UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

ORIENTAÇÃO DE DOCUMENTOS DIGITALIZADOS

TRABALHO DE GRADUAÇÃO

ALUNO: BRUNO TENÓRIO ÁVILA
ORIENTADOR: RAFAEL DUEIRE LINS

RECIFE, SETEMBRO DE 2004

Resumo

Documentos digitalizados por dispositivos de captura, como scanner, podem gerar imagens rotacionadas, acarretando em dificuldades desde a sua visualização até o reconhecimento óptico de caracteres (OCR). A procura por melhores soluções para este tipo de problema tem sido alvo de pesquisa por quase duas décadas. Este trabalho apresenta o problema da orientação e enviesamento de documentos digitalizados, se concentrando em imagens monocromáticas. Após o detalhamento do problema e o estudo do estado da arte, critérios foram definidos, classificados e priorizados para a construção e avaliação de tais métodos. Dois novos algoritmos de abordagens diferentes são propostos e validados em uma série de testes com 27.840 imagens, assim como, foram comparados ao algoritmo proposto por Baird.

Palavras chaves: Análise de documentos digitalizados, detecção de orientação e enviesamento, imagens monocromáticas.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Gráficos

Lista de Equações

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 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 a civilização humana, ele apresenta uma série de desvantagens na sua utilização. Em primeiro lugar, a sua fabricação depende da celulose, material extraído de árvores e, desta forma, leva ao desmatamento desenfreado de grandes áreas florestais. Em segundo lugar, o papel se desgasta com o tempo tornando-o 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.

Uma outra desvantagem é o custo relacionado ao papel. O valor do papel é extremamente barato e, portanto, pode acarretar em um consumo descontrolado. Além disso, o valor gasto com materiais relacionados à sua utilização, como por exemplo, cartucho de impressoras, lápis, pastas, clipes e prancheta, possui um valor elevado.

Por último, a enorme quantidade de papéis 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. As empresas maiores, com grandes quantidades de papéis, terceirizam o serviço de armazenamento e organização dos documentos que são guardados em galpões.

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 [39], 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 dos EUA ser um país moderno e informatizado.

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

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

Vários fatores dificultaram a conversão do papel 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 preço de vendas cai à metade e o poder de processamento duplica – 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, criptografia de 128 bits. Além disso, novas leis foram aprovadas que regularizam e dão valor jurídico aos documentos digitalizados.

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. Segundo uma pesquisa realizada no final de 2002 pelo CENADEM [38], projetou-se que o mercado de GED no Brasil crescerá 51,5% ao ano no período 2003/2004.

Uma das soluções mais comuns para eliminar o papel é a digitalização, ou seja, é um método que toma como entrada documentos físicos e gera imagens de cada um deles e os indexa. 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 digitalização através do uso de códigos de barra;
- **Digitalização:** o papel é digitalizado, ou seja, através de um scanner, o papel é convertido em uma imagem e armazenado em um computador;
- **Indexação:** detecção automática do código de barra e indexação automática do documento;

- **Processamento da Imagem:** o documento digitalizado passa por vários filtros de tratamento para melhorar a qualidade visual da imagem, diminuir o tamanho de armazenamento e executar a transcrição automática do documento (OCR);
- **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 digitalização.

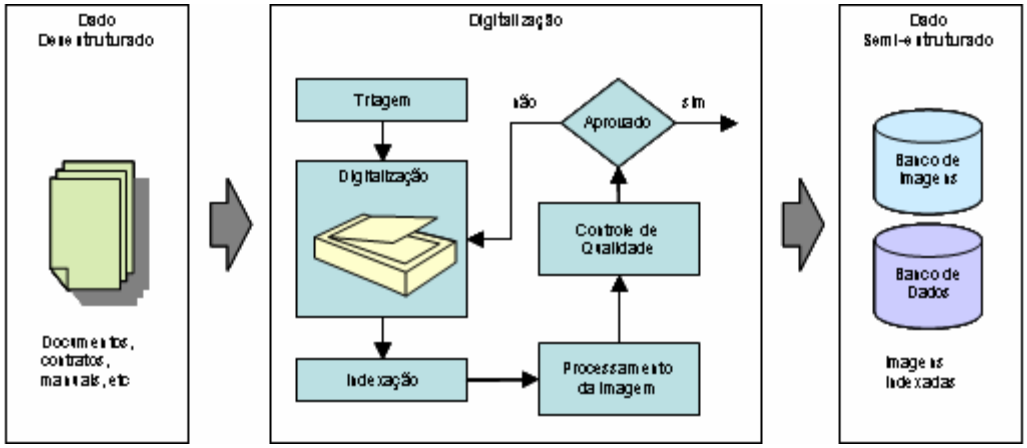


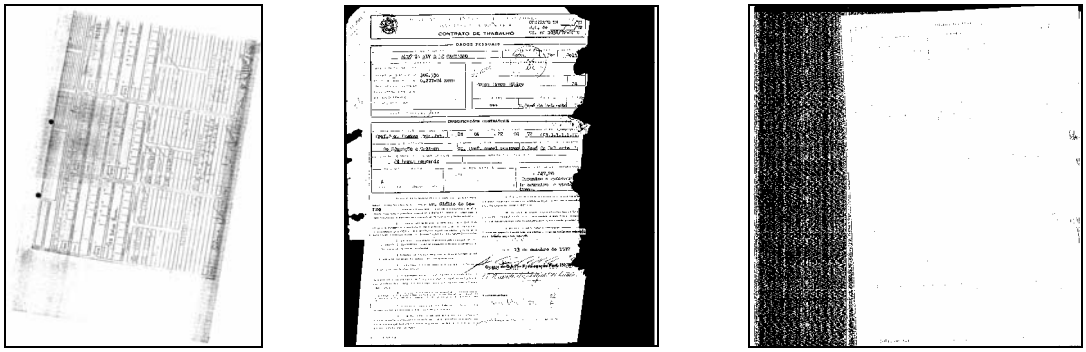
Figura 1 - Processo básico de digitalização.

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 digitalização. Esta fase é assistida pelo computador para a impressão dos códigos de barra que serão incluídos entre os papéis, detectados e utilizados na indexação.

Na fase de d*igitalização*, um scanner de alta produção recebe uma grande carga de papéis de diferentes formatos, tamanhos, estados de conservação e cor. Freqüentemente, o documento é digitalizado 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 2a) e, além disso, pode apresentar bordas pretas nas áreas externas à página e nas partes rasgadas (figura 2b), e; (4) devido à baixa qualidade dos dispositivos de digitalização (figura 2c).

A *indexação* é uma das fases mais importantes do processo. Consiste basicamente de reconhecimento de códigos de barra que contém informações sobre a identificação e localização dos documentos digitalizados e físicos. Em seguida, as imagens são organizadas e separadas de

acordo com a aplicação. As indexações baseadas nos conteúdos dos documentos (OCR) podem ser realizadas após a fase de processamento de imagem.



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

Figura 2 - Exemplos de documentos digitalizados 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 de documento e; (3) compressão de imagem. A subfase de *tratamento de imagem* consiste em aplicar filtros com objetivo de melhorar a imagem em relação à qualidade visual, taxa de compressão e taxa de transcrição. A subfase de *transcrição do documento* refere-se ao reconhecimento óptico de caracteres (OCR - Optical Caracter Recognition), 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. As fases de *indexação* e *processamento de imagem* são automáticas e executadas pelo computador no período ocioso do serviço da digitalização.

A fase de *controle de qualidade* caracteriza-se por ser a mais lenta do processo, em conseqüência da necessidade de inspecionar os resultados cuidadosamente das indexações e processamentos de todas as imagens manualmente auxiliado 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; aplicar filtros específicos para tratamento da imagem; remover as páginas em branco. As imagens com falhas serão rejeitadas e reenviadas à fase de digitalização.

1.1. Objetivo

Este trabalho tem o objetivo de estudar e aprimorar um dos métodos clássicos da subfase de tratamento de imagens: **Orientação de Documentos Digitalizados**.

A pesquisa envolve o detalhamento do problema, o estudo do estado da arte de algoritmos para orientação de documentos digitalizados, o desenvolvimento de novos métodos e a realização de um teste comparativo entre as técnicas existentes na literatura.

1.2. Conceitos

Alguns conceitos devem ser definidos inicialmente para um melhor entendimento do conteúdo ao longo deste documento.

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 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. Existem três tipos de orientação:

- **Retrato**: refere-se a documentos com orientação de 0º (figura 3a);
- **Paisagem**: refere-se a documentos com orientação de 90º e 270º (figura 3b);
- **Invertida**: refere-se a documentos com orientação de 180º (figura 3c).

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

Figura 3 - 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 ±45º 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. Este trabalho *Orientação de Documentos Digitalizados* refere-se ao estudo do problema de orientação e enviesamento de documentos digitalizados.

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.

1.3. Motivação

A principal motivação para o estudo de novas técnicas é a escassez de algoritmos capazes de detectar automaticamente a orientação e enviesamento de documentos digitalizados independente do "layout" do documento, dos símbolos do texto em que está sendo escrito, do estado de conservação, das cores e, principalmente, do grau de rotação do documento. A maioria dos métodos na literatura detecta apenas o enviesamento do documento. Existem poucos métodos para detecção de orientação retrato/paisagem e apenas dois algoritmos encontrados na literatura são capazes de detectar imagens com orientação invertida. Para se ajustarem a uma maior quantidade de documentos, muitos fazem uso de um número excessivo de parâmetros. O desenvolvimento de um algoritmo robusto, menos parametrizado e mais eficiente é, portanto, de interesse técnico-científico com impacto econômico.

Outras motivações são conseqüências dos benefícios gerados por este tipo de algoritmo. Em projetos de digitalização, o alinhamento correto das imagens resulta em uma padronização na visualização dos documentos e na melhora da leitura e navegação das páginas, além de diminuir a quantidade de imagens rejeitadas na fase de controle de qualidade.

Um scanner de alta produção tem capacidade de digitalizar 50 folhas por minuto, frente e verso, quando os papéis estão com orientação retrato. Caso as folhas sejam rotacionadas de forma a ficarem com orientação paisagem, o scanner processa 60 folhas por minuto, frente e verso. Portanto, há um impacto na produtividade de 20% na fase de digitalização apenas ajustando a disposição das páginas no dispositivo.

Outro benefício importante é a melhoria das taxas de OCR e a simplificação para a análise automática do "layout" de páginas. Os métodos de orientação e enviesamento são algoritmos de pré-processamento para as técnicas de OCR. Segundo Alves [2], a taxa de acerto diminui com o aumento do enviesamento. O desempenho da ferramenta comercial Omnipage 12.0 [41] é apresentado na tabela 2.

Rotação	**Sensibilidade à rotação por caractere**		**Sensibilidade à rotação por palavra**	
	Escala de cinza	**Colorido**	**Escala de cinza**	**Colorido**
1 grau	86,15 %	88,38 %	81,68 %	81,88 %
2 graus	86,80 %	87,47 %	75,39 %	75,74 %
3 graus	85,89 %	87,38 %	81,54 %	82,07 %
10 graus	83,85 %	86,75 %	74,66 %	74,08 %

Tabela 2 - Taxa de acerto do Omnipage 12.0 em diferentes ângulos em documentos em escala de cinza e colorido.

1.4. Visão Geral deste Documento

Este documento está organizado em cinco capítulos, além desta introdução e das referências bibliográficas. O capítulo 2 descreve o problema de orientação e enviesamento em detalhes. O capítulo 3 resume os algoritmos existentes na literatura. O capítulo 4 descreve os algoritmos propostos. O capítulo 5 analisa os resultados dos testes entre diversos algoritmos. O capítulo 6 apresenta as conclusões do presente estudo e aponta para linhas de trabalhos futuros.

2. Orientação e Enviesamento de Documentos Digitalizados

Este capítulo descreve em detalhes o problema da orientação e enviesamento de documentos digitalizados. Apresenta algoritmos importantes de pré-processamento e pós-processamento, descreve pontos de referência para diferentes aplicações e, finalmente, discute critérios para os algoritmos de orientação e enviesamento.

2.1. Algoritmos de Pré-processamento

Todos os métodos de orientação e enviesamento de documentos digitalizados demonstraram a necessidade de pré-processamento com a finalidade de facilitar a detecção da rotação. Os algoritmos mais comuns são: binarização, remoção de borda, nomeação de componentes, remoção de ruído e diminuição da resolução.

2.1.1. Binarização

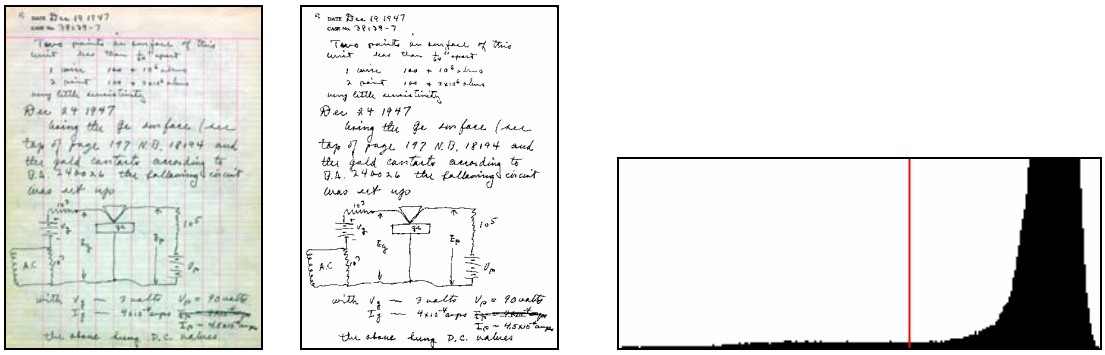
A orientação e enviesamento de documentos digitalizados são simplificadas com o uso de imagens monocromáticas e a quantidade de processamento e armazenamento é bastante comprimida. A idéia da binarização (*Thresholding*) é reduzir o número de cores das imagens coloridas ou em tons de cinza para apenas duas cores, demarcando os pixels das regiões de interesse em preto e os pixels das regiões de fundo em branco.

Em geral, os scanners de média e alta produção já possuem algoritmos de binarização implementados gerando imagens em tons de cinza ou monocromáticas que serão alvo de processamento posterior. Dependendo da aplicação, imagens em mais de duas cores são utilizadas para processamentos específicos.

Binarização é um problema clássico em processamento de imagens e existem dezenas de técnicas diferentes. O recente artigo [29] 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.



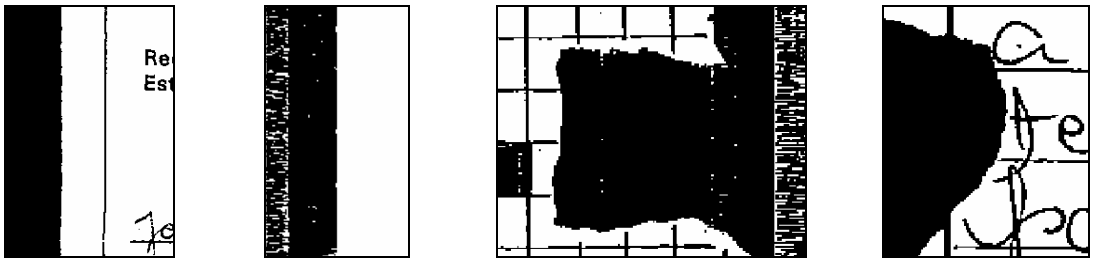
(a) **(b)** **(c)**

Figura 4 – Exemplo de binarização de um manuscrito colorido: (a) imagem original; (b) imagem após binarização; (c) histograma da imagem original com *threshold* global de 150, em vermelho.

No presente trabalho, vamos considerar que os documentos a serem processados para correção da orientação e enviesamento são monocromáticos, ou seja, foram pré-binarizados, estando o estudo detalhado de tais algoritmos fora do escopo. Em especial, assume-se também que a interferência frente-verso oriunda da digitalização de documentos escritos em ambos os lados de papel translúcido foi adequadamente resolvida utilizando algoritmos já descritos na literatura [21][23].

2.1.2. Remoção de Borda

Em projetos de digitalização de documentos burocráticos sem valor iconográfico utilizando scanners de produção industrial com alimentação automática, é comum a área de entrada do papel no scanner ser maior que o papel gerando uma borda preta ruidosa contornando o documento digitalizado. Nestes casos, a borda preta ruidosa acarreta em vários problemas: (1) degrada a imagem visualmente; (2) aumenta o espaço necessário para armazenamento e transmissão via rede; (3) aumenta o gasto de tinta para impressão do documento e; (4) degrada a eficiência dos algoritmos de processamento de imagem.

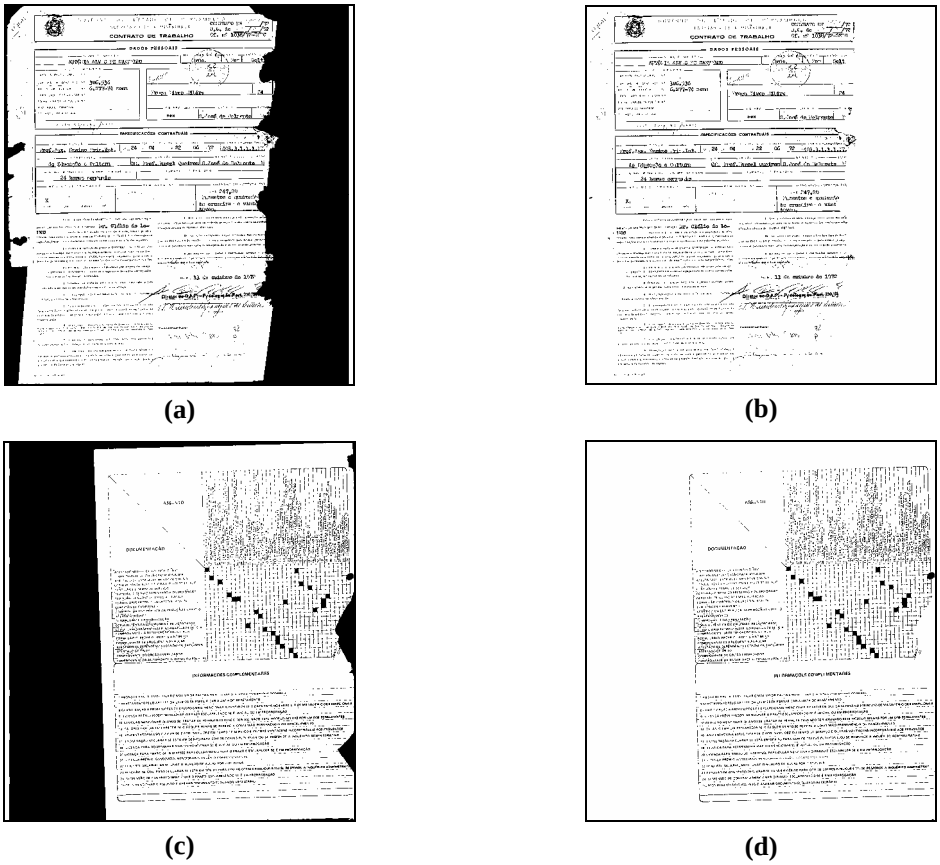


(a) **(b)** **(c)** **(d)**

Figura 5 - Exemplos de características das bordas pretas: (a) borda preta lisa; (b) borda preta com ruído; (c) borda preta com formato irregular devido a um pedaço rasgado no papel; (d) informações do documento conectado à borda preta.

Em geral, a digitalização desses acervos é feita em tons de cinza ou em monocromático. As bordas pretas apresentam as seguintes características: (1) apresentam bastante ruído branco (figura 5b); (2) parte do documento pode estar conectada à borda (figura 5d); (3) o papel pode ter regiões rasgadas fazendo com que a borda tenha um formato irregular (figura 5c). Algoritmos recentes foram propostos por Ávila e Lins [4][5] na tentativa de remover a borda preta com essas características.

A primeira versão do algoritmo [4] propõe um *flood-fill* modificado. Primeiro, para cada pixel preto na margem da imagem inicia-se um *flood-fill* normal, marcando cada pixel como BORDA. Vale ressaltar que, além dos pixels da borda, os pixels de parte do documento conectado à borda também são marcados e, desta forma, devem ser segmentados. Para separar a borda do documento, devem-se localizar todas as carreiras de comprimento horizontal menor ou igual ao parâmetro SEGMENTO e marca-las como ENTRADA_VERTICAL; executar o mesmo procedimento para carreiras verticais e marca-las como ENTRADA_HORIZONTAL. Finalmente, para cada pixel classificado como BORDA na margem da imagem, inicia-se um *flood-fill* que remove os pixels marcados parando nas carreiras marcadas como entrada.



(a) **(b)**

(c) **(d)**

Figura 6 - Exemplo de remoção de borda: (a) imagem original; (b) imagem *a* filtrada; (c) imagem original; (d) imagem *c* filtrada.

A segunda versão do algoritmo [5] otimiza o tempo de processamento do filtro utilizando a mesma idéia do algoritmo original. Primeiro, um *flood-fill* é iniciado a partir dos pixels pretos na margem da imagem e marca os pixels percorridos como BORDA. Em seguida, todas as carreiras verticais e horizontais de comprimento menor ou igual à SEGMENTO são marcadas como ENTRADA_HORIZONTAL e ENTRADA_VERTICAL, respectivamente. O algoritmo de nomeação de componentes é executado e agrupa os pixels vizinhos com a mesma marcação. Iniciando pelos componentes localizados na margem da imagem, prossegue-se da seguinte maneira: todo componente do tipo BORDA é removido; para cada componente classificado como entrada, percorre-se os componentes vizinhos e calcula-se a projeção, se a projeção for menor ou igual a um parâmetro LINE, então o componente é removido.

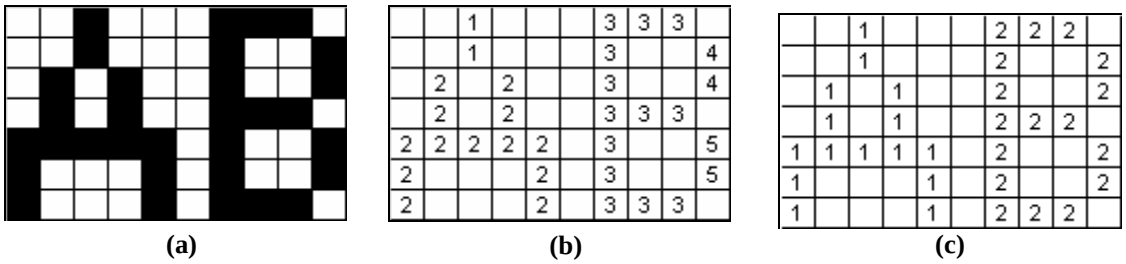
Os dois algoritmos foram testados em 20.000 imagens monocromáticas e concluiu-se que a qualidade da imagem melhorou e atingiu a marca de 29% de redução do tamanho necessário ao armazenamento. Demonstrou-se que o tempo de processamento do segundo algoritmo é mais rápido em relação a primeira versão e aos filtros para remoção de borda de ferramentas comerciais. Nenhum das duas versões do algoritmo removeu informações dos documentos testados, entretanto, a maioria das ferramentas comerciais falhou.

Considera-se que, para os algoritmos de orientação e enviesamento estudados neste trabalho, as imagens já foram filtradas e não apresentam bordas pretas.

2.1.3. Nomeação de Componentes

A nomeação de componentes (*Component Labeling*) é outra ferramenta clássica em processamento de imagens e é bastante utilizada nos algoritmos de orientação e enviesamento de documentos digitalizados. A principal função do método é identificar unicamente conjuntos de pixels conectados entre si de uma imagem (figura 7).

Dois pixels podem estar conectados de duas formas: 4x4, leva em conta apenas os pixels vizinhos na direção vertical e horizontal (figura 7b); 8x8, todos os vizinhos são utilizados, inclusive na diagonal (figura 7c). Em geral, o uso de componentes conectados 8x8 é mais comum nos métodos de orientação, devido aos pixels dos caracteres, após a binarização, estarem dispostos nas diagonais (figura 7a).



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

Figura 7 - Exemplo de nomeação de componentes: (a) imagem original; (b) nomeação de componentes 4x4; (c) nomeação de componentes 8x8.

Existem vários métodos de nomeação de componentes [12][14][25][28][30]. O método mais conhecido é o de *Duas Fases* [30]. 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 desta 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.

2.1.4. 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 barulho; (3) o próprio dispositivo de captura pode ter defeitos.

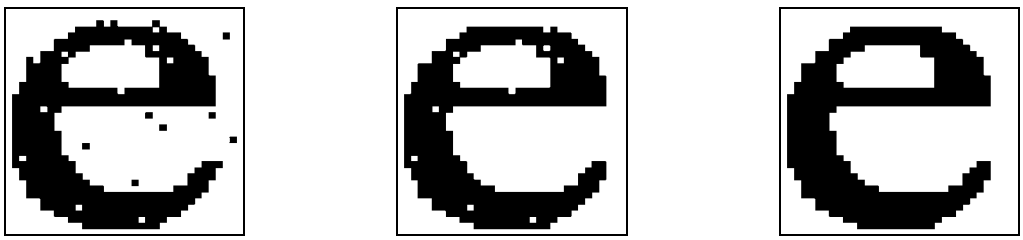
Em qualquer aplicação, os ruídos podem afetar os resultados dos processamentos e, desta forma, devem ser removidos. Em geral, a maioria dos métodos de orientação e enviesamento necessita executar este filtro. Além disso, outras conseqüências da presença de ruídos em imagens são: (1) aumento do tamanho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

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 x k na imagem binária, verifica-se os $4(k+1)$ pixels vizinhos. Se não houver nenhum pixels 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 [25]. O método é uma extensão do filtro da vizinhança. Para cada conjunto de pixels pretos conectados entre si de dimensões *(k-2)* x *(k-2)* na imagem binária, calcula-se: n, número de pixels pretos na vizinhança; c, número de carreiras de pixels pretos conectados na vizinhança e; r, número de pixels pretos nos cantos. A condição para preencher os pontos interiores de branco é:

$$(c=1)\ AND\ \{(n>3k-4)\ OR\ [(n=3k-4)\ AND\ (r=2)]\} \qquad \textbf{(1)}$$

O filtro *K-fill* foi desenvolvido especificamente para textos e gráficos com a finalidade de remover ruído enquanto mantém a legibilidade (figura 8). Outra finalidade é a de suavizar os caracteres na medida em que remove o ruído branco dos caracteres em preto.



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

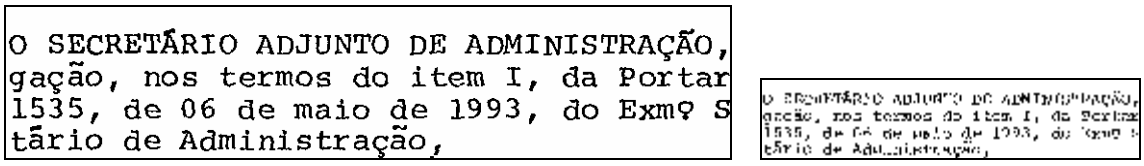
Figura 8 - Exemplo de remoção de ruído utilizando o filtro *K-fill*: (a) imagem original; (b) remoção do ruído preto; (c) remoção do ruído branco.

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 de orientação e enviesamento que utilizam a nomeação de componentes. Após a formação dos blocos, os componentes de dimensões menores à *k*x*k* são classificados como ruído e podem ser removidos ou apenas desconsiderados, desta maneira é efetuada a filtragem de ruído simultaneamente à nomeação de componentes.

2.1.5. Diminuição da Resolução

A digitalização de documentos burocráticos e sem valor iconográfico, se efetuada com resolução de 200dpi, guarda todos os elementos essenciais do documento oferecendo um bom fator qualidade/espaço de armazenamento [23] e transmissão via rede de computadores [22]. Em caso de transcrição automática via OCR, estudos [2] mostram que a resolução de 200dpi oferece uma boa taxa de acertos, embora o pico de acertos tenha sido detectado em 300dpi de resolução. Uma folha A4 digitalizada em 200dpi tem cerca de quatro milhões de pixels; se digitalizada com 300dpi, 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 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, figuras (figura 9). Dependendo da redução e do algoritmo escolhido, os caracteres também podem desaparecer comprometendo a precisão dos algoritmos de orientação e enviesamento.



(a) **(b)**

Figura 9 - Exemplo de diminuição de resolução de imagem: (a) imagem original com 200dpi; (b) imagem reduzida para 50dpi.

Uma técnica para diminuir a resolução de uma imagem é a redução por *threshold*, proposto por Bloomberg [7]. O método de redução por *threshold* é implementada como uma cascata de operações de redução 2x, cada uma com um *threshold* de 1. 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

Na realidade, os algoritmos de orientação e enviesamento não fazem nenhum tipo de filtragem na imagem, apenas calculam ângulos de rotação. Na fase seguinte ao cálculo do ângulo de rotação, a rotação é executada com o ângulo total de rotação detectado como parâmetro de entrada, evitando a degradação da imagem por rotações sucessivas. Outros métodos podem ser executados apenas para melhorar a qualidade da imagem devido a problemas no algoritmo de rotação, entre eles, remoção de margem branca e suavização de caracteres.

2.2.1. Rotação

O algoritmo clássico de rotação é um procedimento simples e deve ser aplicado a todos os pixels pretos da imagem. As coordenadas originais de cada pixels preto devem ser aplicadas à equação 2:

$$\begin{aligned} x' &= (x - c_x)\cos\theta - (y - c_y)\sin\theta + c_x \\ y' &= (y - c_y)\cos\theta + (x - c_x)\sin\theta + c_y \end{aligned} \qquad \textbf{(2)}$$

onde *c* é o ponto médio da imagem. O resultado é uma nova coordenada que deve ser marcado de preto na nova imagem.

Devido aos erros de truncamento dos computadores no cálculo das novas coordenadas, principalmente para ângulos grandes, pode-se resultar em uma imagem de pior qualidade (figura 10b).

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

Figura 10 - Exemplo de rotação de um caractere: (a) imagem original; (b) imagem rotacionada utilizando o algoritmo convencional; (c) imagem rotacionada utilizando o método de Chien.

Uma modificação para o algoritmo clássico de rotação de imagens binárias é proposta por Chien [10]. O algoritmo varre a imagem da esquerda para direita e de cima para baixo e encontra os pontos iniciais e finais de cada carreira preta. Aplica-se a equação 2 apenas nos

pontos iniciais e finais, resultando em *i' e f'*, respectivamente. Na imagem resultante, desenha-se um segmento de reta de dois pixels de espessura de ponto inicial *i'* e ponto final *f'*. Este método proporciona uma imagem de melhor qualidade (figura 10c), além de ser mais rápido.

2.2.2. Remoção de Margem Branca

A remoção de margem branca é um procedimento simples e pode ser integrado ao algoritmo de rotação. Este filtro remove uma margem branca gerada pela rotação da imagem proporcionando, além da já mencionada diminuição no espaço necessário ao armazenamento do documento, uma melhora na qualidade da imagem impressa ou exibida em terminais de vídeo equivalente a um *zoom* na imagem original.

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

Figura 11 - Exemplo de remoção de margem branca: (a) imagem original; (b) imagem rotacionada; (c) imagem com as margens brancas removidas.

No exemplo acima (figura 11), um documento digitalizado com enviesamento de 45º (figura 11a) é rotacionado. A imagem resultante (figura 11b) sofreu um aumento significativo das margens brancas. Ao aplicar o filtro, a margem branca da imagem é removida (figura 11c).

2.2.3. Suavização de Caracteres

A suavização de caracteres tem o objetivo de remover o serrilhado das letras e gráficos preenchendo de branco os pixels pretos serrilhados no perímetro dos símbolos e remover o ruído branco interior, preenchendo de preto. Este filtro melhora a qualidade visual do documento e acarreta em uma diminuição do armazenamento da imagem, porque a grande maioria dos algoritmos de armazenamento e transmissão via rede de imagens estáticas utiliza-se de *run-length encoding*, em alguns estágios da compressão.

Um método de suavização de caracteres é o filtro *K-fill* (citado na seção 2.1.4 Remoção de Ruído). No exemplo acima, a figura 12a é aplicada ao filtro para o ruído preto (figura 12b) e

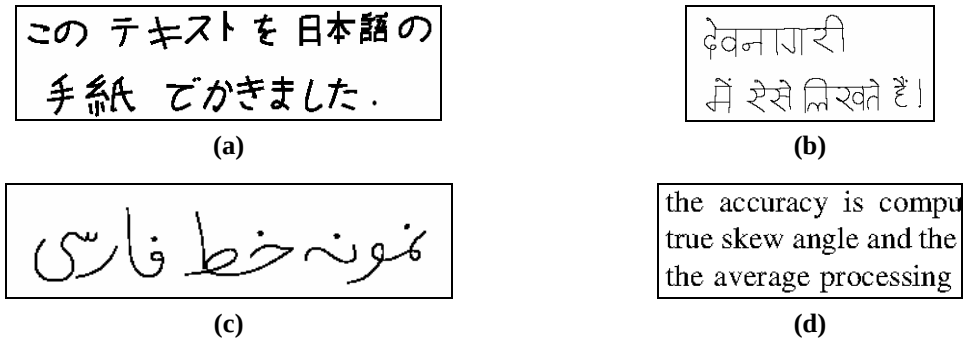
depois para o ruído branco (figura 12c). Vale destacar que este procedimento gerou uma imagem melhor qualidade em relação à figura 10c, contudo, o tempo de processamento é maior.

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

Figura 12 - Exemplo de suavização utilizando o filtro *K-fill*: (a) imagem original; (b) imagem suavizada da cor preta; (c) imagem suavizada da cor branca.

2.3. 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 13). 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)**

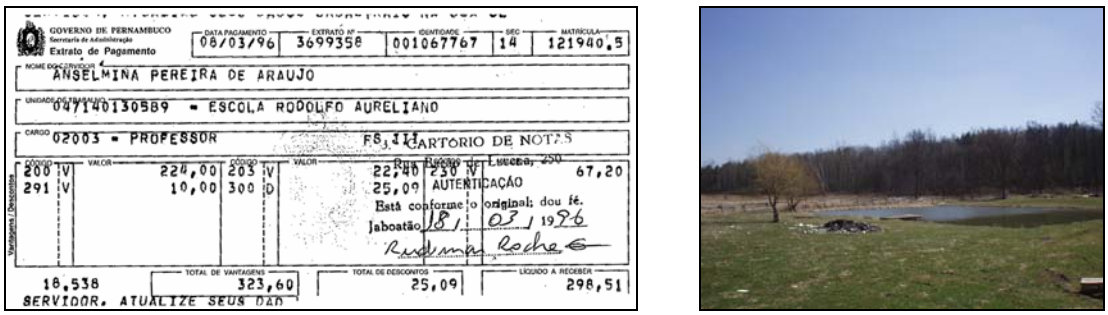
(c) **(d)**

Figura 13 - Exemplos de linhas de texto de diferentes línguas: (a) japonês; (b) hindi; (c) persa; (d) romano.

O conhecimento prévio dos tipos de documentos utilizados pela aplicação torna possível otimizações. A detecção da orientação de formulários digitalizados (figura 14a) 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 utilizados como pontos de referência.



(a) **(b)**

Figura 14 - 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 14b), 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 enviesamento para este tipo de imagem, apenas algoritmos de orientação [34][35][36].

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

2.4. Critérios

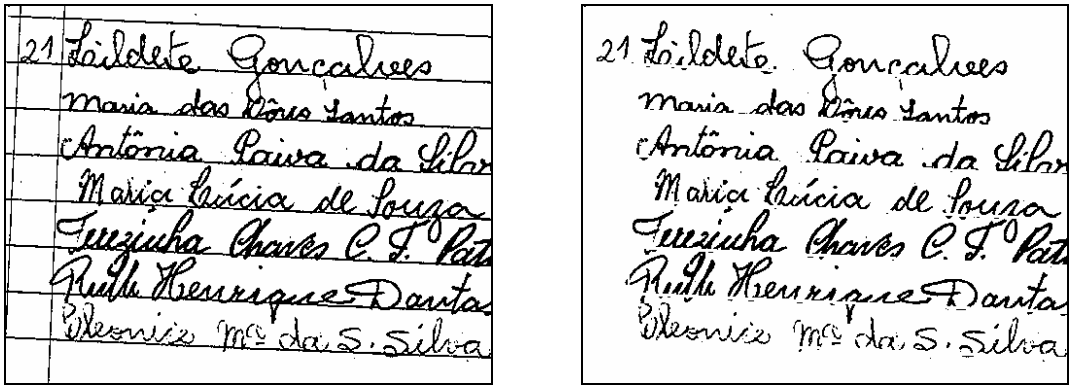
Os critérios para um método de orientação e enviesamento são diferentes para cada tipo de aplicação, contudo, existe um conjunto de critérios que satisfazem à maioria dos casos. Vale a pena a ressaltar a necessidade de discuti-los antes do desenvolvimento de um algoritmo já que afeta a sua construção.

Os critérios para o enviesamento, de acordo com Bloomberg [8], 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 0.1º; (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 [8] 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 necessário para detecção; (6) calcular um valor de confiança entre o valor esperado e o resultado.

Alguns critérios sugeridos por Bloomberg não são válidos. 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 15). 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 15 - 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.

Três características deveriam ser básicas para qualquer método de orientação e enviesamento na maioria dos contextos e nesta ordem de importância: **robustez**, **precisão** e **eficiência**.

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 0.1º;

- **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 à 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. 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.

3. Estado da Arte

Este capítulo apresenta os algoritmos de enviesamento e orientação de documentos digitalizados disponíveis na literatura. Assume-se que os documentos alvo destes algoritmos sofreram os pré-processamentos de remoção de borda, remoção de ruídos e binarização descritos no capítulo anterior. Considera-se aqui que os documentos possuem rotação única e estão excluídos casos mais complexos tais como o apresentado por Spitz em [32] de ângulos múltiplos oriundos, por exemplo da tração não-uniforme durante a impressão ou distorções impostas pelo posicionamento inadequado na digitalização, tal como no caso de livros, etc.

3.1. Algoritmos de Enviesamento

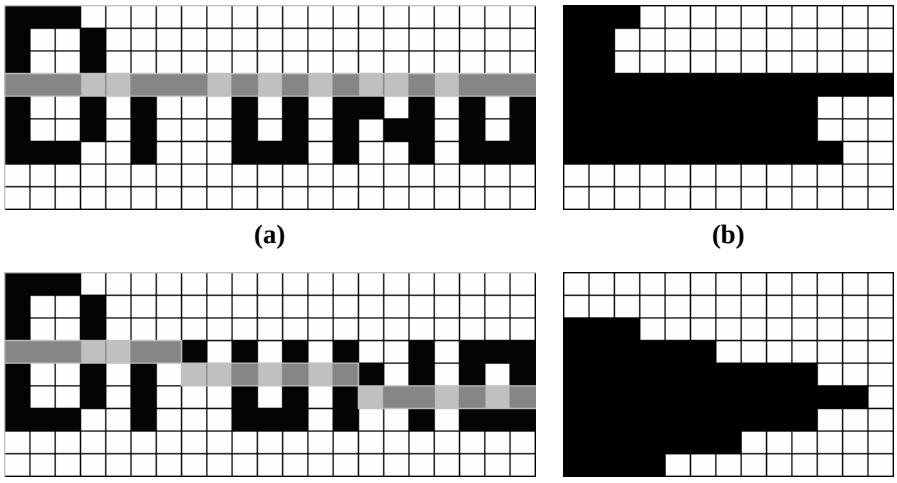
A maioria dos algoritmos de enviesamento pode ser classificada em quatro categorias: projeção de perfil, transformada de Hough, vizinho mais próximo e aproximação da média.

3.1.1. Projeção de Perfil

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 (figura 16). 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 17a), a projeção de perfil horizontal apresenta picos para as linhas de texto e vales para os espaços entre linhas (figura 17b) e a projeção de perfil vertical forma agrupamentos para cada coluna (figura 17c).

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 [27]. 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.

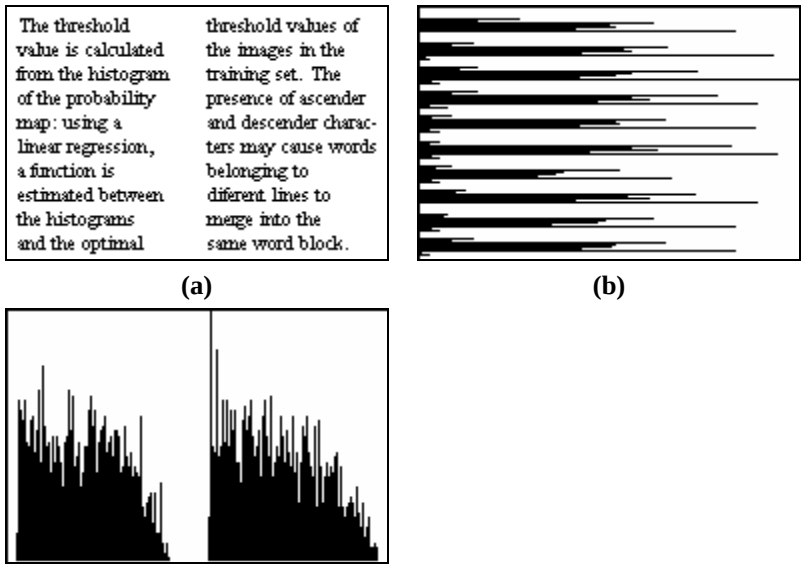


(a) **(b)**

(c) **(d)**

Figura 16 - Exemplo de projeção de perfil: (a) contagem dos pixels pretos de uma linha a 0º; (b) projeção de perfil horizontal da imagem *a*; (c) contagem dos pixels pretos de uma linha rotacionada a 8º; (d) projeção de perfil horizontal da imagem *c*;

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.



(a) **(b)**

(c)

Figura 17 - 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.

O algoritmo mais conhecido desta abordagem foi proposto por Baird [6]. 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 ±15º.

No artigo de Ciardiello *et al* [11], 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 [18], 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).

3.1.2. Transformada de Hough

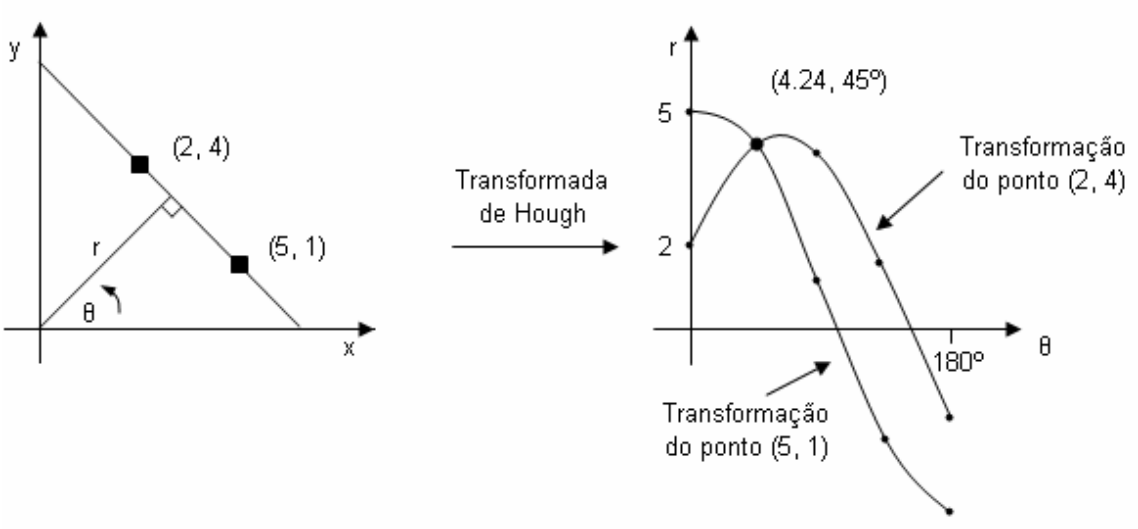
A transformada de Hough é uma técnica conhecida e muito utilizada para aproximação de linhas e curvas [13][17]. Esta abordagem é útil quando o objetivo é aproximar linhas ou curvas utilizando um conjunto de pixels isolados na imagem. O método envolve a transformação do plano de coordenadas da imagem para um espaço parametrizado.

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

$$r = x \cdot \cos\theta + y \cdot \sin\theta \qquad \textbf{(3)}$$

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 intersectarão no mesmo ponto

no plano de Hough. A coordenada do espaço de Hough com o maior acúmulo de pontos mapeados refere-se a uma linha com parâmetros *r* e θ (figura 18). 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 18 - Exemplo de transformada de Hough: (a) imagem com dois pixels pretos; (b) espaço parametrizado de Hough. Ambos os pixels se transformam em uma senóide no espaço de Hough. O ponto de intersecção das senóides refere-se a uma linha com parâmetros *r* e θ passando pelos dois pixels da imagem *a*.

Em aplicações de documentos digitalizados, a transformada de Hough é utilizada para detectar o enviesamento das linhas de texto. Srihari e Govindaraju [33] 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* [16] desenvolveu um algoritmo de enviesamento em que reduz a quantidade de pontos utilizados na transformada de Hough. Primeiro, a resolução da imagem de 300dpi é reduzida por um fator de 4x 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 300dpi), 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 ±45º. A precisão depende da resolução angular atribuída na transformada de Hough.

No trabalho de Le *et al* [19] é descrito um algoritmo de orientação e enviesamento utilizando transformada de Hough. A sub-região melhor 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 ±15º e com uma precisão de 0.5º.

Pal e Chaudhuri [26] 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.

3.1.3. Vizinho mais Próximo

Os métodos desta abordagem 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 ±45º, geralmente até ±90º. Desta forma, estes algoritmos também podem ser considerados algoritmos de orientação retrato/paisagem.

O primeiro trabalho desta abordagem foi proposto por Hashizume *et al* [15]. Para cada componente, computa-se o ângulo formado entre o seu vizinho geometricamente mais próximo (figura 19b) e acumula em um histograma (figura 19c). O pico do histograma refere-se ao ângulo de rotação do documento (figura 19d). Este método é também conhecido como *1-NN* ou *1-Nearest Neighbor*.

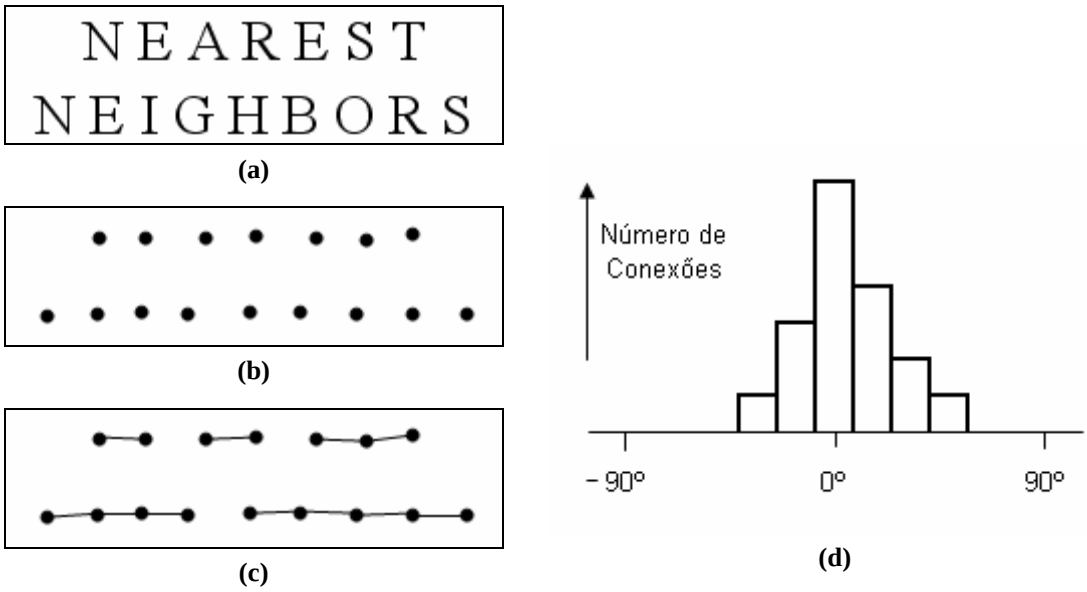


Figura 19 - Exemplo de 1-NN: (a) imagem com texto; (b) centróides, média dos coordenadas dos pixels pretos de cada bloco; (c) conexão de cada bloco com o vizinho mais próximo; (d) histograma dos ângulos formados entre duas conexões. Neste caso, o pico representa o ângulo de 0º.

O'Gorman [24] generaliza o método de Hashizume e estende para *k* vizinhos mais próximos. Este método é conhecido como *k-NN* ou *k-Nearest Neighbor*. Neste 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 20b). 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.

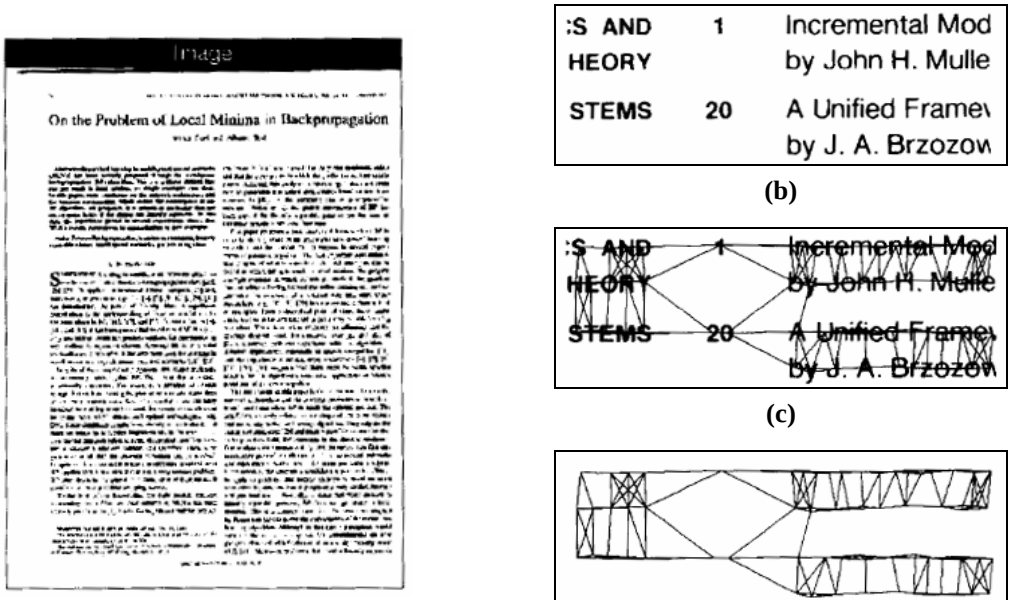


Figura 20 - Exemplo de *k*-NN: (a) imagem original; (b) sub-região da imagem *a*; (c) conexões entre os *k* vizinhos de cada bloco, neste caso, *k* = 5; (d) apenas as conexões entre os vizinhos.

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 21a). 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 21b). O pico do histograma acumulado fornece a distância entre os caracteres da linha de texto.

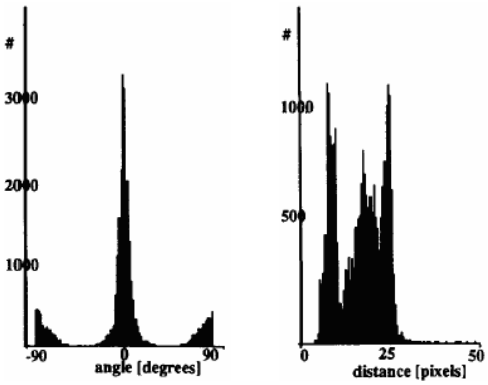
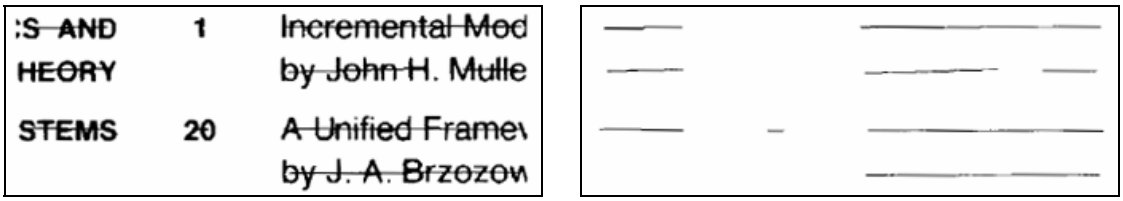


Figura 21 - Histogramas da figura 20a: (a) histograma dos ângulos formado entre os vizinhos; (b) histograma da distância entre vizinhos.

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 22). 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 22 - Formação das linhas de texto: (a) linhas de texto agrupadas; (b) retas formadas pelo método do mínimo quadrado em cada linha de texto agrupado.

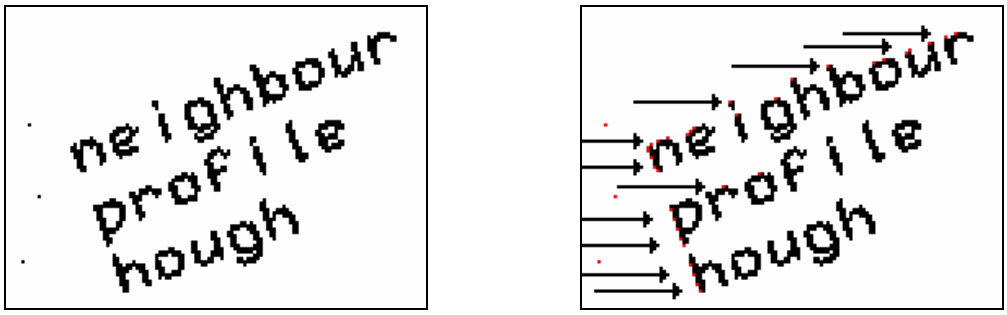
Smith [31] descreve um método baseado na formação de linhas de texto. Como pré-processamento, uma filtragem é aplicada aos componentes cuja altura está fora do intervalo de 20% a 95% da distribuição das alturas dos blocos. Os componentes são ordenados pela coordenada axial e agrupados em linhas de texto 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.

3.1.4. Aproximação da Média

O trabalho de Lins e Ávila [20] representa o último trabalho publicado sobre o assunto, logo, representa o estado da arte dos algoritmos de enviesamento e caracteriza-se por apresentar uma nova abordagem para detecção da rotação.

Uma característica do algoritmo é focar a análise nos pontos pretos mais a esquerda do documento. A média das coordenadas axiais fornece uma linha vertical utilizada como referência para a rotação. A distância horizontal dos pontos mais a esquerda da linha de referência (desvio) é usada para calcular o ângulo de rotação a ser aplicado; a distância horizontal diminui e se aproxima da média representada pela linha de referência. A imagem

rotacionada submete-se ao mesmo processo até que atinja um erro de rotação (condição da terminação).

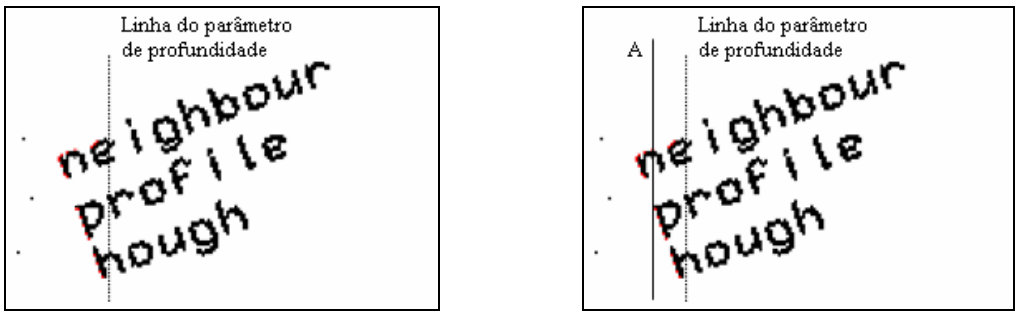


(a) **(b)**

Figura 23 – Exemplo do algoritmo: (a) imagem rotacionada; (b) seleção dos pontos mais à esquerda em vermelho.

O primeiro passo do algoritmo é percorrer de baixo para cima procurando os pontos mais a esquerda de cada linha na imagem (figura 23b). A distância entre o ponto mais a esquerda e os outros pontos selecionados é medida. Um parâmetro de profundidade é usado para evitar os pontos que não pertencem à borda mais a esquerda do documento (figura 24a). A média da abscissa (A) dos pontos restantes mais a esquerda é calculado. A linha vertical A determina a linha de referência para correção da rotação (figura 24b).

Neste passo, os pontos dez por cento mais a esquerda são selecionados e sua média (LA) é calculada (figura 25). A distância entre LA e A é feita como a tangente do ângulo de rotação θ. A posição de LA em relação ao ponto médio de A determina se a rotação é no sentido horário (LA na metade superior de A) ou anti-horário (LA na metade inferior de A).



(a) **(b)**

Figura 24 – Exemplo do algoritmo: (a) pontos selecionados pelo parâmetro de profundidade; (b) calculo da abscissa A.

A imagem formada somente pelos pontos originais mais a esquerda é rotacionada a um ângulo θ utilizando o algoritmo clássico da rotação. O uso de somente dos pontos mais a

esquerda melhora o tempo de processamento, como também preserva a qualidade da imagem evitando a degradação da imagem devido às sucessivas rotações.

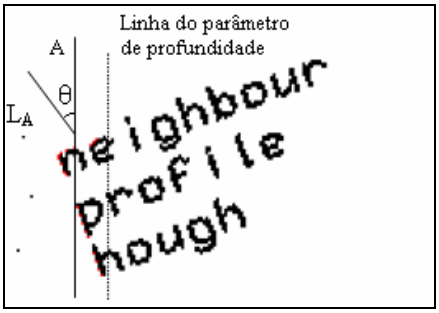


Figura 25 – Exemplo do algoritmo: calculo do ângulo de rotação *θ* e sentido.

A nova imagem é sofre sucessivas iterações até que o novo ângulo de rotação calculado seja menor que o aceito pela aplicação. Este parâmetro depende também da resolução da imagem. Uma outra possibilidade é aplicar o algoritmo até que comece a divergir (a tendência do algoritmo é oscilar em torno de um ponto mínimo). A soma de todos os ângulos sucessivos calculados em cada iteração resulta no ângulo de enviesamento. O algoritmo funciona em uma faixa de ±45º.

3.2. 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.

3.2.1. Algoritmos para Orientação Retrato/Paisagem

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 26a). Primeiro, a projeção de perfil horizontal e vertical é calculado para todo o documento (figura 26b e 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 a vertical, caso contrário, a orientação é paisagem.

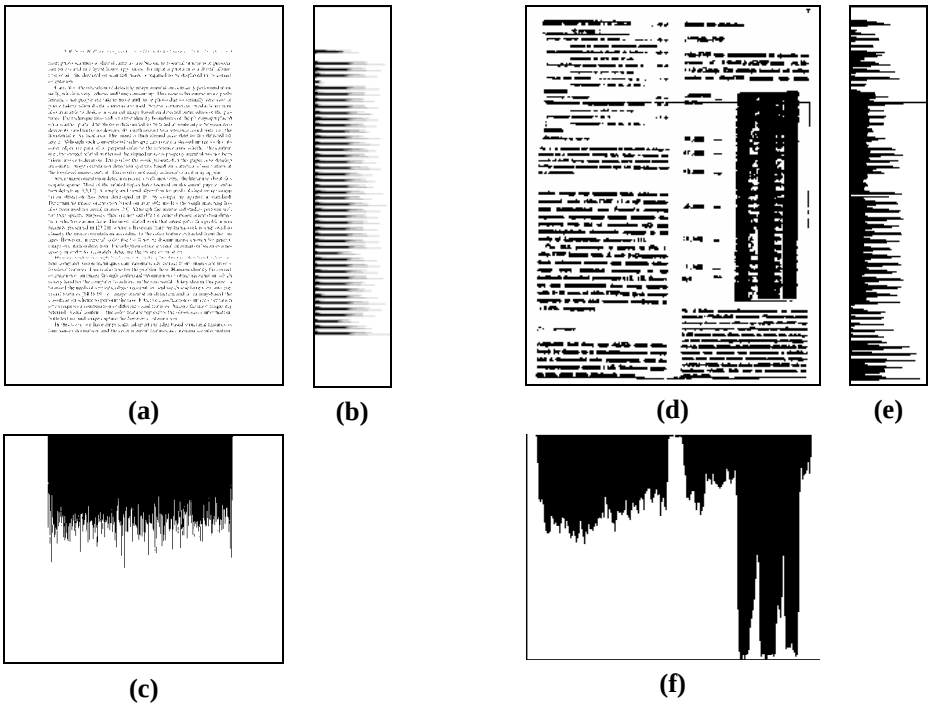


Figura 26 - Dois exemplos do algoritmo de Akiyama: (a) imagem apenas com texto; (b) projeção horizontal; (c) projeção vertical; (d) imagem com figuras; (e) projeção horizontal; (f) projeção vertical.

Este algoritmo fornece uma estimativa global da orientação do documento, logo, é sensível a presença de blocos não considerados texto (figura 26d). Le *et al* [19] 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 27). 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.

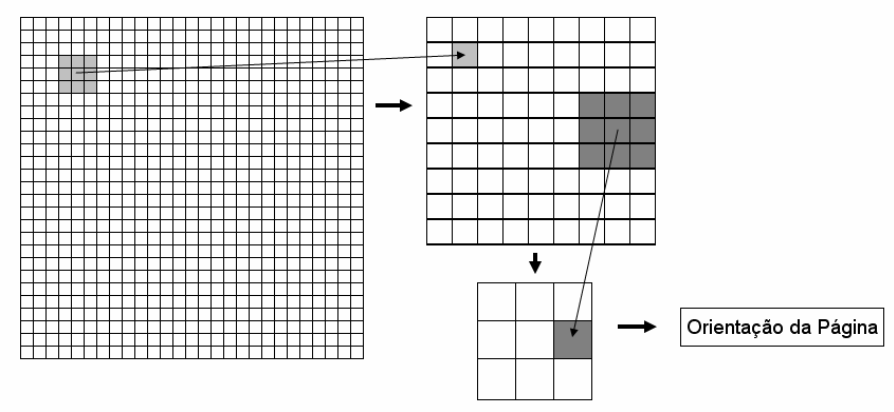


Figura 27 - Estrutura da pirâmide para detectar a orientação retrato/paisagem proposto por Le.

3.2.2. Algoritmos para Orientação Invertida

Algoritmos desta categoria só podem ser aplicados em textos de escrita romana e números arábicos (figura 28), 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.

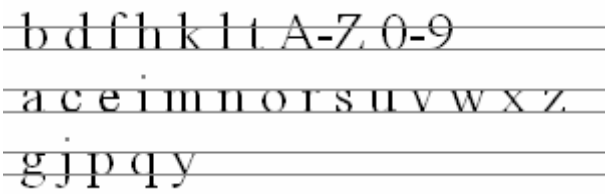


Figura 28 - Caracteres alfanuméricos agrupados de acordo com suas saliências para cima (primeira linha), para baixo (última linha) e sem saliências (linha do meio).

Bloomberg [8] 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 4x e, depois, é dilatada horizontalmente, na tentativa de juntar os caracteres de uma palavra e as palavras de uma linha de texto, no sentido horizontal (figura 29b).

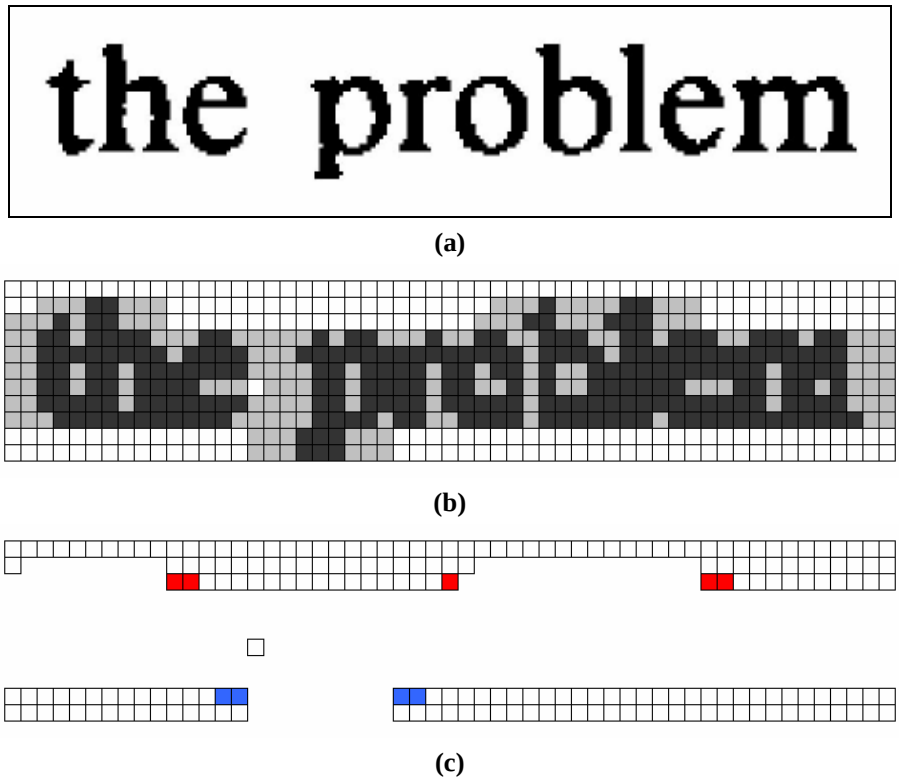


Figura 29 - Exemplo de orientação invertida proposto por Bloomberg: (a) imagem original; (b) imagem reduzida 4x e dilatada horizontalmente com largura sete, os quadrados em cinza claro referem-se aos pixels gerados pela dilatação, os quadrados em cinza escuro referem-se aos pixels gerado pela redução da resolução da imagem original; (c) os quadrados em vermelho são os pixels das letras para cima e os em azul são os pixels das letras para baixo.

Para identificar as letras com saliência para cima e para baixo, quatro estruturas de vizinhança são utilizadas (figura 30). As duas estruturas na esquerda são para contar os pixels na esquerda e na direita das letras com saliência para cima; as duas estruturas da direita são para contar os pixels da esquerda e da direita das letras com saliência para baixo.

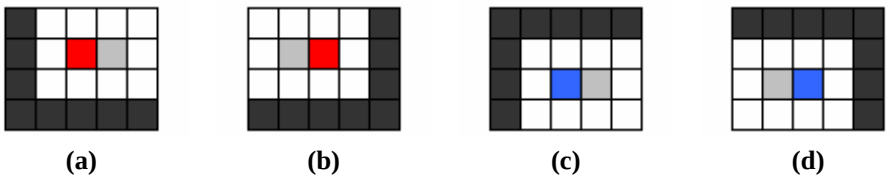


Figura 30 - Estruturas utilizadas para contar os pixels para cima e para baixo. Os quadrados em cinza escuro referem-se aos pixels pretos dilatados, em cinza claro referem-se aos pixels brancos, em branco referem-se aos pixels de qualquer cor, em vermelho e azul referem-se ao centro da estrutura em que os pixels das letras com saliência para cima e para baixo, respectivamente, devem estar.

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 Números Grandes*: a variância esperada em cada um desses números é proporcional às suas raízes quadradas. A probabilidade de que duas populações possam ser distintas pode ser estimada pela raiz quadrada da soma das variâncias individuais, forma:

$$\sigma_o = \frac{\sqrt{N_{cima} + N_{baixo}}}{2} \quad \textbf{(3)}$$

O *sinal de orientação normalizado* é definido como a diferença entre o número de pixels para cima e para baixo, expresso como um múltiplo de σ_o:

$$\overline{S}_{orientação} = \frac{2 \cdot \left| N_{cima} - N_{baixo} \right|}{\sqrt{N_{cima} + N_{baixo}}} \quad \textbf{(4)}$$

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 deve ser rotacionados em 90º.

3.3. Resumo

Um resumo com os algoritmos estudados neste capítulo é organizado segundo as principais características de cada um. As imagens utilizadas pelos algoritmos citados foram pré-processados utilizando os filtros de binarização, remoção de borda e remoção de ruído. O sumário do capítulo para algoritmos de enviesamento é: quatro métodos da abordagem Projeção de Perfil, cinco métodos que utilizam transformada de Hough, três algoritmos da classe Vizinho mais Próximo e o método que representa o estado da arte com nova abordagem Aproximação da Média. O resumo é apresentado na tabela 3.

Abordagem	**Autores**	**Faixa de ângulo / Precisão**	**Tipo de documento**	**Comentários**
Projeção de Perfil (*Projection Profile*)	Postl [27]	±45º / 0.6º	Documentos complexos; texto com direção dominante	
	Baird [6]	±15º / 0.05º	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* [11]	±45º / 0.7º	Documentos complexos	
	Ishitani [18]	±30º / 0.12º	Documentos complexos e com poucas linhas de texto	
Transformada de Hough (*Hough Transform*)	Srihari e Govindaraju [33]	±90º / 1º	Documentos com apenas textos	
	Hinds *et al* [16]	±15º / 0.5º	Documentos complexos	Uma estimativa da altura máxima dos caracteres é necessária
	Le *et al* [19]	±15º / 0.5º	Documentos complexos	
	Pal e Chaudhuri [26]	±45º / 0.2º	Documentos complexos com predominância de textos; texto com apenas uma direção; símbolos romanos	
	Amin e Fisher [3]	±45º	Documentos complexos; texto com direção dominante	
Vizinho mais Próximo (*Nearest Neighbor*)	Hashizume *et al* [15]	±90º / 5º	Documentos simples	Espaços entre linhas de texto devem ser maiores que o espaço entre os caracteres
	O'Gorman [24]	±90º	Documentos complexos; múltiplas direções	Generalização do método de Hashizume
	Smith [31]	±15º / 0.05º	Documentos complexos; texto com apenas uma direção	
Aproximação da Média	Lins e Ávila [20]	±45º / 0.1º	Documentos complexos com predominância de texto	Representa o estado da arte. Apresenta desempenho mais rápido que o método de Baird, contudo os autores afirmam que falta melhorar a precisão do algoritmo

Tabela 3 - Resumo dos algoritmos de enviesamento apresentados neste capítulo.

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 4.

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* [19]	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* [8]	Operações Morfológicas	Documentos complexos com predominância de texto	

Tabela 4 - Resumo dos algoritmos de orientação apresentados neste capítulo.

4. Algoritmos Propostos

Este capítulo apresenta dois novos algoritmos: (1) algoritmo de enviesamento e orientação utilizando a abordagem Vizinho mais Próximo; (2) algoritmo de enviesamento utilizando a abordagem Aproximação da Média. Para ambos, assume-se que os algoritmos funcionam em imagens monocromáticas.

4.1. Algoritmo de Enviesamento e Orientação utilizando Vizinho mais Próximo

Um algoritmo de enviesamento e orientação são propostos 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.

4.1.1. Características

O algoritmo proposto 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 0.1º. Apenas a característica **eficiência** não foi priorizada no desenvolvimento para não haver perdas na robustez e precisão, contudo, todas as otimizações possíveis foram implementadas.

4.1.2. Rationale

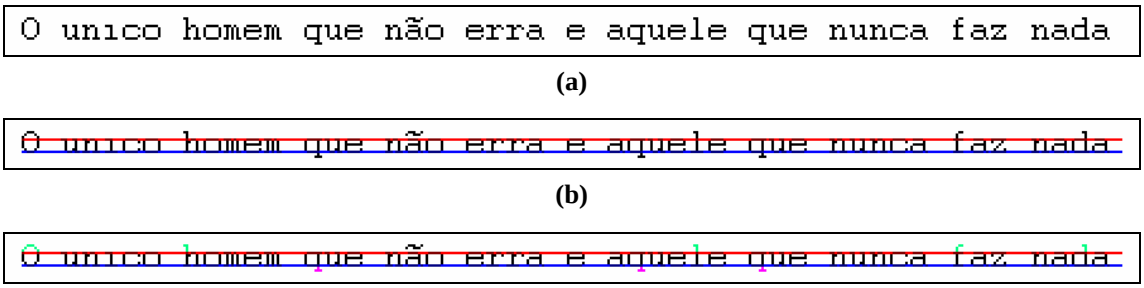
A idéia básica deste algoritmo é utilizar as linhas de texto como ponto de referência garante a detecção de enviesamento e orientação retrato/paisagem para qualquer documento. Fazer uso das características dos símbolos das linhas de texto torna possível o cálculo da orientação invertida para documentos 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 ±90º. 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 utilizado 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.



(a)

(b)

(c)

Figura 31 - 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 31a, 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 31b). O valor 3.698 ($2x43^2$ – 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 31c, 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 31c, é acumulada na variável *down*. Neste caso, a variável *up* é maior que *down*, logo, a orientação não está invertida.

Para linhas de texto com símbolos de dimensões semelhantes, como por exemplo, linhas de texto apenas com letras maiúsculas (figura 32a), 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 32b).

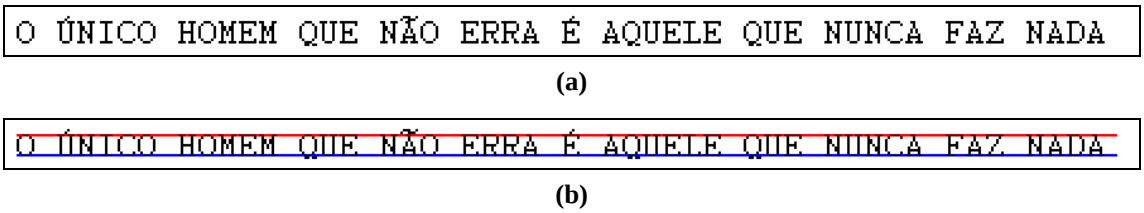


Figura 32 - Outro exemplo do algoritmo proposto: (a) linha de texto apenas com letras maiúsculas; (b) as duas retas formadas utilizando o MMQ nos pontos médios superiores, em vermelho, e nos pontos médios inferiores, em azul.

4.1.3. Algoritmo

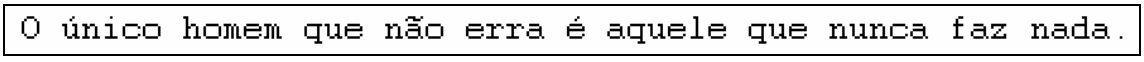
Nesta seção, o algoritmo é apresentado na forma de pseudocódigo e alguns detalhes serão explicitados a seguir.

Os passos principais do algoritmo são: (1) executar a nomeação de componentes 8x8; (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:

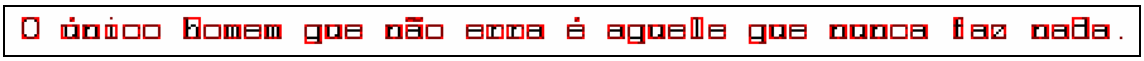
```
blocos = nomeacao_componentes (imagem);
remover_ruido (blocos);
inicializar (histograma_grosso, 0);
inicializar (histograma_fino, 0);
inicializar (up, 0);
inicializar (down, 0);

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);
```

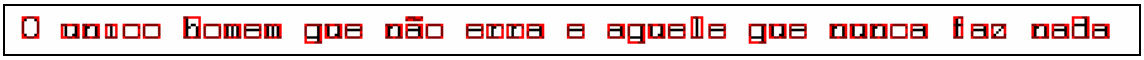
O primeiro passo é executar os algoritmos de pré-processamento nomeação de componentes 8x8 e remoção de ruídos (figura 33). Em seguida, a partir de cada bloco não marcado (*bloco*), um vizinho mais próximo (*localizar_vizinho_mais_proximo*) é localizado e uma linha de texto (*agrupar_linha_texto*) é 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 33 - 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;
```

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 34). 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 34 - Exemplo de localizar o vizinho mais próximo: (a) sub-região do exemplo anterior com o bloco atual (*bloco*) selecionado em azul; (b) a quarta iteração para localizar o vizinho, em azul; (c) a penúltima iteração; (d) o vizinho (*vizinho*), em verde, localizado na esquerda, portanto, a direção (*direcao*) 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:

```
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;
```

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 35b). Estas duas retas formam uma faixa delimitando uma sub-região na imagem para a procura dos próximos dois vizinhos.

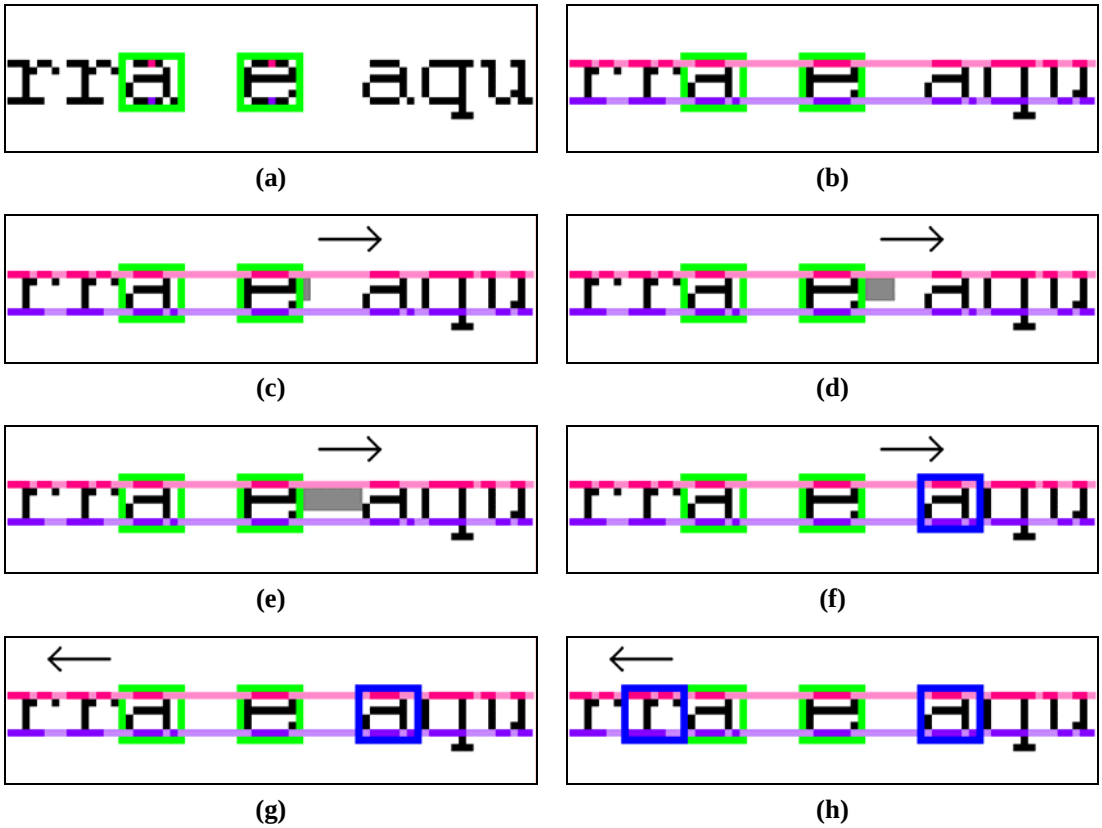


Figura 35 - Exemplo do processo de agrupamento de linha de texto: (a) os dois primeiros elementos da linha de texto, *bloco* e *vizinho*, e os pontos médios superiores e inferiores em rosa e roxo, respectivamente; (b) as retas *reta_superior* e *reta_inferior*, em rosa e roxo, respectivamente; (c) primeira 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;

Em um sentido (figura 35c), inicia-se a busca por um pixel de outro vizinho (*vizinho_fim*); em sentido contrário (figura 35g), 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.

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;
```

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:

```
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;
```

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 a*ngulo_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:

```
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;
```

4.2. Algoritmo de Enviesamento utilizando Aproximação da Média

Um 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.

4.2.1. Características

O algoritmo proposto pertence à classe Aproximação da Média e refere-se à detecção de enviesamento na faixa de ±45º. O método sugerido inicialmente por Lins e Ávila [20], não apresenta uma boa precisão, assim como, uma alta média de erro e, portanto, este algoritmo foi desenvolvido com o objetivo de melhorar a precisão e diminuir os erros, enquanto mantém o excelente desempenho.

Seguindo os critérios e prioridades estabelecidas no capítulo 2, este algoritmo diminui a sua **robustez** fazendo suposições acerca dos tipos de documentos com a finalidade de aumentar a **eficiência** da detecção. A principal suposição é que a parte inferior do documento deve possuir linhas de texto. Isto se deve basicamente a estratégia de escolha dos pontos utilizados pelo algoritmo. Em relação à **precisão**, o algoritmo foi desenvolvido para obter uma precisão de 0.1º.

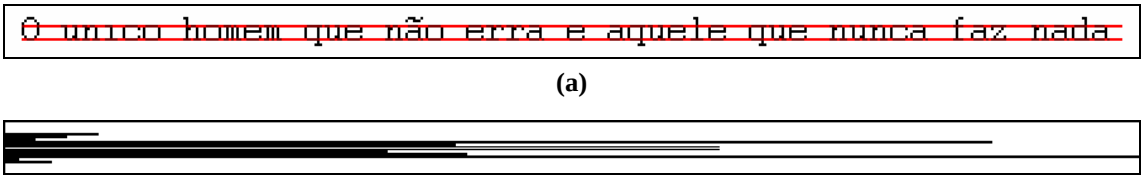
4.2.2. Rationale

Para melhorar a precisão e diminuir o erro de detecção, os pontos são selecionados de baixo para cima, um para cada coluna da imagem, ao contrário, da versão inicial que era da esquerda para a direita. Esta mudança é justificada pelo motivo de 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.

A seleção de uma sub-região é adotada 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 200dpi, a altura da janela é 200 pixels). A sub-região com mais pontos é selecionada e utilizada em todo o algoritmo.

Outra mudança refere-se à utilização da moda para decisão sobre a busca do ângulo. A projeção de perfil horizontal (figura 36b) 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 36a. 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.

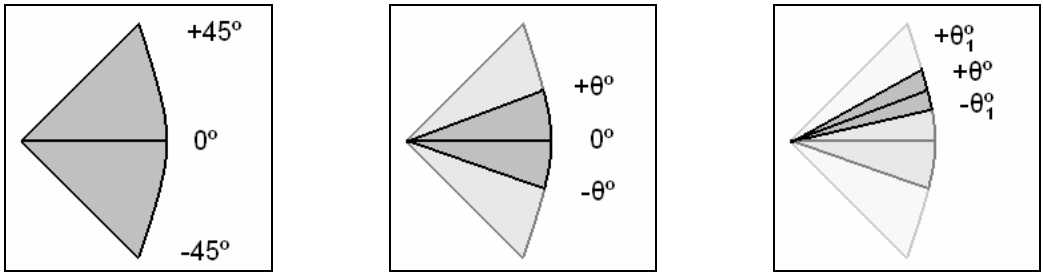


(a)

(b)

Figura 36 - 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.

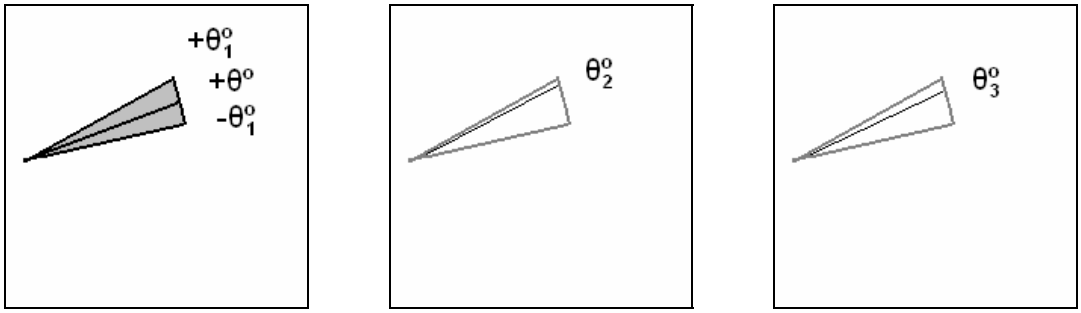
A estratégia de busca do tipo *bifurcação* é novamente utilizada porque é um dos recursos que aumentam o desempenho do algoritmo. A bifurcação consiste em definir dois limites em uma faixa de valores e, a cada iteração sucessiva, diminuir a faixa de valores aproximando os dois limites até o valor desejado (figura 37).



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

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

A principal idéia 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 dos pontos e calcula-se outra média das abscissas dos pontos acima da primeira média. Estas duas médias são utilizadas 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)** **(c)**

Figura 38 - Exemplo de busca utilizando varredura: (a) faixa de ângulo, em cinza; (b) primeira iteração; (c) segunda iteração.

A estratégia de busca do tipo *varredura* é introduzida 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 38).

4.2.3. Algoritmo

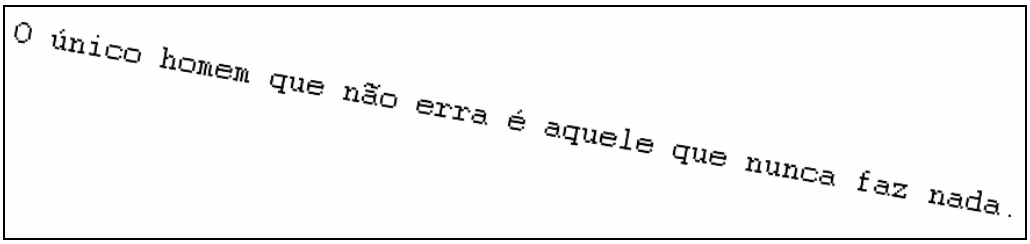
Nesta seção, o algoritmo é apresentado na forma de pseudocódigo e alguns detalhes serão explicados a seguir.

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:

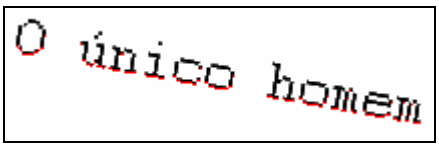
```
pontos          = selecionar_pontos_baixo_cima (imagem);
pontos_subregiao = selecionar_subregiao (imagem, pontos);
eixo_rotacao     = selecionar_eixo_rotacao (pontos_subregiao);
angulo_grosso    = localizar_angulo_bifurcacao
                       (pontos_subregiao, eixo_rotacao);
angulo_fino   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

```
                    (pontos_subregiao, eixo_rotacao, angulo_grosso);
```

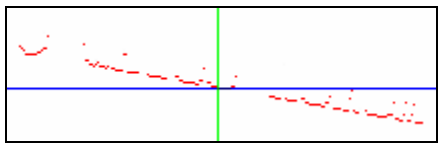
O primeiro passo consiste em selecionar o primeiro pixel preto de baixo para cima de cada coluna da imagem (*pontos*). Em seguida, seleciona-se uma sub-região de altura igual à resolução da imagem contendo a maior quantidade de pixels selecionados no primeiro passo (figura 39b). Todos os passos seguintes 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 39c): 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 0.1º.



(a)



(b)



(c)

Figura 39 - Exemplo do algoritmo proposto: (a) linha de texto rotacionada a -10º; (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 1º é feita através de uma estratégia de busca do tipo *bifurcação*. O pseudocódigo deste passo é:

```
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;
```

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 ±45º formando uma faixa de ângulos com precisão 1º. O *angulo_grosso* é inicializado como o ângulo médio dos dois limites, neste caso, 0º. 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 40, a *media*, 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.

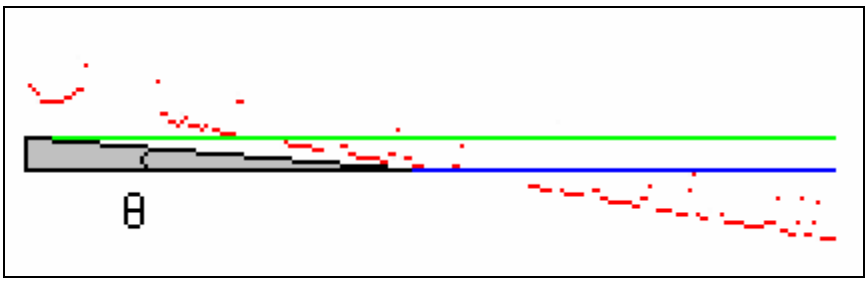


Figura 40 - Exemplo do cálculo de *theta*: *media*, em azul, *media_desvio*, em verde e o triângulo utilizado no cálculo de *theta*, em cinza.

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 0.1º 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, 0.1º);
     end;
     return angulo_fino;
end;
```

A partir do ângulo grosso (*angulo_grosso*) detectado, uma faixa de ângulo é definida como *angulo_grosso* ± *1º* com uma precisão de 0.1º. 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 0.1º. O ângulo, cuja moda é a maior, refere-se ao ângulo fino (*angulo_fino*) ou ângulo de enviesamento.

5. Metodologia e Análise de Resultados

Uma série de testes comparativos entre algoritmos de enviesamento e orientação foi realizada seguindo os critérios definidos no capítulo 2 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.

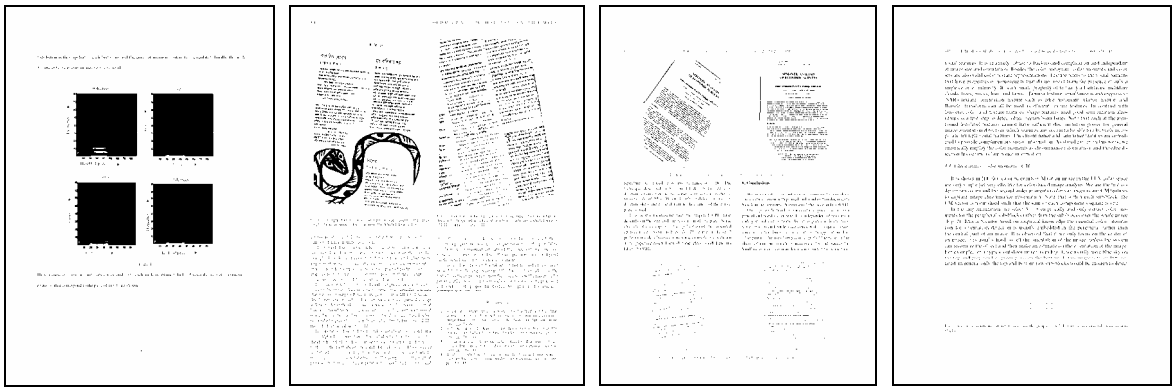
5.1. Metodologia

Para o ambiente de execução dos testes, utilizou-se um computador com processador Pentium IV de 2.4GHz, memória RAM de 512MB 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 [42] utilizando compressão CCITT Group 4. A biblioteca LibTIFF 3.6.1 [40] foi utilizada para manipulação de imagens TIFF.

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) 200dpi, uma vez que resolução guarda todos os elementos essenciais do documento oferecendo um bom fator qualidade/espaço de armazenamento [23] e transmissão via rede de computadores [22].

Os resultados dos testes serão analisados segundo o conjunto de critérios definidos no capítulo 2.

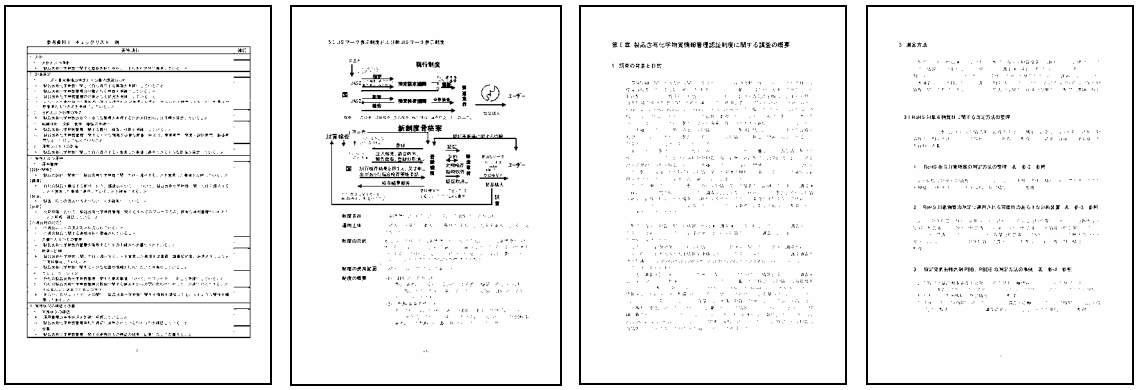
Um algoritmo de detecção de rotação deve ter uma dependência mínima em relação ao “layout” dos documentos e, desta forma, 270 documentos de tamanho A4, retirados de artigos científicos, escritos em inglês e feitos no computador com os mais diversos tipos de “layout” foram utilizados nos testes (figura 41).



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

Figura 41 - 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.

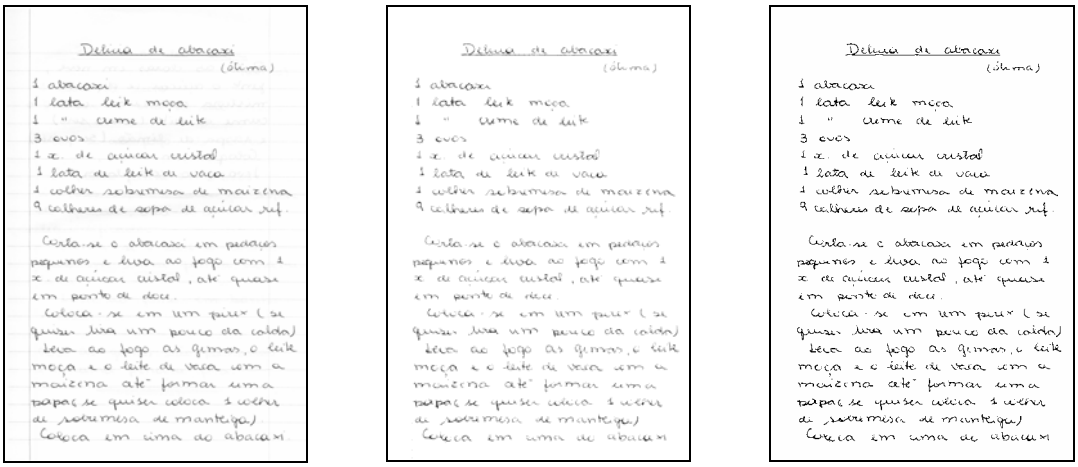
Em relação ao critério de dependência mínima da língua em que o texto foi escrito, 50 documentos em japonês com texto escrito horizontalmente da esquerda para a direita, de tamanho A4 e com vários tipos de “layout” foram testados (figura 42).



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

Figura 42 - 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 [37], 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 43 - Exemplo do tratamento do documento manuscrito utilizado nos testes: (a) documento digitalizado em tons de cinza e guiado pelas linhas do papel para corrigir o alinhamento; (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 [6] e o proposto na seção 4.1. O algoritmo de Baird foi escolhido por ser considerado, pelo próprio autor, o mais eficiente e com menor média de erro entre os outros métodos da literatura. Contudo, testes realizados por Amim 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 0.1º; (3) funcionam em imagens monocromáticas de 200dpi de resolução.

Todas as imagens foram rotacionadas e testadas nos seguintes ângulos: 0.0º, ±0.1º, ±0.2º, ..., ±0.9º, ±1.0º, ±2.0º, ..., ±14.0º, ±15.0º, ±20.0º, ±30.0º, ..., ±170.0º e 180º. Para os documentos em japonês, foram rotacionadas e testadas apenas entre ±90.0º, 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 4.1 refere-se à detecção do enviesamento e está restrito à faixa de ±15º, por causa da limitação do algoritmo de Baird. O algoritmo proposto na seção 4.1 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 mili-segundos (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;
- **Faixa de confiança:** porcentagem de acerto da detecção segundo uma faixa de tolerância de erro (0.0º, 0.1º, 0.2º).

5.2. Análise de 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:

Algoritmo	Tempo médio	Média do erro	Desvio padrão do erro	Erro máximo	Faixa de confiança (%)		
					0.0º	0.1º	0.2º
Baird	95 ms	0,004º	0,020º	0,2º	96,02 %	99,97 %	100 %
Algoritmo proposto	115 ms	0,001º	0,012º	0,1º	98,60 %	100 %	100 %

Tabela 5 - Resultados do primeiro teste entre os dois algoritmos com documentos digitalizados datilografados, em inglês e com vários tipos de "layout".

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:

Algoritmo	Tempo médio	Média do erro	Desvio padrão do erro	Erro máximo	Faixa de confiança (%)		
					0.0º	0.1º	0.2º
Algoritmo proposto	131 ms	0,022º	0,871º	180,0º	98,29 %	99,73 %	99,94 %
Algoritmo proposto sem orientação invertida	131 ms	0,002º	0,013º	0,1º	98,34 %	100 %	100 %

Tabela 6 - Resultados do segundo teste apenas com o algoritmo proposto utilizando a mesma base de documentos do primeiro teste.

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 6 na primeira linha. Corrigindo as imagens e calculando novamente sem os efeitos da orientação invertida (tabela 6, segunda linha), a média de erro caiu em dez vezes e o erro máximo foi reduzido para 0.1º. 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:

Algoritmo	Tempo médio	Média do erro	Desvio padrão do erro	Erro máximo	Faixa de confiança (%)		
					0.0º	0.1º	0.2º
Baird	86 ms	0,005º	0,023º	0,1º	94,20 %	100 %	100 %
Algoritmo proposto	125 ms	0,035º	0,070º	0,8º	71,83 %	95,63 %	98,48 %

Tabela 7 - Resultados do teste dos algoritmos em documentos escritos na lingua japonesa.

Neste teste, o algoritmo de Baird obteve melhores resultados em todos os indicadores, sendo recomendado para detecção de enviesamento na faixa de ±15.0º 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 ±90.0º, totalizando 3.250 imagens. Os resultados foram:

Algoritmo	Tempo médio	Média do erro	Desvio padrão do erro	Erro máximo	Faixa de confiança (%)		
					0.0º	0.1º	0.2º
Algoritmo proposto	155 ms	0,040º	0,075º	0,9º	68,27 %	88,55 %	97,38 %

Tabela 8 - Resultados do teste do algoritmo proposto em documentos escritos na lingua japonesa.

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:

Algoritmo	Tempo médio	Média do erro	Desvio padrão do erro	Erro máximo	Faixa de confiança (%)		
					0.0º	0.1º	0.2º
Baird	24 ms	0,584º	1,376º	15º	13,30 %	35,83 %	52 %
Algoritmo proposto	33 ms	0,611º	4,090º	78,6º	11,22 %	32,32 %	52,85 %

Tabela 9 - Resultados do teste dos algoritmos em documentos manuscritos.

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.

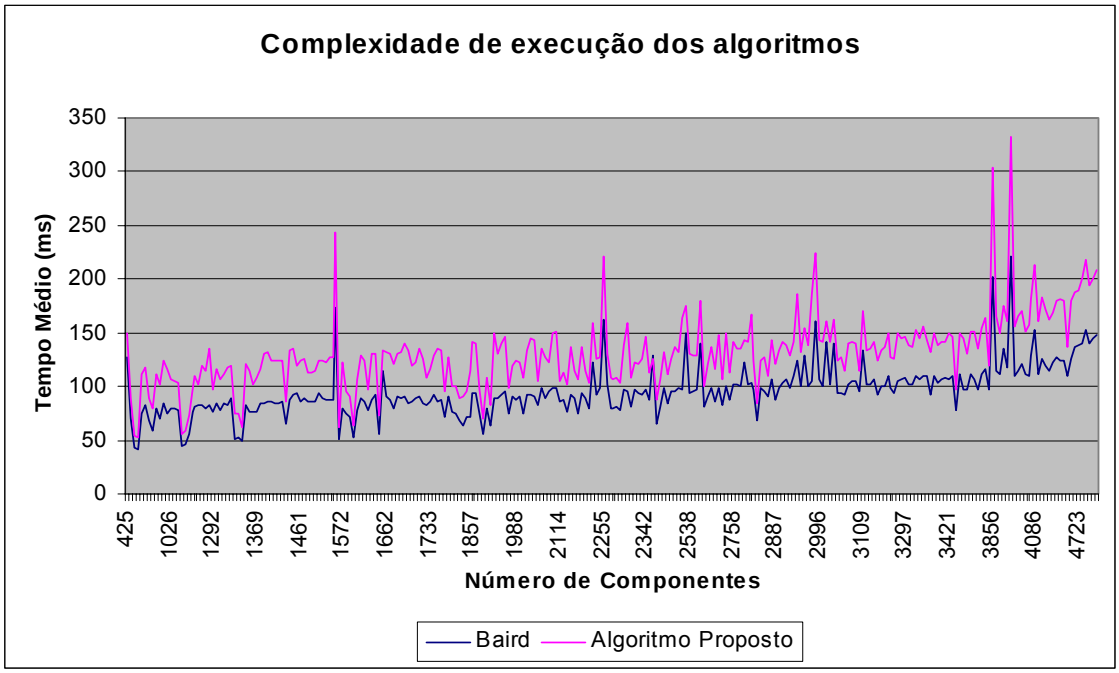


Gráfico 1 - Complexidade de execução do algoritmo de Baird e do proposto.

Para analisar a complexidade de execução de cada algoritmo, 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 [24].

6. Conclusão

Este trabalho tem a finalidade de estudar e aprimorar os algoritmos de orientação e enviesamento de documentos digitalizados apresentando os conceitos relevantes, algoritmos correlatos, critérios para avaliação, além dos algoritmos existentes na literatura e o estado da arte no assunto. Dois novos algoritmos de abordagens diferentes foram demonstrados e testes comparativos foram executados com diversos tipos de documentos.

Baseado nos estudos dos algoritmos existentes na literatura e citados neste trabalho, detectou-se uma particularidade em comum nas técnicas apresentadas: todos utilizam as linhas de texto para servir de ponto de referência no cálculo da rotação do documento. Visto que todas as línguas do mundo organizam os textos em linhas de texto e que todos os documentos digitalizados apresentam um mínimo de linhas de texto, os algoritmos de orientação e enviesamento utilizam alguma característica das linhas de texto para detectar o ângulo de rotação dos documentos. Este ponto de referência também foi utilizado para a construção dos algoritmos propostos.

Parâmetros e limites necessários para a construção e avaliação dos algoritmos de orientação e enviesamento foram apresentados por Bloomberg [8] e complementados neste trabalho. Novos critérios essenciais e relevantes foram definidos, classificados e priorizados em três características distintas: robustez, precisão e eficiência. A construção dos algoritmos propostos e os testes realizados foram guiados pelos critérios e características apresentados.

Dois algoritmos foram propostos utilizando abordagens diferentes: Vizinho mais Próximo e Aproximação da Média. O primeiro algoritmo segue os critérios definidos no capítulo 2 e prioriza a robustez. Este método é o primeiro algoritmo integrado da literatura capaz de detectar a rotação do documento em qualquer ângulo, assim como, é o primeiro com complexidade de execução linear entre os algoritmos da abordagem Vizinho mais Próximo. Baseado no conhecimento prévio das características dos documentos em que será executado, o segundo algoritmo diminui a robustez e prioriza a eficiência. Este algoritmo apresenta ajustes finos para o método inicialmente proposto por Lins e Ávila [20].

Cinco testes foram realizados de forma a comparar e avaliar os algoritmos implementados segundo os critérios apresentados no capítulo 2. O algoritmo de Baird e o proposto na seção 4.1 foram comparados em 27.840 imagens e apresentaram os seguintes resultados:

- O algoritmo de Baird é ligeiramente mais rápido e limitado a detecção de rotação na faixa de ±15.0º, entretanto, o algoritmo proposto demanda maior tempo devido a sua capacidade de detectar a rotação em uma faixa de ângulo doze vezes maior;
- Para documentos com textos datilografados em inglês e com vários tipos de "layout", a taxa de acerto da detecção do enviesamento do algoritmo proposto é melhor e apresenta um erro máximo de 0.1º. As taxas de acerto do algoritmo proposto em relação às orientações retrato/paisagem e invertida são 100% e 99.95%, respectivamente;
- Para documentos datilografados em japonês, o algoritmo de Baird apresentou ligeira melhora em relação ao algoritmo proposto;
- Para documentos manuscritos em letra corrida, o algoritmo de Baird e o proposto obtiveram uma performance razoável e apresentaram uma faixa de confiança com tolerância de 1.0º de 91.18% e 95.67%, respectivamente.

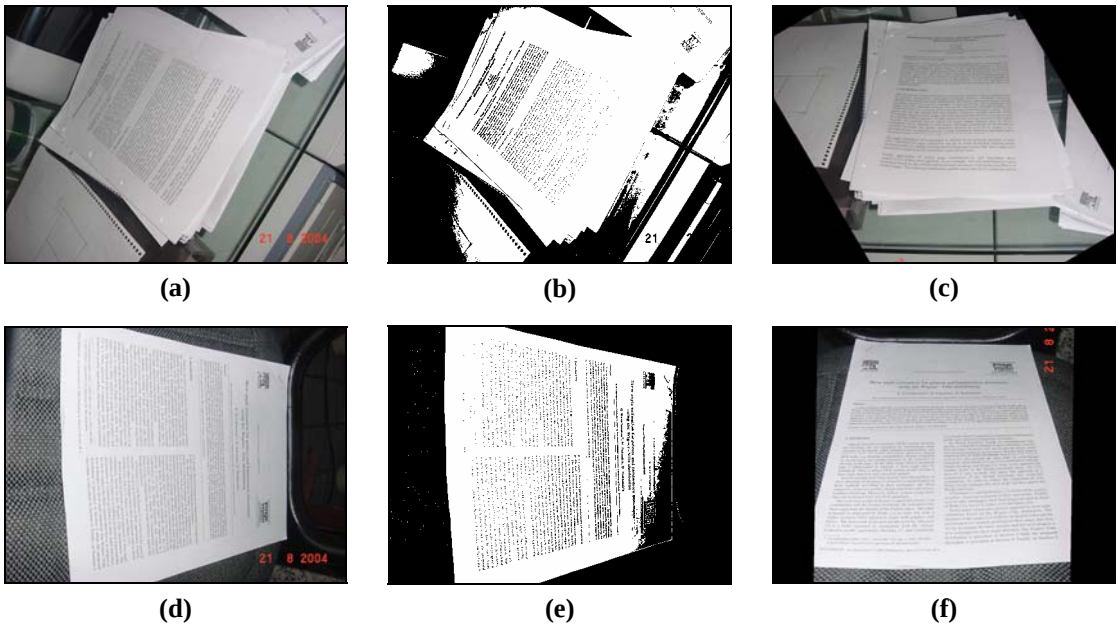
6.1. Dificuldades

Algumas dificuldades foram encontradas no desenvolvimento deste trabalho. O principal problema encontrado foi a indisponibilidade de acesso aos artigos relevantes ao estudo do problema de modo que trabalhos importantes, como o de Baird [6], não foi possível obter detalhes e explicações originais dos autores, entretanto, consultas a outros artigos, possibilitaram o entendimento do método proposto pelos autores.

Outra dificuldade foram as faltas de detalhamento e esclarecimento de partes importantes dos algoritmos encontrados nos artigos da literatura o que inviabilizou a implementação e comparação nos testes.

6.2. Trabalhos Futuros

Tendo em vista os bons resultados obtidos por um dos algoritmos propostos nos testes, ajustes finos serão sugeridos e testados de forma a obter melhores resultados em relação à robustez, precisão e eficiência. Uma das possíveis modificações é criar um fator de confiança para a orientação invertida que funcione corretamente em textos apenas com letras maiúsculas.



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

(d) **(e)** **(f)**

Figura 44 - 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 -59.6º pelo algoritmo proposto na seção 4.1; (d) documento fotografado distorcido e com orientação paisagem; (e) imagem *d* binarizada; (f) imagem *d* rotacionada com o ângulo detectado de 89.1º pelo algoritmo proposto na seção 4.1.

Outra possibilidade de trabalho futuro é o estudo de tipos de documentos originados de fontes ainda não exploradas na literatura, como por exemplo, fotografias de documentos que trazem novos desafios, tais como, regiões distorcidas (figura 44).

7. Referências Bibliográficas

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

[2] N. F. Alves. 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] A. Amin and S. Fisher. A Document Skew Detection Method Using the Hough Transform. *Pattern Analysis & Applications*, vol. 3, no 3, pp 243-253, September, 2000;

[4] B. T. Ávila and R. D. Lins. A New Algorithm for Removing Noisy Borders from Monochromatic Documents. *In Proc. of ACM-SAC'2004*, pp 1219-1225, Chipre, ACM Press, March, 2004;

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

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

[7] D. S. Bloomberg. Image analysis using threshold reduction. *SPIE Conference on Image Algebra and Morphological Image Processing II*, vol. 1568, San Diego, California, pp 38-52, July, 1991;

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

[9] R. S. Caprari. Algorithm for text page up/down orientation determination. *Pattern Recognition Letter*, Elsevier, vol. 21, pp 311-317, 2000;

[10] S. Chien and Y. Baek. A fast black run rotation algorithm for binary images. *Proceedings of Pattern Recognition Letters*, 1998, Elsevier, vol. 19, pp 455-459;

[11] G. Ciardiello, G. Scafuro, M. T. Degrandi, M. R. Spada and M. P. Roccotelli. An experimental system for office document handling and text recognition. *In Proc. of the*

9th International Conference on Pattern Recognition, volume 2, pp 739-743, Roma, Itália, November, 1988;

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

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

[14] M. B. Dillencourt, H. Samet and M. Tamminen. A General Approach to Connected-Component Labeling for Arbitrary Image Representations. *Journal of the Association for Computing Machinery*, vol. 39, nº 2, pp 253-280, April, 1992;

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

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

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

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

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

[20] R. D. Lins and B. T. Ávila. A New Algorithm for Skew Detection in Images of Documents. International Conference on Image Analysis and Recognition, Springer Verlag, LNSC, Porto, Portugal, September, 2004;

[21] R. D. Lins, M. S. Guimarães Neto, L. R. França Neto and L. G. Rosa. An Environment for Processing Images of Historical Documents. *Microprocessing & Microprogramming*, pp. 111-121, North-Holland, January, 1995;

[22] R. D. Lins and D. S. A. Machado. A comparative study of file formats for image storage and transmission. *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 175-183, 2004;

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

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

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

[26] U. Pal and B. B. Chaudhuri. An improved document skew angle estimation technique. *Pattern Recognition Letters*, 17(8):899-904, July, 1996;

[27] W. Postl. 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, pp 687-689, 1986;

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

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

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

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

[32] A. L. Spitz. Correcting for variable skew in document images. *International Journal on Document Image Analysis*, Springer-Verlag, September, 2003;

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

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

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

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

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

[38] CENADEM – http://www.cenadem.com.br;

[39] How much information 2003? – http://www.sims.berkeley.edu/research/projects/how-much-info-2003/;

[40] LibTIFF 3.6.1 – http://www.libtiff.org/;

[41] Omnipage 12.0, ScanSoft Corporation – http://www.omnipage.com;

[42] Tiff Specification Revision 6.0 – http://partners.adobe.com/asn/tech/tiff/specification.jsp;