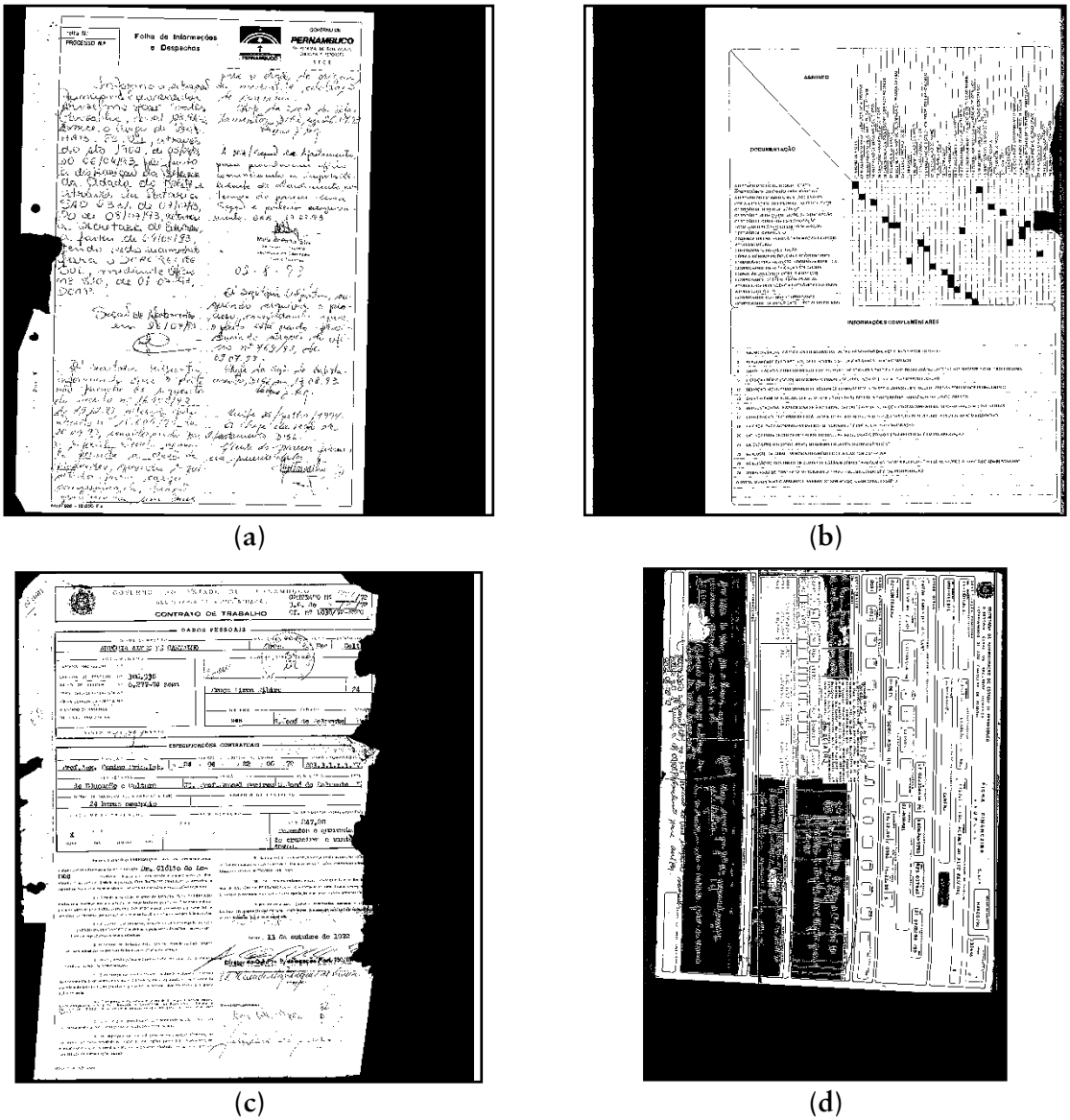
Um método de suavização de caracteres é o filtro *k-fill* (seção 2.1.3). No exemplo acima, a **figura** *8* é aplicada ao filtro para o ruído preto (**Figura** *8*b) e depois para o ruído branco (**Figura** *8*c). Vale destacar que este procedimento gerou uma imagem com menor quantidade de degradações em relação à **figura** *8*c, contudo, o tempo de processamento é maior.

(a) (b) (c)

Figura 8 - *Exemplo de suavização utilizando o filtro k-fill: (a) imagem original; (b) imagem suavizada da cor preta; (c) imagem suavizada da cor branca.*

3. REMOÇÃO DE BORDAS PRETAS

Scanners de alta produção são comumente utilizados para capturar documentos de diversas dimensões em um projeto de digitalização. O espaço do alimentador de papel do scanner é ajustado para o papel de maior dimensão. Portanto, para papéis de dimensões menores, enviesados ou rasgados, surge uma borda preta circundando-a (Figura 9). Devido a outros fatores como brilho e contraste, pode surgir na borda preta uma grande quantidade de ruído.



(a) (b)

(c) (d)

Figura 9 - *Exemplos de documentos burocráticos reais com borda preta, adquiridos por um scanner de produção por firma de digitalização de documentos do Recife.*

A borda preta traz diversos problemas: (a) degrada a qualidade do documento digitalizado produzindo poluição visual, pois parte significativa da área apresentada não possui informação; (b) aumenta significativamente o tamanho comprimido da imagem e; (c) aumento a quantidade de tinta na impressão do documento. A remoção deste elemento do documento trará benefícios, contudo, a sua remoção manual é inviável devido à enorme quantidade de documentos a serem processados. Dessa forma, torna-se importante o desenvolvimento de um filtro automático para remover bordas pretas ruidosas.

A borda preta apresenta as seguintes características: (a) circunda o documento digitalizado (Figura 9); (b) a textura pode ser sólida e/ou apresentar ruídos com tiras brancas (Figura 10a) e; (c) pode apresentar um formato irregular (Figura 10b), devido a uma parte do papel que pode estar rasgada. Uma característica importante foi identificada e deve-se dar destaque: parte da informação do documento pode estar conectada à borda preta (Figura 10c). Desta forma, o filtro de remoção de borda preta deve, essencialmente, evitar que parte da informação do documento seja perdida.

(a) *(b)* **(c)**

Figura 10 - *Características da borda preta: (a) textura sólida e/ou ruidosa; (b) formato irregular; (c) informação conectada à borda.*

Dois termos importantes são utilizados neste capítulo e devem ser definidos: (1) carreira é uma cadeia de pixels conectados de mesmo valor e em uma mesma direção (ex. carreira preta horizontal); (2) bloco é um conjunto de carreiras conectadas de mesmo valor. Este capítulo está organizado da seguinte forma: revisão bibliográfica, idéia básica do filtro proposto, duas diferentes implementações do filtro e os resultados dos testes.

3.1. Revisão Bibliográfica

Em 1996, Le *et al.* [31] propôs um método para remoção de borda que consiste em separar o documento em uma grade de blocos de imagem. Através de análises estatísticas, o algoritmo classifica cada bloco em textual, não-textual (figuras, desenhos), branco (fundo do papel) ou borda. Blocos classificados como borda são removidos. O método estima o tamanho da fonte do texto e, então, utiliza a técnica de *smearing* para segmentar a informação dos blocos classificados como borda.

Em 2001, Fan *et al.* [23] propôs um método semelhante ao de Le *et al.* O primeiro passo consiste em reduzir a resolução da imagem; em seguida, os blocos são segmentados localizando-se possíveis pontos de corte e; finalmente, os blocos são classificados em borda e não-borda. Os blocos classificados com borda são removidos.

Em 2004, Peerawit e Kawtrakul [42] propuseram um método para remover borda baseado em projeção de perfis e densidade de pixels. Após o cálculo e análise da densidade dos pixels, observou-se que o pico da densidade representa a borda e as densidades baixas representam o texto. Desta forma, o algoritmo proposto remove a borda preta guiando-se pela densidade dos pixels percorrendo da esquerda para a direita até atingir o pico que representa o ponto de corte. O mesmo é feito para a direita para a esquerda.

Existem diversas ferramentas comerciais com um filtro para remover borda preta, entre eles: Scanfix [64], Leadtools [62], BlackIce [56], ClearImage [59] e Skyline [65]. Esses filtros são proprietários e os algoritmos implementados não foram publicados.

3.2. Algoritmo Proposto

O filtro de remoção de borda preta com ruídos é baseado no algoritmo de preenchimento e adaptado para segmentar a informação da borda preta [4][5]. Consideram-se, neste estudo, que as informações de um documento monocromático apresentam-se principalmente nas letras ou caracteres do texto. Outros elementos não-textuais, como figuras, tabelas, gráficos, podem apresentar informação, contudo, são menos freqüentes.

Os principais passos do filtro proposto são explicados a seguir. Tendo em vista que a borda preta caracteriza-se por estar nos limites da imagem, inicia-se o filtro pelo o preenchimento dos pixels pretos conectados entre si em pixels brancos a partir dos limites da imagem (Figura 11a). Observa-se que todos os elementos do documento, exceto as figuras, são formados por carreiras pretas que, em média, têm um tamanho relativamente pequeno e constante dado uma resolução. Portanto, para segmentar a informação da borda preta, o filtro verifica estatisticamente se, para cada carreira vertical ou horizontal da borda

preta de tamanho igual ou inferior ao parâmetro SEGMENT, o componente segmentado é informação (Figura 11b,c): (a) caso seja, o algoritmo de preenchimento não ultrapassa esta carreira; (b) caso contrário, o algoritmo de preenchimento prossegue pela carreira.

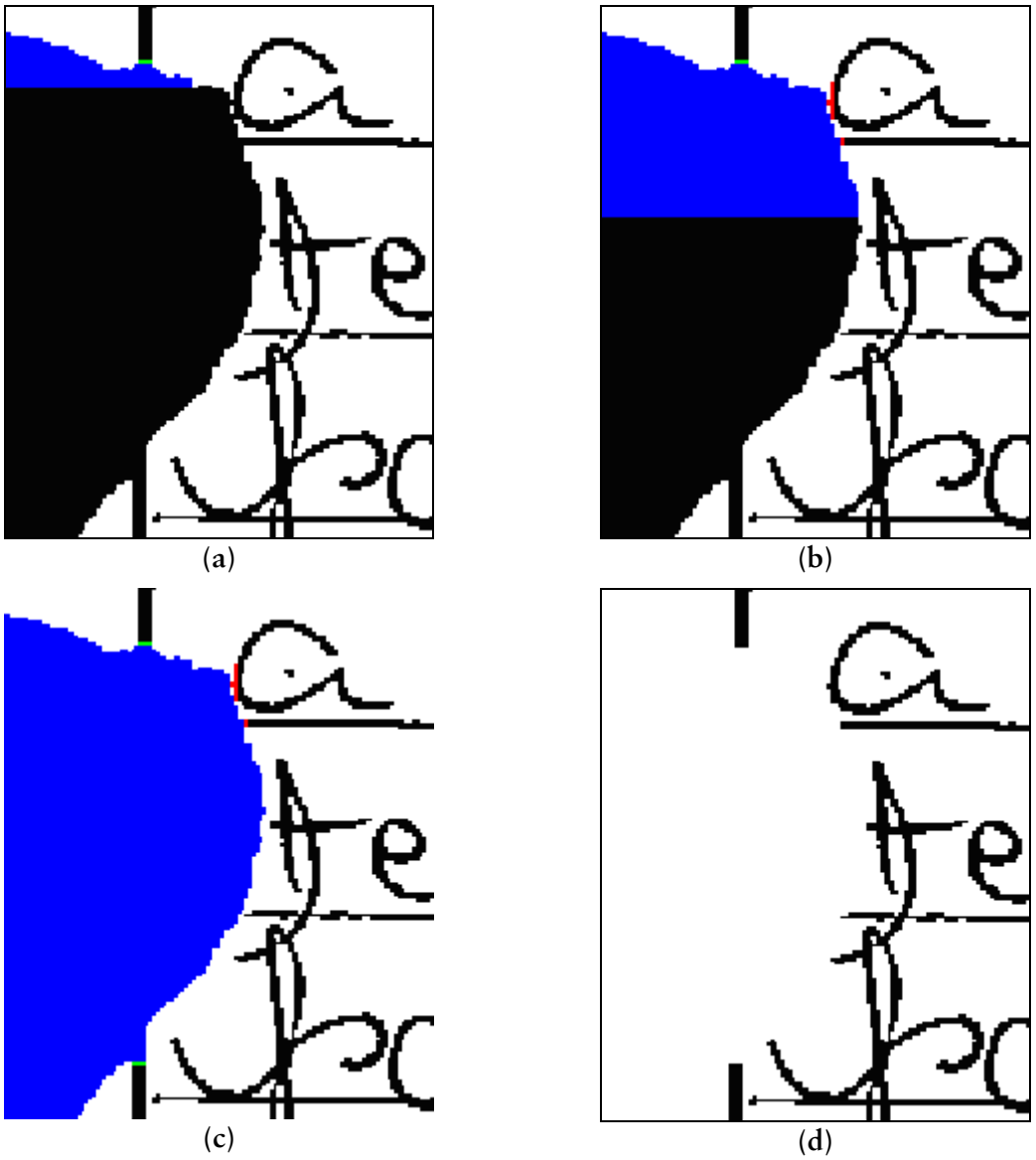


Figura 11 – *Passos do filtro para remoção de borda preta: (a) pixels em azul classificados como borda preta; (b, c) carreiras em vermelho e verde segmentam a informação da borda preta; (d) borda preta removida.*

A análise estatística realizada para classificar o componente segmentado como informação deve levar em conta outros elementos não-textuais que aparecem na borda preta, como por exemplo, pequenos rasgões e excessivos ruídos. Nestes casos, estes elementos devem ser removidos. De fato, uma vez que o componente é segmentado, é

possível extrair diversas características, como área, número de carreiras, tamanho médio das carreiras, etc. e aplicá-las em classificadores mais complexos, como redes neurais. No entanto, o filtro proposto utiliza apenas uma característica: o tamanho da projeção horizontal ou vertical contados da posição da carreira ao final do componente segmentado (Figura 12). Se o tamanho da projeção for maior que o parâmetro LINE, então o componente segmentado é classificado como informação, caso contrário, é classificado como borda preta.

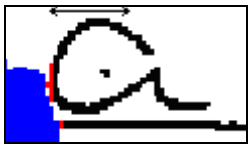


Figura 12 – *Tamanho da projeção horizontal a partir da carreira marcada é utilizado para classificar o componente segmentado.*

Nesta dissertação, os pixels da borda preta são conectados 4×4 entre si. No entanto, alguns ruídos podem aparecer na borda de modo a se isolar (Figura 13). Este tipo de ruído pode ser caracterizado como borda preta se os pixels em sua volta também forem classificados como tal. Para garantir que apenas os componentes menores sejam classificados como borda preta, o filtro classifica apenas os ruídos que estão entre dois componentes classificados como borda preta a uma distância menor que o parâmetro CONNECT.

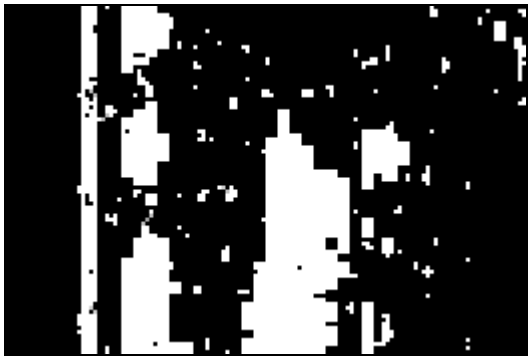


Figura 13 – *Borda preta com ruídos pretos isolados (ilhados).*

As duas próximas seções apresentam duas implementações diferentes do filtro proposto cuja principal diferença é a velocidade de execução. As implementações diferem na forma de localizar a segmentação e na classificação do bloco segmentado. Um ponto em comum é que ambas as implementações do filtro proposto processam apenas a borda preta

e a informação conectada a ela; os outros elementos do documento digitalizado não são processados.

3.2.1. Algoritmo 1

Quanto à forma de localizar a segmentação, o primeiro algoritmo do filtro [4] procura por carreiras menores que representam o início da segmentação à medida que preenche a borda preta. Ao final de cada carreira preta, verifica-se se o tamanho dela é menor que o parâmetro SEGMENT. Se sim, o bloco segmentado é classificado como informação ou borda. Este procedimento é feito para as carreiras verticais e horizontais.

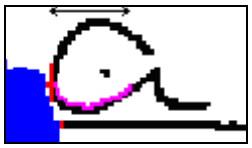


Figura 14 – *Os pixels em roxo foram percorridos através da busca em profundidade e a projeção atingiu o valor do parâmetro LINE.*

Quanto à forma de classificar o bloco segmentado, o algoritmo executa uma busca em profundidade através dos pixels do bloco (Figura 14). Caso a projeção (vertical ou horizontal), ou seja, a distância entre a carreira segmentada e o pixel mais profundo percorrido, for maior que o parâmetro LINE, então o bloco é classificado como informação; caso contrário, é classificado como borda preta.

O pseudocódigo do primeiro algoritmo é descrito a seguir (**Pseudocódigo *1***). A função *create_list_of_black_pixels_at_limit* cria uma lista e adiciona todas as coordenadas dos pixels pretos que estão nas extremidades da imagem. O primeiro algoritmo do filtro é executado enquanto houver um *pixel* em *border_list*. Uma característica do primeiro algoritmo é que o procedimento é executado para cada pixel preto que tenha possibilidade de ser classificado como borda preta. Portanto, o pixel selecionado *p* é inicialmente marcado como borda (BORDER) e pode, no entanto, ser marcado como informação posteriormente. Para cada pixel branco vizinho *w* ao atual pixel *p*, as carreiras horizontais e verticais pretas que tem como ponto inicial o pixel *w* são percorridas para calcular os seus comprimentos. Se o comprimento da carreira horizontal é menor que o parâmetro SEGMENT, então o bloco segmentado por essa carreira horizontal é classificado. A projeção vertical do bloco é medida e se for maior que o parâmetro LINE, então a carreira é marcado como possível informação (HORIZONTAL_MARK); caso contrário, todos os pixels pretos do bloco são marcados como borda (BORDER).

Procedimento semelhante é executado para a carreira vertical. Após o processamento dos pixels brancos vizinhos, todos os pixels vizinhos pretos são adicionados na lista *border_list*.

```
procedure remove_black_border(image, SEGMENT, LINE, CONNECT)
begin
  border_list := create_list_of_black_pixels_at_limit(image);
  while there is pixel p not marked in border_list do
    begin
      mark_pixel(image, p, BORDER);
      for all white neighbour pixels w of p do
        begin
          hor := calc_horizontal_black_run(image, w);
          ver := calc_vertical_black_run(image, w);
          if hor.length < SEGMENT then
            begin
              prj := calc_vertical_projection(image, hor);
              if prj > LINE then
                mark_run(image, hor, HORIZONTAL_MARK);
              else
                flood_fill(image, hor, BORDER);
            end;
          if ver.length < SEGMENT then
            begin
              prj := calc_horizontal_projection(image, ver);
              if prj > LINE then
                mark_run(image, ver, VERTICAL_MARK);
              else
                flood_fill(image, ver, BORDER);
            end;
        end;
      add_black_neighbour_pixels(image, p, border_list);
    end;
  process_special_pattern(image, CONNECT);
  substitute_marks(image);
end;
```

Pseudocódigo 1 – *Primeiro algoritmo do filtro proposto.*

O procedimento acima remove os padrões mais comuns de bordas pretas e, portanto, um pós-processamento pode ocorrer para detectar e marcar outros padrões, como, por exemplo, pixels isolados (Figura 13).

Finalmente, todos os pixels marcados como borda (BORDER) são substituídos por pixels brancos e os outros tipos de marcações (HORIZONTAL_MARK e VERTICAL_MARK) são substituídos por pixels pretos.

3.2.2. Algoritmo 2

O segundo algoritmo do filtro proposto [5] foi construída de forma a remover os gargalos de desempenho do primeiro algoritmo. A abordagem adotada foi construir, de baixo para cima (*bottom-up*), objetos abstratos representando partes da borda preta. A abordagem de baixo para cima é iniciada agrupando-se os pixels pretos vizinhos com características similares em um objeto. Esse objeto pode ser agrupado com outros objetos similares, até chegar a um nível de abstração suficiente para a aplicação desejada. A vantagem prática desta abordagem é diminuir, em ordens de grandeza, o número de pixels processados para alguns objetos que os representa.

Quanto à forma de localizar a segmentação, o segundo algoritmo do filtro proposto marca todas as carreiras horizontais e verticais após o preenchimento da borda preta (Figura 15) e não durante. Em seguida, um *Component Labelling* 4×4 modificado para agrupar pixels com cores iguais é aplicado na borda preta de forma a construir blocos identificando-os unicamente e armazenando suas dimensões (Figura 15c). Finalmente, o próximo passo é construir um grafo de vizinhança dos blocos (*Region Adjacency Graph* – Figura 15d) [47]. Os nós do grafo representam os blocos e as arestas as relações de vizinhança entre os blocos. A representação abstrata da borda preta e suas partes estão finalizadas.

Quanto à forma de classificar o bloco segmentado, este algoritmo calcula para cada bloco marcado como entrada vertical ou horizontal do grafo de vizinhança a projeção. Executa-se uma busca em profundidade através dos blocos (nós) do grafo e a projeção é o somatório das dimensões de cada bloco percorrido. A projeção é calculada no sentido dos limites da imagem para o seu interior, ou seja, inicialmente, o algoritmo calcula a projeção dos blocos nos limites da imagem e segue classificando os blocos em sentido ao centro do documento digitalizado.

O pseudocódigo do segundo algoritmo do filtro é apresentado a seguir (**Pseudocódigo 2**). O primeiro passo é criar uma lista de carreiras horizontais pretas dos pixels pretos que tem a possibilidade de serem classificados como borda ou informação, iniciando a partir das extremidades da imagem. Todos os segmentos são marcados inicialmente como borda (BORDER). Este passo faz parte da abordagem de criar objetos a partir dos pixels, onde os objetos são carreiras horizontais pretas. A vantagem é reduzir em ordens de grandeza o número de objetos a ser processados.

O próximo passo é segmentar as carreiras horizontais e verticais com comprimento menor que o parâmetro SEGMENT. Essas carreiras são marcadas na imagem e é possível

que um mesmo pixel seja marcado com HORIZONTAL_MARK e VERTICAL_MARK. Neste caso, as marcações horizontais foram escolhidas. Neste momento, os valores dos pixels são HORIZONTAL_MARK, ou VERTICAL_MARK, ou BORDER.

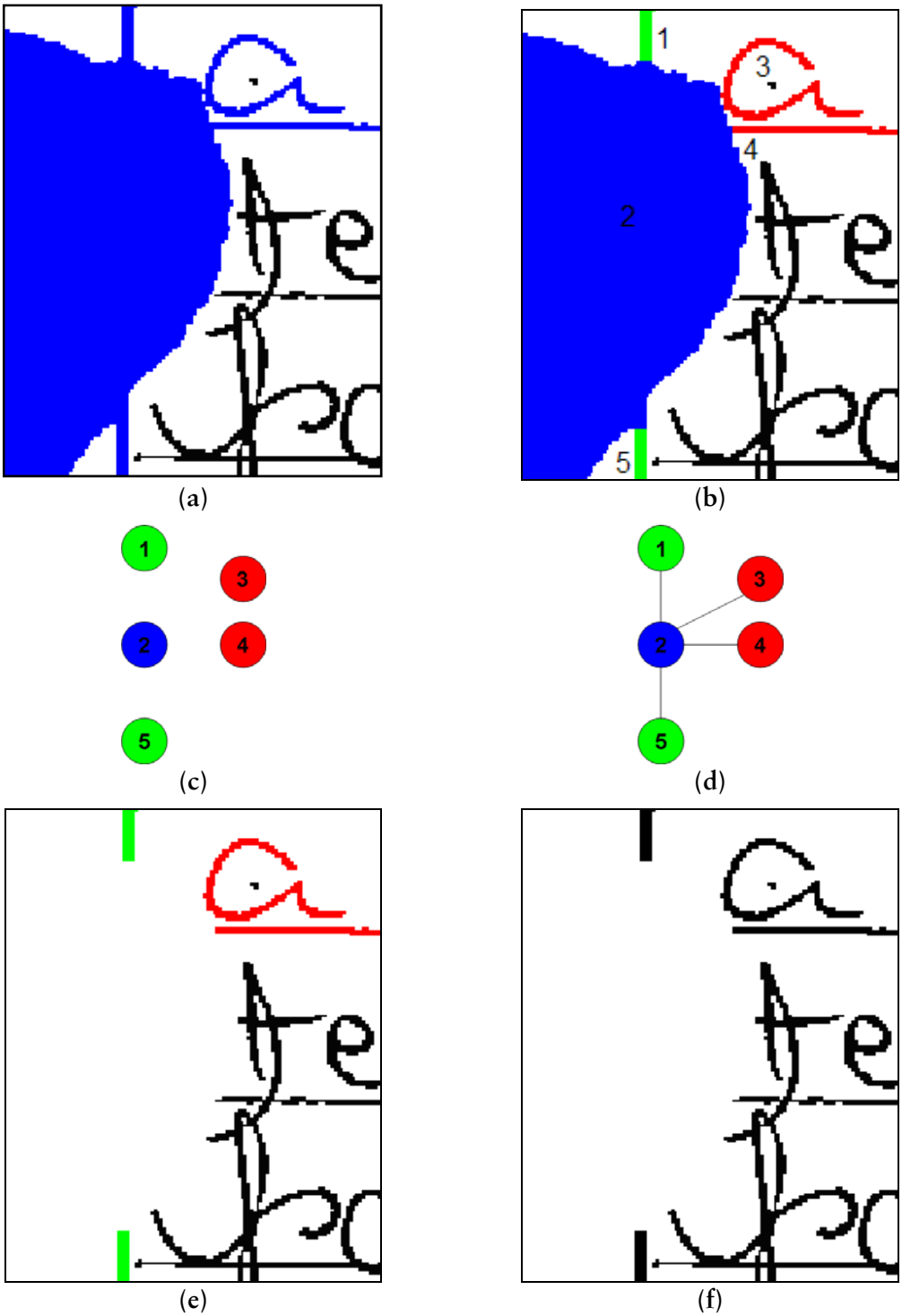


Figura 15 – *Passos do segundo algoritmo: (a) preenchimento da borda preta, incluído a informação conectada; (b) segmentação da borda e da informação; (c) construção e identificação dos blocos; (d) grafo de vizinhança da borda; (e) apenas o bloco em azul*

foi removido, pois os outros foram classificados como informação; (f) resultado final.

```
procedure remove_black_border(image, SEGMENT, LINE, CONNECT)
begin
  segment_list := create_horizontal_runs(image);
  mark_horizontal_runs(image, segment_list, SEGMENT);
  mark_vertical_runs(image, segment_list, SEGMENT);
  block_list := component_labelling_4×4(image, segment_list);
  adj_graph := region_adjacency_graph(block_list);
  classify_list := create_list_of_blocks_at_limit(block_list);
  while there exists block b in classify_list do
    begin
      if b.mark is HORIZONTAL_MARK then
        begin
          prj := calc_vertical_projection(block_list,
                                          adj_graph, b);
          if prj < LINE then
            b.mark := BORDER;
        end
      else
      if b.mark is VERTICAL_MARK then
        begin
          prj := calc_horizontal_projection(block_list,
                                            adj_graph, b);
          if prj < LINE then
            b.mark := BORDER;
        end;
      b.visited := true;
      if b.mark is BORDER then
        add_marked_neighbour_blocks(block_list, adj_graph,
                                    classify_list, b);
    end;
  process_special_pattern(image, blocks, adj_graph,
                          segment_list, CONNECT);
  substitute_marks(image, blocks);
end;
```

Pseudocódigo 2 – *Segundo algoritmo do filtro proposto.*

Em seguida, um *component labelling* modificado de 4×4 foi aplicado às carreiras segmentadas e agrupadas de acordo com a marcação para a formação de blocos. Os blocos foram marcados com o mesmo tipo dos seus segmentos. Este é o próximo passo da abordagem hierárquica. Um passo auxiliar é a formação de um grafo conectando os blocos vizinhos entre si chamado de g*rafo de adjacência de região*. Os blocos são representados por um nó e as relações de vizinhança são representadas por arestas.

O próximo passo é a classificação dos blocos marcados como HORIZONTAL_MARK ou VERTICAL_MARK. Uma lista (*classify_list*) é criada com

todos os blocos nas extremidades da imagem. Para cada bloco com marcação horizontal na lista, é calculada a projeção vertical. Se a projeção for menor que o parâmetro LINE, então o bloco é classificado e marcado como borda (BORDER). A projeção vertical é calculada somando-se as alturas dos blocos percorridos em profundidade no grafo de adjacência partindo do bloco atual. Blocos já visitados não são utilizados para o cálculo da projeção. Procedimento semelhante é executado para blocos com marcações verticais. Finalmente, o bloco atual é marcado como visitado e se tiver sido marcado como borda (BORDER), então os blocos vizinhos a ele não visitados são adicionados à lista (*classify_list*).

O próximo passo é detectar e classificar padrões especiais de borda preta, como por exemplo, pixels isolados. Finalmente, para os pixels dos blocos visitados e classificados como borda (BORDER) são substituídos por pixels brancos e para os pixels dos outros blocos são substituídos por pixels pretos.

3.3. Resultado dos Testes

A base de imagens utilizada para testar os algoritmos de remoção de borda foi dividida em quarto grupos: CD1, CD2, CD3, CD4 (**Tabela *3***). Todas as imagens são monocromáticas e armazenadas utilizando o formato TIFF com o algoritmo de compressão CCITT Group IV. As imagens do CD1 possuem bordas pretas sem ruídos excessivos com a resolução de 200 dpi. As imagens dos grupos CD2, CD3 e CD4 possuem bordas pretas ruidosas digitalizadas com resolução 300 dpi. Cerca de 30% da área é ocupada pela borda preta.

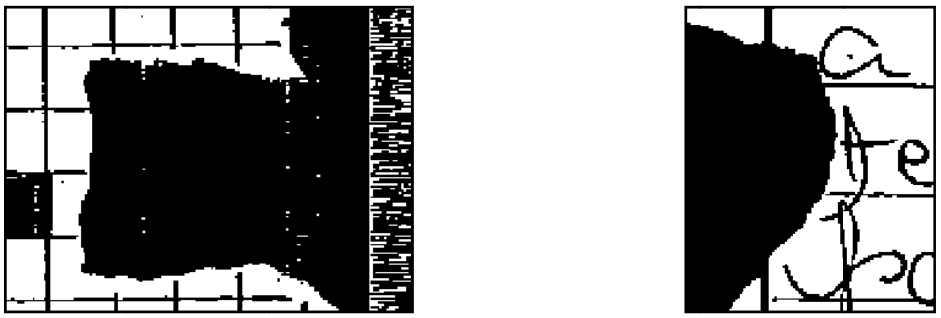
Tabela 3 – *Testes da base de imagens.*

Base	Número de imagens	Média do tamanho original	Altura / Largura (pixels)
CD1	6.020	44 KB	2.336 / 2.578
CD2	5.614	123 KB	4.960 / 3.507
CD3	5.096	124 KB	4.960 / 3.507
CD4	4.537	152 KB	4.960 / 3.507

Nenhum pré ou pós-processamento foi executado nos documentos digitalizados. No entanto, a remoção de ruído antes da remoção da borda preta ajudaria a reduzir o tempo de processamento.

Os algoritmos propostos foram implementados em C e executados em um Intel Pentium IV 2.4 GHz e 512MB de RAM. Ambas as versões foram comparadas a várias ferramentas comerciais: Scanfix, Leadtools, BlackIce, Skyline Tools e ClearImage. Os parâmetros utilizados são: para ambas as versões do filtro proposto, connect = 25 pixels, segment = 10 pixels, line = 10 pixels; para o Leadtools, border percent = 40, white noise length = 40, variance = 40; para o SkyLine, threshold = 100. As ferramentas não especificadas não utilizam parâmetros.

Os critérios utilizados para comparação foram priorizados de acordo com qualidade da imagem final, o tempo de processamento e o tamanho comprimido da imagem final. Infelizmente, a medição de qualidade de imagem ainda é um problema em aberto e, portanto, apenas a inspeção visual das imagens foi utilizada para "medir" a qualidade. Deste modo, as figura 16a, b foram extraídas de dois documentos diferentes e servem apenas de exemplo do efeito de cada filtro de remover borda preta utilizado na comparação. A figura 16a possui uma borda preta ruidosa, com formato irregular e conectado à informação do documento. A figura 16b possui uma borda preta de formato irregular e conectado à informação do documento, contudo, não apresenta ruído branco.



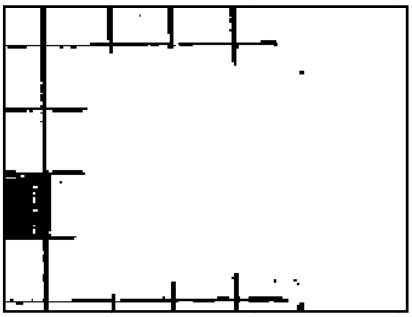
(a) (b)

Figura 16 – *Exemplos utilizados para inspeção visual do processamento dos filtros.*

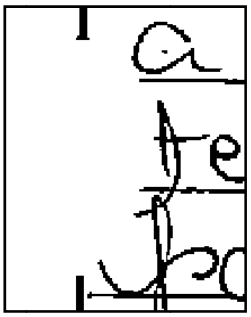
O tamanho de um documento monocromático comprimido com o algoritmo CCITT Group IV pode dar uma idéia da quantidade de ruído na imagem, uma vez que este algoritmo de compressão se degrada na presença de ruído. Portanto, este critério pode ser utilizado para dar apenas uma idéia do quanto dos documentos foi removido de borda preta.

3.3.1. Qualidade da Imagem Final

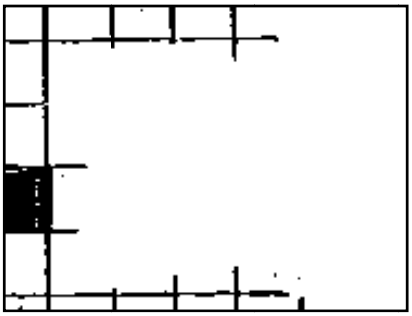
As imagens processadas pelos algoritmos e ferramentas testadas podem ser vistas para inspeção visual na figura abaixo:



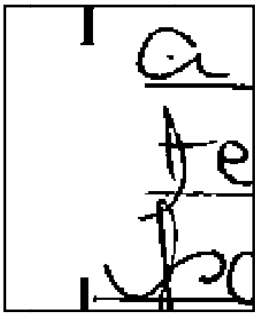
(**a**) *Primeiro algoritmo*



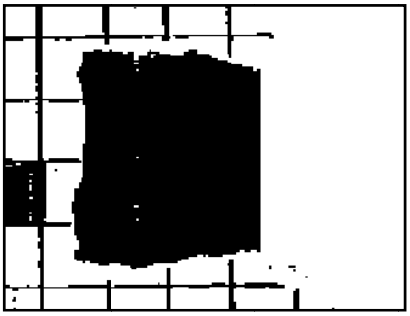
(**b**) *Primeiro algoritmo*



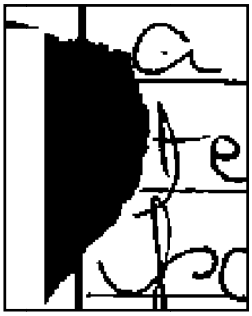
(**c**) *Segundo algoritmo*



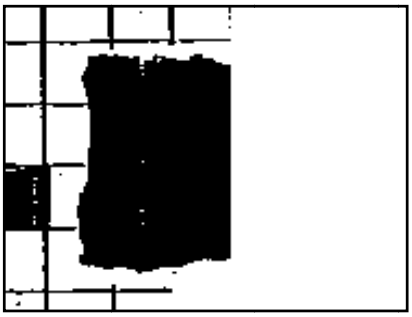
(**d**) *Segundo algoritmo*



(**e**) *Scanfix*

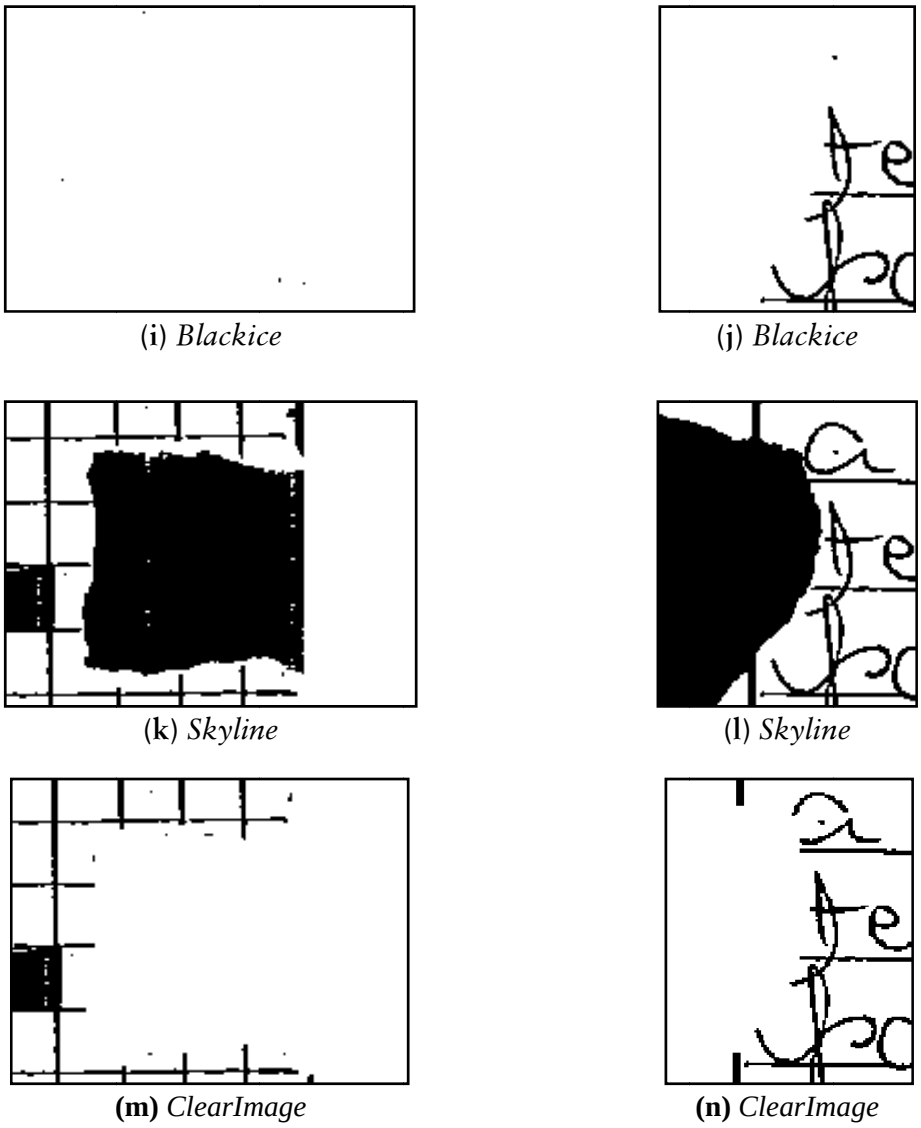


(**f**) *Scanfix*



(**g**) *Leadtools*

(**h**) *Leadtools*



(i) *Blackice* (j) *Blackice*

(k) *Skyline* (l) *Skyline*

(m) *ClearImage* **(n)** *ClearImage*

Figura 17 – *Imagens após o processamento dos algoritmos e ferramentas.*

As imagens finais dos algoritmos propostos (Figura 17a,c e b,d) possuem pequenas diferenças. Vale ressaltar que a borda preta nas imagens foi segmentada corretamente sem que nenhuma informação do documento tenha sido removida. Devido à grande quantidade de imagens da base, não foi contado precisamente quantas bordas pretas não foram corretamente segmentadas, no entanto, estima-se que em cerca de 95% dos casos, os filtros funcionaram com sucesso.

A ferramenta Scanfix [64] (Figura 17e,f) mostrou-se eficiente em bordas pretas com formato regular (mesmo formato do documento). No entanto, falhou para remover as bordas com formato irregular como apresentado, resultando em um pior desempenho.

A ferramenta Leadtools [62] (Figura 17g,h) apresentou um pior desempenho que Scanfix. Na figura 17g, não apenas parte da borda não foi removida como também parte de informação do documento foi removido, na figura 17h, nenhuma parte da borda foi removida.

A ferramenta Blackice [56] (Figura 17i,j) não necessitou de parâmetros, contudo, pelos resultados finais das imagens pôde-se perceber que o algoritmo utilizado é o de preenchimento, mas sem nenhum sinal de segmentação. Este representa o pior das ferramentas, pois destrói toda a informação conectada à borda preta.

A ferramenta Skyline [65] (Figura 17k,l) necessita um parâmetro (*Threshold*) que indica a quantidade de pixels a partir dos limites da imagem para cortar, não importando se têm informação ou borda. Essa ferramenta também apresenta um dos piores resultados em relação à qualidade da imagem.

Entre as ferramentas testadas, a ClearImage [59] (Figura 17m,n) apresentou o melhor desempenho removendo bordas irregulares e segmentando a informação da borda. No entanto, comparando com os algoritmos propostos, percebe-se que a segmentação do ClearImage ocorre um pouco depois do inicio da informação, removendo parte dela. Portanto, os algoritmos propostos apresentam uma qualidade superior ao ClearImage, assim como, a todas as outras apresentadas.

3.3.2. Resultados dos Testes

Os resultados de tempo de processamento estão apresentados na tabela 4. O tempo de processamento de ambas as versões do filtro proposto é proporcional ao tamanho da borda preta e à quantidade de ruído nela. O primeiro algoritmo se degrada bastante nas imagens das bases CD2, CD3 e CD4, onde o ruído é mais forte. No entanto, o segundo algoritmo diminui significativamente o tempo (até cinco vezes) em relação ao segundo algoritmo (CD4). Esta diminuição ocorre devido à abordagem de segmentar antes da classificação dos blocos (algoritmo 2) e não na medida em que segmenta (algoritmo 1). Deste modo, os blocos segmentados são substituídos por representações abstratas (grafos), o que reduz, em ordens de grandeza, a quantidade de componentes (pixels) para ser processado.

Os testes realizados no Scanfix foram executados em uma versão de demonstração, em que a escolha da imagem a ser processada era manual e, portanto, não permitiu a medição do tempo de processamento nem do tamanho comprimido das imagens.

As ferramentas Leadtools e BlackIce apresentaram comportamento semelhante aos propostos, na medida em que aumenta o tempo proporcionalmente ao tamanho da borda e à quantidade de ruído.

A Skyline apresentou o melhor tempo com uma ordem de grandeza menor. No entanto, ela não faz qualquer tipo de processamento ou segmentação, e simplesmente corta a imagem de acordo com o parâmetro especificado.

Tabela 4 – *Resultados do tempo de processamento dos algoritmos e ferramentas, em segundos.*

Base	Algoritmo 1	Algoritmo 2	Scanfix	Leadtools	BlackIce	Skyline	ClearImage
CD1	0,26	0,17	-	0,35	0,91	0,03	0,30
CD2	1,41	0,44	-	1,67	2,96	0,07	0,78
CD3	1,92	0,44	-	1,64	3,10	0,06	0,82
CD4	2,31	0,46	-	1,93	3,04	0,06	0,80

Uma melhor comparação com a ferramenta ClearImage é necessária, uma vez que apresentou a melhor qualidade da imagem final entre as ferramentas. Em relação ao segundo algoritmo proposto, ela é cerca de duas vezes mais lenta.

Os resultados dos tamanhos comprimidos das imagens processadas estão apresentados na tabela 5. Os tamanhos comprimidos de ambas as versões são relativamente iguais e demonstram o mesmo comportamento para ambas as versões. Este resultado também serve como evidência da motivação da necessidade de remoção da borda preta, pois a redução do custo de armazenamento pode chegar a até 29% do tamanho original (CD3).

Tabela 5 - *Resultados do tamanho comprimido das imagens após processamento, em KB.*

Base	Algoritmo 1	Algoritmo 2	Scanfix	Leadtools	BlackIce	Skyline	ClearImage
CD1	38	38	-	36	35	38	37
CD2	95	94	-	91	88	114	95
CD3	88	87	-	85	81	116	88
CD4	113	112	-	108	102	142	115

As ferramentas Leadtools, BlackIce e ClearImage apresentaram comportamentos semelhantes ao filtro proposto. A ferramenta Skyline, devido ao algoritmo implementado, não remove toda a borda acarretando em um maior tamanho de arquivo.

4. DETECÇÃO DE ROTAÇÃO

Este capítulo aborda o problema da detecção automática do ângulo de rotação de documentos digitalizados e monocromáticos. Formalmente, a detecção da rotação é uma função em que uma imagem é passada como parâmetro e é retornado um ângulo. A correção da rotação é outro problema a ser abordado no próximo capítulo.

A detecção e correção da rotação são consideradas pré-processamentos essenciais das ferramentas de OCR, devido às redes neurais terem sido treinadas com os caracteres sem rotação. Estudos mostram que a taxa de acerto cai com o aumento da rotação [2]. O reconhecimento do layout dos documentos também é simplificado se considerar apenas os documentos sem rotação. Outros estudos também mostram que a taxa de compressão aumenta para os documentos sem rotação [27].

Os documentos digitalizados rotacionados com mais de 10 graus trazem dificuldade de percepção aos leitores provocando stress e fadiga [39] e, portanto, a detecção e correção padronizam a sua visualização facilitando a navegação e leitura do documento.

A revisão bibliográfica apresenta os algoritmos encontrados na literatura, assim como, as principais técnicas. Dois algoritmos diferentes são propostos: o primeiro detecta orientação e enviesamento e o segundo detecta apenas o enviesamento. Finalmente, o resultado dos testes do primeiro algoritmo é comparado com outro algoritmo da literatura.

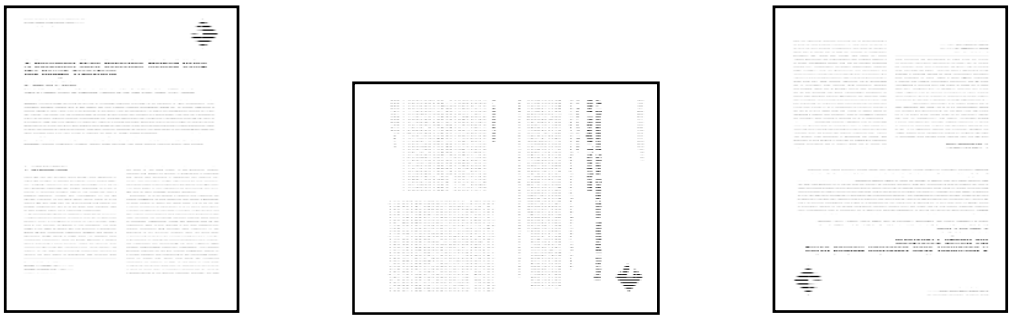
4.1. Conceitos

Alguns conceitos devem ser definidos inicialmente para um melhor entendimento deste capítulo.

Uma **linha de texto** é um grupo de símbolos, caracteres e palavras relativamente próximas e dispostas em uma direção.

A **orientação de uma linha de texto** refere-se à direção e sentido formado pelos elementos da linha de texto em relação a um leitor posicionado à frente do documento digitalizado. Em um documento, pode existir mais de uma linha com orientações diferentes. A **orientação de um documento** é a orientação dominante das linhas de texto do documento. Segundo Bloomberg [12], os múltiplos de 90° descrevem a orientação da imagem. Portanto, definem-se três tipos de orientação:

- **retrato**: refere-se a documentos com orientação de 0° (**Figura *18***a);
- **paisagem**: refere-se a documentos com orientação de 90° e 270° (**Figura *18***b);
- **invertida**: refere-se a documentos com orientação de 180° (**Figura *18***c).



(a) (b) (c)

Figura 18 - *Tipos de orientação: (a) retrato; (b) paisagem; (c) invertida.*

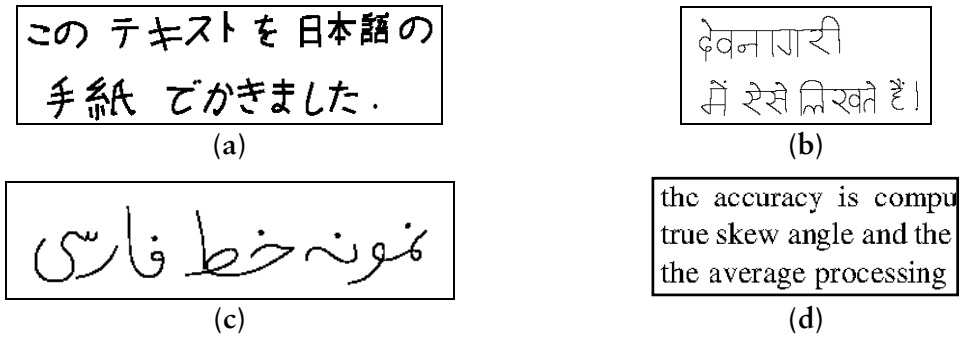
Após a correção da orientação, o documento ainda pode estar desalinhado a um ângulo na faixa de ±45° chamado de **ângulo de enviesamento** (*skew angle*). O **enviesamento de uma linha de texto** é o desvio angular formado pelos elementos da linha de texto em relação a uma linha horizontal. O **enviesamento de um documento** é o desvio angular dominante das linhas de texto do documento.

A **rotação de um documento** refere-se ao ângulo total de desalinhamento do documento incluindo a orientação e enviesamento.

Um **algoritmo, método ou técnica de orientação** refere-se à detecção automática da orientação do documento. Um **algoritmo, método ou técnica de enviesamento** refere-se à detecção automática e correção do ângulo de enviesamento do documento.

4.2. Pontos de Referência

A maioria dos documentos digitalizados possui uma característica em comum: **linhas de texto**. Todas as línguas do mundo organizam os símbolos, letras e palavras em linhas de texto, independente do formato e da maneira de desenhá-los (**Figura *19***). Vale lembrar que em línguas distintas podem ter direções e sentidos diferentes na escrita. Por exemplo, nas línguas ocidentais, o sentido de escrita é da esquerda para a direita de cima para baixo em linhas horizontais, enquanto que no chinês (mandarim) a escrita é feita de baixo para cima da direita para a esquerda em linhas verticais. Todos os algoritmos estudados neste trabalho se baseiam em alguma característica da linha de texto para determinar a orientação e enviesamento do documento.



(a) (b) (c) (d)

Figura 19 - *Exemplos de linhas de texto de diferentes línguas: (a) japonês; (b) hindi; (c) persa; (d) latino.*

O conhecimento prévio dos tipos de documentos utilizados pela aplicação torna possível otimizações. A detecção da orientação de formulários digitalizados (**Figura *20***a) pode ser otimizada e calculada através da localização dos campos que são bem definidos e posicionados. A detecção do enviesamento pode se basear nas linhas divisórias dos campos que, neste caso, levam a uma excelente precisão.

Em geral, a maioria das aplicações utiliza documentos digitalizados em duas cores. As imagens coloridas podem ser filtradas e convertidas para duas cores. As fotos e imagens pertencentes ao conteúdo do documento digitalizado são, em geral, ignorados e apenas as linhas de texto são utilizadas como pontos de referência.

(a) (b)

Figura 20 - *Exemplos de imagens digitalizadas: (a) formulário; (b) imagem colorida de paisagem.*

Em imagens coloridas sem texto, como paisagens e pessoas, o ponto de referência muda em cada caso. Por exemplo, em fotos digitalizadas de paisagens (**Figura *20***b), pode-se basear no fato de que a parte superior deve ser o céu, onde há maior luminosidade, e a parte inferior deve ser a terra, com menor luminosidade. Outro exemplo são as imagens com pessoas que podem se basear na detecção dos olhos ou do rosto. Vale a pena ressaltar que

não faz sentido desenvolver algoritmos de detecção e correção de enviesamento para este tipo de imagem, apenas algoritmos de orientação [51][52][53].

Neste trabalho, estudaram-se apenas documentos digitalizados em duas cores e com um mínimo de linhas de texto para servir de ponto de referência.

4.3. Critérios

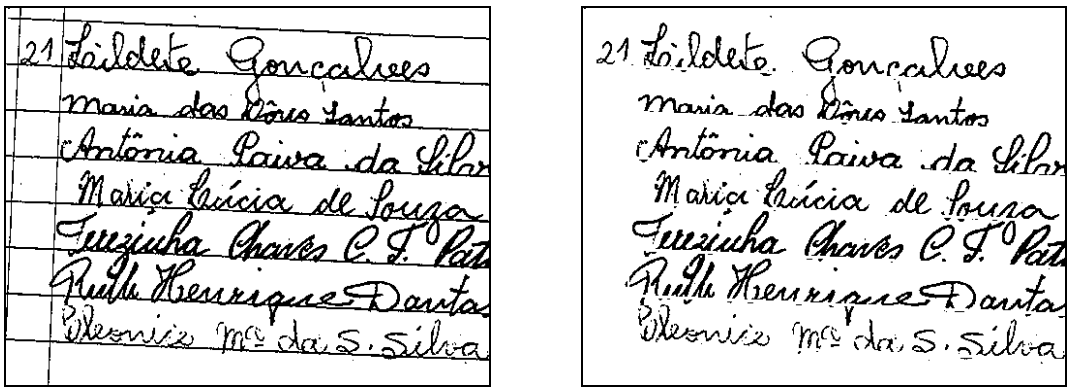
Os critérios para um método de orientação e enviesamento são diferentes para cada tipo de aplicação, contudo, existe um conjunto de critérios que satisfazem à maioria dos casos. Vale a pena a ressaltar a necessidade de discuti-los antes do desenvolvimento de um algoritmo já que afeta a sua construção.

Os critérios para o enviesamento, de acordo com Bloomberg *et al.* [12], são: (1) as operações devem ser rápidas, com o tempo computacional relativamente independente do conteúdo da imagem; (2) deve ser preciso, com uma margem de erro menor que 0,1°; (3) não deve necessitar de segmentar a imagem para isolar as linhas de texto; (4) a detecção não deve ser afetada por imagens de mais de duas cores; (5) para documentos de várias colunas, o enviesamento deve ser independente do alinhamento das linhas de texto entre as colunas; (6) capacidade de detectar o enviesamento localmente e globalmente; (7) o método deve realizar uma medição de confiança ou uma estimativa da probabilidade de erro.

No caso de orientação, os critérios segundo Bloomberg *et al.* [12] são: (1) o tempo de processamento deve ser independente do conteúdo do documento; (2) as operações não devem necessitar de segmentação da imagem; (3) o processamento deve ser relativamente rápido para ser executado antes do OCR; (4) a presença de uma pequena quantidade de enviesamento não deve afetar a orientação; (5) apenas uma quantidade mínima de texto deve ser necessária para detecção; (6) calcular um valor de confiança entre o valor esperado e o resultado.

Alguns critérios sugeridos por Bloomberg não são difíceis de serem implementados. O tempo de processamento pode ser dependente do conteúdo da imagem, uma vez que um documento com um texto maior possui mais linhas de texto, inclusive documentos com várias colunas, e todas devem ser consideradas para aumentar a precisão da detecção da rotação. O método pode fazer uso da segmentação da imagem para isolar as linhas de texto, uma vez que o algoritmo de orientação e enviesamento que se baseiam pelo alinhamento das linhas de texto, devem ser capazes de localizar todas elas e, dessa forma, pode ser necessário segmentar a imagem (**Figura *21***). Para documentos com linhas de texto em várias direções, apenas uma quantidade mínima de texto não é suficiente para detectar orientação ou enviesamento. O método deve se utilizar de uma quantidade máxima de linhas de texto que

for capaz de localizar, de forma a detectar a orientação e enviesamento dominante do documento. Caso o documento tenha um número reduzido de linhas de texto, o algoritmo deve ser capaz de realizar a detecção com esta quantidade mínima de texto disponível.



(a) (b)

Figura 21 - *Exemplo de documento digitalizado que necessita ser segmentado: (a) imagem original com linhas do papel mesclado com as linhas de texto; (b) linhas de texto separadas das linhas do papel.*

Portanto, seguindo os três critérios citados (seção 4.3), os métodos de orientação e enviesamento devem seguir os seguintes critérios priorizados: robustez, precisão e eficiência. Neste caso, não existe a modificação dos pixels do documento, portanto, não cabe o critério de qualidade final do documento. Todavia, ele foi substituído pelo critério de precisão, pois quanto menor o erro, melhor será a qualidade final da imagem.

A característica robustez refere-se à necessidade do método de detectar a orientação e enviesamento em uma quantidade máxima de diferentes tipos de documentos. Os critérios relacionados a esta característica são:

- **Dependência mínima em relação ao "layout" do documento**: os algoritmos não devem fazer suposições sobre o "layout" do documento (ex: várias colunas, tabelas) de forma que não afetem a detecção da rotação. A presença de componentes como imagens e gráficos também não devem alterar o resultado;
- **Dependência mínima em relação à língua do texto**: os diferentes símbolos das diversas línguas utilizados para escrever o texto não devem degradar a detecção da rotação. Vale ressaltar que a direção da linha de texto em algumas línguas (ex. japonês) pode ser vertical, desta forma, afeta apenas a referência da orientação retrato/paisagem. Assim como, não é possível detectar orientação invertida em algumas línguas (ex. japonês);

- **Dependência mínima da forma em que o texto foi escrito**: a forma de escrever os símbolos não deve influenciar no resultado do algoritmo. O texto pode ser escrito de forma datilografada ou manuscrita com letra de forma ou corrida;
- **Dependência mínima do número de cores da imagem**: a quantidade de cores não deve afetar os resultados dos algoritmos. Este problema pode ser resolvido facilmente através da binarização;
- **Quantidade mínima de parâmetros**: o uso excessivo de parâmetros para ajustar o algoritmo aos tipos de documento deve ser evitado;
- **Maior faixa de ângulo**: os algoritmos devem abranger ao máximo a faixa de ângulo que é capaz de detectar.

A característica precisão refere-se à necessidade do método de detectar orientação e enviesamento com uma pequena margem de erro. Os critérios correspondentes a esta característica são:

- **Menor margem de erro**: a margem de erro varia inversamente ao aumento das dimensões do documento. Para documentos de tamanho A4, a margem de erro do enviesamento deve ser menor que 0,1°;
- **Presença de enviesamento**: a presença de uma pequena quantidade de enviesamento não deve afetar a detecção da orientação.

A característica eficiência refere-se à necessidade do método de detectar a orientação e enviesamento ser executado no menor tempo possível, na medida em que não prejudique a robustez e a precisão.

Dois contextos são analisados de forma a justificar a organização e prioridade dos critérios acima: aplicações de OCR e soluções de digitalização de alta produção.

No primeiro contexto, as aplicações de OCR possuem dois detalhes relevantes: (1) recebem de entrada qualquer tipo de documento digitalizado e; (2) são sensíveis às imagens rotacionadas. Neste caso, fica clara a necessidade de priorizar os critérios de robustez e precisão e, portanto, a eficiência é praticamente irrelevante.

No segundo contexto, as soluções de digitalização de alta produção possuem três detalhes importantes: (1) indexação é uma fase crítica do processo; (2) grande diversidade de tipos de documentos; (3) grande volume de documentos. Os fatos 1 e 2 priorizam os critérios de robustez e precisão. O terceiro fato torna a eficiência relevante, contudo, não mais importante pelo fato de que se a maioria das imagens forem corretamente indexadas e processadas, menor é o número de imagens rejeitadas na fase de controle de qualidade, aumentando a velocidade de todo o processo e não apenas de um dos filtros utilizados. Isto

possibilita uma maior automatização do processo. O problema do grande volume de documentos pode ser solucionado com a distribuição da indexação e do processamento das imagens entre os computadores utilizados na digitalização resultando no aumento da velocidade do processo (capítulo 6).

4.4. Revisão Bibliográfica

O problema da detecção da rotação de um documento digitalizado é um problema clássico no campo de engenharia de documentos. Portanto, existem dezenas de algoritmos propostos e diferentes abordagens, em que apenas os principais serão apresentados. Os diversos algoritmos serão classificados em detecção de enviesamento e de orientação.

4.4.1. Algoritmos de Enviesamento

A maioria dos algoritmos de enviesamento pode ser classificada em três categorias: projeção de perfil, transformada de Hough e vizinho mais próximo.

A projeção de perfil (*Projection Profile*) é o histograma do total dos pixels pretos de cada linha paralela da imagem rotacionada em um determinado ângulo. A suposição feita para esta abordagem é que os textos dos documentos estão alinhados em linhas de texto paralelas. Para documentos com linhas de texto dispostos horizontalmente (**Figura 22**a), a projeção de perfil horizontal apresenta picos para as linhas de texto e vales para os espaços entre linhas (**Figura 22**b) e a projeção de perfil vertical forma agrupamentos para cada coluna (**Figura 22**c).

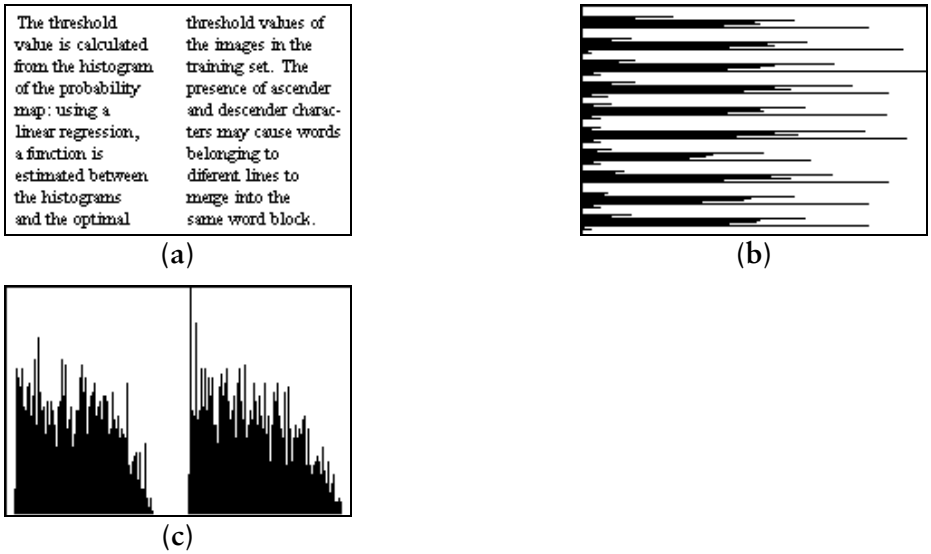


Figura 22 - *Exemplo de projeção de perfil calculada a um ângulo de 0º: (a) documento com duas colunas; (b) projeção de perfil horizontal; (c) projeção de perfil vertical.*

Os passos básicos são: (1) calcular a projeção de perfil da imagem para cada ângulo; (2) definir uma função chamada de *Premium* e; (3) selecionar um ângulo que maximiza a função.

O primeiro trabalho sobre esta abordagem foi de Postl [43]. Todos os pontos pretos da imagem são utilizados e, em cada iteração, o ângulo utilizado na projeção é incrementado a um valor fixo. A função *Premium* é a soma do quadrado da diferença entre os valores adjacentes do histograma. O ângulo que maximizar a função é o ângulo de enviesamento.

Apesar da simplicidade desta técnica, o custo computacional é alto e então, variante foram propostos na tentativa de reduzir a quantidade de pontos utilizados e aprimorar a estratégia de busca do ângulo de enviesamento.

O algoritmo mais conhecido desta abordagem foi proposto por Baird [9]. Ele sugere que seu método é o mais rápido e preciso entre os algoritmos de enviesamento e funciona em uma grande quantidade de diferentes "layouts", incluindo múltiplas colunas, tabelas, espaçamento variável das linhas e várias fontes. Primeiro, a nomeação dos componentes é aplicada ao documento como algoritmo de pré-processamento. Para a redução da quantidade de pontos, é utilizado apenas o ponto médio inferior de cada bloco formado. Componentes maiores são ignorados durante o processo de projeção, enquanto outros, como caracteres, fragmento de caracteres e ruído de margem, são projetados no histograma. Para acelerar a busca do ângulo, um método iterativo é proposto: na primeira iteração, toda a faixa de busca é utilizada com uma precisão pequena; nas iterações seguintes, a faixa de busca é restrita aos valores vizinhos do ângulo melhor classificado na iteração anterior e a precisão é aumentada. Para cada ângulo projetado, a soma dos quadrados (função *Premium*) dos valores do histograma é calculada. O ângulo, na qual a função é maximizada, corresponde ao ângulo de enviesamento. Para melhor precisão, o autor recomenda que a faixa do ângulo seja limitada a ±15°.

No artigo de Ciardiello *et al.* [18], apenas uma sub-região de maior densidade de pixels preto por linha é selecionado do documento digitalizado para ser projetado. A função Premium é desvio padrão do histograma.

No trabalho de Ishitani [28], uma sub-região da imagem é selecionada e os valores acumulados no histograma referem-se ao número de transições de preto/branco das linhas da sub-região. A função *Premium* é a variância dos valores do histograma. O método é robusto na presença de grandes regiões sem texto (imagens, gráficos, tabelas).

A transformada de Hough é uma técnica conhecida e muito utilizada para aproximação de linhas e curvas [21][26]. Esta abordagem é útil quando o objetivo é

aproximar linhas ou curvas utilizando um conjunto de pixels isolados na imagem. O método envolve a transformação do plano de coordenadas da imagem para um espaço parametrizado.

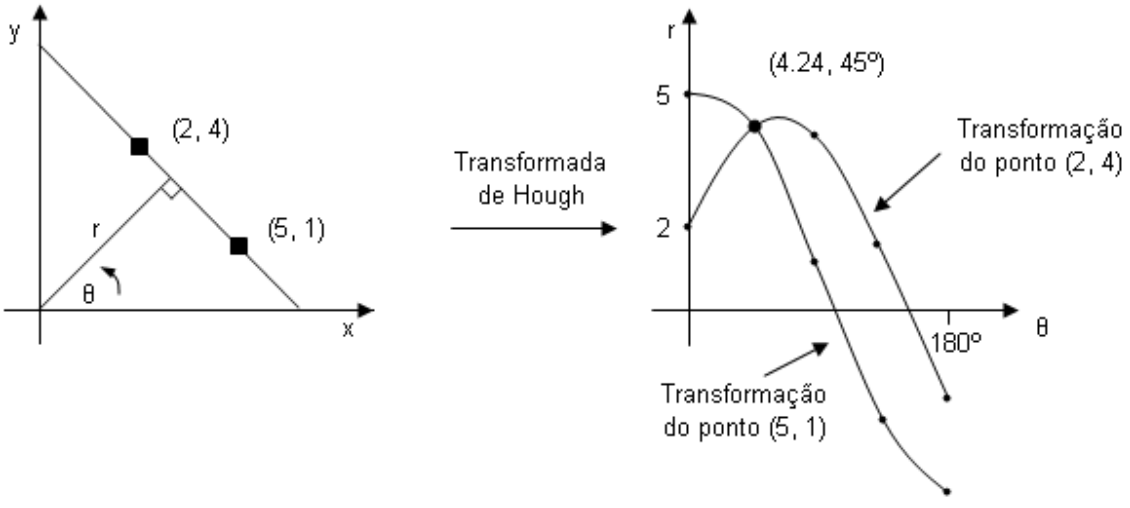
Considerando a aproximação de retas, a equação da linha pode ser expressa como:

$$r = x \cdot \cos\theta + y \cdot sen\theta \qquad (2)$$

onde r é a distância do ponto à origem, θ é o ângulo formado entre r e o eixo axial e, assim, o par (r, θ) define espaço de Hough. Cada pixel preto da imagem de coordenada (x, y) é mapeada com uma nova coordenada (r, θ) no plano de Hough para todos os valores possíveis de θ. Quando vários pontos são colineares, as novas coordenadas se interceptarão no mesmo ponto no plano de Hough. O ângulo é calculado por:

$$\theta^* = \tan^{-1}\left(\frac{x_1 - x_2}{y_1 - y_2}\right) \qquad (3)$$
$$r^* = x_1 \cos\theta + y_1 sen\theta$$

A coordenada do espaço de Hough com o maior acúmulo de pontos mapeados refere-se a uma linha com parâmetros r e θ (**Figura 23**). Na prática, devido a erros de truncamento e ruídos, os pontos não são exatamente colineares e, logo, não será mapeado exatamente com o mesmo ponto no plano de Hough.



(**a**) (**b**)

Figura 23 - *Exemplo de transformada de Hough: (a) imagem com dois pixels pretos; (b) espaço parametrizado de Hough.*

Em aplicações de documentos digitalizados, a transformada de Hough é utilizada para detectar o enviesamento das linhas de texto. Srihari e Govindaraju [49] aplicaram a técnica de transformação de Hough em documentos digitalizados cujo conteúdo contenha apenas textos e com uma única orientação. Todos os pixels pretos da imagem são mapeados no espaço de Hough e o ângulo de enviesamento é estimado como o ângulo que maximiza a soma dos quadrados do gradiente ao longo do eixo r no espaço parametrizado.

Mesmo que a complexidade dos métodos que utilizam a transformada de Hough seja linear em relação ao número de pontos transformados e a faixa de ângulo (θ), o custo computacional é alto. Desta forma, variantes foram desenvolvidas na tentativa de reduzir o número de pontos utilizados.

Hinds *et al*. [25] desenvolveu um algoritmo de enviesamento em que reduz a quantidade de pontos utilizados na transformada de Hough. Primeiro, a resolução da imagem de 300 dpi é reduzida por um fator de 4× e, em seguida, transformada em uma *burst image*. Esta imagem é construída substituindo cada carreira vertical preta pelo pixel mais inferior dela atribuindo o valor do comprimento da carreira. A transformada de Hough é aplicada aos pixels da *burst image* que possuem valor menor a 25 (para 300 dpi), na tentativa de descartar os componentes que não representam texto. Ao contrário do convencional, cada mapeamento no espaço de Hough é incrementado com o valor associado pixel da *burst image*, e não incrementado com valor um. O maior pico no espaço de Hough determina o ângulo de enviesamento. O autor indica que o método funciona em uma faixa de ±45°. A precisão depende da resolução angular atribuída na transformada de Hough.

No trabalho de Le *et al*. [30] é descrito um algoritmo de orientação e enviesamento utilizando transformada de Hough. A sub-região mais bem classificada pelo algoritmo de orientação é passada pela nomeação dos componentes e apenas os pixels da última carreira de horizontal de cada bloco são mapeados no espaço de Hough, na tentativa de reduzir o efeito dos blocos não considerados texto. O método de enviesamento deve ser limitado à faixa de ±15° e com uma precisão de 0,5°.

Pal e Chaudhuri [41] propuseram um algoritmo de enviesamento fazendo uso de nomeação de componentes. A idéia é remover os componentes classificados como ruído: sinais de pontuação, caracteres com saliência para cima (t, f, h, k, b) e para baixo (p, q, y, j, g). Componentes menores com altura abaixo da média são filtrados. Dos blocos resultantes, dois pontos são utilizados na transformada de Hough: o ponto mais a esquerda da carreira mais acima e o ponto mais a direita da carreira mais abaixo.

Amin e Fisher [3] determinam o ângulo de enviesamento de um documento identificando blocos de texto. Primeiro, o algoritmo de nomeação de componentes é aplicado. Os componentes são classificados pelas dimensões em três grupos: ruído, pequeno e grande. Os blocos são agrupados a outros de mesma classificação distanciados até um limite pré-definido. Para a estimativa de enviesamento, cada grupo é dividido em segmentos verticais e apenas o retângulo inferior é utilizado na transformada de Hough. O pico detectado no espaço de Hough representa a parte inferior de uma linha de texto.

Os métodos da abordagem de Vizinho mais Próximo exploram o fato dos caracteres estarem próximos uns dos outros e alinhados em uma linha de texto. Eles são caracterizados por um processo *bottom-up*, em que pixels são agrupados em componentes e, então, agrupados em linhas até um nível superior de abstração, e utilizam a distância entre eles e as relações espaciais para estimar o ângulo de enviesamento. Todos os algoritmos desta classe fazem uso da nomeação de componentes como pré-processamento. Este grupo possui a capacidade inerente de detectar ângulos maiores à faixa de ±45°, geralmente até ±90°. Desta forma, esses algoritmos também podem ser considerados algoritmos de orientação retrato/paisagem.

O primeiro trabalho desta abordagem foi proposto por Hashizume *et al.* [24]. Para cada componente, computa-se o ângulo formado entre o seu vizinho geometricamente mais próximo (**Figura *24***b) e acumula em um histograma (**Figura *24***c). O pico do histograma refere-se ao ângulo de rotação do documento (**Figura *24***d). Este método é também conhecido como *1-NN* ou *1-Nearest Neighbor*.

O'Gorman [38] generaliza o método de Hashizume e estende para *k* vizinhos mais próximos. Este método é conhecido como *k-NN* ou *k-Nearest Neighbor*. Nesse trabalho, o autor descreve o espectro de um documento chamado *Docstrum*, como a representação do documento e que descreve as características da estrutura global da página. O pré-processamento é feito para remover os ruídos e componentes indesejáveis. O filtro *k-fill* é executado para remover o ruído e apenas os componentes de dimensões mais comuns na página serão utilizados em todo o processo.

O primeiro passo do processo é localizar todos os *k* vizinhos de cada componente (**Figura *25***b). O'Gorman recomenda que o valor de *k* seja cinco com a intenção de localizar os vizinhos da mesma linha de texto e de outras adjacentes que fornecerão informações sobre a distância entre caracteres e a distância entre duas linhas, respectivamente. Este passo necessita de um tempo quadrático $O(N^2)$, onde N é o número de blocos, contudo, é possível otimizá-lo realizando a ordenação dos componentes.

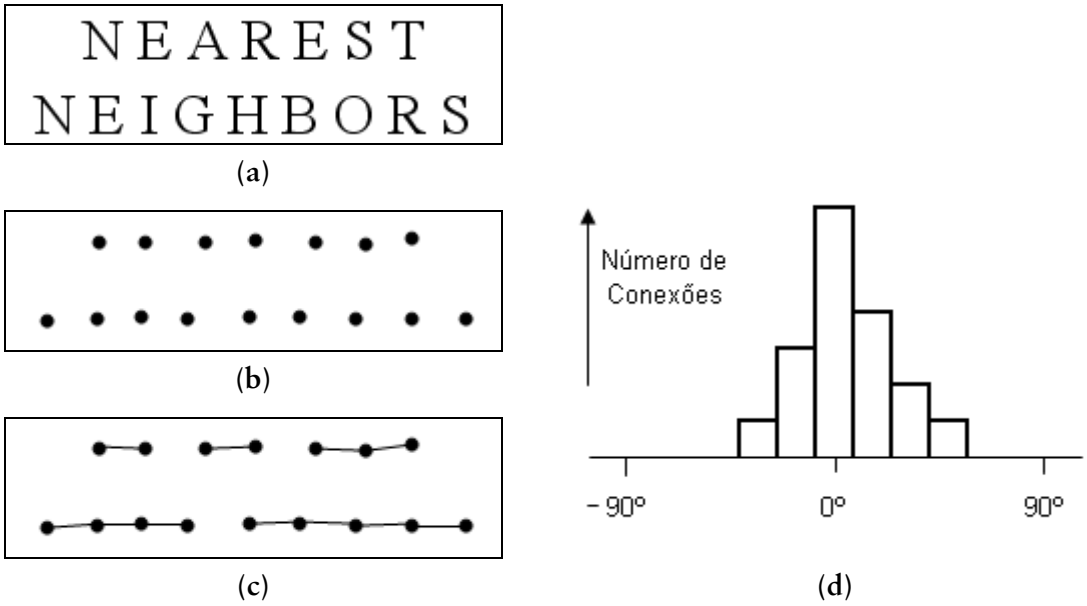


Figura 24 - *Exemplo de 1-NN: (a) imagem com texto; (b) centróides, média dos coordenadas dos pixels pretos de cada bloco; (c) conexão de cada bloco com o vizinho mais próximo; (d) histograma dos ângulos formados entre duas conexões. Neste caso, o pico representa o ângulo de 0°.*

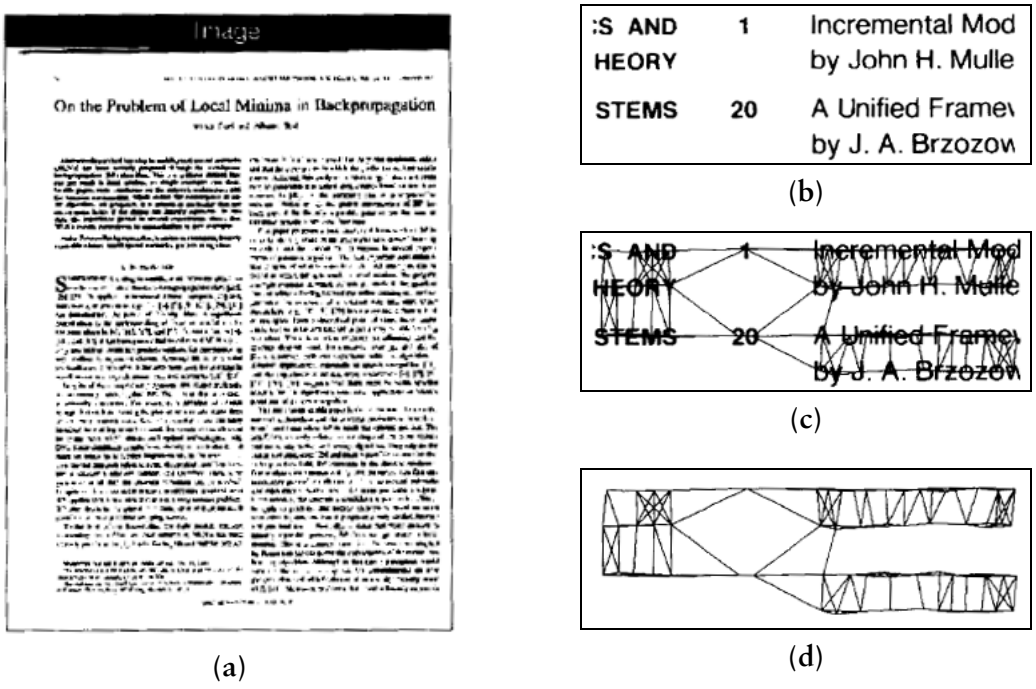
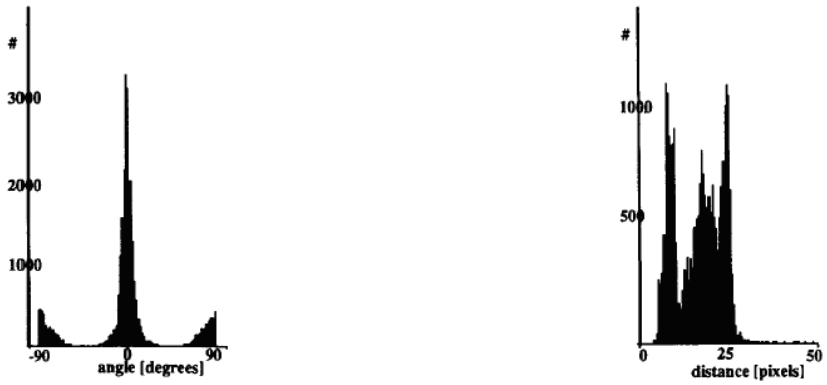


Figura 25 - *Exemplo de k-NN: (a) imagem original; (b) sub-região da imagem a; (c) conexões entre os k vizinhos de cada bloco, neste caso, k = 5; (d) apenas as conexões entre os vizinhos.*

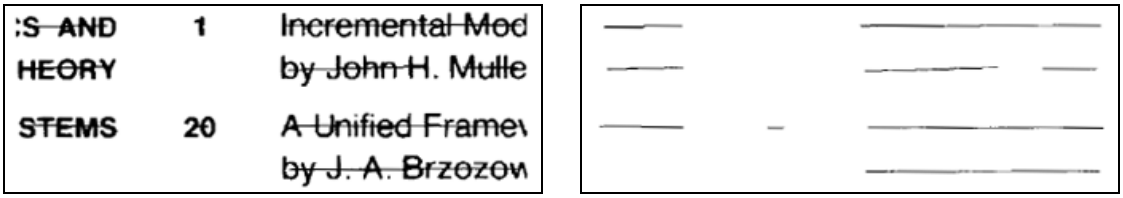
O próximo passo é estimar a orientação do documento. Para cada componente, o ângulo entre o centróide do componente e cada um dos *k* vizinhos é acumulado em um histograma (**Figura *26***a). O pico do histograma suavizado representa a orientação do documento. Em seguida, para cada componente, as distâncias euclidianas entre ele e cada um dos *k* vizinhos com mesmo ângulo da orientação estimada é acumulada em outro histograma (**Figura *26***b). O pico do histograma acumulado fornece a distância entre os caracteres da linha de texto.



(a) **(b)**

Figura 26 - *Histogramas da figura 25a: (a) histograma dos ângulos formado entre os vizinhos; (b) histograma da distância entre vizinhos.*

O passo final é formar as linhas de texto de forma a obter uma melhor precisão do ângulo de rotação. Uma associação transitiva é realizada entre os componentes para agrupar os vizinhos da mesma linha de texto (**Figura *27***). Os critérios de agrupamento entre dois blocos são: ter o mesmo ângulo da orientação estimada e estar a uma distância igual ou menor ao do estimado entre caracteres. Por último, um ângulo mais preciso é formado pelo método do mínimo quadrado aplicado aos centróides da linha de texto e acumulado em outro histograma. O pico do histograma fornece o ângulo de rotação do documento.



(a) **(b)**

Figura 27 - *Formação das linhas de texto: (a) linhas de texto agrupadas; (b) retas formadas pelo método do mínimo quadrado em cada linha de texto agrupado.*

Smith [48] descreve um método baseado na formação de linhas de texto. Como pré-processamento, uma filtragem é aplicada aos componentes cuja altura está fora do intervalo de 20% a 95% da distribuição das alturas dos blocos. Os componentes são ordenados pela coordenada axial e agrupados em linhas de texto da seguinte maneira: para cada componente, o ângulo da sobreposição vertical com linhas existentes, se houver, é calculado. Ele leva em conta a distância horizontal entre o componente e a linha, e a atual estimativa do desvio angular da linha (inicialmente horizontal e atualizado a cada associação). O componente atual é associado a uma nova linha ou para uma existente, dependendo do seu ângulo de sobreposição vertical. Finalmente, para cada linha formada, o ângulo é calculado pelo método do mínimo quadrado. O ângulo de rotação global da página é a mediana dos ângulos calculados.

4.4.2. Algoritmos de Orientação

Os algoritmos de orientação são técnicas de detecção automática de orientação retrato/paisagem e invertida de documentos digitalizados. Os algoritmos de enviesamento da classe Vizinho mais Próximo são capazes de detectar a orientação retrato/paisagem ao mesmo tempo. Na literatura, existem métodos específicos para detectar orientação retrato/paisagem e/ou invertida. Contudo, não há nenhum capaz de detectar um documento rotacionado em qualquer ângulo.

Akiyama e Hagita [1] propuseram um método para orientação retrato/paisagem baseado nas variâncias das projeções verticais e horizontais. A idéia é baseada no fato de o texto possuir espaços regulares entre as linhas de texto com a altura similar a dos caracteres e os espaços entre caracteres serem menores e verticalmente desalinhados (**Figura *28***a). Primeiro, a projeção de perfil horizontal e vertical é calculado para todo o documento (**Figura *28***b,c). Em seguida, a variância para cada projeção é calculada. A orientação do documento escrito em língua ocidental é retrato se a variância da projeção horizontal for maior em relação à vertical, caso contrário, a orientação é paisagem.

Este algoritmo fornece uma estimativa global da orientação do documento, logo, é sensível a presença de blocos não considerados texto (**Figura *28***d). Le *et al.* [30] propôs uma estimativa local para diminuir a sensibilidade de documentos com gráficos ou tabelas. A orientação da página é detectada dividindo a imagem em pequenos quadrados, cada um classificado como texto ou não-texto de acordo com várias heurísticas que leva em consideração a densidade e a distribuição dos pixels pretos. Cada quadrado classificado como texto é classificado como retrato ou paisagem analisando as projeções verticais e horizontais. A classificação depende, em primeiro lugar, da presença de picos alternados

com vales e, em segundo lugar, na comparação das variâncias dos perfis. Esses quadrados constituem o primeiro nível da pirâmide; cada camada superior é constituída por quadrado maiores construídos a partir da junção de noves quadrados vizinhos da camada imediatamente inferior (**Figura *29***).

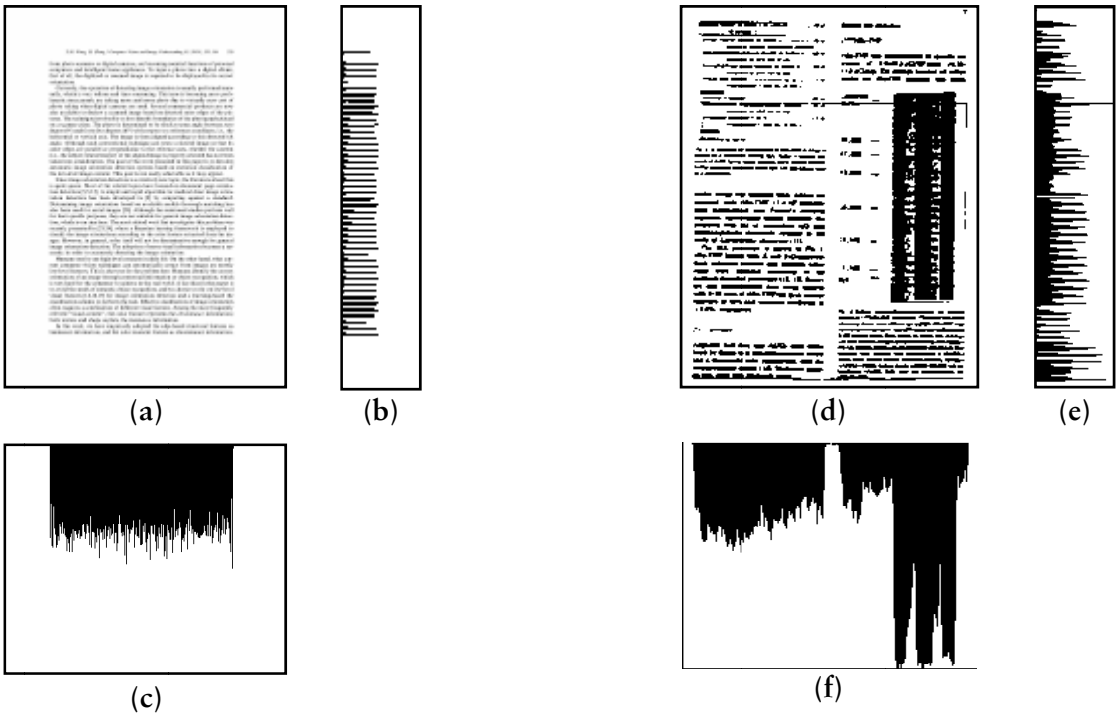


Figura 28 - *Dois exemplos do algoritmo de Akiyama: (a) imagem apenas com texto; (b) projeção horizontal; (c) projeção vertical; (d) imagem com figuras; (e) projeção horizontal; (f) projeção vertical.*

A classificação é propagada para os quadrados das camadas superiores até o topo; cada quadrado é classificado, retrato ou paisagem, de acordo com a classificação da maioria dos nove quadrados inferiores. O topo da pirâmide é constituído de nove quadrados que são utilizados para determinar a orientação da página. Para estimar o enviesamento, a sub-região com melhor classificação dos nove quadrados da última camada é selecionada. A transformada de Hough é aplicada aos pontos da última carreira de cada bloco.

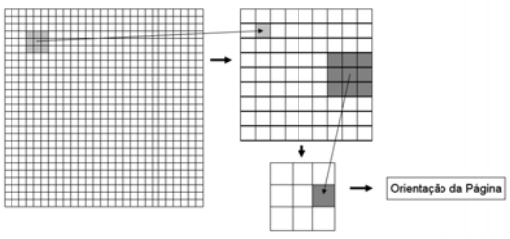


Figura 29 - *Estrutura da pirâmide para detectar a orientação retrato/paisagem.*

Os algoritmos para orientação invertida só podem ser aplicados em textos de escrita romana e números arábicos (**Figura *30***), em que são formados por letras com saliências para cima (b, d, f, h, k, l, t, A-Z, 0-9), para baixo (g, j, p, q, y) e sem saliências (a, c, e, i, m, n, o, r, s, u, v, w, x, z). As letras com saliência para cima, para baixo e sem saliências representam 69%, 8% e 23% do total, respectivamente. A idéia dos algoritmos para orientação invertida é explorar o fato de não só existirem mais letras com saliência para cima do que para baixo, mas também que tais caracteres são mais freqüentes em textos em relação aos caracteres com saliência para baixo.

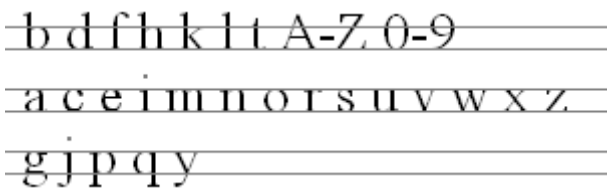


Figura 30 - *Caracteres alfanuméricos agrupados de acordo com suas saliências para cima (primeira linha), para baixo (última linha) e sem saliências (linha do meio).*

Bloomberg *et al.* [12] utilizou técnicas de diminuição de resolução e operações morfológicas para detectar orientação invertida. Ele ressalva que seu método também detecta orientação retrato/paisagem. Primeiro, a imagem tem resolução reduzida em quatro vezes (cinza escuro) e, depois, é dilatada horizontalmente com largura sete (cinza claro), na tentativa de juntar os caracteres de uma palavra e as palavras de uma linha de texto, no sentido horizontal (**Figura *31***b).

Para identificar as letras com saliência para cima e para baixo, quatro estruturas de vizinhança são utilizadas (**Figura *31***c). As duas estruturas na esquerda são utilizadas para contar os pixels na esquerda e na direita das letras com saliência para cima e; as duas estruturas da direita são utilizadas para contar os pixels da esquerda e da direita das letras com saliência para baixo. Os quadrados em cinza escuro referem-se aos pixels pretos dilatados; em cinza claro referem-se aos pixels de qualquer cor; em branco referem-se aos pixels brancos; em vermelho e azul referem-se ao centro da estrutura em que os pixels das letras com saliência para cima e para baixo, respectivamente, devem estar.

Quando um pixel satisfizer a uma destas estruturas, ele é devidamente marcado de acordo com a estrutura utilizada (**Figura *31***d).

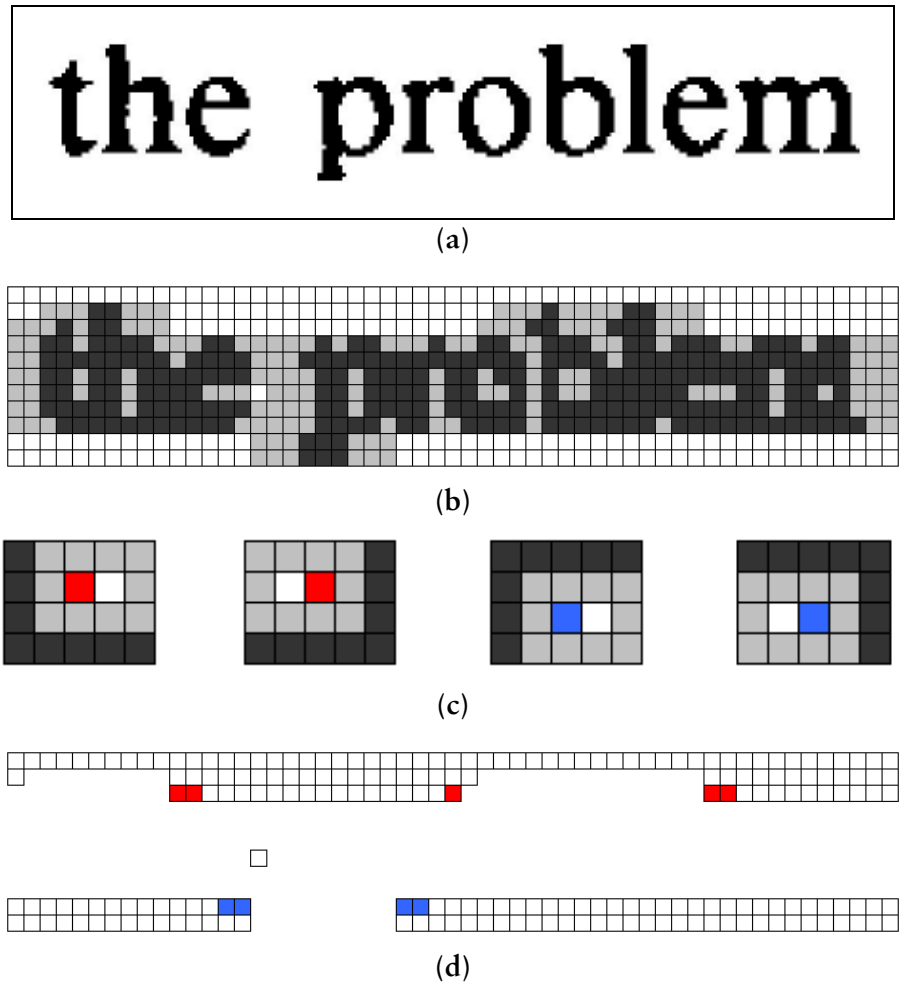


Figura 31 - *Exemplo de orientação invertida proposto por Bloomberg: (a) imagem original; (b) imagem reduzida e dilatada horizontalmente; (c) estruturas utilizadas para contar os pixels para cima e para baixo; (d) pixels detectados com as estruturas do item c.*

Os pixels são contados e os dados estatísticos da diferença entre os pixels das letras com saliência para cima e para baixo são calculados utilizando a *Lei dos Grandes Números*. Para determinar a orientação retrato/paisagem, a imagem reduzida deve ser dilatada verticalmente e as quatro estruturas utilizadas para contar os pixels para cima e para baixo devem ser rotacionadas em 90°.

4.4.3. Sumário

Um resumo com os algoritmos estudados neste capítulo é organizado segundo as principais características de cada um. Cattoni *et al.* [14] apresenta um sumário mais completo e detalhado com outros métodos. O sumário do capítulo para algoritmos de enviesamento é apresentado na **tabela 6**.

Tabela 6 - *Resumo dos algoritmos de enviesamento apresentados neste capítulo.*

Abordagem	Autores	Faixa de ângulo / Precisão	Tipo de documento	Comentários
Projeção de Perfil (*Projection Profile*)	Postl [43]	±45° / 0,6°	Documentos complexos; texto com direção dominante	
	Baird [9]	±15° / 0,05°	Documentos com pré-dominância de textos com direção dominante	O autor considera o seu método o mais rápido e preciso de todos
	Ciardiello *et al.* [18]	±45° / 0,7°	Documentos complexos	
	Ishitani [28]	±30° / 0,12°	Documentos complexos e com poucas linhas de texto	
Transformada de Hough (*Hough Transform*)	Srihari e Govindaraju [49]	±90° / 1°	Documentos com apenas textos	
	Hinds *et al.* [25]	±15° / 0,5°	Documentos complexos	Uma estimativa da altura máxima dos caracteres é necessária
	Le *et al.* [30]	±15° / 0,5°	Documentos complexos	
	Pal e Chaudhuri [41]	±45° / 0,2°	Documentos complexos com predominância de textos; texto com apenas uma direção; símbolos romanos	
	Amin e Fisher [3]	±45°	Documentos complexos; texto com direção dominante	
Vizinho mais Próximo (*Nearest Neighbor*)	Hashizume *et al.* [24]	±90° / 5°	Documentos simples	Espaços entre linhas de texto devem ser maiores que o espaço entre os caracteres
	O'Gorman [38]	±90°	Documentos complexos; múltiplas direções	Generalização do método de Hashizume
	Smith [48]	±15° / 0,05°	Documentos complexos; texto com apenas uma direção	

Os algoritmos de orientação são menos freqüentes na literatura e, portanto, apenas três técnicas foram apresentadas. O sumário do capítulo é: três algoritmos para orientação retrato/paisagem e um para orientação invertida. Vale ressaltar que para a orientação invertida, os caracteres do documento devem ser romanos e possuírem letras com e sem saliência para a detecção. O resumo é apresentado na **tabela 7**.

Tabela 7 - *Resumo dos algoritmos de orientação apresentados neste capítulo.*

Orientação	Autor	Abordagem	Tipo de documento	Comentários
Paisagem/Retrato	Akiyama e Hagita [1]	Variância global das linhas de texto	Documentos complexos	Método sensível a documentos com imagens
Paisagem/Retrato	Le *et al.* [30]	Variância local das linhas de texto de cada sub-região	Documentos complexos	Tenta resolver o problema de Akiyama e Hagita
Paisagem/Retrato e Invertida	Bloomberg *et al.* [12]	Operações Morfológicas	Documentos complexos com predominância de texto	

4.5. Algoritmo Proposto de Enviesamento e Orientação Utilizando Vizinho Mais Próximo

Um algoritmo de enviesamento e orientação é proposto e as suas características são descritas, bem como as idéias utilizadas na formulação do método. Por último, o algoritmo é detalhado na forma de pseudocódigo.

4.5.1. Características

O algoritmo proposto [6] pertence à classe do Vizinho mais Próximo utilizando uma abordagem *bottom-up*, iniciando pelo agrupamento de pixels vizinhos e terminando com a formação de linhas de texto.

Este algoritmo é utilizado para detecção de orientação retrato/paisagem, invertida e enviesamento ao mesmo tempo, portanto, trata-se do primeiro algoritmo integrado conhecido na literatura capaz de detectar qualquer rotação de um documento.

Ele também é o primeiro algoritmo da abordagem Vizinho mais Próximo com uma complexidade de execução linear em relação ao número de blocos, ao contrário dos outros que são quadráticos.

O algoritmo foi desenvolvido seguindo os critérios e as prioridades para técnicas de orientação e enviesamento discutidos no capítulo 2. Em relação à robustez, este método é capaz de detectar corretamente documentos: (1) com diferentes tipos de "layout", como por exemplo, múltiplas colunas, com gráficos, com tabelas, com outras linhas de texto em múltiplas direções, com diferentes fontes e tamanhos; (2) com escritas que utilizam diferentes símbolos, como por exemplo, a inglesa e a japonesa; vale ressaltar que as escritas orientais possuem linhas de texto na direção vertical e, desta forma, mudam a referência da orientação paisagem/retrato, assim como, não apresentam características suficientes (maiúsculas e minúsculas) para detectar a orientação invertida; (3) com diferentes formas de escrever os símbolos, como por exemplo, a datilografada e a manuscrita com letra de forma. A suposição feita neste algoritmo é cada componente representa uma letra. No caso de textos em manuscrito com letra corrida, um componente representa uma palavra, portanto, o algoritmo proposto não é recomendado para este caso. O método caracteriza-se por não utilizar nenhum tipo de parâmetro, tornando-o mais robusto e automatizado. Em relação à precisão, o algoritmo foi desenvolvido para obter uma precisão de 0,1°. Apenas a característica eficiência não foi priorizada no desenvolvimento para não haver perdas na robustez e precisão, contudo, todas as otimizações possíveis foram implementadas.

4.5.2. Idéia Básica do Algoritmo

A idéia básica deste algoritmo é utilizar as linhas de texto como ponto de referência garante a detecção de enviesamento e orientação retrato/paisagem para qualquer documento. Fazer uso das características dos símbolos das linhas de texto torna possível o cálculo da orientação invertida para documentos com textos formados por caracteres romanos.

O pré-processamento necessário é a nomeação de componentes e a remoção de ruído. Em seguida, para cada componente não marcado, localizam-se seus vizinhos e agrupam-nos na tentativa de formar uma linha de texto. Para cada agrupamento, constroem-se duas retas utilizando o método do mínimo quadrado (MMQ): (1) para linhas de texto na direção horizontal, a primeira reta é formada a partir dos pontos médios superiores da caixa de cada elemento do agrupamento e a segunda com o pontos médios inferiores; (2) para linhas de texto na direção vertical, constrói-se uma reta com os pontos médios esquerdos e outra reta com os pontos médios direitos. As duas retas formadas acima

possuem características ideais para a detecção da rotação. Para o resto do capítulo, as explicações referem-se às linhas de texto na direção horizontal; para as linhas na direção vertical, a idéia é a mesma.

A idéia é se aproveitar do fato das letras sem saliência serem mais freqüentes em textos. O MMQ serve para minimizar o efeito das letras com saliências e aproximar as retas para as letras sem saliência. Isto faz com que uma das retas passe por cima das letras sem saliência e no meio das letras com saliência para cima; e a outra reta passará em baixo das letras sem saliência no meio das letras com saliência para baixo. Estas duas retas são geralmente paralelas entre si e, então, o quadrado do número de elementos da linha de texto é acumulado em um histograma de ângulos entre ±90°. Este procedimento é realizado para todas as linhas de textos formadas no documento. O pico do histograma refere-se ao enviesamento e à orientação retrato/paisagem do documento.

A outra idéia é a mesma utilizada pelos algoritmos de orientação invertida: existem mais letras com saliência para cima do que para baixo, além de serem mais freqüentes. Utilizando as mesmas duas retas construídas anteriormente, para cada componente da linha de texto, o quadrado da distância do ponto mais acima para a reta superior e o quadrado da distância do ponto mais baixo para a reta inferior são acumuladas em duas variáveis globais *up* e *down*, respectivamente. Este procedimento é executado para todas as linhas de texto formadas no documento e, se *down* for maior que *up*, então a orientação é invertida.

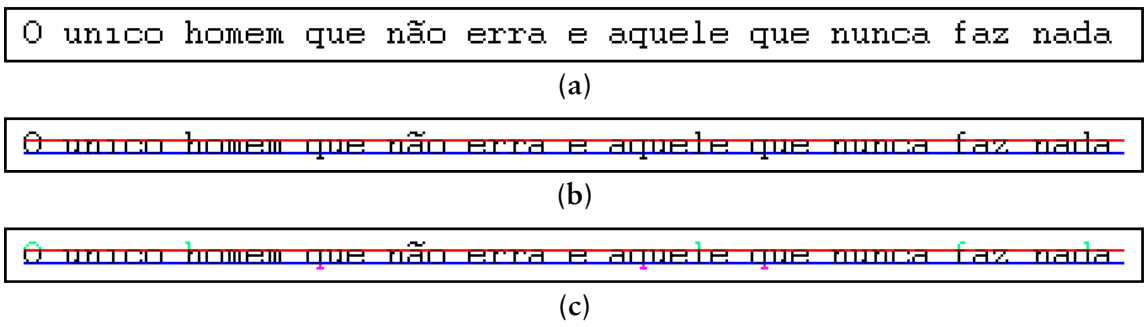


Figura 32 - *Exemplo do algoritmo proposto: (a) linha de texto agrupado pelo algoritmo; (b) a reta superior, em vermelho, e a reta inferior, em azul, formados pelo MMQ aplicado aos pontos médios superiores e inferiores, respectivamente; (c) letras com saliências para cima, em verde, e letras com saliência para baixo, em rosa.*

No exemplo da **figura 32**a, existem 43 letras romanas, entre eles, 5 letras com saliência para cima, 2 letras com saliência para baixo e 36 letras sem saliência. As letras são agrupadas em uma linha de texto e duas retas são construídas utilizando o MMQ aplicado: aos pontos médios superiores de cada bloco formando a reta superior, em vermelho; aos

pontos médios inferiores de cada bloco formando a reta inferior, em azul (**Figura *32***b). O valor 3.698 (2x432 – duas retas com 43 elementos) é acumulado no histograma no ângulo 0º. O pico do histograma representa a orientação retrato/paisagem e o ângulo de enviesamento, neste caso, em 0º. Para detectar a orientação invertida, o quadrado da distância do ponto mais acima de cada uma das 5 letras com saliência para cima à reta superior, em verde na **figura *32***c, é acumulada na variável *up*; o quadrado da distância do ponto mais abaixo de cada uma das 3 letras com saliência para baixo à reta inferior, em rosa na **figura *32***c, é acumulada na variável *down*. Neste caso, a variável *up* é maior que *down*, logo, a orientação não está invertida.

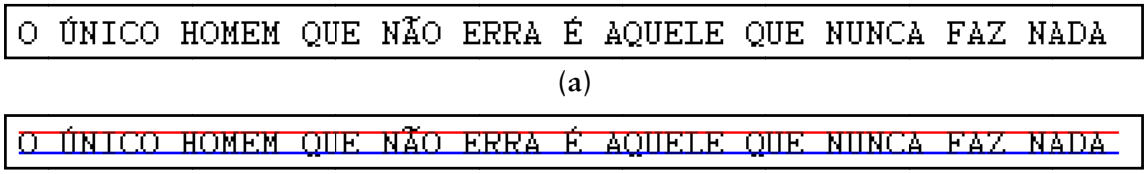


Figura 33 - *Outro exemplo do algoritmo proposto: (a) linha de texto apenas com letras maiúsculas; (b) as duas retas formadas utilizando o MMQ nos pontos médios superiores, em vermelho, e nos pontos médios inferiores, em azul.*

Para linhas de texto com símbolos de dimensões semelhantes, como por exemplo, linhas de texto apenas com letras maiúsculas (**Figura *33***a), o algoritmo funciona corretamente apenas para a orientação retrato/paisagem e enviesamento. Para orientação invertida, não faz sentido a detecção uma vez que as dimensões das letras são próximas (**Figura *33***b).

4.5.3. Algoritmo

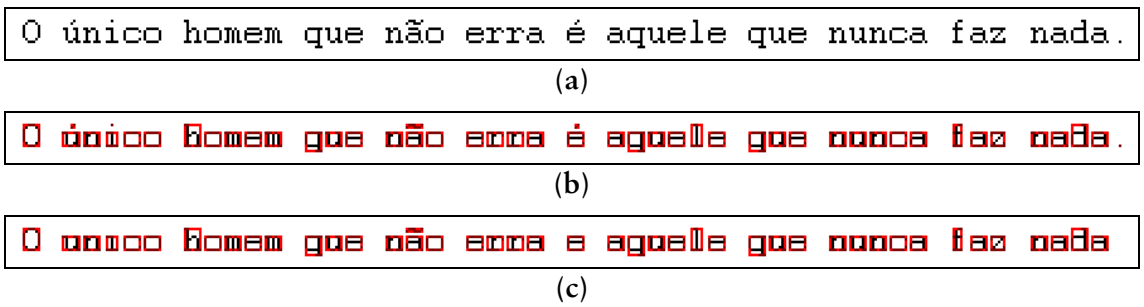
Nesta seção, o algoritmo é apresentado na forma de pseudocódigo e alguns detalhes serão explicitados a seguir.

Os passos principais do algoritmo são: (1) executar a nomeação de componentes 8×8; (2) remover de ruídos; (3) agrupar uma linha de texto; (4) detectar enviesamento e orientação retrato/paisagem da linha de texto; (4) detectar orientação invertida da linha de texto; (5) detectar rotação do documento. O pseudocódigo do procedimento principal é:

```
blocos = nomeacao_componentes (imagem);
remover_ruido (blocos);
inicializar (histograma_grosso, histograma_fino, 0);
inicializar (up, down, 0);

for each bloco não marcado em blocos do
begin
     {vizinho, direcao} = localizar_vizinho_mais_proximo (bloco);
     if (vizinho não for nulo) then
     begin
          {linha, reta_superior, reta_inferior}
             = agrupar_linha_texto (bloco, vizinho, direcao);

          detectar_enviesamento_retrato_paisagem
             (linha, reta_superior, reta_inferior,
              histograma_grosso, histograma_fino);

          detectar_invertida (linha, direcao, reta_superior,
                              reta_inferior, up, down);
     end;
end;

angulo_total = detectar_rotacao (histograma_grosso, histograma_fino,
                                 up, down);
```

Pseudocódigo 3 – *Procedimento principal do algoritmo de detecção de orientação e enviesamento.*

O primeiro passo é executar os algoritmos de pré-processamento nomeação de componentes 8×8 e remoção de ruídos (**Figura *34***). Em seguida, a partir de cada bloco não marcado (bloco), um vizinho mais próximo (localizar_vizinho_mais_proximo) é localizado e uma linha de texto (agrupar_linha_texto) é formada na direção detectada. O enviesamento e a orientação (detectar_enviesamento_retrato_paisagem e detectar_invertida) são calculados a partir da linha de texto formada. Quando todos os blocos estiverem marcados, o ângulo total da rotação é calculado (detectar_rotacao).



(a)

(b)

(c)

Figura 34 - *Exemplo do pré-processamento do algoritmo proposto: (a) linha de texto original; (b) caixa dos blocos formados na nomeação dos componentes, em vermelho; (c) componentes classificados como ruído foram removidos.*

A partir do bloco selecionado (bloco), o vizinho mais próximo (vizinho) é localizado e determinado a direção formada entre eles (direcao). O pseudocódigo da função para localizar o vizinho mais próximo é:

```
function localizar_vizinho_mais_proximo (bloco): {vizinho, direcao};
begin
     inicializar(raio, 1);
     while (vizinho não encontrado) do
     begin
          vizinho = procurar_bloco (raio, esquerda);
          if (vizinho não for nulo) then return {vizinho, HORIZONTAL};
          vizinho = procurar_bloco (raio, direita);
          if (vizinho não for nulo) then return {vizinho, HORIZONTAL};
          vizinho = procurar_bloco (raio, em_cima);
          if (vizinho não for nulo) then return {vizinho, VERTICAL};
          vizinho = procurar_bloco (raio, em_baixo);
          if (vizinho não for nulo) then return {vizinho, VERTICAL};
          incrementar(raio, 1);
     end;
end;
```

Pseudocódigo 4 – *Localizar vizinho mais próximo do bloco atual.*

O vizinho (vizinho) é localizado procurando-se por um pixel preto dele ao redor do bloco atual (bloco) em um raio (raio) incrementado a cada iteração (**Figura *35***). Se um vizinho for encontrado no lado esquerdo ou direito, a direção (direcao) é horizontal; caso tenha sido encontrado em cima ou em baixo, a direção é vertical.

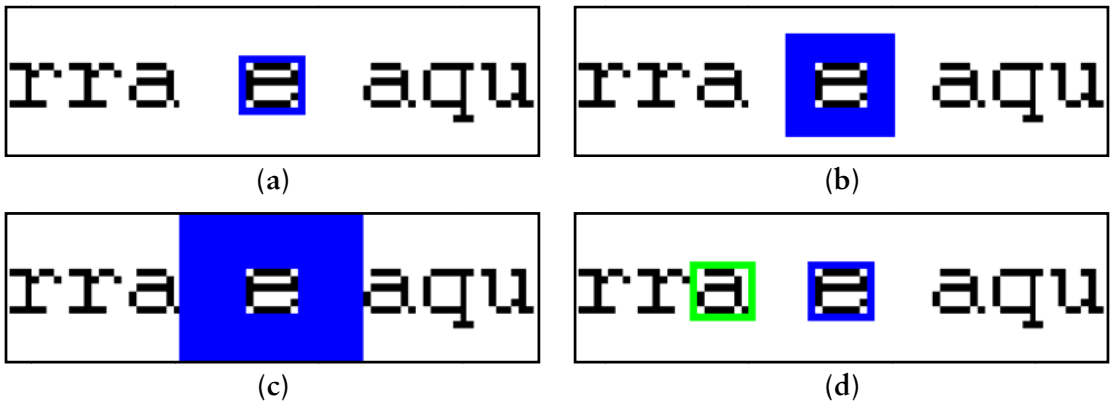


Figura 35 - *Exemplo de localizar o vizinho mais próximo: (a) sub-região do exemplo anterior com o bloco atual selecionado, em azul; (b) a quarta iteração para localizar o vizinho, em azul; (c) a penúltima iteração; (d) o vizinho, em verde, localizado na esquerda, portanto, a direção é horizontal.*

O próximo passo é utilizar o bloco atual (bloco) e o vizinho (vizinho) para iniciar o agrupamento de uma linha de texto (linha). O pseudocódigo do agrupamento da linha de texto é apresentado a seguir (**Pseudocódigo *5***).

```
function agrupar_linha_texto (bloco, vizinho, direcao):
   {linha, reta_superior, reta_inferior};
begin
     inicializar (linha, {bloco, vizinho});
     inicializar (faixa_altura, altura(bloco), altura(vizinho));
     inicializar (faixa_largura, largura(bloco), largura(vizinho));
     inicializar (faixa_distancia, distancia(bloco, vizinho));
     while (existir vizinho) do
     begin
          if (direcao = HORIZONTAL) then
          begin
               reta_superior = metodo_minimo_quadrado
                                  (linha, ponto_medio_superior);
               reta_inferior = metodo_minimo_quadrado
                                 (linha, ponto_medio_inferior);
          end
          else if (direcao = VERTICAL) then
          begin
               reta_superior = metodo_minimo_quadrado
                                  (linha, ponto_medio_esquerda);
               reta_inferior = metodo_minimo_quadrado
                                  (linha, ponto_medio_direita);
          end;

          {vizinho_inicio, vizinho_fim}
             = procurar_vizinhos (linha, reta_inferior, reta_superior,
                                  faixa_altura, faixa_largura,
                                  faixa_distancia);

          inserir_no_inicio (linha, vizinho_inicio);
          inserir_no_fim (linha, vizinho_fim);

          atualizar (faixa_altura, altura(vizinho_inicio),
                     altura(vizinho_fim));
          atualizar (faixa_largura, largura(vizinho_inicio),
                     largura(vizinho_fim));
          atualizar (faixa_distancia,
                     distancia(vizinho_inicio, first(linha)),
                     distancia(vizinho_fim, last(linha)));
     end;
     marcar (linha);
     return {linha, reta_superior, reta_inferior};
end;
```

Pseudocódigo 5 – *Agrupamento da linha de texto.*

A linha de texto (linha) é uma seqüência e, inicialmente, possui dois elementos: o bloco atual (bloco) e o vizinho mais próximo (vizinho). Em seguida, duas retas são construídas, reta_superior e reta_inferior, utilizando o método do mínimo quadrado aplica

aos pontos médios superiores e inferiores, respectivamente (Figura 36b). Estas duas retas formam uma faixa delimitando uma sub-região na imagem para a procura dos próximos dois vizinhos.

Em um sentido (Figura 36c), inicia-se a busca por um pixel de outro vizinho (vizinho_fim); em sentido contrário (Figura 36g), inicia-se outra busca por um pixel de outro vizinho (vizinho_inicio). Os vizinhos, vizinho_inicio e vizinho_fim, são inseridos no inicio e no fim da linha de texto (linha), respectivamente. Se não existirem dois novos vizinhos, a iteração pára. Ao final do agrupamento, cada elemento da linha de texto (linha) é marcado e é retornada a linha (linha), a reta superior (reta_superior) e inferior (reta_inferior) da última iteração.

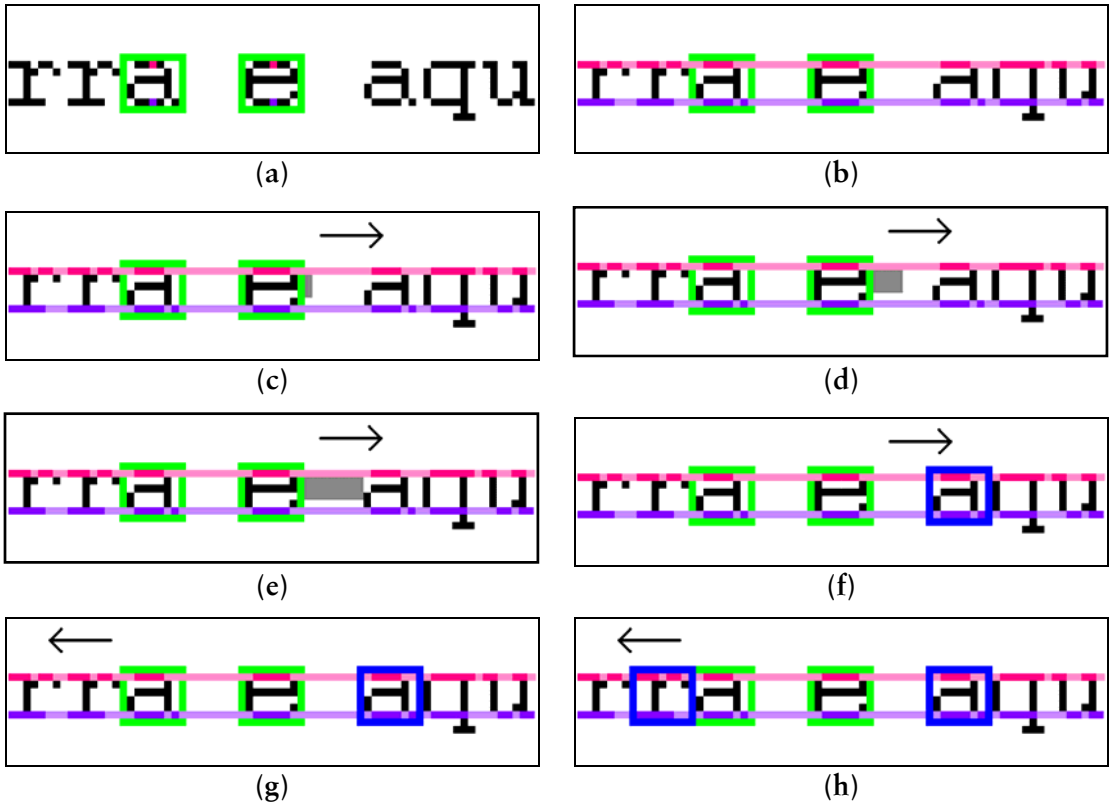


Figura 36 - Exemplo do processo de agrupamento de linha de texto: (a) os dois primeiros elementos da linha de texto, bloco e vizinho, e os pontos médios superiores e inferiores em rosa e roxo, respectivamente; (b) as retas reta_superior e reta_inferior, em rosa e roxo, respectivamente; (c) primeira iteração a procura de um vizinho no sentido da direita; (d) quarta iteração; (e) penúltima iteração; (f) novo vizinho localizado (vizinho_fim), em azul; (g) o mesmo procedimento é executado para a esquerda; (h) na primeira iteração, o novo vizinho é encontrado (vizinho_inicio), em azul.

Existem três critérios a serem considerados para permitir a adição de um novo vizinho à linha de texto: altura, a largura e a distância entre dois vizinhos adjacentes. Na primeira iteração, as faixas de valores de cada critério são ajustadas de acordo com os atributos dos dois primeiros elementos da linha: bloco e vizinho. Nas iterações seguintes, a cada inserção de um novo vizinho, as faixas de valores são atualizadas. A média de cada atributo é utilizada a cada iteração para definir uma nova faixa de valores. A faixa da distância varia entre zero e duas vezes a média da distância entre dois componentes adjacentes da linha de texto. A faixa da altura e da largura varia entre a metade da média e três vezes a média. Os critérios relacionados à altura e à largura têm o objetivo de evitar a adição de um componente que representa uma imagem, uma linha, uma tabela, um fragmento de caractere ou qualquer outro componente que não tenha as dimensões semelhantes aos elementos da linha de texto que está sendo formada. O critério da distância tem a finalidade de evitar que linhas de texto de duas colunas sejam unidas ou que um componente mais distante seja unido à linha de texto atual. Além disso, determinar uma faixa de distância acelera a procura dos dois próximos vizinhos (procurar_vizinhos), uma vez que delimita a sub-região já formada pelas duas retas (reta_superior e reta_inferior).

O próximo passo é detectar o enviesamento e orientação retrato/paisagem. O quadrado do número de elementos da linha de texto formada (linha) é acumulado nos histogramas (histograma_grosso e histograma_fino) em cada um dos ângulos formados pelas duas retas (reta_superior e reta_inferior) construídas no passo anterior. O histograma_grosso é utilizado para determinar o ângulo de rotação com uma precisão de 1º; e o histograma_fino para o ângulo de rotação com precisão 0,1º. O pseudocódigo deste procedimento é:

```
procedure detectar_enviesamento_retrato_paisagem
   (linha, reta_superior, reta_inferior,
    histograma_grosso, histograma_fino);
begin
     angulo_superior = desvio_angular(reta_superior);
     angulo_inferior = desvio_angular(reta_inferior);
     valor = (#linha)²;
     acumular (histograma_grosso, histograma_fino, angulo_superior,
               angulo_inferior, valor);
end;
```

Pseudocódigo 6 – *Detectar enviesamento e orientação retrato/paisagem.*

Em seguida, a orientação invertida é detectada para a linha de texto. Na verdade, os quadrados das distâncias do ponto mais longe de cada reta são acumulados nas variáveis *up*

e *down*, para que ao final de todo o processamento do documento, poder detectar a orientação invertida. O pseudocódigo deste passo é apresentado a seguir (**Pseudocódigo** *7*).

Ao final de todo o documento ser processado e todos os componentes estarem marcados, a detecção do enviesamento e orientação retrato/paisagem é realizado procurando o pico do histograma_grosso e atribuído à variável angulo_grosso. Para selecionar um ângulo com a precisão de 0,1º (angulo_total), seleciona-se o pico no histograma_fino apenas entre a faixa de ±1º em relação ao angulo_grosso. Para a orientação invertida, se down for maior que up, então, adiciona 180º ao angulo_total. Vale ressaltar que os histogramas não precisam ser suavizados. O pseudocódigo deste último passo é apresentado a seguir (**Pseudocódigo** *8*).

```
procedure detectar_invertida (linha, direcao, reta_superior,
                              reta_inferior, up, down);
begin
     for each bloco de linha do
     begin
          if direcao = HORIZONTAL then
          begin
               ponto_superior = procurar_ponto_superior (bloco);
               ponto_inferior = procurar_ponto_inferior (bloco);
          end
          else if direcao = VERTICAL then
               ponto_superior = procurar_ponto_esquerdo (bloco);
               ponto_inferior = procurar_ponto_direito (bloco);
          end;

          distancia_superior = distancia (ponto_superior,
                                          reta_superior);
          distancia_inferior = distancia (ponto_inferior,
                                          reta_inferior);

          acumular (up, distancia_superior²);
          acumular (down, distancia_inferior²);
     end;
end;
```

Pseudocódigo 7 – *Detectar orientação invertida.*

```
function detectar_rotacao (histograma_grosso, histograma_fino, up,
                           down): angulo_total;
begin
     angulo_grosso = procurar_pico (histograma_grosso);
     angulo_total  = procurar_pico (histograma_fino, angulo_grosso);
     if down > up then
          incrementar (angulo_total, 180);
     return angulo_total;
end;
```

Pseudocódigo 8 – *Detectar rotação do documento.*

4.6. Algoritmo Proposto de Enviesamento Utilizando Aproximação da Média

Outro algoritmo de enviesamento é proposto e as suas características são descritas, bem como as idéias utilizadas na formulação do método. Por último, o algoritmo é detalhado na forma de pseudocódigo.

4.6.1. Características

O algoritmo proposto pertence à classe Aproximação da Média e foi sugerido inicialmente por Lins e Ávila [32].

A principal suposição é que determinada região do documento deve possuir elementos suficientes para guiar o processo de detecção sem ter que analisar o documento como um todo. Portanto, seguindo os critérios e prioridades estabelecidas, este algoritmo diminui a sua robustez fazendo suposições acerca do tipo de documento com a finalidade de aumentar a eficiência da detecção. Este algoritmo é recomendado para formulários e documentos com grande quantidade de linhas de texto.

No entanto, o método proposto tem a capacidade de detectar o ângulo de enviesamento em uma faixa de ±45°. Em relação à precisão, o algoritmo foi desenvolvido para obter uma precisão de 0,1°.

4.6.2. Idéia Básica do Algoritmo

O primeiro passo do algoritmo proposto é a escolha dos pontos de uma determinada região em que se supõe que possui elementos suficientes para guiar o processo de detecção. Este passo é critico, pois uma má escolha acarretará no cálculo incorreto do ângulo de enviesamento do documento. No método inicialmente proposto em [32], os pontos escolhidos eram os primeiros de cada linha no sentido da direita para a esquerda. Para melhorar a precisão e diminuir o erro de detecção, os pontos podem ser também selecionados de baixo para cima, um para cada coluna da imagem, ao contrário da versão inicial. Esta sugestão é justificada pelo motivo de que a parte inferior das linhas de texto corresponder melhor ao enviesamento do documento. A seleção desses pontos também representa um aumento no desempenho do algoritmo, já que há menos pontos a serem processados.

No método inicialmente proposto em [32], utilizou-se um parâmetro de profundidade limitar as distâncias dos pontos selecionados (**Figura 38**a). Outra sugestão é o

uso de uma sub-região é adotado de forma a evitar a influência dos pontos selecionados em outras linhas de texto. A sub-região tem altura igual à resolução da imagem (ex. para uma imagem de resolução de 200 dpi, a altura da janela é 200 pixels) e a mesma largura da imagem. A sub-região com mais pontos é selecionada e utilizada nos outros passos do algoritmo.

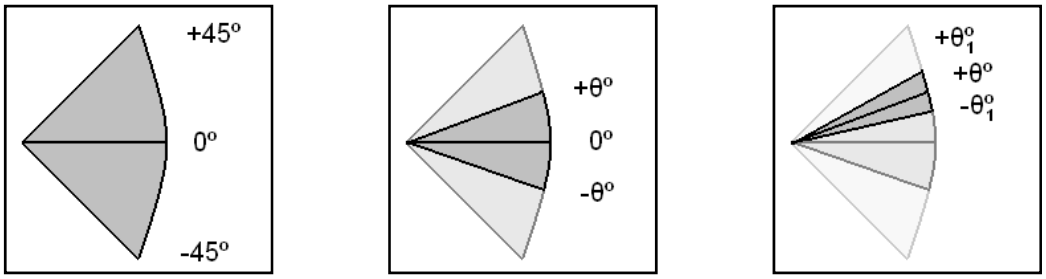


Figura 37 - *Exemplo de busca utilizando bifurcação: (a) faixa de ângulo na primeira iteração, em cinza escuro; (b) segunda iteração; (c) terceira iteração.*

O próximo passo do algoritmo é aplicar a estratégia de busca do tipo bifurcação. Esta estratégia é um dos recursos que aumentam o desempenho do algoritmo. A bifurcação consiste em definir dois limites em uma faixa de valores e, a cada iteração sucessiva, diminuir a faixa de valores aproximando os dois limites até um valor desejado (**Figura 37**). Vale a pena ressaltar que este método se diferencia das técnicas que tentam, a cada intervalo de uma determinada faixa de ângulo, maximizar uma função *Premium* aplicada a projeção do perfil da imagem.

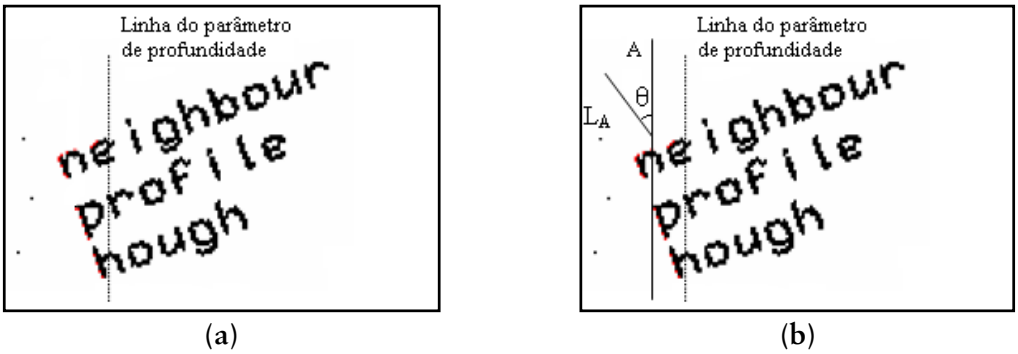
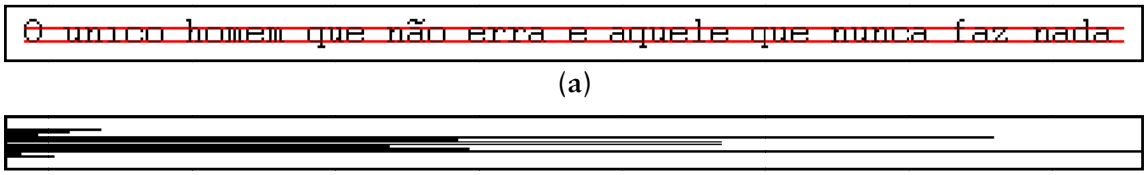


Figura 38 – *Passos do algoritmo: (a) seleção dos pontos pelo parâmetro de profundidade; (b) cálculo do ângulo theta.*

O próximo passo do algoritmo é estimar o ângulo de rotação de acordo com os pontos selecionados. O principal diferencial do algoritmo está na forma de calcular o

ângulo (*theta*) utilizado na busca com bifurcação. Após a seleção dos pontos de uma sub-região, calcula-se a média das abscissas (A) dos pontos e calcula-se outra média das abscissas (L_A) dos pontos a esquerda de A. Os pontos mais acima (P_C) e mais abaixo (P_B) de todos os pontos selecionados junto com as duas médias são utilizados para calcular o ângulo (*theta*). A cada iteração, a tendência é o ângulo (*theta*) diminuir uma vez que as médias se aproximam. Este fato justifica a classificação do algoritmo na classe Aproximação da Média.

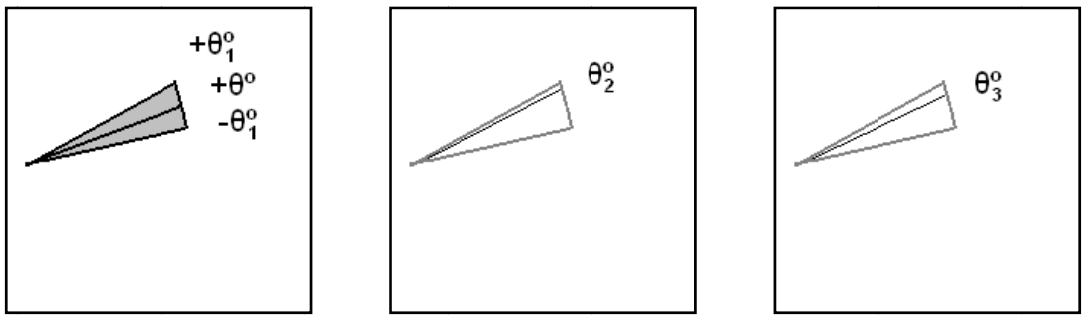


(a)

(b)

Figura 39 - *Exemplo de projeção de perfil horizontal de uma linha de texto: (a) linha de texto com duas linhas em vermelho de maior número de pixels pretos; (b) projeção de perfil horizontal apresenta dois picos quando a linha de texto não está rotacionada.*

O cálculo da moda também pode ser utilizado para decisão sobre a busca do ângulo. A projeção de perfil horizontal (**Figura *39***b) de uma linha de texto com símbolos romanos e datilografados possui dois picos quando não apresenta rotação e referem-se às linhas em vermelho na **figura *39***a. O algoritmo proposto, ao selecionar os pontos de baixo para cima, procura pelo ângulo em que o pico da linha inferior em vermelho é máximo, ou seja, a moda é máxima.



(a) (b) (c)

Figura 40 - *Exemplo de busca utilizando varredura: (a) faixa de ângulo, em cinza; (b) primeira iteração; (c) segunda iteração.*

Outra sugestão é o uso da estratégia de busca do tipo varredura com o objetivo de diminuir os erros de detecção e aumentar a precisão do enviesamento. A varredura consiste em percorrer uma faixa de valores a passos incrementados por um valor fixo, de um limite ao outro (**Figura *40***). A fase de bifurcação seria terminada após o cálculo de o ângulo resultar em um valor menor que 1° e, em seguida, a busca em varredura decidiria o melhor ângulo com uma precisão de 0,1°.

4.6.3. Algoritmo

Nesta seção, o algoritmo proposto com as modificações sugeridas é apresentado na forma de pseudocódigo e alguns detalhes serão explicados a seguir.

Os principais passos do algoritmo são: (1) selecionar os pontos de baixo para cima; (2) selecionar uma sub-região contendo a maior quantidade de pontos selecionados; (3) definir um eixo de rotação; (4) localizar o ângulo grosso com maior moda no eixo da rotação através da bifurcação; (5) localizar o ângulo fino através da varredura próximo ao ângulo grosso detectado. O pseudocódigo do procedimento principal é:

```
pontos           = selecionar_pontos_baixo_cima (imagem);
pontos_subregiao = selecionar_subregiao (imagem, pontos);
eixo_rotacao     = selecionar_eixo_rotacao (pontos_subregiao);
angulo_grosso    = localizar_angulo_bifurcacao
                       (pontos_subregiao, eixo_rotacao);
angulo_fino      = localizar_angulo_varredura
                       (pontos_subregiao, eixo_rotacao, angulo_grosso);
```

Pseudocódigo 9 – *Procedimento principal do algoritmo de detecção de enviesamento.*

O primeiro passo consiste em selecionar o primeiro pixel preto de baixo para cima de cada coluna da imagem (pontos). Em seguida, seleciona-se uma sub-região de altura igual à resolução da imagem contendo a maior quantidade de pixels selecionados no primeiro passo (**Figura *41***b).

Todos os passos seguintes utilizarão apenas os pontos selecionados no primeiro passo e que estejam contidos na sub-região selecionada (pontos_subregiao). Um eixo de rotação (eixo_rotacao) é calculado e utilizado em todo o processo (**Figura *41***c): define-se um ponto de coordenadas (x, y) que será calculado através da média das coordenadas axiais e das abscissas dos pontos, respectivamente. Em seguida, um ângulo grosso é localizado através de uma bifurcação (angulo_grosso) e, finalmente, realiza-se uma varredura nos ângulos vizinhos ao ângulo grosso em busca de um ângulo fino (angulo_fino), isto é, um ângulo de enviesamento com precisão de 0,1°.

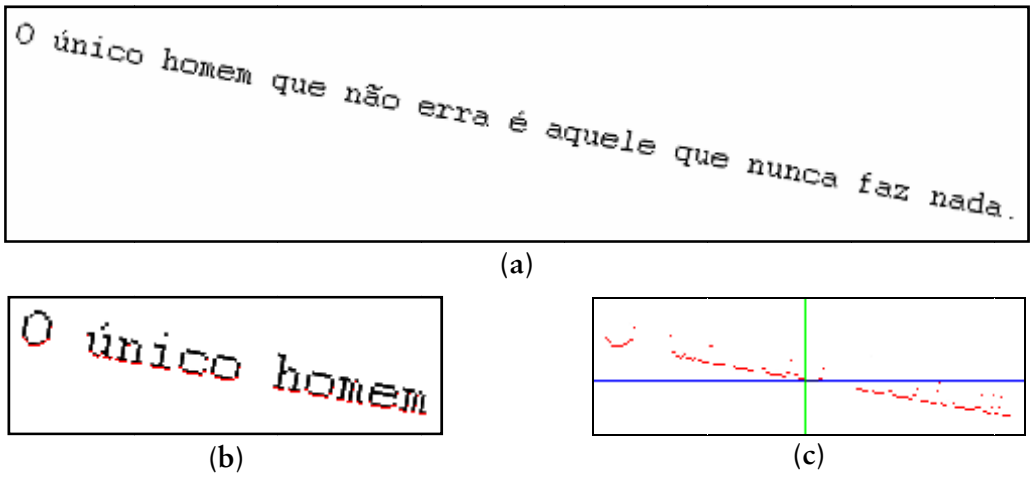


Figura 41 - *Exemplo do algoritmo proposto: (a) linha de texto rotacionada a -10°; (b) pontos selecionados de baixo para cima, em vermelho; (c) a média das coordenadas axiais e das abscissas (media), em verde e azul, respectivamente. O ponto de intersecção define o eixo de rotação.*

A localização do ângulo grosso com precisão de 1° é feita através de uma estratégia de busca do tipo bifurcação. O pseudocódigo deste passo é apresentado no pseudocódigo 10.

Neste passo, a moda dos pontos selecionados (pontos_subregiao) é utilizada como critério de decisão para a busca. Na primeira iteração, os limites iniciais são ±45° formando uma faixa de ângulos com precisão 1°. O angulo_grosso é inicializado como o ângulo médio dos dois limites, neste caso, 0°. Nas iterações sucessivas, um ângulo theta é calculado e novos limites são definidos como angulo_grosso ± theta. A moda do ângulo atual (angulo_grosso) e a moda de cada limite (angulo_grosso + theta e angulo_grosso - theta) são calculadas. Se a moda de algum dos limites for maior que a moda do angulo_grosso, então, o novo angulo_grosso é definido como o ângulo do limite de maior moda; caso contrário, o processo é parado.

O ângulo *theta* é calculado levando em conta a média das abscissas dos pontos (media) e a média das abscissas dos pontos acima da media (media_desvio). Na **figura 42**, a média, a media_desvio e o eixo_rotacao formam um triângulo que é utilizado para o cálculo de *theta*. Nas iterações seguintes, o valor de *theta* tende a diminuir por causa do fato de a media_desvio se aproximar de media, justificando a classificação do algoritmo na abordagem Aproximação da Média.

```
function localizar_angulo_bifurcacao (pontos_subregiao, eixo_rotacao):
    angulo_grosso;
begin
     inicializar (angulo_grosso, 0);
     inicializar (parar, false);
     while não parar do
     begin
          theta = calcular_angulo (pontos_subregiao);

          moda_atual    = calcular_moda (pontos_subregiao,
                                         eixo_rotacao,
                                         angulo_grosso);
          moda_positiva = calcular_moda (pontos_subregiao,
                                         eixo_rotacao,
                                         angulo_grosso + theta);
          moda_negativa = calcular_moda (pontos_subregiao,
                                         eixo_rotacao,
                                         angulo_grosso - theta);

          if moda_positiva > moda_atual and
             moda_positiva > moda_negativa then
          begin
               moda_atual = moda_positiva;
               incrementar (angulo_grosso, theta);
          end;

          if moda_negativa > moda_atual and
             moda_negativa > moda_positiva then
          begin
               moda_atual = moda_negativa;
               decrementar (angulo_grosso, theta);
          end;

          if moda_atual > moda_positiva and
             moda_atual > moda_negativa then parar = true;
     end;
     return angulo_grosso;
end;
```

Pseudocódigo 11 – *Localizar ângulo de enviesamento utilizando a estratégia de bifurcação.*

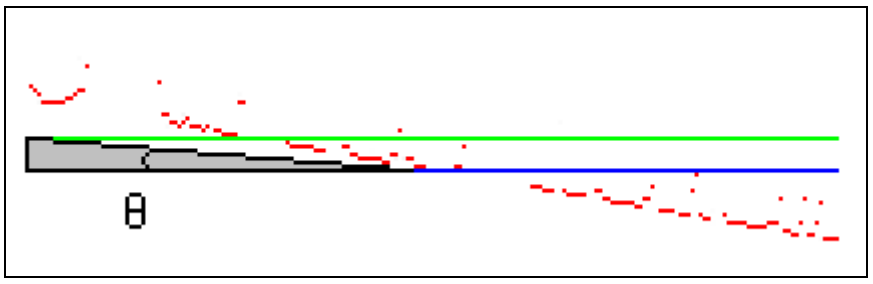


Figura 42 - *Exemplo do cálculo de theta: media, em azul, media_desvio, em verde e o triângulo utilizado no cálculo de theta, em cinza.*

Para calcular a moda, os pontos são rotacionados a um ângulo fornecido utilizando o eixo de rotação. Os pixels pretos são contados do eixo de rotação e de cada linha adjacente. O maior número de pixels entre essas linhas representa a moda dos pontos no ângulo fornecido.

O pseudocódigo do procedimento para localizar o ângulo de enviesamento com uma precisão de 0,1° a partir do ângulo grosso (angulo_grosso) utilizando uma estratégia de busca do tipo varredura é:

```
function localizar_angulo_varredura (pontos_subregiao, eixo_rotacao,
    angulo_grosso): angulo_fino;
begin
     inicializar (angulo, angulo_grosso + 1);
     inicializar (angulo_fim, angulo_total – 1);
     inicializar (moda_max, 0);
     inicializar (angulo_fino, 0);
     while angulo >= angulo_fim do
     begin
          moda = calcular_moda (pontos_subregiao, eixo_rotacao,
                                angulo);
          if moda > moda_max then
          begin
               moda_max = moda;
               angulo_fino = angulo;
          end;
          decrementar (angulo, 0.1°);
     end;
     return angulo_fino;
end;
```

Pseudocódigo 12 – *Localizar ângulo de enviesamento através da estratégia de varredura.*

A partir do ângulo grosso (angulo_grosso) detectado, uma faixa de ângulo é definida como angulo_grosso ± 1° com uma precisão de 0,1°. O ângulo da primeira iteração (angulo) é inicializado com o valor do limite superior. Enquanto o ângulo atual (angulo) não atinge o limite inferior, a moda em cada ângulo é calculada e o ângulo atual é decrementado em 0,1°. O ângulo, cuja moda é a maior, refere-se ao ângulo fino (angulo_fino) ou ângulo de enviesamento.

4.7. Resultados dos Testes

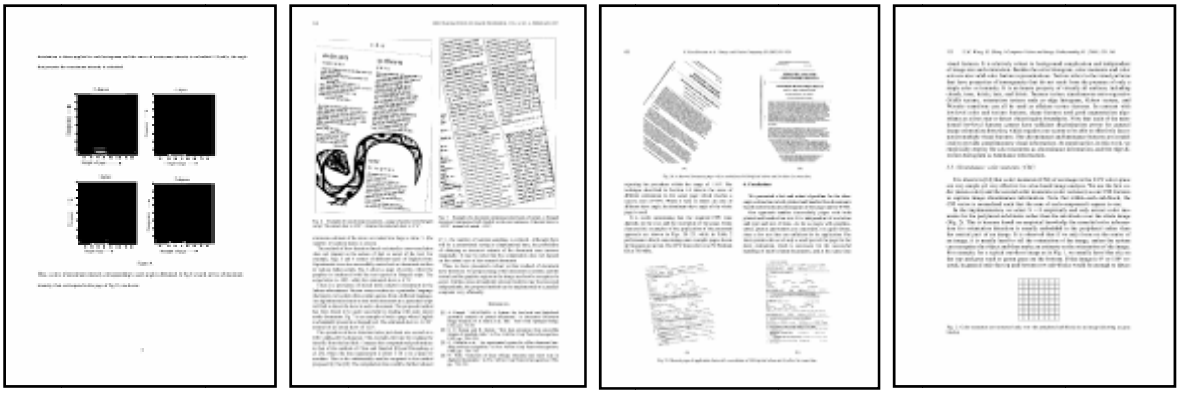
Uma série de testes comparativos entre algoritmos de enviesamento e orientação foi realizada seguindo os critérios definidos com o objetivo de avaliar as características de cada método em vários contextos. Primeiro, a metodologia utilizada nos testes é detalhada e, em seguida, os resultados são analisados. O algoritmo proposto de detecção de enviesamento

(seção 4.6) não utilizado nas comparações e os resultados dos testes são apresentados separadamente ao final desta seção.

4.7.1. Metodologia

Para o ambiente de execução dos testes, utilizou-se um computador com processador Pentium IV de 2.4 GHz, memória RAM de 512 MB e sistema operacional Windows XP. Os algoritmos foram desenvolvidos utilizando a linguagem ANSI-C e o compilador do Visual C++ 6.0 da Microsoft. As imagens foram armazenadas e lidas no formato TIFF utilizando compressão CCITT Group 4.

Assume-se apenas que os documentos digitalizados são: (1) monocromáticos, uma vez que a quantidade de cores pode ser facilmente reduzida com a binarização e, portanto, evita a influência de outras cores e simplifica os testes; (2) 200 dpi, uma vez que resolução guarda todos os elementos essenciais do documento oferecendo um bom fator qualidade/espaço de armazenamento [36] e transmissão via rede de computadores [35]. Os resultados dos testes serão analisados segundo o conjunto de critérios definidos.

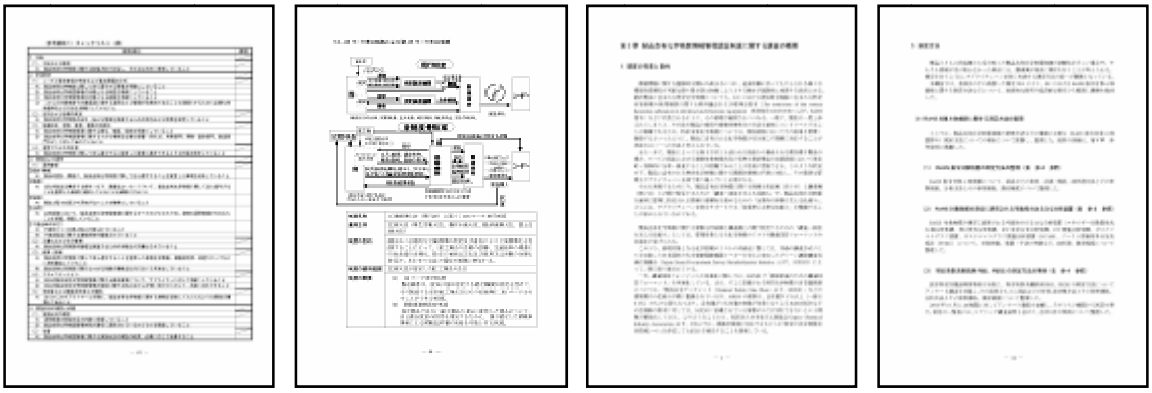


(a) (b) (c) (d)

Figura 43 - *Exemplos de tipos de "layout" utilizado nos testes: (a) documento com poucas linhas e com imagens; (b, c) texto separado em duas colunas e com linhas de texto em outras direções; (d) documento com uma coluna.*

Um algoritmo de detecção de rotação deve ter uma dependência mínima em relação ao "layout" dos documentos e, desta forma, 270 documentos de tamanho A4, retirados de artigos científicos, escritos em inglês e feitos no computador com os mais diversos tipos de "layout" foram utilizados nos testes (**Figura *43***).

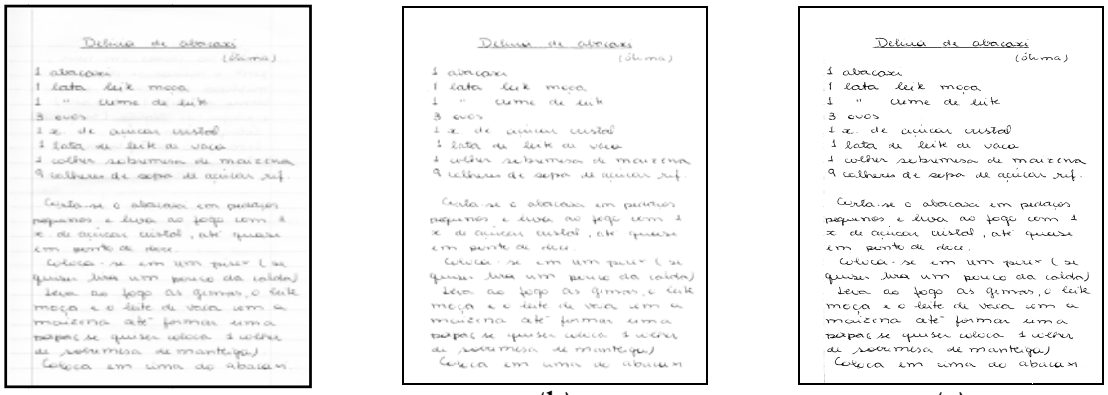
Em relação ao critério de dependência mínima da língua em que o texto foi escrito, 50 documentos em japonês com texto escrito horizontalmente da esquerda para a direita, de tamanho A4 e com vários tipos de "layout" foram testados (**Figura *44***).



(a) **(b)** **(c)** **(d)**

Figura 44 - *Exemplo de documentos em inglês utilizados nos testes: (a) documento com tabela; (b) documento com imagens; (c, d) texto em uma coluna.*

Em relação ao critério de dependência mínima da forma em que o texto foi escrito, 50 documentos em português escritos à mão em letra corrida foram coletados, tratadas e utilizadas nos testes. Os documentos foram digitalizados em tons de cinza, a rotação foi corrigida manualmente e guiado pelas linhas do papel e, em seguida, utilizando o Adobe Photoshop CS 8.0 [55], aplicaram-se os filtros de *posterize* com parâmetro 3 e de binarização (figura 43). As linhas do papel conectam as palavras de modo a formarem apenas um bloco após a nomeação de componentes, degradando os dois algoritmos e, desta forma, foram removidos aplicando-se os filtros citados.



(a) **(b)** **(c)**

Figura 45 - *Exemplo do tratamento do documento manuscrito utilizado nos testes: (a) documento digitalizado em tons de cinza; (b) imagem após a execução do filtro posterize com parâmetro 3; (c) imagem após a execução da binarização.*

Os algoritmos testados são: o proposto por Baird [9] e o proposto na seção 4.5. O algoritmo de Baird foi escolhido por ser considerado o mais eficiente e com menor média de erro entre os outros métodos da literatura. Testes realizados por Amin e Fisher [3] mostram que o algoritmo de Baird foi o mais rápido e o segundo com menor média de erro. Ambos os algoritmos apresentam as seguintes semelhanças: (1) nenhum utiliza parâmetros e, portanto, os resultados não podem ser afetados por uma escolha incorreta; (2) foram desenvolvidos para obter uma precisão de 0,1°; (3) funcionam em imagens monocromáticas de 200dpi de resolução.

Todas as imagens foram rotacionadas e testadas nos seguintes ângulos: 0.0°, ±0.1°, ±0,2°, ..., ±0,9°, ±1,0°, ±2,0°, ..., ±14,0°, ±15,0°, ±20,0°, ±30,0°, ..., ±170,0° e 180°. Para os documentos em japonês, foram rotacionadas e testadas apenas entre ±90,0°, já que não é possível detectar orientação invertida neste idioma. Vale ressaltar que a comparação entre os algoritmos de Baird e o proposto na seção 4.5 refere-se à detecção do enviesamento e está restrito à faixa de ±15°, por causa da limitação do algoritmo de Baird. O algoritmo proposto na seção 4.5 é testado para as outras faixas de ângulos. Finalmente, um teste é executado para verificar a complexidade de execução dos algoritmos.

Desta forma, os testes foram executados em 370 diferentes documentos em 82 ângulos diferentes, exceto para os documentos japoneses que foram 65 ângulos e documentos manuscritos em 49 ângulos, totalizando 27.840 imagens.

Os indicadores utilizados para medição e análise dos testes são:

- **Tempo médio**: o tempo médio, em milissegundos (ms), de execução do algoritmo, incluindo algoritmos de pré-processamento;
- **Média do erro**: média da diferença entre o ângulo rotacionado e o ângulo detectado pelos algoritmos;
- **Desvio padrão do erro**: fornece uma noção da homogeneidade dos erros em relação à média do erro;
- **Erro máximo**: indica a maior diferença entre o ângulo rotacionado e o ângulo detectado no teste, ou seja, é a tolerância de erro em que o algoritmo apresenta 100% de acerto;
- **Faixa de confiança**: porcentagem de acerto da detecção segundo uma faixa de tolerância de erro (0,0°, 0,1°, 0,2°).

4.7.2. Resultados

O primeiro teste envolve os documentos digitalizados com texto datilografado em inglês e com vários tipos de "layout" aplicados aos dois algoritmos citados. O total de

documentos utilizados foram 270 e rotacionadas em 49 ângulos diferentes entre a faixa de ±15.0º e, portanto, o total de imagens testadas foram 13.230. Os resultados foram:

Tabela 8 - *Resultados do primeiro teste entre os dois algoritmos com documentos digitalizados datilografados, em inglês e com vários tipos de "layout".*

Algoritmo	Tempo médio	Média do erro	Desvio padrão do erro	Faixa de confiança (%)			Erro máximo
				0,0º	0,1º	0,2º	
Baird	95 ms	0,004º	0,020º	96,02%	99,97%	100%	0,2º
Algoritmo proposto	115 ms	0,001º	0,012º	98,60%	100%	100%	0,1º

O método de Baird demonstrou ser mais rápido, contudo, devido ao fato da faixa de detecção do algoritmo proposto ser doze vezes maior, ele executa processamento extra para a detecção de outros ângulos tornando-o 21% mais lento. Apesar de o algoritmo proposto ter a média de erro quatro vezes menor que o de Baird, são dois números muito pequenos e, portanto, são ambos robustos em relação ao critério de dependência mínima do tipo de "layout".

O segundo teste envolve apenas o algoritmo proposto na mesma base de documentos acima, porém, utilizando os 270 documentos rotacionadas em 82 ângulos diferentes totalizando 22.140 imagens testadas. Os resultados foram:

Tabela 9 - *Resultados do segundo teste apenas com o algoritmo proposto utilizando a mesma base de documentos do primeiro teste.*

Algoritmo	Tempo médio	Média do erro	Desvio padrão do erro	Faixa de confiança (%)			Erro máximo
				0,0º	0,1º	0,2º	
Algoritmo proposto	131 ms	0,022º	0,871º	98,29%	99,73%	99,94%	180,0º
Algoritmo proposto sem orientação invertida	131 ms	0,002º	0,013º	98,34%	100%	100%	0,1º

A detecção da orientação invertida errou em apenas 12 imagens (0,05%), desta forma, o algoritmo para orientação invertida comprovou a sua eficácia experimentalmente para documentos com as características descritas. Contudo, para cada imagem não detectada corretamente, acrescentava-se um erro de 180º e influenciou bastante os

resultados fornecidos na **tabela 9** na primeira linha. Corrigindo as imagens e calculando novamente sem os efeitos da orientação invertida (**tabela 9**, segunda linha), a média de erro caiu em dez vezes e o erro máximo foi reduzido para 0,1°. Os algoritmos de enviesamento e orientação retrato/paisagem foram validados com sucesso na prática para documentos com as características descritas. O algoritmo é rápido levando em consideração suas capacidades e processou as 22.140 imagens em 48 minutos.

O terceiro teste tem a finalidade de testar os algoritmos em documentos com textos de diferentes idiomas. Neste caso, 50 documentos japoneses foram rotacionadas em 49 ângulos diferentes totalizando 2.450 imagens. Os resultados foram:

Tabela 10 - *Resultados do teste dos algoritmos em documentos escritos na língua japonesa.*

Algoritmo	Tempo médio	Média do erro	Desvio padrão do erro	Faixa de confiança (%)			Erro máximo
				0,0°	0,1°	0,2°	
Baird	86 ms	0,005°	0,023°	94,20%	100%	100%	0,1°
Algoritmo proposto	125 ms	0,035°	0,070°	71,83%	95,63%	98,48%	0,8°

Neste teste, o algoritmo de Baird obteve melhores resultados em todos os indicadores, sendo recomendado para detecção de enviesamento na faixa de ±15,0° para documentos japoneses. O algoritmo proposto obtém piores resultados devido ao fato de alguns símbolos japoneses serem formados por pequenos traços (strokes) separados e, desta forma, o agrupamento da linha é prejudicado, assim como, a detecção da rotação.

O quarto teste refere-se à aplicação do algoritmo proposto nos 50 documentos do terceiro teste rotacionados em 65 ângulos diferentes na faixa de ±90,0°, totalizando 3.250 imagens. Os resultados foram:

Tabela 11 - *Resultados do teste do algoritmo proposto em documentos escritos na língua japonesa.*

Algoritmo	Tempo médio	Média do erro	Desvio padrão do erro	Faixa de confiança (%)			Erro máximo
				0,0°	0,1°	0,2°	
Algoritmo proposto	155 ms	0,040°	0,075°	68,27%	88,55%	97,38%	0,9°

Neste teste, o algoritmo proposto foi testado em relação à detecção de enviesamento e orientação retrato/paisagem para o critério de dependência mínima do idioma do texto. O

erro máximo de 0,9º indica que a detecção de orientação retrato/paisagem funcionou em 100% das imagens, contudo, a detecção do enviesamento teve a precisão degradada, mesmo assim, obteve uma taxa de acerto em 97% das imagens com uma tolerância de 0,2º, faixa aceitável para OCR e projetos de digitalização.

No último teste, o critério de dependência mínima em relação à forma em que o texto foi escrito, 50 documentos manuscritos com letra corrida foram rotacionados em 49 ângulos diferentes na faixa de ±15,0º, totalizando 2.450 imagens. Os resultados foram:

Tabela 12 - *Resultados do teste dos algoritmos em documentos manuscritos.*

Algoritmo	**Tempo médio**	**Média do erro**	**Desvio padrão do erro**	**Faixa de confiança (%)**			**Erro máximo**
				0,0º	**0,1º**	**0,2º**	
Baird	24 ms	0,584º	1,376º	13,30 %	35,83 %	52 %	15º
Algoritmo proposto	33 ms	0,611º	4,090º	11,22 %	32,32 %	52,85 %	78,6º

Ambos os algoritmos apresentaram um razoável desempenho para documentos manuscritos com letra corrida. O motivo decorre da semelhança de ambos os algoritmos em utilizar apenas um ponto de cada componente formado devido à suposição feita de que cada bloco representaria uma letra, contudo, em documentos manuscritos, pode representar uma palavra. Entretanto, a faixa de confiança com uma tolerância de 1,0º atinge 91,18% e 95,67% para o algoritmo de Baird e o proposto, respectivamente.

Para analisar o tempo médio de execução de cada algoritmo relativo ao número de componentes do documento, o **gráfico *1*** foi construído associando o número de componentes sem ruídos e o tempo médio de processamento. Teoricamente, o algoritmo de Baird é linear, porque, em cada cálculo da projeção de perfil, os componentes são percorridos uma vez e o número de cálculo de projeção é fixo. Este fato é verificado também na prática. Teoricamente, o algoritmo proposto também executa em tempo linear que é verificado na prática. No **gráfico *1***, o tempo de execução do algoritmo proposto exibe uma tendência linear semelhante ao de Baird, apresentando uma pequena sobrecarga. Vale ressaltar que o número de componentes sem ruído de um documento digitalizado de tamanho A4 em inglês varia entre 1.000 e 5.000, também verificado por O'Gorman [38].

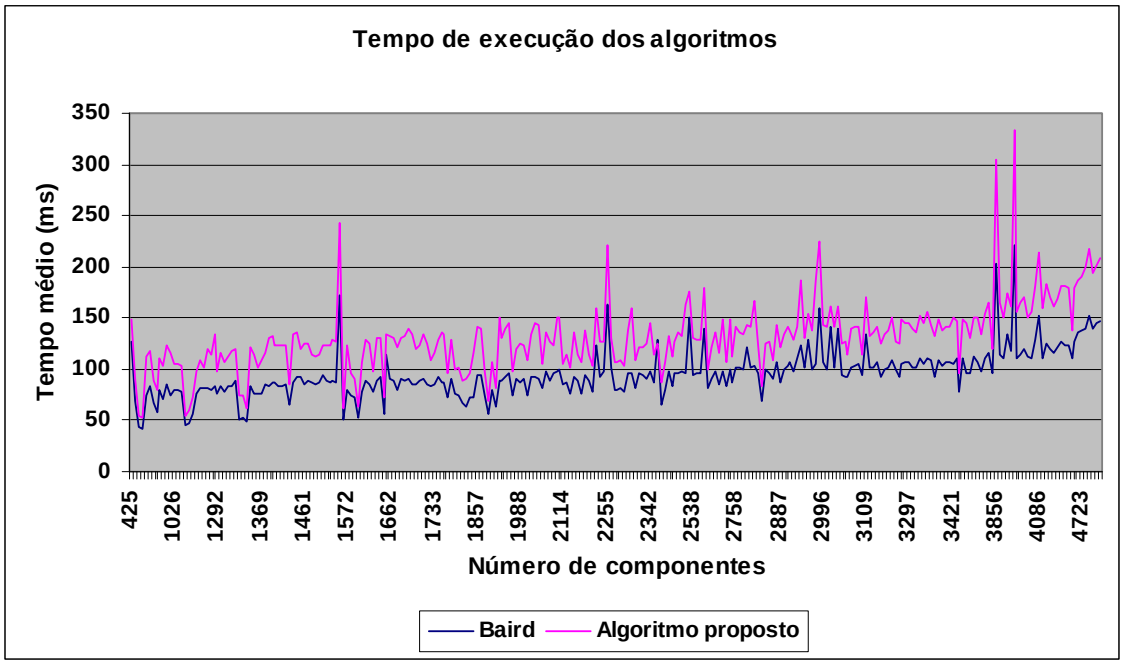


Gráfico 1 – *Tempo médio de execução do algoritmo de Baird e do proposto: notar as similaridades da curvas.*

Finalmente, um último teste foi realizado com apenas quatro documentos capturados de câmeras digitais (**Figura *46***). Para os quatros documentos, a imagem foi binarizada e em seguida aplicada ao algoritmo de detecção de rotação. Os ângulos foram detectados corretamente para todos os casos. Por último, a imagem colorida foi rotacionada com o ângulo detectado na imagem monocromática.

4.7.3. Resultados dos Testes do Algoritmo Proposto de Enviesamento

O algoritmo de detecção de enviesamento exatamente como proposto em [32] foi testado em 200 documentos digitalizados e monocromáticos retirados de artigos científicos e rotacionados em 0,1° a 0,9° com passos de 0,1°, em 1° a 9° com passos de 1°, e de 10° a 45° com passos de 5°, para ambos os lados (horário e anti-horário), totalizando 10.400 documentos digitalizados. O algoritmo foi implementado com a linguagem C ANSI e testado em um computador com processador Intel Pentium IV 2.4 GHz e 512 MB RAM.

O tempo médio de processamento foi 9 ms para pontos selecionados da esquerda para a direita e 8 ms para pontos selecionados de baixo para cima. O tempo máximo de processamento de uma imagem foi de 30 ms. O número médio de iterações foi quatro com o máximo de quatorze. O menor erro detectado foi 0,004°. No entanto, para imagens com rotação acima de 20°, o algoritmo detectou incorretamente os ângulos de enviesamento,

gerando uma média de erro alta. Isto sugere trabalhos futuros para melhor seleção dos pontos do documento.

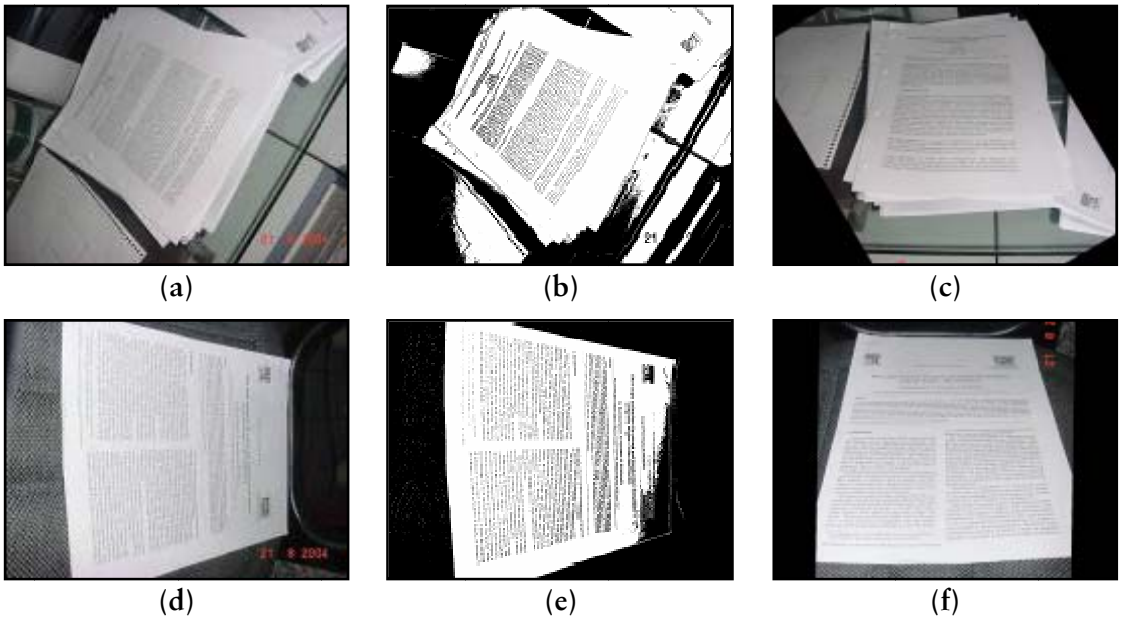


Figura 46 - *Exemplo de fotografias de documentos: (a) documento fotografado com distorções e outros objetos ao redor; (b) imagem a binarizada; (c) imagem a rotacionada com o ângulo detectado de -59,6° pelo algoritmo proposto seção 4.5; (d) documento fotografado distorcido e com orientação paisagem; (e) imagem d binarizada; (f) imagem d rotacionada com o ângulo detectado de 89,1° pelo algoritmo proposto seção 4.5.*

5. CORREÇÃO DE ROTAÇÃO

No capítulo anterior, foi apresentada a importância da detecção da orientação e enviesamento do documento digitalizado. O passo seguinte é rotacionar os *pixels* da imagem dado o ângulo detectado. Um *pixel* representa um ponto com coordenadas (x, y) em uma imagem. O procedimento para rotacionar um *pixel* é aplicar as suas coordenadas a uma matriz de transformação linear de rotação de um ângulo θ e de eixo de rotação (c_x, c_y). A equação de rotação é:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} - \begin{pmatrix} c_x \\ c_y \end{pmatrix} = \left\lfloor \begin{pmatrix} \cos\theta & -sen\theta \\ sen\theta & \cos\theta \end{pmatrix} \left[\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} c_x \\ c_y \end{pmatrix} \right] \right\rfloor$$

$$\therefore \qquad (4)$$

$$x' = \left\lfloor (x - c_x) \cdot \cos\theta - (y - c_y) \cdot sen\theta \right\rfloor + c_x$$

$$y' = \left\lfloor (y - c_y) \cdot \cos\theta + (x - c_x) \cdot sen\theta \right\rfloor + c_y$$

em que $\lfloor \cdot \rfloor$ é a função piso.

O algoritmo clássico de rotação de um documento é aplicar a equação (***4*** em todos os pixels, onde (c_x, c_y) é o centro do documento. Vale a pena ressaltar que: (a) o algoritmo de rotação não está levando em consideração o valor ou cor de cada *pixel*, apenas as suas coordenadas e, portanto, documentos monocromáticos, escala de cinza e coloridos podem ser rotacionados utilizando o algoritmo clássico e; (b) as coordenadas são números inteiros e a equação de rotação envolve quatro multiplicações e duas adições de pontos flutuantes, logo, é necessário o arredondamento dos resultados.

No entanto, o algoritmo clássico apresenta vários problemas relacionados ao tempo de execução e à qualidade do documento rotacionado. O algoritmo clássico de rotação aplica a equação de rotação para todos os pontos do documento e, além disso, para cada *pixel*, seis operações de ponto flutuante têm que ser executadas.

Não obstante, o algoritmo clássico degrada fortemente o documento rotacionado. Isto ocorre por causa da necessidade de arredondamento das coordenadas dos *pixels*. Três tipos de degradações são identificados (**Figura *47***): buracos (ruídos) brancos aparecem no interior dos objetos; bordas lisas tornam-se acidentadas e; partes dos objetos originalmente conectados podem se desconectar.

(a) (b)

Figura 47 - *Dois quadrados pretos de 10x10 pixels aplicados ao algoritmo clássico de rotação com 25°.*

O uso do algoritmo clássico de rotação em regiões preenchidas gera buracos (ruídos) brancos entre dois pixels vizinhos devido ao fato de muitos pixels serem mapeados em um mesmo pixel depois de rotacionados e arredondados. Os problemas dos contornos acidentados e dos objetos desconectados são conseqüências do problema do buraco branco, devido ao fato que todos os pixels são arredondados, incluindo as bordas. Estes problemas ocorrem em documentos monocromáticos, escala de cinza e colorido, devido à equação de rotação depender apenas das coordenadas e não do valor do pixel. Foi observado que, para imagens de pequenas dimensões, estes problemas ocorrem apenas em ângulos maiores. Contudo, para imagens maiores, esses problemas ocorrem em qualquer ângulo.

5.1. Revisão Bibliográfica

Existem diversos algoritmos de rotação de documentos propostos na literatura que tratam apenas do problema do buraco branco. Em 1986, Paeth [40] propôs um algoritmo de três passos em que a matriz de rotação é decomposta em três matrizes de corte. Em 1990, Cheng *et al.* [15] observaram que os buracos aparecem quando a distância Euclidiana entre dos pixels vizinhos após a rotação é dois ou raiz quadrada de cinco e propôs que os pontos do meio fossem preenchidos para eliminar os buracos brancos e manter a conexão entre os objetos. Em 1992, Danielsson e Hammerin [20] também propuseram um algoritmo de três passos em que a imagem é deslocada três vezes utilizando diferentes matrizes. Em 1996, Shah e Aggrawal [46] propuseram um método de mapeamento inverso em que cada valor do pixel é determinado por um mapeamento inverso de todos os pixels da imagem rotacionada para a imagem original. Em 1997, Jiang *et al.* [29] otimizou o método de mapeamento inverso utilizando uma tabela de mapeamento construída pelo algoritmo de detecção de enviesamento do documento e fazendo a transformação para 16 pixels de uma vez.

Os algoritmos acima podem ser utilizados tanto para documentos coloridos quanto para monocromáticos. Contudo, existem na literatura algoritmos específicos para documentos monocromáticos. Em 1998, Chien e Baek [16] propuseram um algoritmo rápido para carreiras pretas (Fast Black Run) em que, para cada carreira preta, é rotacionado apenas o pixel inicial e final da carreira e então, é a linha entre os dois pixels é desenhada. Para evitar o problema dos buracos brancos, se a carreira preta tiver uma carreira adjacente em baixo dela, então uma nova linha é desenhada abaixo da linha atual. Este método é mais rápido que o algoritmo clássico de rotação. Em 2001, Chien e Baek [17] propuseram um método rápido e hierárquico de comparação de blocos para extração de padrões de pixels da imagem original e calcula, para cada bloco, a sua rotação com o ângulo especificado. Desta forma, o algoritmo clássico de rotação pode ser substituído pela simples comparação de blocos com diferentes padrões de pixels. Este método é mais rápido que o algoritmo *Fast Black Run*.

Vale a pena ressaltar que nenhum dos métodos propostos acima não cita nem trata do problema de bordas acidentadas. Além disso, nenhum método garante que os objetos não serão desconectados, apesar de tentarem tratar deste problema. Existe a possibilidade de executar um pós-processamento *k-fill* (seção 2.1.3) que serve para suavizar as bordas sem desconectar os objetos, contudo não reconecta os objetos já desconectados pelo método de rotação e não é adequado para imagem que contenham figuras *halftone*.

5.2. Algoritmo Proposto

Um algoritmo de rotação de documentos monocromático é proposto de forma a tratar dos três problemas identificados gerando uma imagem de melhor qualidade [7][8].

A idéia básica do algoritmo é minimizar o número de pixels aplicados à equação de rotação (Equação 1). Desta forma, seleciona-se, para cada objeto do documento, um número mínimo de pontos que possa representá-lo com a fidelidade necessária. Apenas os pontos selecionados são rotacionados e cada objeto é reconstruído em uma nova imagem. Os passos básicos do algoritmo de rotação são apresentados a seguir (**Pseudocódigo *13***).

O primeiro passo é a vetorização da imagem, em que é feito um processo *bottom-up* dos pixels em um grafo da representação abstrata da imagem. O passo 1.1 localiza as bordas dos componentes da imagem (Figura 48b) através do uso de uma tabela de máscaras e aplicando em todos os pixels pretos. O próximo passo (1.2) determina os pontos críticos (Figura 48c), o conjunto mínimo de pontos necessário para representar a imagem, através do uso de uma tabela de máscaras e aplicando em todos os pixels pretos.

```
1. Construir vetores V:
      1.1. Localizar as bordas;
      1.2. Localizar os pontos críticos PC;
      1.3. Construir um grafo de vetores V;
      1.4. Otimizar número de vetores V;
2. Localizar polígonos preenchidos P;
3. Aplicar a equação de rotação nos pontos críticos PC;
4. Desenhar os vetores V na imagem rotacionada;
5. Preencher os polígonos P na imagem rotacionada.
```

Pseudocódigo 13 – *Procedimento principal do algoritmo de correção de rotação.*

O passo 1.3 consiste em criar um grafo de vetores, onde os nós representam um ponto crítico e as arestas representam os vetores (**Figura *48***d). Um vetor representa dois pontos críticos vizinhos conectados através de uma aresta definida no passo 1.1. O passo 1.3 funciona da seguinte forma: para cada ponto crítico, percorra através da aresta até localize um novo ponto crítico, então crie um vetor com os dois pontos e adicione ao grafo.

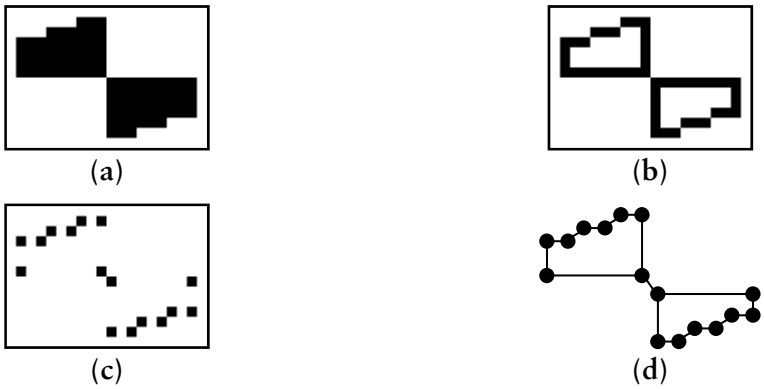


Figura 48 – *Passos do algoritmo: (a) imagem original; (b) bordas; (c) pontos críticos; (d) grafo de vetores.*

Em muitas situações, uma aresta possui pontos críticos intermediários que podem ser representados por apenas dois deles nas extremidades. Portanto, o passo 1.4 consiste em remover estes pontos críticos redundantes e unir os vetores colineares um único vetor (**Figura *49***). Este passo ajuda a reduzir o número de pontos críticos e funciona da seguinte forma: (a) para cada três vetores consecutivos, faça uma linha entre o ponto inicial do primeiro vetor e o ponto final do terceiro vetor e; (b) verifica se os dois pontos interiores pertencem à reta; (c) se sim, então una os três vetores em um, remova os dois pontos críticos intermediários, pegue os dois próximos vetores e segue para o passo *a*; (d) se não, simplesmente segue para o passo a com o próximo vetor.

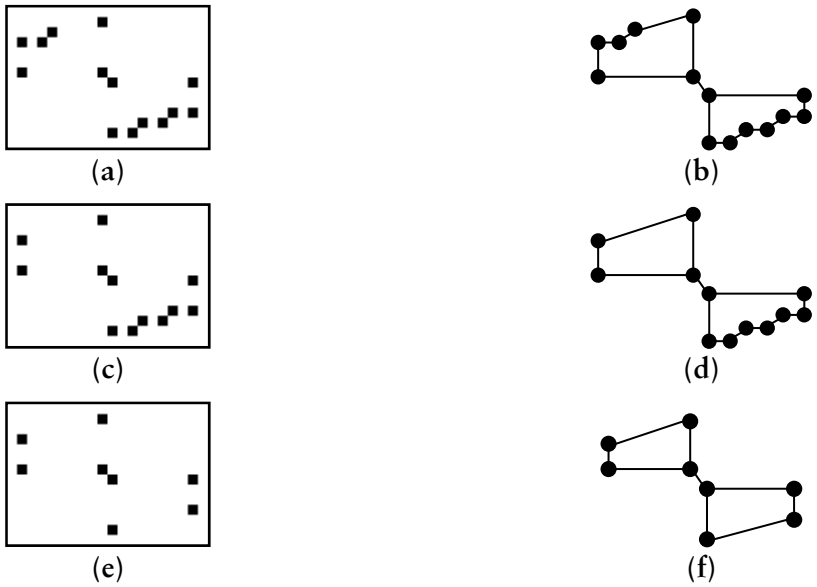


Figura 49 – *Passos para removes pontos críticos redundantes: (a, c, e) pontos críticos sendo removidos; (b, d, f) grafo de vetores sem os pontos redundantes.*

De modo a completar a representação da imagem original, o segundo passo identifica os vetores que pertencem a uma região fechada e preenchida e funciona como segue: (a) o algoritmo *component labelling* 4×4 (seção 2.1.2) é executado; (b) usando o mapa de bordas (arestas), o de pontos críticos e o mapa de identificadores feito pelo *component labelling*, extraia todos os vetores que pertencem de cada região preenchida. Os vetores extraídos serão utilizados no ultimo passo para preencher a região que eles representam.

O terceiro passo simplesmente rotaciona todos os pontos críticos usando a equação de rotação (Equação 1) dado um ângulo especificado (**Figura *50***a). O quarto passo desenha uma linha para cada vetor rotacionado no *buffer* da imagem rotacionada (**Figura *50***b). Finalmente, o quinto passo, usa os vetores identificados no passo 2 e aplica o algoritmo de *Scan Flood-fill* [10] (**Figura *50***c) na região delimitada por eles.

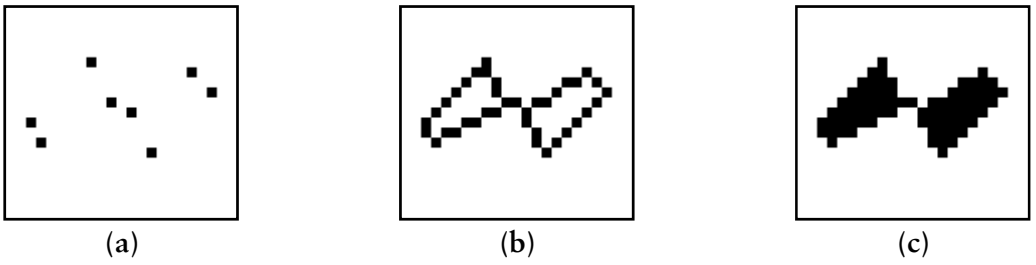


Figura 50 – *Passos do algoritmo: (a) pontos críticos rotacionados em 30º; (b) vetores desenhados; (c) regiões preenchidas e imagem final rotacionada.*

O problema do ruído branco é resolvido pelo algoritmo *Scan Flood-Fill*, porque ele preenche todos os pixels do interior de uma região delimitada por um conjunto de vetores. O problema da borda acidentada é resolvido em conseqüência do quarto passo, porque o algoritmo de desenhar linha cria uma borda (aresta) uniforme para cada vetor. Os elementos são mantidos conectados, porque o grafo de vetores criado também representa a ligação entre duas partes de um componente, logo, estes vetores são desenhados no quarto passo.

Observa-se que todos os algoritmos de rotação degradam a imagem original e, portanto, o primeiro passo do algoritmo pode ser substituído por qualquer outro algoritmo de vetorização existente [50][54], incluindo tipos com perda e sem perda. Deste modo, é viável aceitar uma tolerância de erro no passo 1.4b. No entanto, esta possibilidade não foi implementada neste trabalho. De modo a melhorar a qualidade da imagem rotacionada, implementações futuras podem incluir a vetorização de arcos como arestas curvadas. Neste trabalho, apenas arestas retas foram vetorizadas.

Na próxima seção, um novo método de medição de qualidade das imagens rotacionadas é apresentado.

5.3. Medição da Degradação dos Algoritmos de Correção de Rotação

Todos os algoritmos de rotação na literatura degradam a imagem original durante a rotação. Inspeção visual da qualidade da imagem resultante é útil apenas nos casos de imagens simples tal como na **figura *47***, usada por muitos algoritmos para demonstrar os seus resultados. Documentos digitalizados reais são muito mais complexos do que os "quadrados-conectados" (**Figura *47***a). O algoritmo proposto não apenas preenche os buracos brancos introduzidos na rotação, como garante a conexão entre componentes vizinhos e mantêm a suavização das bordas, recursos que os antecessores não ofereceram.

Inspeção visual das imagens rotacionadas oferecem uma fraca avaliação qualitativa da performance dos algoritmos em questão. Portanto, é necessário quantificar o quanto esses algoritmos degradam a imagem original durante o processo de rotação. Logo, um método quantitativo para medir o nível de degradação dos algoritmos de rotação em imagens monocromáticas é introduzido.

Uma metodologia para a análise quantitativa é apresentada a seguir. No primeiro passo, a imagem original é rotacionada no sentido horário em um ângulo de θº e então, é rotacionada no sentido anti-horário para a posição original em -θº, usando o mesmo algoritmo de rotação ambas as vezes. Então, o número de pixels que diferem da imagem original para a imagem duplamente rotacionada é contado, iniciando do centro da imagem.

Vale a pena ressaltar que a rotação altera a imagem original provocando um aumento em suas dimensões: a imagem rotacionada é mais larga que a imagem original. Cada algoritmo de rotação produz uma imagem de tamanho diferente. Portanto, uma comparação quantitativa está longe de ser uma tarefa trivial e não foi apresentada em nenhum outro trabalho na literatura. A análise PSNR (Peak Signal to Noise Ratio), que é aplicada com sucesso para comparar imagens com perdas [35], não é eficaz neste contexto. Existe também uma dificuldade para alinhar as imagens para estabelecer um ponto de referência fixo de modo a contar os pixels diferentes. Para resolver este problema, a imagem original é cortada para remover a borda branca adicionada antes e depois de cada rotação. Se a imagem rotacionada e cortada final estiver com diferentes dimensões comparadas com a imagem original cortada, então os pixels excedentes das linhas e/ou colunas são contados como incorretos (diferentes). Logo, a porcentagem entre os pixels incorretos e o total de pixels é usada como medida da degradação do algoritmo de rotação. Menor a porcentagem obtida, melhor a qualidade resultante da imagem e, conseqüentemente, melhor o algoritmo.

Um exemplo é dado para exibir o método de medida proposto comparando os resultados do algoritmo proposto e do algoritmo clássico de rotação. Usando o algoritmo proposto, a **figura *51***a foi rotacionado em 30º (**Figura *51***b), então foi rotacionado de volta em -30º e cortada resultando na figure 5c. Os mesmos passos foram executados para o algoritmo clássico de rotação (**Figura *51***d,e).

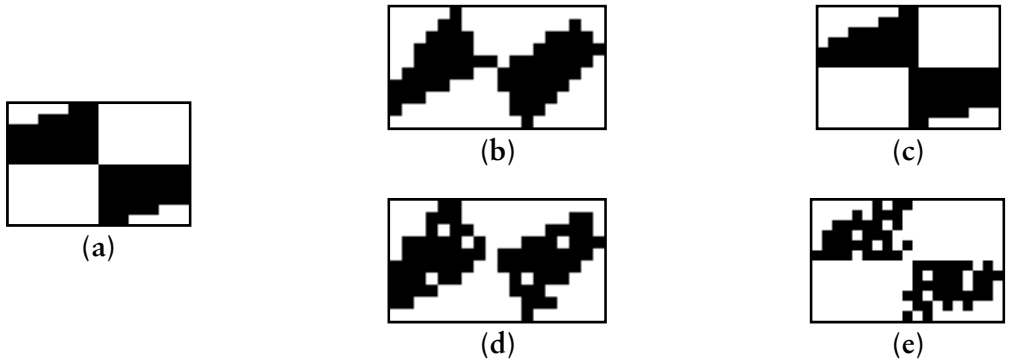


Figura 51 – *Um exemplo para medir a degradação produzida pelos algoritmos de rotação proposto e clássico: (a) imagem original; (b) rotação de 30º da imagem original utilizando o algoritmo proposto; (c) rotação de -30º da imagem do item b; (d) rotação de 30º da imagem original utilizando o algoritmo clássico; (e) rotação de -30º da imagem do item d.*

Figura 5c tem 18x12 pixels e tem as mesmas dimensões da imagem original cortada (**Figura *51***a). Contudo, **figura *51***e tem 19x12 pixels e, conseqüentemente, resultará em uma

quantidade maior de número de pixels incorretos. O numero total de pixels na **figura *51*c** é 216, logo, existem 16 pixels incorretos indicando uma degradação de 7,4%. O total de pixels na **figura *51*e** é 228, dos quais 40 são considerados errados resultando em 17,5% de degradação. Estes resultados estão sumarizados na tabela abaixo:

Tabela 13 - *Resultados da medição da degradação do algoritmo proposto e clássico de rotação do exemplo da figura 51.*

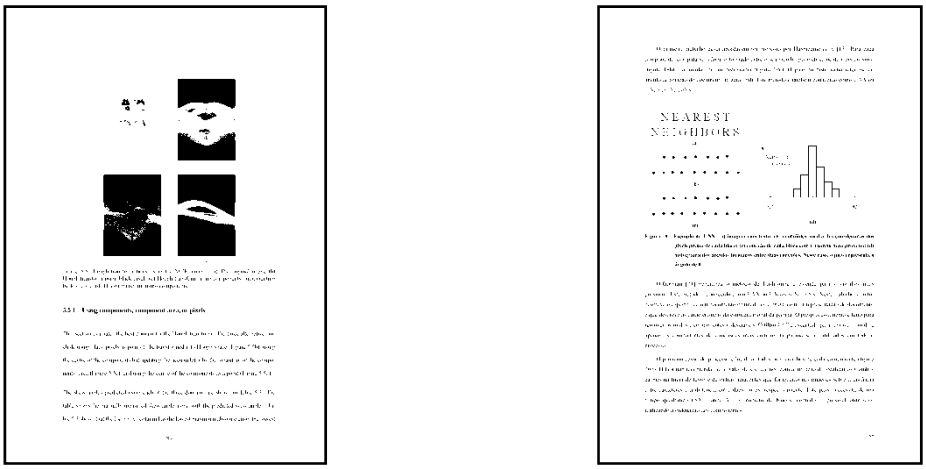
Algoritmo	Total de pixels	Pixels incorretos	Degradação
Proposto	216	16	7,4%
Clássico	228	40	17,5%

O algoritmo de rotação proposto produz uma imagem de melhor qualidade do que o algoritmo de rotação clássico. Os resultados do método quantitativo proposto são consistentes com os resultados qualitativos obtidos pela inspeção visual. Na próxima seção, o algoritmo de rotação proposto é comparado e medido com outros algoritmos de rotação em 2.000 documentos digitalizados.

5.4. Resultados dos Testes

Esta seção analisa o tamanho e a qualidade da imagem resultante dos algoritmos de rotação proposto, clássico e *Fast Black Run*. Nenhum pré ou pós-processamento foi executado excetuando-se o corte da margem branca necessário ao método de medição proposto. Todos os algoritmos foram implementados na linguagem ANSI-C e o ambiente de execução foi um Intel Pentium IV 3 GHz e 512 MB RAM.

Uma base de imagens foi criada com 2.000 documentos monocromáticos digitalizados extraídos de livros, teses e artigos. A base inclui documentos digitalizados pelo *scanner* e documentos sintéticos com elementos textuais e não-textuais, tal como, figuras *halftone* (**Figura *52***). Todas as imagens são 300 dpi e armazenadas usando o formato de arquivo TIFF usando o algoritmo de compressão CCITT G4.



(a) (b)

Figura 52 – *Exemplos da base de imagens: (a) imagens com figures em halftone; (b) texto com gráficos.*

5.4.1. Qualidade da Imagem

Esta seção avalia a qualidade das imagens rotacionadas por inspeção visual e pela medição do nível de degradação de todos os algoritmos de rotação, de acordo com a metodologia proposta na seção anterior.

De modo a medir o nível de degradação, todas as imagens foram cortadas para remover a borda branca, rotacionadas em 45°, cortadas novamente, rotacionadas de volta em -45° e cortadas uma última vez. Os resultados estão sumarizados na **tabela *14***.

Tabela 14 – *Medição da qualidade dos algoritmos de rotação em 2.000 documentos digitalizados.*

Algoritmo	Nível de Degradação
Clássico	3,50%
Fast Black Run	3,01%
Proposto	2,52%

Os resultados mostram que o algoritmo proposto degrada menos que os outros algoritmos. O algoritmo clássico de rotação resultou no pior algoritmo de rotação. Este resultado é compatível com a inspeção visual das imagens.

De modo a prover ao leitor um exemplo mais real dos efeitos dos algoritmos de rotação, a palavra "Using" foi cortada da versão rotacionada da **figura *52***a de todos os algoritmos, aumentada e apresentada na **figura *53***.

A imagem resultante do algoritmo clássico (**Figura *53***b) apresenta os problemas de buraco branco e bordas dentadas, produzindo um tipo de textura dentro das letras. O algoritmo *Fast Black Run* (**Figura *53***c) soluciona o problema do buraco branco, com esperado. Contudo, as bordas perdem a suavização. O algoritmo proposto soluciona ambos os problemas e produz uma imagem de melhor qualidade.

5.4.2. Tamanho da Imagem

Esta seção avalia o tamanho das imagens resultantes dos algoritmos de rotação aplicados à base de imagens original armazenadas em TIFF comprimidas com o algoritmo CCITT G4.

O tamanho total da base de imagens original comprimida é 105.944.920 bytes. O tamanho das imagens comprimida na primeira rotação de 45° e a segunda rotação de -45° obtidos pelo algoritmo de rotação são:

Tabela 15 - *Resultados do tamanho, em bytes, da imagem comprimida.*

Algoritmo	**Tamanho da imagem comprimida**	
	1ª rotação	**2ª rotação**
Clássico	281.370.126	273.545.102
Fast Black Run	152.947.758	150.904.026
Proposto	129.710.828	104.225.298

O tamanho total da base de imagens comprimida do algoritmo clássico de rotação é o maior entre eles. Isto ocorre por causa dos buracos brancos e das bordas dentadas que produzem mais e menores carreiras pretas, os quais degradam o algoritmo de compressão CCITT G4. O tamanho resultante gerado pelo algoritmo *Fast Black Run* [16] é menor que o clássico, porque ele preenche os buracos brancos nas áreas pretas e, conseqüentemente, reduz o número de carreiras pretas. Contudo, o tamanho total ainda é maior que o do algoritmo proposto, por causa das bordas dentadas que criam carreiras pretas menores. O algoritmo de rotação proposto resulta em um tamanho de imagem comprimida menor, por causa da menor quantidade de carreiras pretas geradas. Vale a pena ressaltar que o tamanho final das imagens comprimidas geradas pelo algoritmo proposto é menor que o tamanho total original. Esta redução deve-se ao fato do algoritmo proposto suavizar alguns contornos, diminuindo a quantidade de carreiras pretas.

(a)

(b)

(c)

(d)

Figura 53 – *A palavra Using da figura 52a para cada algoritmo rotacionado em 45° e -45°: (a) imagem original; (b) algoritmo clássico; (c) algoritmo Fast Black Run; (d) algoritmo proposto.*

Os resultados acima indicam uma correlação entre o tamanho das imagens comprimidas com o nível de degradação dela.

6. ARQUITETURAS PARA PROCESSAMENTO DE DOCUMENTOS DIGITALIZADOS MONOCROMÁTICOS

Este capítulo apresenta diversos tipos de arquiteturas para processamento de documentos digitalizados monocromáticos reunidos em um ambiente específico intitulado *BigBatch*.

6.1. BigBatch

O *BigBatch* é um ambiente de processamento especializado para processar automaticamente um grande volume de documentos digitalizados monocromáticos gerados por *scanners* de alta produção [33][34].

O *BigBatch* remove de ruídos brancos e pretos, remove bordas pretas, detecta rotação e corrige o documento, remove bordas brancas e, finalmente, salva o documento processado no formato TIFF com o algoritmo de compressão CCITT Group IV. Os filtros utilizam os parâmetros fixos com o objetivo de automatizar o processamento e evitar a necessidade de um operador. Deste modo, assume-se que os documentos foram digitalizados com 200 ou 300 dpi.

Os filtros incorporados ao ambiente foram implementados na ferramenta Microsoft Visual Studio 6 com a linguagem C e a interface do ambiente foi desenvolvida na ferramenta Borland Delphi 2005 com a linguagem Object Pascal. Desta forma, o *software* pode ser executado apenas na plataforma Windows 32 bits. Um recurso implementado é a possibilidade de organizar os documentos em projetos agrupados pelo modo de operação.

A **figura *54*** ilustra cada fase do processamento de um documento ao ser aplicado aos filtros. Percebe-se a redução da poluição visual e a visualização padronizada do documento. A qualidade dos documentos processados são superiores aos outros filtros. Esta afirmação é possível de ser feita, pois já foi confirmada a qualidade final das imagens após a aplicação de cada filtro desenvolvido neste trabalho.

O *BigBatch* pode operar em quatro formas: manual, seqüencial, *cluster* e *grid*. Cada modo será detalhado nas próximas seções.

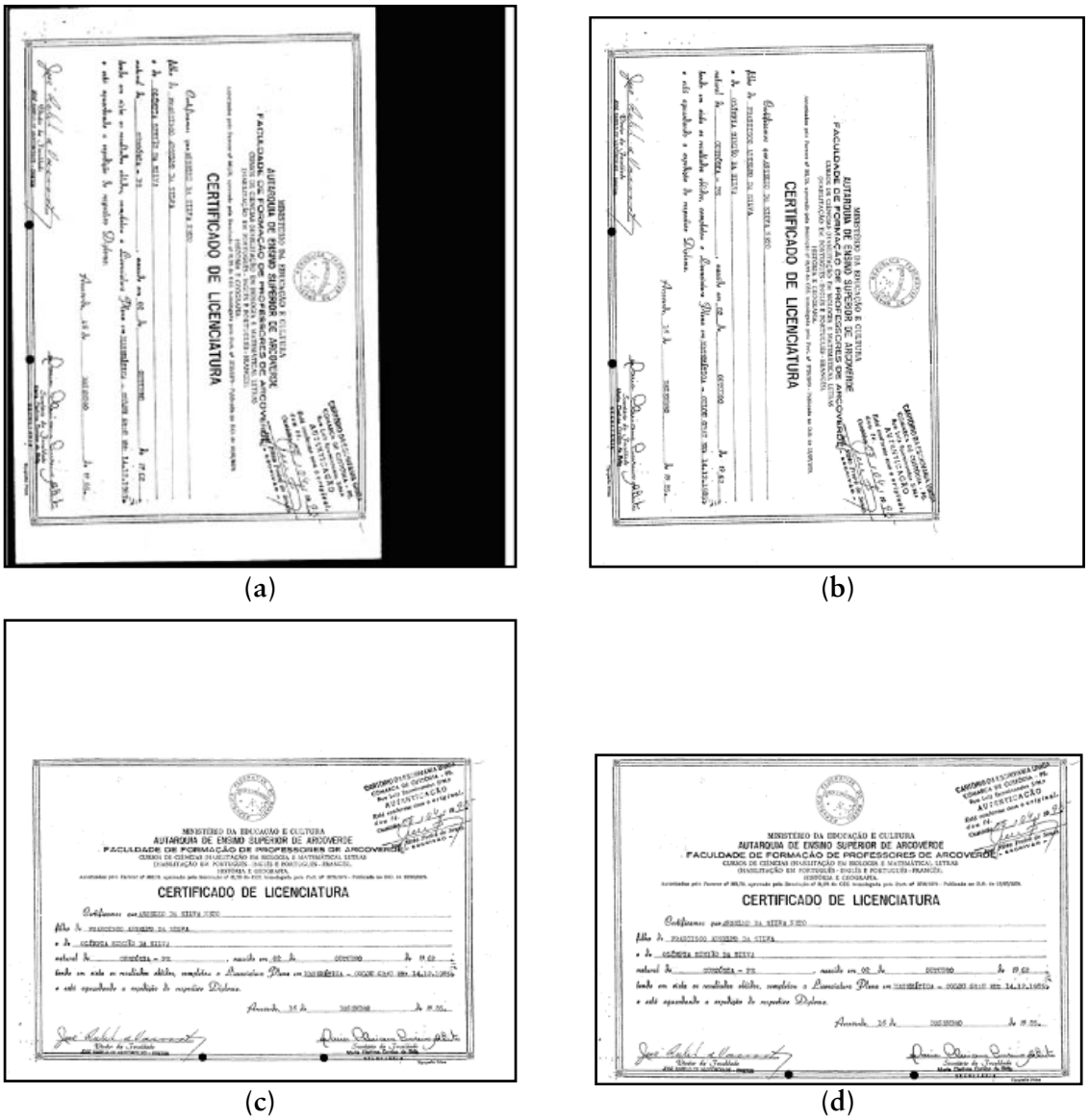


Figura 54 - *Documentos após cada filtragem: (a) remoção de ruído; (b) remoção de borda preta; (c) detecção e correção da rotação; (d) remoção da borda branca.*

6.2. Modo Manual e Seqüencial

O modo manual do *BigBatch* (**Figura 55**) permite que o usuário interaja com o ambiente e possa escolher o documento a ser processado, assim como, escolher quais filtros e executá-los em qualquer ordem.

O modo seqüencial do *BigBatch* (**Figura 56**) permite que o usuário escolha um diretório com os documentos digitalizados e também escolher um diretório para armazenados os documentos processados. No entanto, todos os filtros são executados automaticamente para cada imagem.

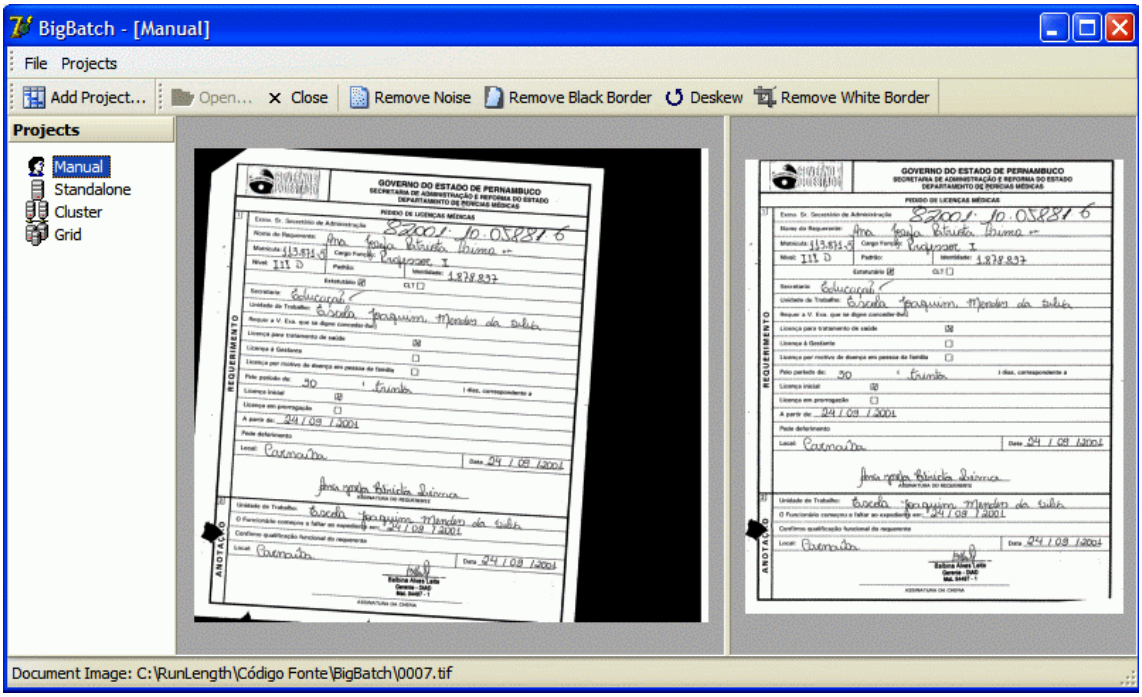


Figura 55 - *Interface do modo manual do BigBatch.*

As 21.267 imagens utilizadas no capítulo 3 foram utilizadas para testar o *BigBatch* no modo seqüencial. Pode-se observar também, na **tabela *16***, a redução do tamanho comprimido das imagens de até 36% do tamanho original. O tempo total de processamento das imagens foi de 10 horas e 48 minutos, ou seja, 1,82 segundos por imagem.

Tabela 16 – *Resultados do processamento dos filtros do BigBatch no modo seqüencial.*

Base	Número de Imagens	Tamanho Comp. Original	Tamanho Comp. Processado
CD1	6.020	252 MB	228 MB
CD2	5.615	706 MB	436 MB
CD3	5.097	647 MB	380 MB
CD4	4.538	705 MB	417 MB

A empresa de prestação de serviços de digitalização ArtDigital, situada no Recife/PE, utiliza o *BigBatch* na modalidade seqüencial em produção para tratamento automático dos documentos digitalizados de seus clientes.

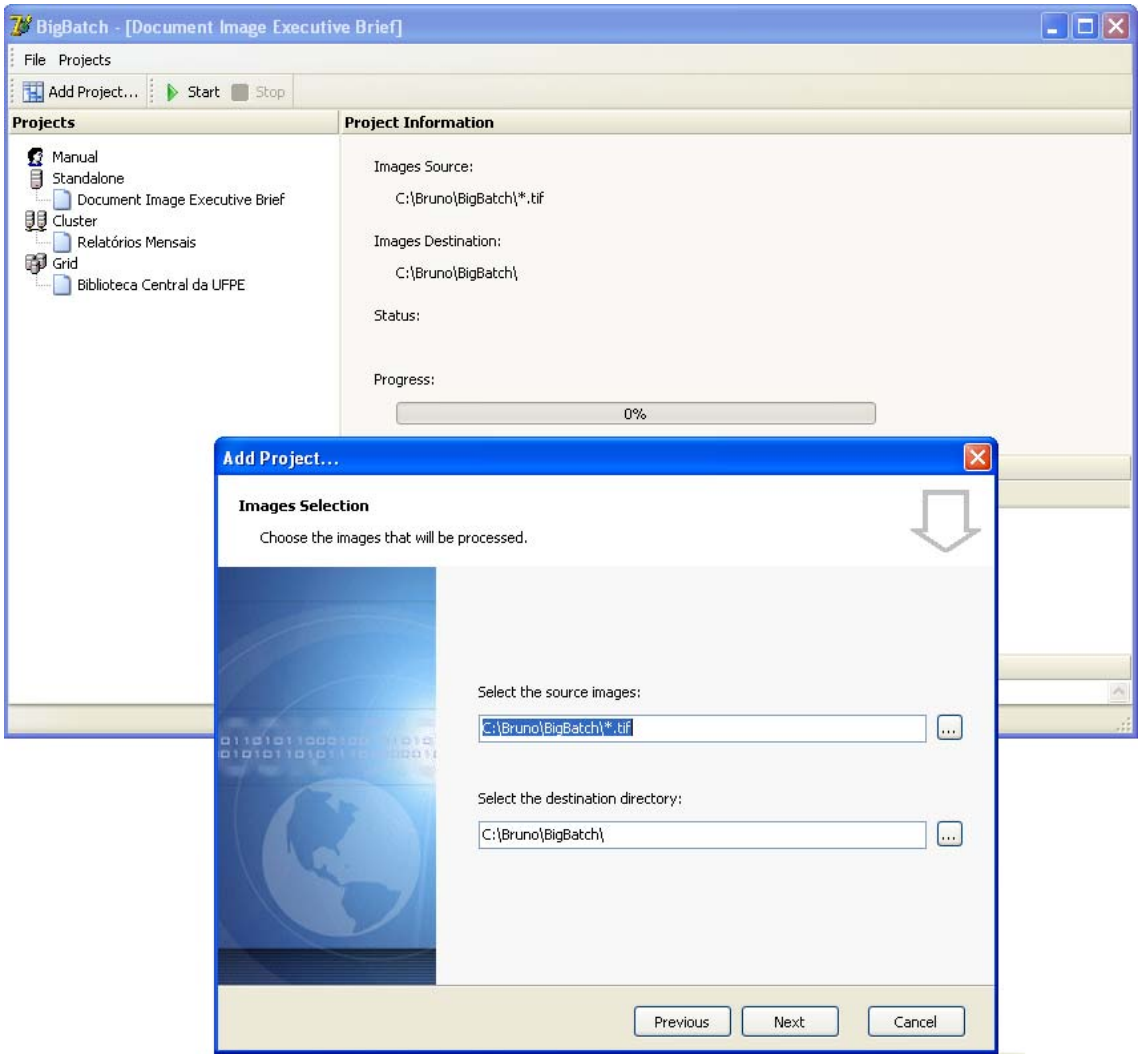


Figura 56 - *Modo seqüencial do BigBatch.*

6.3. Modo Cluster

O processamento de uma grande quantidade de documentos digitalizados é uma tarefa intensa e que pode requerer várias horas ou até dias. Em grandes projetos de digitalização, a quantidade de documento pode chegar a 10 milhões de documentos e o tempo de processamento gasto seria cerca de 210 dias (Estimativa com 1,82 segundos por documento). Estes números podem piorar se outras tarefas forem incorporadas como OCR e indexação automática do conteúdo. Testes preliminares mostraram que a transcrição automática utilizando o Omnipage 15 em um computador Intel Pentium IV 1.6 GHz e 512 MB de RAM levaram até um minuto para processamento por página tamanho A4. Um dos recursos pioneiros do BigBatch é a possibilidade de utilizar um *cluster* para processamento de documentos digitalizados.

A arquitetura do *cluster* é do tipo mestre/escravo. Atualmente, a maioria dos *clusters* é implementada para Linux ou Solaris e utilizam MPI para comunicação interprocessual. O cluster do *BigBatch* for arquitetado para a plataforma Windows 32 bits e utiliza o protocolo TCP/IP para comunicação. Assim como, suporta apenas um servidor mestre que envia pacotes para os trabalhadores e espera pela sua resposta. O mestre não executa qualquer outra operação a não ser supervisionar e alimentar os trabalhadores.

Um documento digitalizado é independente do outro (salvo na compressão de documentos multipaginados), o balanceamento de carga do *cluster* pode ser simplificado. O servidor constrói um pacote de dados contendo 10 documentos digitalizados e envia para os trabalhadores. O empacotamento das imagens tem duas vantagens: (1) reduzir a latência e a sobrecarga e; (2) balancear a carga de trabalho.

Vale a pena ressaltar que, na prática, os documentos e os computadores não são homogêneos. Os documentos podem ter dimensões e resoluções diferentes, diferentes quantidades e tipos de bordas pretas, ruídos, etc. Assim como, os computadores podem ter diferentes configurações de processador e memória. Estes fatores podem levar a uma variação no tempo de processamento dos algoritmos.

O *cluster* do *BigBatch* não é limitado a uma arquitetura homogênea. O pacote tem uma quantidade razoável de imagens para manter o cliente ocupado por alguns segundos e pequeno o suficiente para balancear a carga. Em um ambiente heterogêneo, quanto mais rápido o cliente, mais pacotes ele recebe, enquanto que os mais lentos receberão menos pacotes. Cada cliente mantém uma fila de pacote de forma a iniciar o processamento de um novo pacote enquanto que o antigo é enviado pela rede e, desta forma, manter seus recursos ao máximo.

Atualmente, não foi implementada nenhuma estratégia de tolerância à falha. Caso ocorra uma falha no *master*, todo o *cluster* pára. Assim como, se um nó cliente deixar de funcionar, o servidor não redistribui os seus pacotes. Planeja-se para o futuro, a implementação deste simples controle de realocação dos pacotes.

O *BigBatch* foi bastante testado em um cluster homogêneo com até 35 nós clientes. Uma rede Ethernet não dedicada de 100 Mbps com um *switch* 3Com com capacidade para 48 nós e um *uplink* de 1.0 GBps. Cada cliente, incluindo o servidor, era configurado com um Intel Pentium IV 1.6 GHz e 512 MB de RAM.

O **gráfico 2** mostra os resultados de desempenho em 2, 3, 8, 16, 32 e 35 nós. Com apenas um nó, as imagens foram processadas em 10 horas e 47 minutos. O *cluster* obteve um desempenho praticamente linear apresentando pouca sobrecarga nas tarefas de distribuição e coleta. Com 35 nós, atingiu o desempenho equivalente de um computador

com processador de 52 GHz, processando mais de 21 mil imagens em pouco mais de 20 minutos, ou seja, mais de 1.000 imagens por minuto.

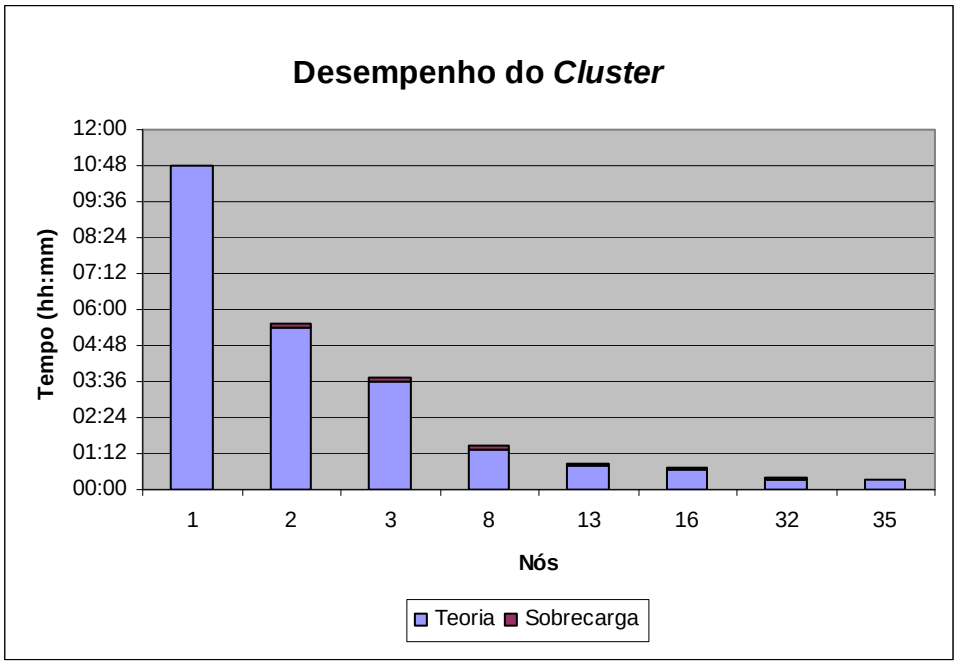


Gráfico 2 - *Desempenho do modo cluster do BigBatch*

Para a próxima versão do *cluster*, planeja-se extrair características do documento digitalizado para estimar e classificá-los com o objetivo de melhorar a estratégia de balanceamento e reduzir o *overhead* de distribuição. Além disso, um controle dinâmico e não-uniforme do tamanho do pacote pelo servidor pode aumentar a vazão da arquitetura. E mais: o uso de um único servidor *master* pode representar um gargalo no desempenho do *cluster*, logo, seria interessante investigar outras arquiteturas de redes, como por exemplo, *peer-to-peer*. Outra interessante análise é a comparação de desempenho do *BigBatch* entre o Windows e o Linux.

6.4. Modo Grid

Outra arquitetura interessante é o *grid*. A partir da mesma motivação da necessidade de desempenho devido ao grande volume de documentos, o *grid* possibilita o uso de todos os recursos computacionais da empresa, inclusive de recursos externos, para esta tarefa.

As tarefas do *BigBatch* apresentam paralelismo de dados e, portanto, são independentes. O *middleware OurGrid* foi projetado exatamente para aplicações que são *bag-of-tasks*, ou seja, aplicações com tarefas independentes. O servidor deve enviar tarefas

para o *grid* e detectar quando elas foram finalizadas. Isto pode ser feito comunicando com o *peer* do *OurGrid*, o qual é responsável por distribuir as tarefas no *grid* e coletar os resultados. Atualmente, a única interface para uma aplicação externa se comunicar com o *OurGrid* é através do RMI (*Remote Method Invocation*), um protocolo de chamada remota de procedimentos específico para a linguagem Java. Como o *BigBatch* foi desenvolvido em Delphi, é necessário criar um módulo de comunicação com o *peer* do *OurGrid*. Isto pode ser feito através de um aplicativo escrito em Java para se comunicar com o *OurGrid* através de RMI e com o *BigBatch* através do *Java Native Interface* (JNI). Vale a pena ressaltar que o modo *grid* ainda não foi incorporado ao *BigBatch*.

Uma questão importante deve ser discutida em relação às políticas de balanceamento das imagens entre os pares. O *OurGrid* inclui um algoritmo de balanceamento desenvolvido para aplicações *bag-of-tasks* que são sobrecarregadas de dados [37], mas que não necessita de informações extras sobre o tipo de processamento. No entanto, não está claro se é o melhor algoritmo de distribuição possível. O *BigBatch* poderia recolher informações sobre as imagens e que ajudaria a prever o tempo de processamento de cada tarefa e, assim, um melhor algoritmo de balanceamento seria desenvolvido.

7. CONCLUSÕES

Esta dissertação teve a finalidade de estudar e aprimorar os algoritmos e arquiteturas para processamento e tratamento de documentos digitalizados monocromáticos. A principal motivação deste estudo é a necessidade de automação dos projetos de digitalização devido ao grande volume de documentos. Os filtros abordados foram remoção de borda preta, detecção e correção de rotação e as arquiteturas estudadas foram seqüencial e *cluster*. A arquitetura *grid* foi apenas apresentada.

Um novo filtro para remoção de borda preta foi proposto baseado no algoritmo de preenchimento que foi adaptado para segmentar a informação da borda. Priorizou-se a capacidade de segmentação sobre a velocidade, de modo a evitar a remoção de informação do documento. Dois algoritmos diferentes do filtro foram desenvolvidos e comparados com cinco ferramentas comerciais existentes em mais de 21 mil documentos. O segundo algoritmo do filtro proposto foi cerca de duas vezes mais rápido que a ferramenta *ClearImage*, cuja remoção da borda apresentou os melhores resultados entre as ferramentas comerciais. Observou-se também que é possível reduzir em até 29% o custo de armazenamento dos documentos digitalizados apenas com a remoção das bordas pretas.

O clássico problema de detectar a rotação de documentos digitalizados foi abordado. Um algoritmo capaz de detectar o enviesamento e a orientação ao mesmo tempo foi proposto pela primeira vez na literatura. Este algoritmo é baseado na abordagem do Vizinho mais Próximo e tem as seguintes características: a única suposição que faz sobre o documento é que deve ter um número mínimo de linhas de texto suficientes para detectar a rotação; não faz uso de parâmetros; detecta orientação invertida apenas para documentos de língua ocidental; tem uma precisão de 0,1° e; detecta a rotação em qualquer ângulo. Comparado ao método de Baird em uma base de 22 mil documentos de língua inglesa, a nova técnica apresentou uma taxa de erro ligeiramente menor, tem uma faixa de detecção 12 vezes maior e complexidade linear, sendo apenas 21% mais lento. Para documentos de língua oriental ou manuscritos, a nova técnica apresentou comportamento semelhante, no entanto, inferior ao de Baird. A taxa de erros para estes documentos pode ser diminuída se os pré-processamentos de união de fragmentos e separação de símbolos forem executados. Para apenas quatro documentos capturados com câmeras digitais, o algoritmo proposto foi capaz de detectar satisfatoriamente a rotação deles.

Um segundo algoritmo para detectar apenas o enviesamento de documentos digitalizados foi proposto. A nova técnica faz uso de projeção das modas para detectar a

rotação. Desta vez, o método proposto faz suposições sobre o documento e é apropriado para formulários e documentos com muitas linhas de texto. Apresentou um excelente desempenho com uma taxa de erro razoável.

A detecção de rotação apenas fornece um ângulo correspondente à possível rotação do documento e, portanto, não rotaciona os pixels da imagem. A correção da rotação é outro problema com diferentes causas e conseqüências. O algoritmo clássico de rotação utiliza uma série de transformações lineares que envolvem o uso de pontos flutuantes e, portanto, ao arredondar, ruídos surgem gerando buracos no interior dos objetos. As técnicas na literatura tratam apenas do problema dos buracos brancos no interior dos objetos. Outros dois problemas foram detectados e tratados pela primeira vez: a borda acidentada e a desconexão de objetos. Um algoritmo foi proposto para rotacionar documentos digitalizados monocromáticos que trata pela primeira vez destes três problemas. Um método para medir a degradação dos métodos de rotação também foi proposto. O algoritmo de correção de rotação proposto foi comparado com o algoritmo *Fast Black Run* e apresentou uma menor taxa de degradação gerando, assim, uma imagem de melhor qualidade. Observou-se que o custo de armazenamento das imagens comprimidas após serem rotacionadas pelo algoritmo proposto foi reduzido em 62% e 31% em relação ao algoritmo clássico e o *Fast Black Run*, respectivamente.

As arquiteturas seqüencial e *cluster* para processamento de documentos digitalizados foram estudadas e incorporadas a um ambiente chamado *BigBatch*. Os filtros incorporados ao *BigBatch* foram: remoção de ruído, remoção de borda preta, detecção e correção de rotação e remoção de borda branca. Ao serem aplicados a uma base de 21 mil imagens, o processamento em modo seqüencial durou cerca de 11 horas e enquanto que o modo *cluster* com 35 nós durou pouco mais de 21 minutos, processando cerca de mil documentos por minuto. A possibilidade da execução do *BigBatch* em arquitetura distribuída do tipo *Grid* foi apenas especulada.

A empresa de prestação de serviços de digitalização ArtDigital, situada no Recife/PE, utiliza o *BigBatch* na modalidade seqüencial em produção para tratamento automático dos documentos digitalizados de seus clientes.

7.1. Trabalhos Futuros

A principal motivação deste trabalho é a necessidade de automatização devido ao grande volume de documentos e, portanto, uma das formas de automatizar é evitar o uso de parâmetros nos filtros aplicados. Neste trabalho, apenas o filtro de remoção de borda preta ainda apresenta a necessidade de ajustar parâmetros. Portanto, o cálculo automático dos

parâmetros do filtro de remoção de borda preta é proposto como trabalho futuro. O parâmetro LINE pode ser removido do filtro se outros métodos de classificação do bloco segmentado forem utilizados, como por exemplo, redes neurais. O parâmetro SEGMENT pode ser estimado através da medição da espessura dos componentes textuais do documento.

Um método quantitativo de medição de degradação do algoritmo de detecção e remoção de borda preta pode ser proposto e funcionaria da seguinte forma: (1) seria aplicada a um documento "limpo" (figura 57a) uma borda preta retirada de outro documento (figura 57b) formando um documento sintético (figura 57c); (2) o algoritmo de remoção de borda preta seria aplicado ao documento sintético e; (3) a imagem filtrada seria sobreposta com o documento "limpo" e então, seria contado o número de pixels pertencentes à borda preta que não foram removidos e o número de pixels pertencentes à informação que foram removidos; (4) o nível de degradação do algoritmo seria dado em função desses dois números.

(a) (b) (c)

Figura 57 – *Método quantitativo de medição de degradação do algoritmo de detecção e remoção de borda preta: (a) documento "limpo"; (b) borda preta extraída; (c) documento sintético.*

O método de detecção de rotação pode ser melhorado se apenas aplicado aos componentes textuais do documento. Isto aumentaria o desempenho e diminuiria a taxa de erros. O uso de pré-processamento de união de fragmentos e separação de componentes melhoraria o desempenho da detecção da rotação para documentos manuscritos e de línguas orientais. Um fator de confiança pode ser proposto para evitar estimar a orientação invertida em linhas de texto com apenas letras maiúsculas.

O método de correção de rotação pode ser integrado com um suavizador de componentes. A vetorização utilizada para a representação dos objetos do documento pode

ser do tipo com perdas. Deste modo, os pixels que tornam as bordas do objeto acidentadas já podem ser removidos. Portanto, ao invés do método de correção degradar a imagem após a rotação, a imagem rotacionada poderá estar com melhor qualidade que a original. Além disso, a tempo de processamento do documento seria diminuída uma vez que dois filtros foram aplicados de uma vez.

O balanceador de carga do *cluster* poderá extrair características da imagem comprimida de forma a estimar o tempo de processamento e, portanto, melhorar o balanceamento das imagens, principalmente para ambientes heterogêneos. Além disso, o controle de fluxo pode ser melhorado aumentando-se ou diminuindo o tamanho do pacote de modo a aumentar a vazão do servidor. A arquitetura *Grid* também poderia ser implementada.

8. PUBLICAÇÕES

Segue abaixo a relação das publicações, em ordem alfabética pelo sobrenome do primeiro autor e em ordem cronológica, gerados por esta dissertação:

[1] Ávila, B.T., Lins, R.D. A New Algorithm for Removing Noisy Borders from Monochromatic Documents. *In: 20th ACM Symposium on Applied Computing*, v. 2, p. 1219-1225, ACM Press, Nicósia, Chipre, 2004;

[2] Ávila, B.T., Lins, R.D. Efficient Removal of Noisy Borders from Monochromatic Documents. *In: International Conference on Image Analysis and Recognition*, LNCS, Springer, Porto, Portugal, 2004;

[3] Ávila, B.T., Lins, R.D. A New and Fast Orientation and Skew Detection Algorithm for Monochromatic Document Images. *In: ACM International Conference on Document Engineering*, ACM Press, Bristol, Inglaterra, 2005;

[4] Ávila, B.T., Lins, R.D., Neto, L.A.O. A New Rotation Algorithm for Monochromatic Images. *In: XXII Simpósio Brasileiro de Telecomunicações (SBrT)*, Campinas, São Paulo, 2005;

[5] Ávila, B.T., Lins, R.D., Neto, L.A.O. A New Rotation Algorithm for Monochromatic Images. *In: ACM International Conference on Document Engineering*, ACM Press, Bristol, Inglaterra, 2005;

[6] Lins, R.D., Ávila, B.T. A New Algorithm for Skew Detection in Images of Documents. *In: International Conference on Image Analysis and Recognition*, LNCS, Springer, Porto, Portugal, 2004;

[7] Lins, R.D., Ávila, B.T. BigBatch - A Toolbox for Monochromatic Documents. *In: ACM International Conference on Document Engineering*, ACM Press, Bristol, Inglaterra, 2005;

[8] Lins, R.D., Ávila, B.T., Formiga, A.A. BigBatch: An Environment for Processing Monochromatic Documents. *In: International Conference on Image Analysis and Recognition*, v. 41142. p. 886-896, LNCS, Springer, Póvoa de Varzim, Portugal, 2006.

9. REFERÊNCIAS

[1] Akiyama, T., Hagita, N. Automated entry system for printed documents. *Pattern Recognition*, v. 23, p. 1141-1154, 1990;

[2] Alves, N.F. Estratégias para melhoria do desempenho de ferramentas comerciais de reconhecimento óptico de caracteres. Dissertação de Mestrado em Engenharia Elétrica, Departamento de Eletrônica e Sistemas, Universidade Federal de Pernambuco, 2003;

[3] Amin, A., Fisher, S. A Document Skew Detection Method Using the Hough Transform. *Pattern Analysis & Applications*, v. 3(3), p. 243-253, setembro, 2000;

[4] Ávila, B.T., Lins, R.D. A New Algorithm for Removing Noisy Borders from Monochromatic Documents. *In: 20th ACM Symposium on Applied Computing*, v. 2, p. 1219-1225, ACM Press, Nicósia, Chipre, 2004;

[5] Ávila, B.T., Lins, R.D. Efficient Removal of Noisy Borders from Monochromatic Documents. *In: International Conference on Image Analysis and Recognition*, LNCS, Springer, Porto, Portugal, 2004;

[6] Ávila, B.T., Lins, R.D. A New and Fast Orientation and Skew Detection Algorithm for Monochromatic Document Images. *In: ACM International Conference on Document Engineering*, ACM Press, Bristol, Inglaterra, 2005;

[7] Ávila, B.T., Lins, R.D., Neto, L.A.O. A New Rotation Algorithm for Monochromatic Images. *In: XXII Simpósio Brasileiro de Telecomunicações (SBrT)*, Campinas, São Paulo, 2005;

[8] Ávila, B.T., Lins, R.D., Neto, L.A.O. A New Rotation Algorithm for Monochromatic Images. *In: ACM International Conference on Document Engineering*, ACM Press, Bristol, Inglaterra, 2005;

[9] Baird, H.S. The Skew Angle of Printed Documents. *Proc. Conf. Society of Photographic Scientists and Engineers*, p. 14-21, 1987;

[10] Berger, M. Computer Graphics with Pascal. The Benjamin/Cummings Publishing Company Inc., 1a edição, 1986;

[11] Bloomberg, D.S. Image analysis using threshold reduction. *SPIE Conference on Image Algebra and Morphological Image Processing II*, v. 1568, San Diego, California, p. 38-52, julho, 1991;

[12] Bloomberg, D.S., Kopec, G.E., Dasari, L. Measuring document image skew and orientation. *SPIE Conference on Document Recognition II*, p. 302-316, San Jose, Califórnia, EUA, fevereiro, 1995;

[13] Bottou, L., Haffner, P., Howard, P.G., Simard, P., Bengio, Y., LeCun, Y. High Quality Document Image Compression with DjVu. *Journal of Electronic Imaging*, v. 7(3), p. 410-425, SPIE,1988;

[14] Cattoni, R., Coianiz, T., Messelodi, S., Modena, C.M. Geometric Layout Analysis Techniques for Document Image Understanding: a review. ITC_IRST, janeiro, 1998;

[15] Cheng, H.D., Tang, Y.Y., Suen, C.Y. Parallel image transformation and its VLSI implementation. *Pattern Recognition*, 23, 1113-1129, 1990;

[16] Chien, S., Baek, Y. A fast black run rotation algorithm for binary images. *Pattern Recognition Letters*, Elsevier Science Inc., New York, NY, EUA, v. 19, p. 455-459, 1998;

[17] Chien, S., Baek, Y. Hierarchical block matching method for fast rotation of binary images. *IEEE Transactions on Image Processing*, v. 10(3), p. 483-489, 2001;

[18] Ciardiello, G., Scafuro, G., Degrandi, M.T., Spada, M.R., Roccotelli, M.P. An experimental system for office document handling and text recognition. *In Proc. of the 9th International Conference on Pattern Recognition*, v. 2, p. 739-743, Roma, Itália, novembro, 1988;

[19] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. *Introduction to algorithms*, MIT Press, Second Edition, 2001;

[20] Danielsson, P., Hammerin, M. High accuracy rotation of images. *CVGIP: Graphical Model and Image Processing*, v. 54(4), p. 340-344, 1992;

[21] Davies, E.R. Machine Vision: Theory, Algorithms, Practicalities. Academic Press, 1992;

[22] Dillencourt, M.B., Samet, H., Tamminen, M. A General Approach to Connected-Component Labeling for Arbitrary Image Representations. *Journal of the Association for Computing Machinery*, v. 39(2), p. 253-280, abril, 1992;

[23] Fan, K.C., Wang, Y.K., Lay, T.R. Marginal noise removal of document images. *Pattern Recognition,* v. 35, p. 2593-2611, 2002;

[24] Hashizume, A., Yeh, P.S., Rosenfeld, A. A method of detecting the orientation of aligned components. *Pattern Recognition Letters*, v. 4, p. 125-132, 1986;

[25] Hinds, S., Fisher, J., D'Amato, D. A document skew detection method using run-length encoding and the Hough transform. *In Proc. of the 10th International Conference on Pattern Recognition*, p. 464-468, Atlantic City, NJ, junho, 1990;

[26] Hough, P.V.C. Methods and means for recognizing complex patterns. US Patent #3.069.654, dezembro, 1962;

[27] Inglis, S. Lossless Document Image Compression. PhD thesis, University of Waikato, Hamilton, Nova Zelândia, março, 1999;

[28] Ishitani, Y. Document Skew Detection Based on Local Region Complexity. *In Proc. of the 2nd International Conference on Document Analysis and Recognition*, IEEE Computer Society, p. 49-52, Tsukuba, Japão, outubro, 1993;

[29] Jiang, H.D., Han, C.C., Fan, K.C. A fast approach to the detection and correction of skew documents. *Pattern Recognition Letters*, v. 18, p. 675-686, 1997;

[30] Le, D.S., Thoma, G.R., Wechsler, H. Automated Page Orientation and Skew Angle Detection for Binary Document Images. *Pattern Recognition*, v. 27(10), p. 1325-1344, 1994;

[31] Le, D.X., Thoma, G.R., Wechsler, H. Automated Borders Detection and Adaptive Segmentation for Binary Document Images. *13th International Conference on Pattern Recognition (ICPR'96)*, v. 3, p. 737, 1996;

[32] Lins, R.D., Ávila, B.T. A New Algorithm for Skew Detection in Images of Documents. *In: International Conference on Image Analysis and Recognition*, LNCS, Springer, Porto, Portugal, 2004;

[33] Lins, R.D., Ávila, B.T. BigBatch - A Toolbox for Monochromatic Documents. *In: ACM International Conference on Document Engineering*, ACM Press, Bristol, Inglaterra, 2005;

[34] Lins, R.D., Ávila, B.T., Formiga, A.A. BigBatch: An Environment for Processing Monochromatic Documents. *In: International Conference on Image Analysis and Recognition*, v. 41142. p. 886-896, LNCS, Springer, Póvoa de Varzim, Portugal, 2006;

[35] Lins, R.D., Machado, D.S.A. A comparative study of file formats for image storage and transmission. *Journal of Electronic Imaging*, v. 13(1), p. 175-183, 2004;

[36] Mello, C.A.B., Lins, R.D. Image Segmentation of Historical Documents. *Visual 2000*, México, agosto, 2000;

[37] Mowbray, M. OurGrid: a Web-based Community Grid. *In Proc. of the IADIS International Conference on Web Based Communities*, 2006;

[38] O'Gorman, L. The Document Spectrum for Page Layout Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 15(11), p. 1162-1173, 1993;

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

[40] Paeth, A.W. A fast algorithm for general raster rotation. *In Proceedings of the Graphics Interface '86 / Vision Interface '86*, Vancouver, Canada, Canadian Information Processing Society, p. 77-81, 1986;

[41] Pal, U., Chaudhuri, B.B. An improved document skew angle estimation technique. *Pattern Recognition Letters*, v. 17(8), p. 899-904, julho, 1996;

[42] Peerawit, W., Kawtrakul, A. Marginal Noise Removal from Document Images Using Edge Density. *In Proceeding of Information and Computer Engineering Workshop*, janeiro, 2003;

[43] Postl, W. Detection of Linear Oblique Structures and Skew Scan in Digitized Documents. *Proc. 8th Int'l Conf. Pattern Recognition (ICPR)*, IEEE CS Press, Los Alamitos, Califórnia, p. 687-689, 1986;

[44] Ronse, C., Divijver, P.A. Connected Components in Binary Images: The Detection Problem. Research Studies Press Ltd., Letchworth, England, 1984;

[45] Sezgin, M., Sankur, B. Survey over Image Thresholding Techniques and Quantitative Performance Evaluation. *Journal of Eletronic Imaging*, v. 13(1), p. 145-165, janeiro, 2004;

[46] Shah, S., Aggarwal, J.K. Intrinsic parameter calibration procedure for a (high-distorced) fish-eye lens camera with distortion model and accuracy estimation. *Pattern Recognition*, v. 29, p. 1775-1778, 1996;

[47] Shapiro, L.G., Stockman, G.C. Computer Vision, março, 2000 – http://www.cse.msu.edu/~stockman/Book/book.html;

[48] Smith, R. A Simple and Efficient Skew Detection Algorithm via Text Row Accumulation. *In Proc. of the 3th International Conference on Document Analysis and Recognition*, p. 1145-1148, Montreal, Canadá, agosto, 1995;

[49] Srihari, S.N., Govindaraju, V. Analysis of Textual Images Using the Hough Transform. *Machine Vision and Applications*, v. 2(3), p. 141-153, 1989;

[50] Teh, C.H., Chin, R.T. On the Detection of Dominant Points on Digital Curves. *IEEE Trans. Pattern Anal. Mach. Intell., IEEE Computer Society*, v. 11(8), p. 859-872, 1989;

[51] Vailaya, A., Zhang, H., Jain, A. Automatic Image Orientation Detection. *IEEE International Conference on Image Processing*, Kobe, Japão, outubro, 1999;

[52] Zhang, L., Li, M., Zhang, H. Boosting Image Orientation Detection with Indoor vs. Outdoor Classification. *IEEE Workshop on Applications of Computer Vision*, 2002;

[53] Wang, Y.M., Zhang, H. Detecting image orientation based on low-level visual content. *In Proc. Computer Vision and Image Understanding*, Elsevier, v. 93, p. 328-346, 2004;

[54] Wen-Yen, W., Mao-Jiun, J.W. Document image analysis. Detecting the dominant points by the curvature-based polygonal approximation. *IEEE Computer Society Press,* 1995;

[55] Adobe Photoshop CS 8.0 – http://www.adobe.com;

[56] BlackIce Document Imaging SDK 10. BlackIce Software Inc – http://www.blackice.com;
[57] BRACELPA – Associação Brasileira de Celulose e Papel – http://www.bracelpa.org.br;
[58] CENADEM – http://www.cenadem.com.br;
[59] ClearImage 5. Inlite Research Inc. – http://www.inliteresearch.com;
[60] How much information 2003? – http://www.sims.berkeley.edu/research/projects/how-much-info-2003;
[61] Kodak Digital Science Scanner 1500 – http://www.kodak.com/global/en/business/docimaging/1500002;
[62] Leadtools 13. Leadtools Inc. – http://www.leadtools.com;
[63] ICP Brasil – http://www.icpbrasil.gov.br/legisla.htm;
[64] ScanFix Bitonal Image Optimizer 4.21. TMS Sequóia – http://www.tmsinc.com;
[65] Skyline Tools Corporate Suite 7. Skyline Tools Imaging – http://www.skylinetools.com;

APÊNDICE A – CÓDIGO FONTE DOS ALGORITMOS E PROGRAMAS

Os filtros implementados geraram muitas linhas de código e, portanto, inviabilizou a sua impressão nesta tese. Um CD com os filtros e programas, além de algumas imagens utilizadas, é disponibilizado e os códigos fontes estão organizados em diretórios com segue abaixo:

Tabela 17 - *Organização dos diretórios do CD com o código fonte dos algoritmos e programas implementados nesta tese.*

Diretório	Descrição
Remoção de borda preta	Filtro de remoção de borda preta
- Algoritmo 1	Primeiro algoritmo do filtro (seção 3.2.1)
- Algoritmo 2	Segundo algoritmo do filtro (seção 3.2.2)
Detecção de rotação	Algoritmos de detecção de rotação
- Algoritmo 1	Primeiro algoritmo de rotação (seção 4.5)
- Algoritmo 2	Segundo algoritmo de rotação (seção 4.6)
Correção de rotação	Filtro de correção de rotação (seção 5.2), incluído o método de medição de degradação dos algoritmos de correção de rotação (seção 5.3)
Remoção de borda branca	Filtro de remoção de borda branca (seção 2.2.1) utilizado no BigBatch
Remoção de ruído	Filtro de remoção de ruído (seção 2.1.3) utilizado no BigBatch
BigBatch	Ambiente de processamento de documentos digitalizados e monocromáticos (seção 6.1)
- Ambiente	Interface gráfica do BigBatch (seção 6.2)
- Cluster	Cluster do BigBatch (seção 6.3)
Imagens	Alguns documentos digitalizados em 200 dpi e 300 dpi e monocromáticos.