Resumo

A transposição de documentos históricos para o meio digital é um importante modo de preservação e distribuição de informações da cultura de uma sociedade. O armazenamento das imagens digitalizadas desses documentos resolve o problema da preservação histórica, já que a fragilidade dos meios de comunicação impressos provoca uma baixa durabilidade desses documentos. Mas, tão importante quanto à preservação histórica, é a divulgação dessas informações. A transposição exclusivamente do texto desses documentos para o meio digital facilita o armazenamento e distribuição das informações, além de possibilitar a busca por algum dado específico. Uma etapa crítica do processo de transposição do texto de imagens de documentos é a segmentação de linhas. Quando se tratam de imagens de documentos históricos manuscritos essa etapa se torna ainda mais complexa. As manchas do papel, as caligrafias fora de padrão e a presença de elementos não-textuais na imagem são as principais dificuldades da segmentação de linhas. Este trabalho apresenta um estudo sobre o processo de segmentação de linhas de texto de documentos históricos. Mostra algumas técnicas existentes com essa finalidade, sendo que uma delas foi implementada, em Matlab, para um estudo mais aprofundado. A partir da implementação foram feitos os estudos de casos, com um acervo de documentos históricos manuscritos. O estudo teve seus resultados comparados com os de uma outra técnica, utilizando os mesmos critérios de avaliação.

Abstract

The transposition of historical documents to the digital media is an important way of preserving and distributing cultural information about a society. The storage of the digitalized images of these documents solve the historical preservation problem, since the fragility of the printed communication medias determine the low durability of these documents. However, as important as the historical conservation, is the acknowledgement of these data. The transposition, exclusively of the text of these documents to the digital media makes easier the storage and distribution of the information, besides giving the possibility to search for specific data. A critical stage in the process of transposition of images of the text from documents is the line segmentation. In the case of images from handwritten historical documents this stage becomes even more complex. The spots on the paper, the calligraphy out of pattern and the presence of non-textual elements on the image are the main difficulties of the line segmentation process. This work presents a study on the process of line segmentation of texts from historical documents. It demonstrates some techniques already in use with this purpose; one of them was implemented in Matlab, for a deeper study. Using our implementation, the case studies were carried out, using a collection of handwritten historical documents. The study had its results compared to another technique, using the same evaluation criteria.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

OCR – *Optical Character Recognition* (Reconhecimento Optico de Caracteres).

ASCII – *American Standard Code for International Interchange* (Padrão de Codificação Americano para Intercâmbio Internacional).

ALCM – *Adaptive Local Connectivity Map* (Mapa de Conectividade Local Adaptativo).

RLSA – *Run-Length Smoothing Algorithm.*

Agradecimentos

Gostaria de agradecer primeiramente a meus pais, por todo o esforço e sacrifícios para sempre me garantir uma boa educação. Agradeço a minha namorada Sílvia por todo o apoio e compreensão nesta fase de minha vida, bem como por ajudar tanto na correção do português deste trabalho. Agradeço também a minha cunhada Lucy pela ajuda na correção do texto, e um agradecimento especial a sua amiga Mayara, que me ajudou bastante com o Abstract.

Um obrigado muito especial para meus melhores amigos da faculdade: George, Elias, Raúl, Flávio, Mateus, Rubens e Lumadaiara, que são grandes pessoas e foram muito importantes durante minha vida universitária e tenho certeza de que continuarão sendo. Agradeço também a todos os meus outros amigos, tanto da faculdade, quanto fora dela, por todo o apoio e ajuda quando necessário.

Por fim, gostaria de agradecer aos professores Adriano Lorena e Carlos Alexandre pela confiança e oportunidade de trabalhar com temas tão maravilhosos desde a iniciação científica e, agora, neste trabalho de conclusão de curso. Muito obrigado pelas incontáveis orientações.

Capítulo 1

Introdução

A mídia impressa foi uma das maiores revoluções tecnológicas da humanidade, tendo substituído todas as outras formas de armazenamento e difusão de informação, sendo usada até hoje. Porém, sua fragilidade, rápida ocupação de grandes espaços físicos e dificuldade na busca por dados específicos são as suas principais desvantagens. Essas características são fatores preocupantes quando tratamos de documentos de importância histórica, nos quais estão armazenadas diversas informações valiosas para a cultura e história de uma sociedade. Uma alternativa viável e vantajosa a esse tipo de mídia, nos dias atuais, é o uso de recursos computacionais.

A criação de dispositivos digitalizadores tornou possível a transposição de documentos para computadores na forma de imagens. Dessa forma, é possível o armazenamento de imagens de documentos, possibilitando uma proteção mais eficiente aos desgastes provocados pelo tempo. Também, no caso de documentos históricos, proporciona maior facilidade na divulgação de seu conteúdo.

Um ponto crítico dessa tecnologia é a grande quantidade de espaço em *Bytes* necessário para armazenar as imagens. Uma imagem de um documento digitalizado, no formato padrão do Microsoft Windows, o BMP (*bitmap*), pode chegar a ocupar alguns milhares de *Kilobytes* de memória. Quando estas imagens são relativas a um texto, algumas técnicas de análise de documentos [1][2][3] podem ser aplicadas para convertê-las em textos editáveis, preservando a informação, diminuindo o espaço de armazenamento e facilitando a busca por algum dado específico do documento. Por exemplo, a informação contida na imagem de um documento que ocupa cerca de 4 *Megabytes* pode passar a ocupar cerca de 80 *Kilobytes* [2] somente separando-se o texto do papel, e menos de 10 *Kilobytes* se for transposto para o formato de texto computacional padrão.

O esforço despreendido no processo de conversão de documentos históricos para o formato de texto computacional é maior que para documentos contemporâneos, devido às condições físicas desses documentos, como a degradação e envelhecimento do papel, clareamento da tinta, documentos escritos nos dois lados do papel com a tinta de um lado transpondo para o outro, entre outros. Em documentos manuscritos, diferente de datilografados, acrescentam-se ainda as variações de caligrafia e disposição do texto fora de qualquer padrão, dificultando o processo de localização textual [4]. A Figura 1 ilustra algumas dessas características dos documentos históricos.

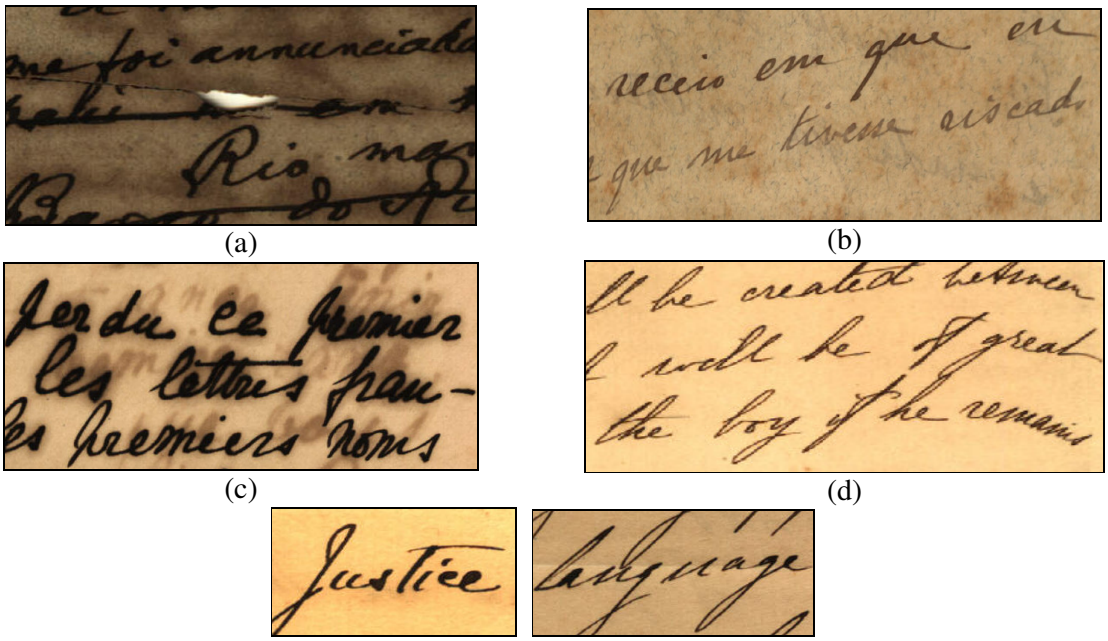


Figura 1. Exemplos das principais dificuldades do tratamento de documentos históricos: (a) amostra de degradação do papel (b) amostra de clareamento da tinta (c) amostra de tinta do verso aparecendo na frente da folha (d) uma amostra de texto fora dos padrões convencionais e (e) exemplos de variação de caligrafia de uma mesma pessoa.

Para a transposição de imagens para o formato de texto são utilizados sistemas conhecidos como *Optical Character Recognition* (OCR – Reconhecimento Óptico de Caracteres) [1]. Um sistema de OCR recebe uma imagem como entrada e faz a transposição do texto contido nessa imagem para um arquivo com caractere na codificação *American Standard Code for Information Interchange* (ASCII), padrão computacional para representação de caracteres. Em um sistema desse tipo existem várias etapas de processamento da imagem, as quais visam determinar a estrutura física do documento, o relacionamento entre os objetos que o compõem, e por fim o reconhecimento e recomposição de seu conteúdo [5]. As principais etapas [1][6] de um sistema de OCR são: limiarização, pré-processamento, segmentação do documento, segmentação do texto, extração de características e, por fim, a classificação do conteúdo encontrado. A Figura 2 ilustra essas principas etapas de um sistema de OCR.

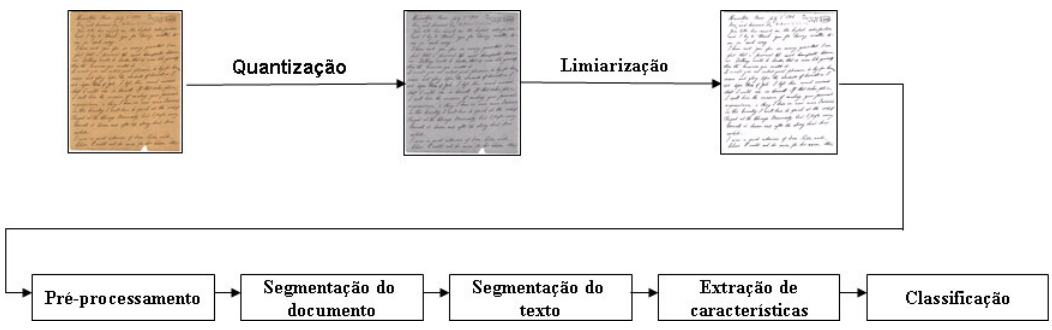
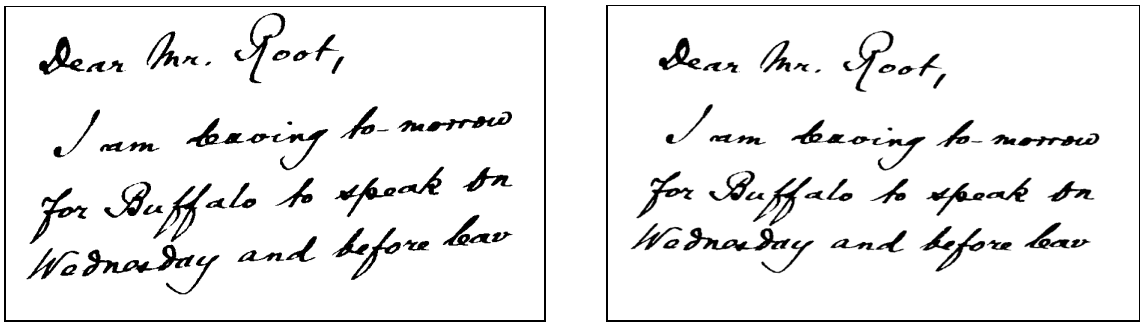


Figura 2. Diagrama de blocos mostrando as principais etapas do processo de reconhecimento.

A operação de quantização serve para reduzir o universo de cores da imagem. Documentos históricos normalmente são digitalizados no formato *true colour*[1], para fins de preservação, por isso é necessária a quantização para reduzir a quantidade de dados a serem processados. Após a operação de quantização, a imagem do documento passa a ter 256 tons de cinza. A limiarização, ou binarização, consiste na conversão de uma imagem para duas cores apenas, preto e branco. Para documentos, esse processo pode ser entendido como segmentação quando serve para separar os objetos (texto, imagens, desenhos, etc) do papel ao fundo.

O pré-processamento visa tornar a imagem binária livre das impurezas comuns em documentos (manchas) que são capturadas no processo de digitalização, mantendo-se na imagem digital. Muitas impurezas não são eliminadas no processo de limiarização, sendo necessária a aplicação de algoritmos que removam esses elementos, como filtros digitais [5]. Outro tratamento que é feito nesta etapa é a correção da orientação da imagem. Muitas vezes, durante o processo de digitalização, ou mesmo de acordo com a caligrafia, a disposição do texto da imagem não possui um enquadramento perfeito. Assim, nesta etapa são encontrados erros com relação a orientação da imagem e são feitas as devidas correções. Várias técnicas podem ser usadas para a detecção e correção de inclinação em imagens, como a transformada de Hough [5]. A Figura 3 (a) mostra um exemplo de parte de documento com orientação errada e (b) após a correção.



(a) (b)

Figura 3. (a) Um exemplo de pedaço de documento com a orientação errada. (b) a imagem com sua orientação corrigida.

A segmentação do documento objetiva, dada a imagem com as correções realizadas nas fases anteriores, organizar os objetos do documento em duas classes distintas: texto e não-texto. Já a etapa de segmentação do texto é responsável pela identificação no texto das linhas, palavras e caracteres.

Antes do sistema fazer a classificação dos caracteres ou palavras, comumente se utiliza algum processo de extração de características nessas unidades textuais. Utilizar todos os dados como entrada dos algoritmos de classificação muitas vezes podem apresentar informação desnecessária e aumentar o tempo de processamento [7]. Essa etapa procura coletar os dados mais significativos dessas unidades textuais, que servem como entrada para a etapa de classificação, a qual utiliza normalmente algoritmos de aprendizado de máquina para reconhecer o conteúdo dos caracteres ou palavras.

---

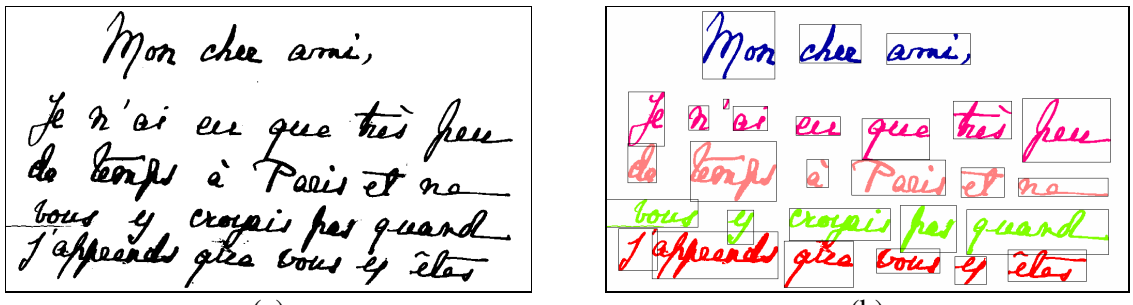
[1] *True colour* é um formato de codificação de cores em imagens digitais. Indica que cada *pixel* da imagem é representado por 24 *bits*, possibilitando cerca de 16,7 milhões de cores possíveis para ele.

1.1 Motivação

Diversas pesquisas sobre as etapas de um sistema de reconhecimento de caracteres são realizadas na tentativa de melhorar a análise e transposição de documentos que ainda não são bem tratados nos sistemas comerciais. Algumas abordagens de análise e reconhecimento de imagens de documentos podem ser encontradas em Marinai [8].

Dentre as pesquisas para aprimoramento dos sistemas de reconhecimento de documentos, há um enfoque neste trabalho para os estudos na área de segmentação. A segmentação de documentos, como já foi explanado anteriormente, visa separar e organizar os elementos textuais dos não-textuais, deixando-os de tal modo que o processo de reconhecimento seja realizado de maneira conveniente. O aumento da granularidade dos dados a serem tratados pelos classificadores torna mais simples os seus treinamentos. Um exemplo dessa afirmação é o fato de ser mais simples e rápido treinar um classificador para reconhecer um caractere entre os 24 da língua portuguesa, do que treiná-lo para classificar uma palavra inteira de uma vez, entre milhões de palavras existentes. Por isso é preferível, na maioria dos casos, identificar todos os caracteres da imagem para só então reconhecer seu conteúdo.

Para simplificar a identificação dos caracteres, normalmente identifica-se primeiro todas as regiões de texto, depois as linhas que formam esse texto e, em seguida, as palavras que compõem cada linha, para só então localizar os caracteres. No universo de documentos manuscritos a segmentação de linhas apresenta grande complexidade e é foco de muitas pesquisas [9]. A segmentação para documentos datilografados já é um problema praticamente resolvido, enquanto que para documentos manuscritos as pesquisas ainda precisam avançar mais, devido aos problemas já mencionados anteriormente, de difícil solução. Devido a esses problemas, a segmentação de linhas é um dos passos mais complexos e importantes para um sistema de reconhecimento de texto ser bem sucedido [9]. Um exemplo de segmentação de texto pode ser visto na Figura 4. As palavras caixas correspondem às palavras segmentadas. Palavras da mesma cor indicam que elas foram detectadas como de uma mesma linha [10].



(a) (b)

Figura 4. (a) Imagem original e (b) documento após a segmentação com palavras detectadas. Essa imagem foi retirada da referência [10].

1.2 Objetivos

Todas as etapas precedentes ao reconhecimento de caracteres de uma imagem de documento influenciam decisivamente no resultado final. Quanto melhor identificado um elemento textual, melhor poderá ser o grau de acerto da classificação deste elemento. Dessa forma, os objetivos principais desse trabalho são: estudos sobre segmentação, focado na identificação de linhas em documentos manuscritos; como estudo de caso, aplicar a técnica de segmentação de linhas proposta por Basu *et al.* [11] em documentos históricos, analisando seu desempenho; construir uma biblioteca contendo a técnica utilizada no estudo de caso; e contribuir com o desenvolvimento de um sistema mais eficiente para reconhecimento de textos manuscritos de documentos.

Este trabalho faz parte do projeto PROHIST [12], que visa o desenvolvimento de técnicas de análise e tratamento de documentos históricos. A maior contribuição deste trabalho será uma análise comparativa de desempenho entre a técnica de Basu *et al.* e uma técnica desenvolvida no PROHIST, chamada de Novo Método de Segmentação de Linhas e Palavras, que será mostrado no Capítulo 2 (Seção 2.3.4) e pode ser visto com mais detalhes em Mello *et al.* [10].

1.3 Descrição dos capítulos

Este trabalho está organizado da seguinte forma:

- **Capítulo 1 - Introdução:** Contextualiza o problema a ser tratado, mostrando sua importância e dificuldades, bem como os objetivos deste trabalho.
- **Capítulo 2 - Segmentação de Linhas de Documentos:** Fornece a base científica para o entendimento do resto do trabalho, bem como uma visão geral do estado da arte de segmentação de linhas de documentos.
- **Capítulo 3 - Modelo de Segmentação de Documentos Manuscritos por Fluxo Hipotético de Água:** Apresenta o algoritmo de segmentação por Fluxo Hipotético de Água, a técnica implementada neste trabalho.
- **Capítulo 4 - Estudo de Casos:** Apresenta os resultados obtidos, bem como a análise desses resultados.
- **Capítulo 5 - Conclusões:** Apresenta as considerações finais, bem como as contribuições e trabalhos futuros.

Capítulo 2

Segmentação de Linhas de Documentos

A segmentação é uma das principais etapas na maioria das aplicações de análise de documentos e representa um dos maiores desafios no processamento de imagens. A principal razão dessa dificuldade está na falta de informação sobre os objetos nas imagens [13]. A segmentação consiste em duas tarefas básicas: identificação e delineamento. A identificação indica a localização aproximada do objeto de trabalho na imagem, enquanto o delineamento extrai sua extensão da imagem [14]. Os seres humanos realizam a primeira tarefa com relativa facilidade, enquanto o computador é capaz de realizar a segunda com muito mais precisão que os humanos. A dificuldade da máquina na localização dos objetos na imagem se deve ao fato não existir um modelo matemático que descreva de forma global esses objetos [2].

Quando se trata de análise e processamento de documentos, o desafio é identificar a posição de cada caractere e separá-lo numa sub-imagem, a qual possa ser classificada. Uma estratégia clássica [15] para se chegar aos caracteres de um documento é primeiro encontrar todo o texto, depois as linhas com compõem o texto, depois as palavras, para só então localizar os caracteres. A Figura 5 ajuda a entender esse raciocínio.

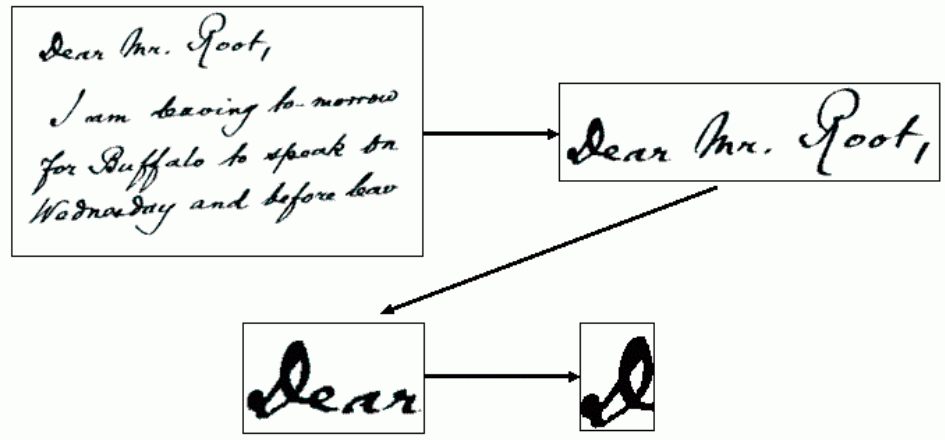


Figura 5. Estratégia clássica de segmentação do texto de um documento.

Este trabalho abordará a etapa de segmentação de linhas de documentos, tarefa importante para o desempenho final do sistema de reconhecimento de texto. Segmentação de linhas de texto é o processo de separação das unidades de texto espacialmente ligadas, seja por *pixels*, componentes conectados ou pontos característicos em comum [9]. Existem duas categorias de abordagens para segmentação de linha: a busca por espaços entre o texto e a busca por componentes conectados pelo texto. Neste capítulo mostraremos as principais características, dificuldades e técnicas para segmentação de linhas de documentos, com ênfase em documentos manuscritos.

2.1 Características e Representação de Linhas

Em uma imagem de documento, as linhas são agrupamentos em *pixels* distribuídos em uma mesma direção na imagem. Elas apresentam as seguintes características e elementos em comum [9]:

- Linha-base: linha imaginária que segue e une a parte inferior do corpo dos caracteres em uma linha de texto, considerando a posição inferior média dos caracteres;
- Linha-média: linha imaginária que segue e une a parte superior do corpo dos caracteres numa linha de texto, considerando a posição superior média dos caracteres;
- Linha superior: linha imaginária que acompanha os caracteres ascendentes, delimitando superiormente a linha;
- Linha inferior: linha imaginária que acompanha os caracteres descendentes, delimitando inferiormente a linha.

A Figura 6 mostra um exemplo de linha de um documento, ilustrando os quatro principais elementos descritos acima.

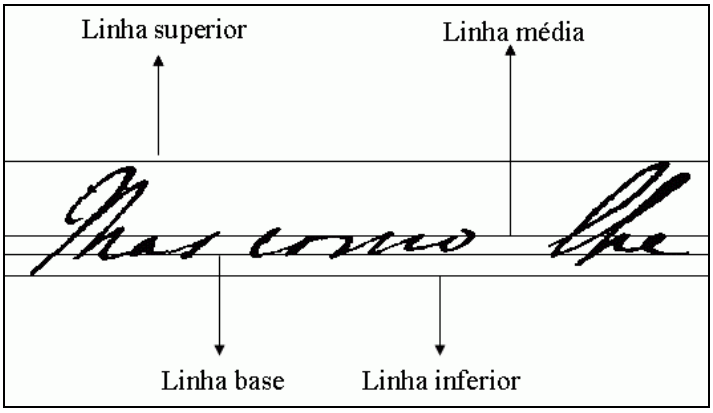
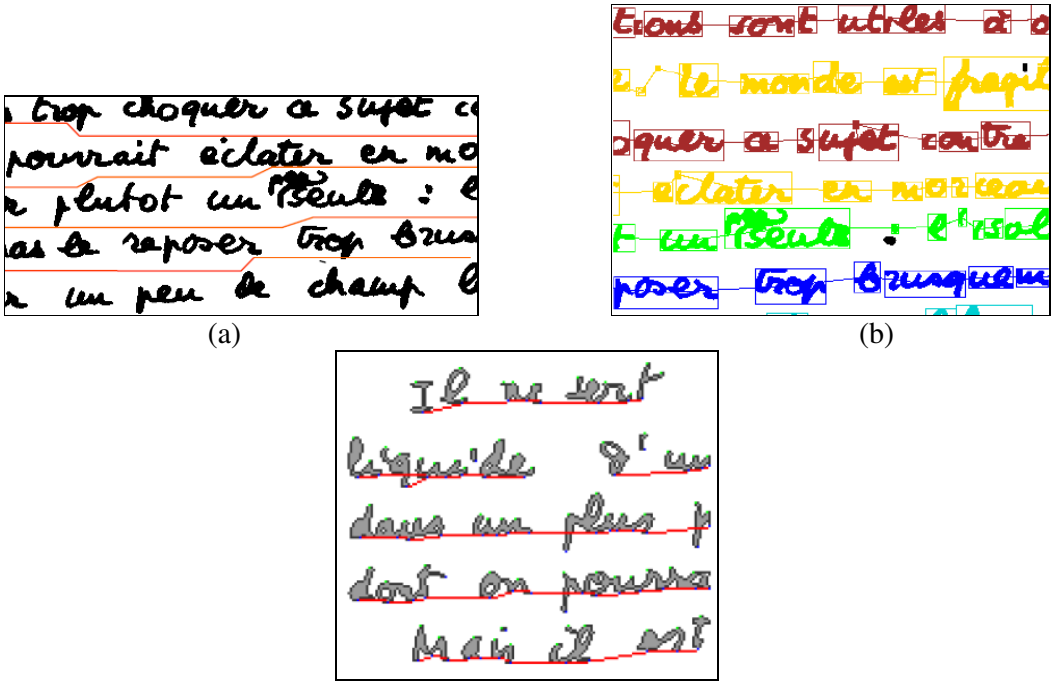


Figura 6. Principais componentes de uma linha de texto.

Dentre os tipos de representação de linhas de um documento digitalizado, destacam-se [9]: representação pelos caminhos de separação entre as linhas, por *strings* e por linhas-base. Os caminhos são linhas fictícias contínuas que seguem o fluxo da escrita, podendo ser linhas retas ou curvas. As *strings* são conjuntos de elementos relacionados entre si, seja por conexão ou proximidade. O conjunto de *strings* forma uma linha. A representação por linhas-base forma uma linha seguindo e unindo o fluxo da linha base do texto. A Figura 7 ilustra esses conceitos.



(a) (b)

(c)

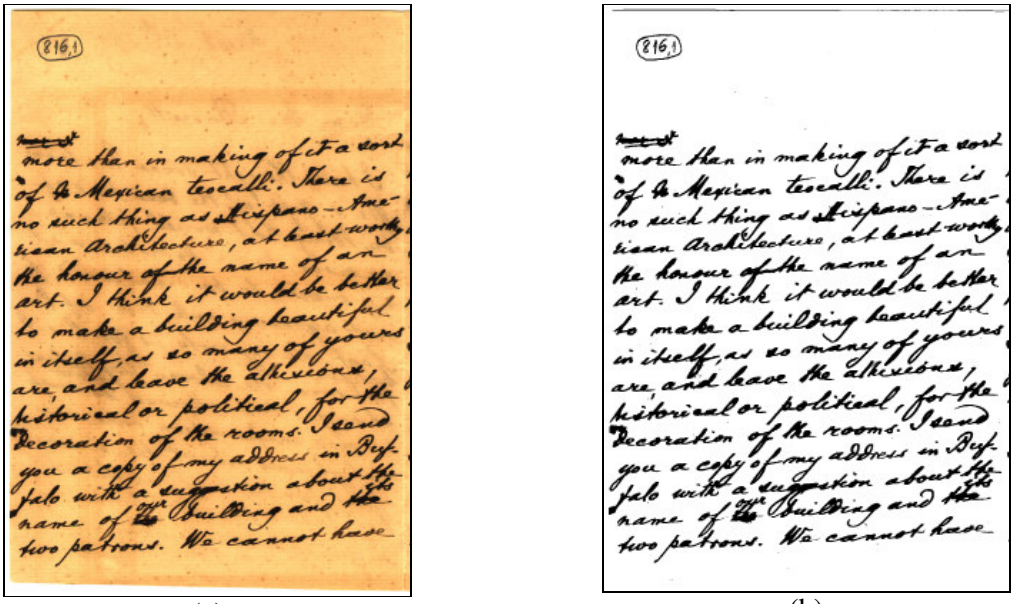
Figura 7. Exemplos de representações de linhas. (a) Representação por caminhos entre as linhas. (b) Representação por *strings*. (c) Representação por linhas-base. Essa imagem foi retirada da referência [9].

2.2 Pré-processamento

O processo ideal para extração de linhas exigiria que a imagem do documento estivesse livre de ruídos e elementos não-textuais, com o ângulo da caligrafia corretamente orientado, além de possuir o mínimo de degradação do texto possível. Mas, em casos práticos, isso é praticamente impossível de acontecer. Por isso é necessário fazer um pré-processamento no documento antes de aplicar alguma técnica de segmentação de linhas. Nesta seção serão descritos rapidamente alguns tipos de pré-processamento que podem ser aplicados numa imagem antes da extração de linhas.

2.2.1 Binarização

Normalmente, antes de aplicar qualquer técnica para remoção de elementos não-textuais da imagem, aplica-se a binarização no intuito de remover o máximo possível de elementos do plano de fundo do documento [5]. Esta técnica converte a imagem para preto e branco, separando o texto dos elementos ao fundo. Isso ocorre através da definição de um limiar, onde todos os *pixels* com valores abaixo dele são convertidos para preto e os demais *pixels* são convertidos para branco. A Figura 8 mostra um exemplo de binarização de um documento colorido. Esse processo se torna mais difícil quando temos imagens de baixo contraste, ou seja, imagens onde a cor do papel é próxima a cor do texto.



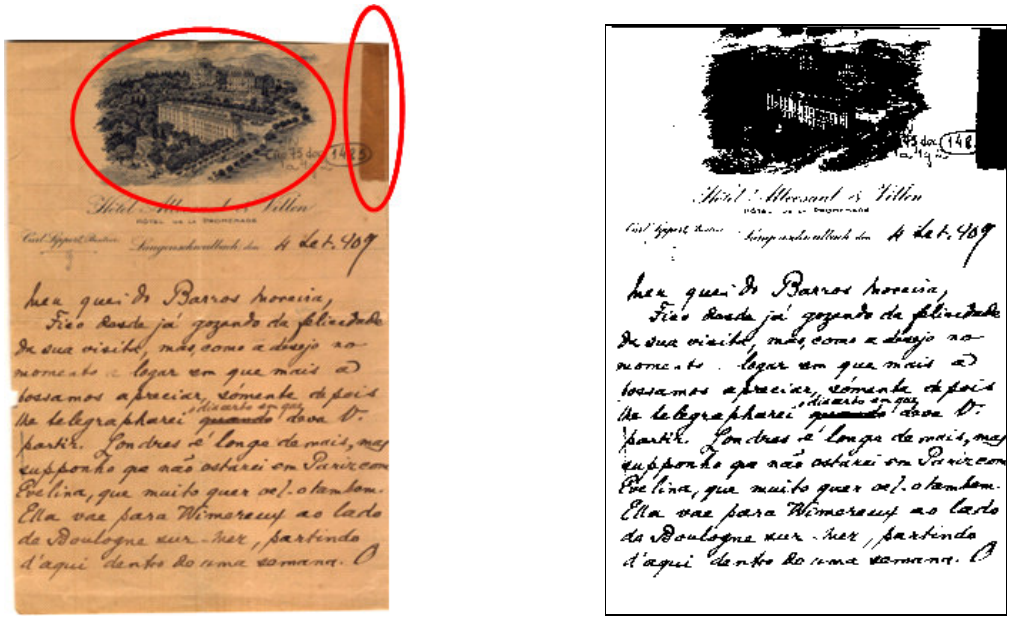
(a) (b)

Figura 8. Exemplo de binarização de imagem colorida: (a) mostra a imagem original e (b) a mesma imagem binarizada.

Existem duas abordagens para binarização: a global e a local. A primeira define um único limiar de segmentação para todos os *pixels* da imagem, enquanto a segunda determina valores de limiar localmente, *pixel* por *pixel,* ou região por região, e produz resultados relativamente melhores, se comparados com a abordagem global [9]. Geralmente não é aplicada binarização global em documentos históricos, devido ao papel de fundo ser bastante heterogêneo; existem muitas pesquisas nessa área visando melhorar o desempenho [16][17].

2.2.2 Redução de Elementos Não-textuais

Elementos não-textuais como manchas e marcas no papel do documento digitalizado – conhecidos como ruídos, muitas vezes não são eliminados totalmente pelo processo de binarização. Eles podem ser removidos através da utilização de filtros digitais específicos, ou até mesmo durante o processo de digitalização do documento [5][13]. Outros tipos de elementos, como figuras, podem ser retirados utilizando conhecimento de sua forma, cor ou posição na imagem [18]. Elementos textuais, mas indesejados, como palavras do verso do documento transpondo o outro lado do papel, podem ser retirados com alguns filtros e técnicas *wavelet* [19][20]. A Figura 9 ilustra alguns tipos de elementos não-textuais que precisariam ser removidos do documento, mesmo após binarização.

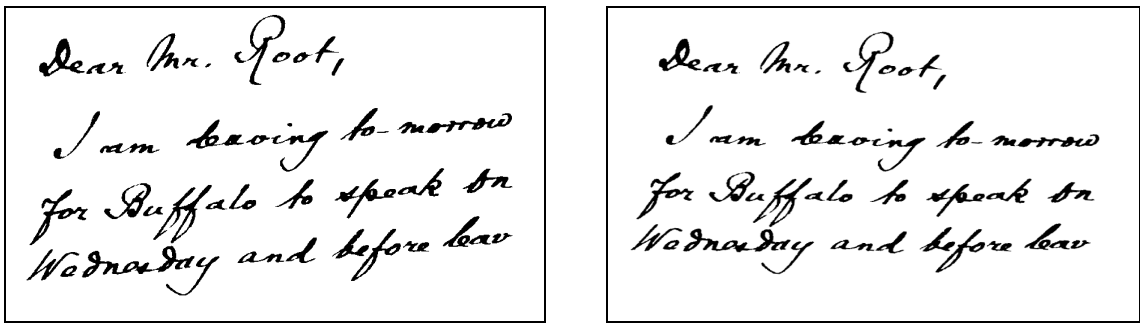


(a) (b)

Figura 9. Exemplo de imagem com elementos não-textuais: (a) mostra a imagem original com os elementos não-textuais destacados em vermelho e (b) mostra a imagem binarizada, com a permanência desses elementos.

2.2.3 Correção da Orientação da Imagem

É muito comum também em imagens de documentos a orientação do texto não ter um enquadramento perfeito. Isso pode ocorrer por falta de cuidado no momento da digitalização ou mesmo devido a caligrafia do autor do documento. O processo de reconhecimento de caracteres requer um bom enquadramento da imagem para obter bons resultados. A orientação de um documento é o ângulo de inclinação predominante nas linhas do texto. Os algoritmos de detecção de orientação encontram o ângulo de inclinação predominante na imagem, o qual podemos utilizar para aplicar uma rotação no sentido inverso nela para corrigi-la. A Figura 10 (a) mostra um documento com a orientação imperfeita, enquanto a Figura 10 (b) mostra a mesma imagem com a orientação corrigida.



(a) (b)

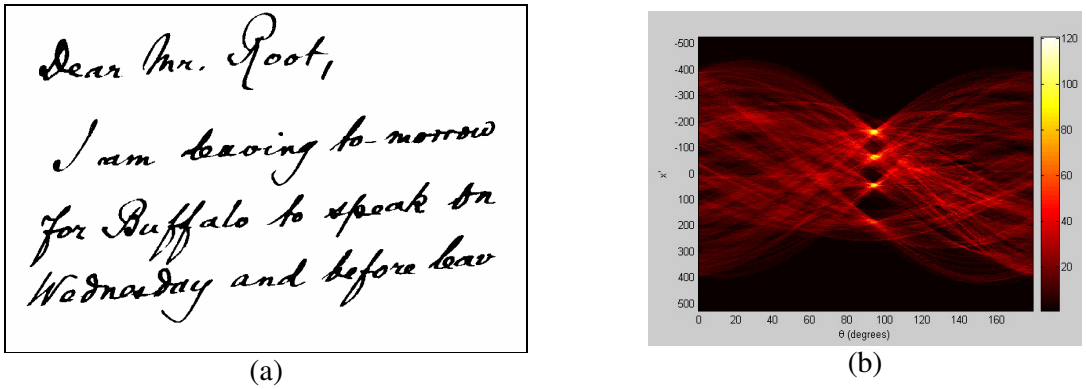
Figura 10. Exemplo de aplicação de correção de orientação em documentos: (a) a imagem original e (b) a imagem após a correção de angulação.

Um algoritmo muito utilizado para detecção de inclinação na imagem é a transformada de Hough [3]. Quando aplicada numa imagem, a transformada pode identificar um conjunto de *pixels* que pertençam a uma mesma forma parametrizável, como retas. A transformada mapeia as coordenadas da imagem em curvas no espaço de Hough, definindo o ângulo de orientação da imagem como sendo o ponto de maior conversão das curvas do espaço de Hough.

Para detecção de orientação em imagens de documentos, a idéia da transformada de Hough é mapear todos os pontos pertencentes a uma mesma reta, num único ponto de um novo espaço de parametrização. Uma reta pode ser parametricamente definida tanto em coordenadas cartesianas quanto em coordenadas polares. A transformada faz o mapeamento entre esses dois espaços. Por exemplo, dada uma imagem I(x,y), onde as coordenadas (x,y) representam a posição de um *pixel* na imagem, a equação da reta pode ser definida como

$$\rho = x \cos \varsigma + y \, sen \, \varsigma \qquad (2.1)$$

onde ρ e ς são constantes. Para cada ponto da imagem I(x,y), são definidas todas as retas de parâmetros ρ e ς que resolvam a equação polar (2.1) que passam pelo ponto I(x,y). O espaço de parametrização, definido pelos parâmetros ρ e ς, é discretizado em uma tabela de acumulação. Cada elemento da tabela corresponde a um intervalo de ρ e de ς. Os valores finais desses dois parâmetros são obtidos pela informação do gradiente. A Figura 11 (a) ilustra uma amostra de imagem e na Figura 11 (b) a representação de suas coordenadas no espaço de Hough.



(a) (b)

Figura 11. Exemplo da representação das coordenadas no espaço de Hough. (a) Amostra de documento. (b) Representação no espaço de Hough dessa imagem.

Após a definição do ângulo de orientação da imagem, basta executar uma rotação no sentido inverso, corrigindo-a [3][21], obtendo o resultado da Figura 10 (b).

2.3 Técnicas de Segmentação de Linhas

Imagens de documentos históricos apresentam complexa estruturação, pois em suas confecções não havia uma boa aderência da tinta no papel, resultando em fragmentação do texto com o passar do tempo. Somado a isso, grande porcentagem de documentos históricos são manuscritos, com linhas com pouca orientação coesa. Dessa forma, torna-se mais complexo estimar o espaçamento e a orientação das linhas do texto. Algumas técnicas foram desenvolvidas visando a extração de linhas de documentos históricos, algumas utilizando características locais das linhas, outras utilizando características globais.

2.3.1 Análise de Projeção

A análise de projeção (*projection profile*) [9][15][22] consiste em projetar a quantidade de *pixels* pretos da imagem binária de um documento, seja horizontalmente ou verticalmente. A Figura 12 mostra um exemplo de um histograma resultante da projeção horizontal e vertical. Nota-se, no histograma horizontal, picos de *pixels* pretos, seguidos de ausência dos mesmos. A quantidade de picos indica a quantidade de linhas no documento, enquanto cada ponto de mínimo indica um potencial ponto de segmentação.

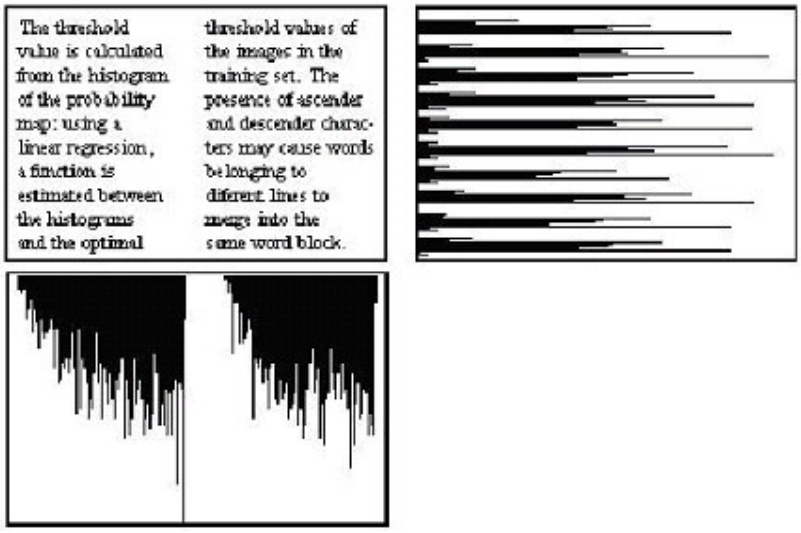


Figura 12. Exemplo de análise de projeção. A direita do texto encontra-se a projeção horizontal e abaixo a projeção vertical. Essa imagem foi retirada da referência [25]

A análise de projeção é uma técnica usada principalmente em documentos datilografados, pois estes apresentam um espaçamento coeso entre os objetos do texto. Pode ser adaptada para textos manuscritos, aumentando a complexidade, como por exemplo o *smoothed projection profile* [23], em que a função de projeção é filtrada por um filtro passa-baixa para eliminar os máximos locais e reduzir os ruídos, aumentando a eficiência da segmentação. Outra variante são as projeções curvas [24], que projetam as transições de *pixels* preto/branco ou os componentes conectados, em vez dos *pixels* pretos. A Figura 13 mostra a segmentação das linhas de um documento utilizando a projeção horizontal simples, enquanto a Figura 14 mostra os passos de segmentação utilizando o *smoothed projection profile*, no qual os pontos de segmentação são

mais bem definidos. Um estudo aprofundado da aplicação da análise de projeção em documentos históricos pode ser visto em Lima [25].

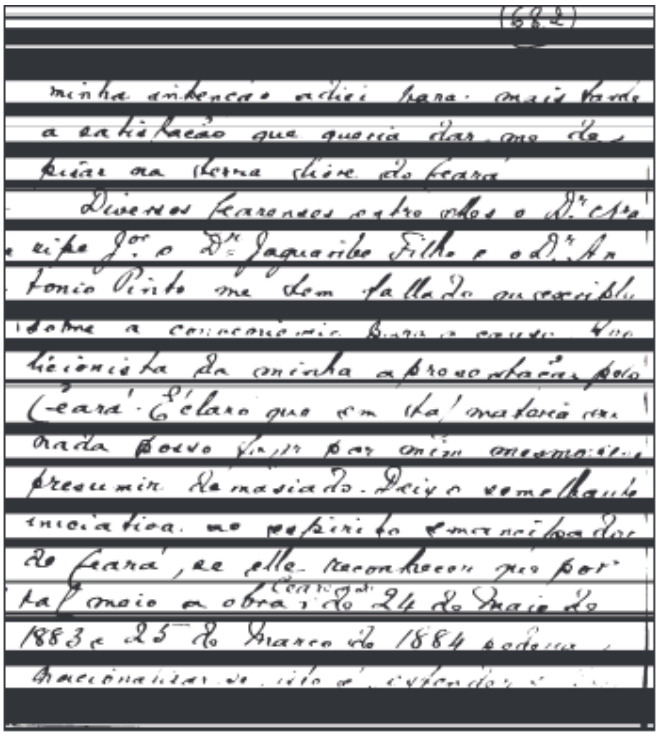
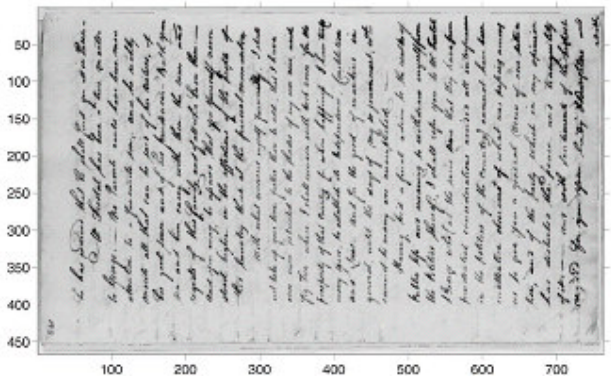
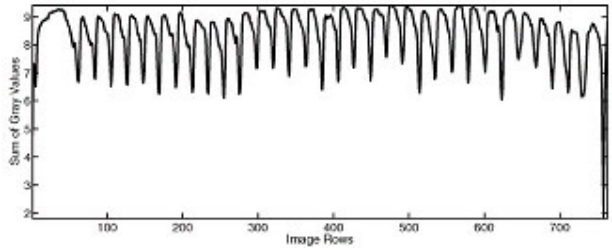


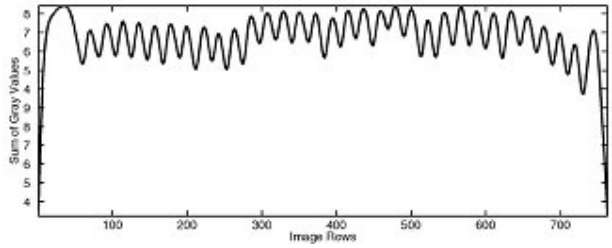
Figura 13. Resultado da aplicação da projeção horizontal num documento manuscrito. Essa imagem foi retirada da referência [25].



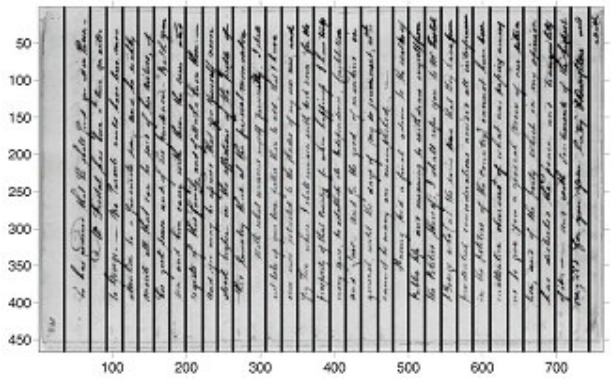
(a)



(b)



(c)



(d)

Figura 14. Passos da técnica *smoothed projection profile* em um documento. (a) Uma imagem rotacionada. (b) *Projection profile*. (c) *Smoothed projection profile*, observar os picos únicos. (d) Segmentação das linhas. Essa imagem foi retirada da referência [23].

2.3.2 Mapas de Conectividade Locais Adaptativos

Um algoritmo usando mapas de conectividade locais adaptativos (*Adaptive Local Connectivity Map* - ALCM) foi proposto por Shi *et al.* [26] para extração de linhas de documentos históricos particularmente complexos. É baseado nas características de conectividade entre os *pixels*.

Dada a função $f : R^2 \rightarrow R$ representando um dado sinal. Sua versão discreta com o domínio limitado a $\{0,1,...,n-1\}\times\{0,1,...,m-1\}$ e valores entre 0 e 255 é a imagem em tons de cinza. Então a ALCM é definida como a seguinte transformada

$$\text{ALCM: } f \rightarrow A$$

de convolução unidirecional:

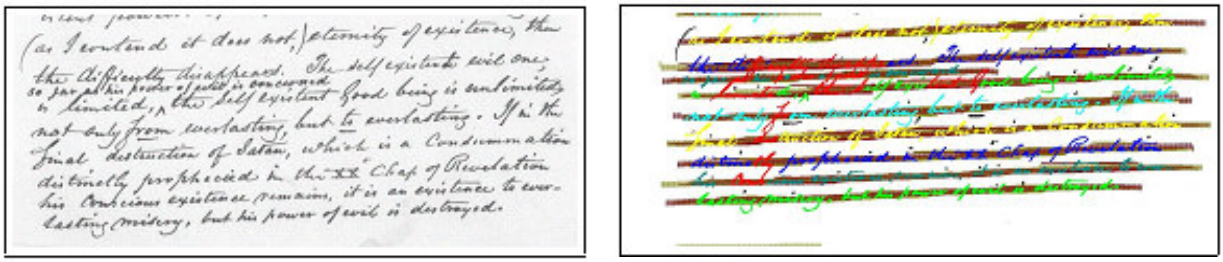
$$A(x, y) = \int_R f(x, y)G_c(t - x, y)dt \tag{2.2}$$

onde,

$$G_c(x, y) = \begin{cases} 1, & se|x| < c \\ 0, & se|x| \geq c \end{cases} \tag{2.3}$$

onde c é o tamanho da janela que vai percorrer a imagem.

É aplicada a transformada ALCM na imagem do documento em tons de cinza, depois a binarização e por fim agrupam-se os elementos próximos. Isso gera uma máscara para o documento original, que possibilita a separação das linhas. A Figura 15 ilustra a aplicação dessa técnica num documento.

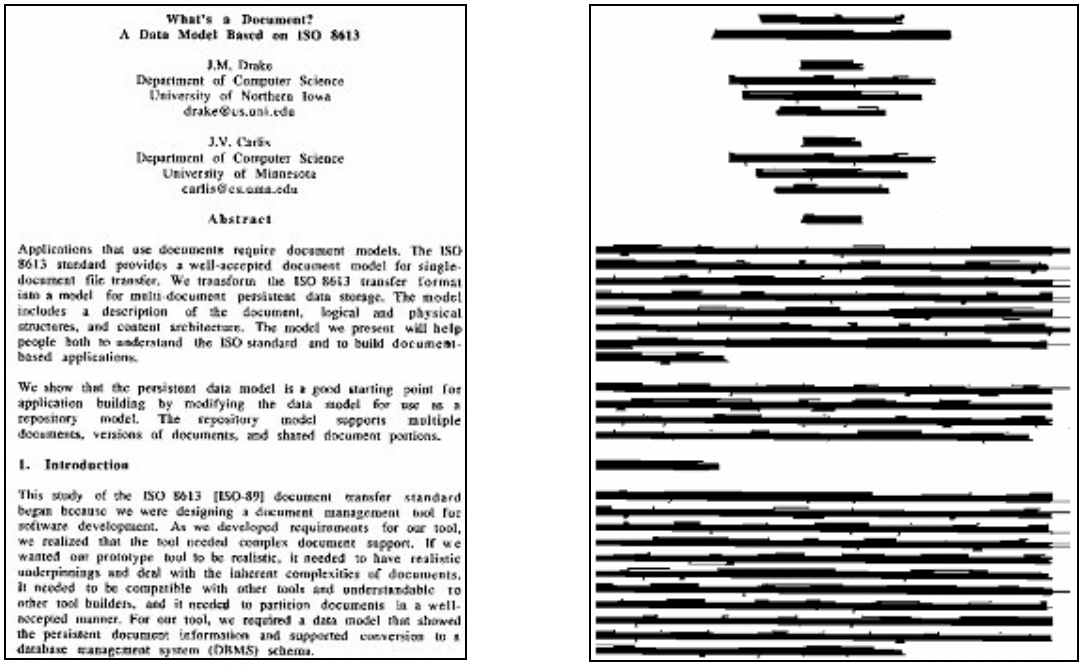


(a) (b)

Figura 15. Exemplo de aplicação do método ALCM para segmentação de linhas: (a) mostra um pedaço de documento e (b) o resultado da aplicação da técnica neste documento. Essa imagem foi retirada da referência [26].

2.3.3 *Run-Length Smoothing*

O algoritmo *run-length smoothing* (RLSA) foi proposto por Wong *et al.* [27] com a finalidade de segmentar regiões do texto. Neste método, o documento é "manchado" pela conversão de uma seqüência de *pixels* brancos, de tamanho menor ou igual a um dado limiar, para *pixels* pretos. Fazendo essa operação unicamente no sentido horizontal, podemos obter as regiões das linhas de um documento, como mostra a figura Figura 16.



What's a Document?
A Data Model Based on ISO 8613

J.M. Drake
Department of Computer Science
University of Northern Iowa
drake@us.uni.edu

J.V. Carlis
Department of Computer Science
University of Minnesota
carlis@cs.umn.edu

Abstract

Applications that use documents require document models. The ISO 8613 standard provides a well-accepted document model for single-document file transfer. We transform the ISO 8613 transfer format into a model for multi-document persistent data storage. The model includes a description of the document, logical and physical structures, and content architecture. The model we present will help people both to understand the ISO standard and to build document-based applications.

We show that the persistent data model is a good starting point for application building by modifying the data model for use as a repository model. The repository model supports multiple documents, versions of documents, and shared document portions.

1. Introduction

This study of the ISO 8613 [ISO-89] document transfer standard began because we were designing a document management tool for software development. As we developed requirements for our tool, we realized that the tool needed complex document support. If we wanted our prototype tool to be realistic, it needed to have realistic underpinnings and deal with the inherent complexities of documents. It needed to be compatible with other tools and understandable to other tool builders, and it needed to partition documents in a well-accepted manner. For our tool, we required a data model that showed the persistent document information and supported conversion to a database management system (DBMS) schema.

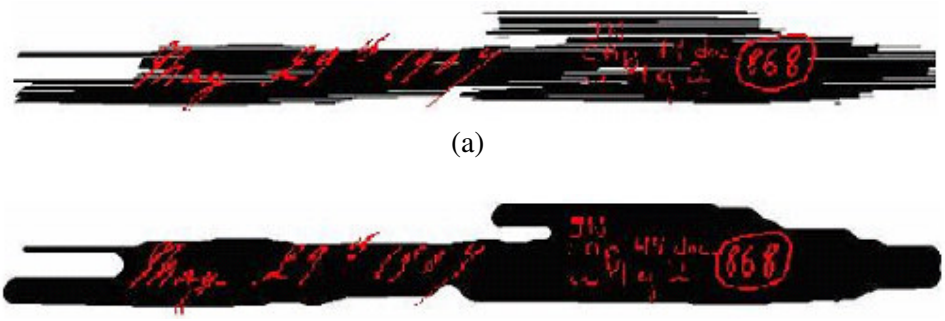
(a) (b)

Figura 16. Aplicação do algoritmo RLSA num documento. (a) Imagem original. (b) Resultado da aplicação do algoritmo horizontalmente. Essa imagem foi retirada da referência [28].

É importante ressaltar que a escolha do limiar é um fator importante no processo, pois ele definirá a qualidade do resultado obtido, já que pode unir elementos indesejadamente, caso o limiar seja muito alto [28].

2.3.4 Novo Algoritmo para Segmentação de Documentos Históricos

Neste método [10][29], procura-se encontrar os *pixels* que fazem parte do texto através de binarização e, a partir de um mapa de transições de *pixels*, obtém as zonas de linha, isto é, as áreas onde se encontram a maior parte do texto. Após a suavização desse mapa, é traçado um esqueleto dessas áreas de linhas, que são usados posteriormente para unir fisicamente as palavras que formam essas linhas, simplificando as zonas de linha. A Figura 17 mostra uma zona de linha original e após suavização (o texto em vermelho serve apenas para mostrar onde os elementos textuais estavam na imagem).



(a)

(b)

Figura 17. Zona de linha da imagem de um documento, onde o texto está sobreposto em vermelho. (a) Zona de linha e (b) zona de linha suavizada. Essa imagem foi retirada da referência [29].

Aplicando alguns algoritmos de correção de linhas partidas [30], tem-se como resultado a imagem mostrada na Figura 18. Posteriormente a segmentação de palavras também é feita, mas foge ao foco deste trabalho, aplicando dilatação no documento e um algoritmo de tratamento de componentes conectados [30]. O resultado final deste algoritmo é mostrado na Figura 19.

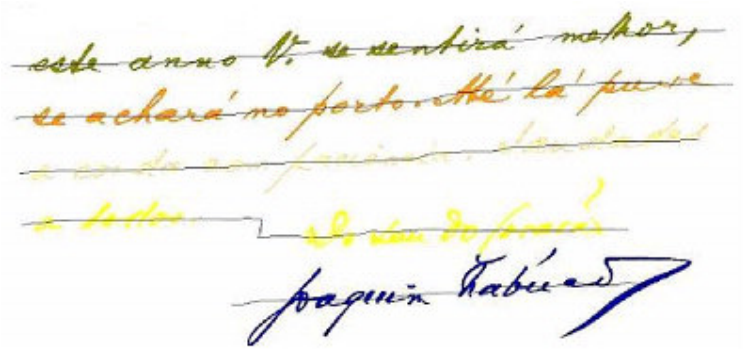


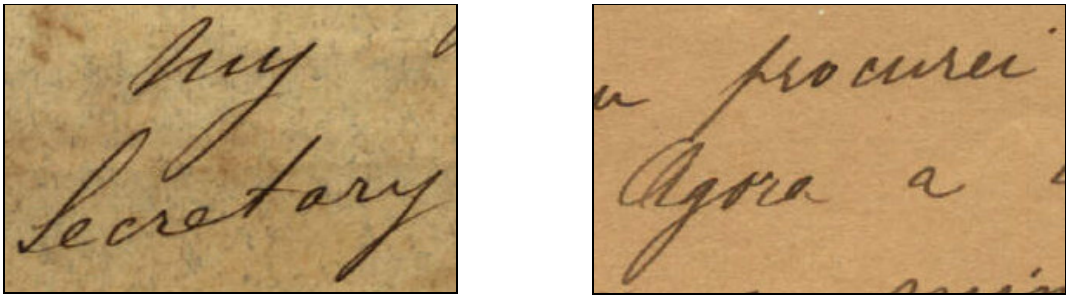
Figura 18. Segmentação de linhas feita pelo algoritmo descrito. Essa imagem foi retirada da referência [29].

Figura 19. Resultado final do algoritmo, com as linhas segmentadas (cada linha de uma cor) e as palavras segmentadas. Essa imagem foi retirada da referência [29].

2.3.5 Processamento de Linhas Conectadas e Sobrepostas

O tratamento de linhas conectadas e sobrepostas representa o principal desafio na segmentação de linhas de documentos [9]. Alguns métodos já possuem características que previnem esse tipo de ambigüidade, mas a maioria necessita do processamento dos componentes conectados/sobrepostos antes ou após a segmentação das linhas. Algumas heurísticas para tratamento desses casos existem, mas não é conhecida nenhuma técnica que seja realmente eficaz, exceto para alguns casos mais simples. O desafio maior aparece no processamento de linhas conectadas, que é fonte de várias pesquisas.

Alguns critérios podem ser usados para a identificação desses componentes, como, por exemplo, suas dimensões serem maiores que a média dos outros componentes, seus alinhamentos serem diferentes dos demais, etc. Uma vez que o componente é detectado como ambíguo, ou seja, possuir a probabilidade de ser um componente conectado ou sobreposto com outro, ele é classificado em duas categorias: o componente é formado por objetos que se tocam, ou é formado por objetos sobrepostos. A Figura 20 exemplifica os conceitos de componentes sobrepostos e conectados. A separação dessas estruturas pode ser feita de modo "bruto", com um corte na horizontal, ou de modo mais preciso, analisando-se o contorno das estruturas que compõem o componente.



(a) (b)

Figura 20. (a) Linhas sobrepostas. (b) Linhas conectadas.

Em Zahour *et al.* [31][32] o documento primeiro é dividido em oito colunas de dimensões iguais. A projeção horizontal é feita para cada coluna. Em cada histograma obtido, dois mínimos consecutivos delimitam um bloco de texto. Para detecção de sobreposição ou toques de linhas é usada a técnica de aprendizagem de máquina, não supervisionada, *k-means*[2]. Após os componentes serem encontrados, um outro sistema de agrupamento *k-means* é utilizado para encontrar o melhor ponto de separação entre os componentes.

No trabalho de Piquin *et al.* [33] a segmentação é feita a partir da esqueletização de caracteres que se tocam e também através de um dicionário de possíveis configurações de conexões (Figura 21).

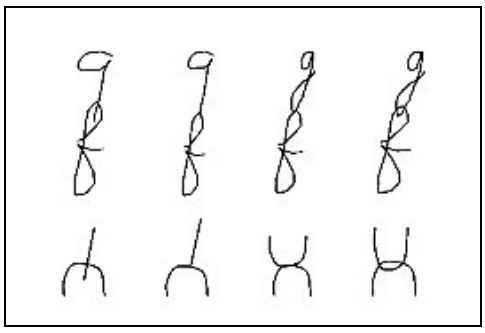


Figura 21. Configurações típicas de componentes conectados. Essa imagem foi retirada da referência [9].

Em Bruzzone e Coffetti [34], o ponto de contato entre componentes ambíguos é detectado e processado através de sua borda externa. Uma análise precisa do contorno do ponto de contato é feita de modo a separar o traço do caractere de acordo com duas configurações específicas: um *loop* em contato com o traçado ou dois *loops* em contato com o traçado. Em casos mais simples de documentos manuscritos (Marti e Bunke [24]), o centro de gravidade do componente conectado é usado para determinar se ele pertence a linha atual ou a próxima linha do documento, ou mesmo para cortar o componente em duas ou mais partes. Isso funciona muito bem para caracteres isolados, mas pode falhar se o componente for uma palavra ou conjunto de palavras [9].

[2] Algoritmo não-supervisionado que analisa os dados de entrada e os agrupa em K categorias, por semelhança.

Capítulo 3

Modelo de Segmentação de Documentos Manuscritos por Fluxo Hipotético de Água

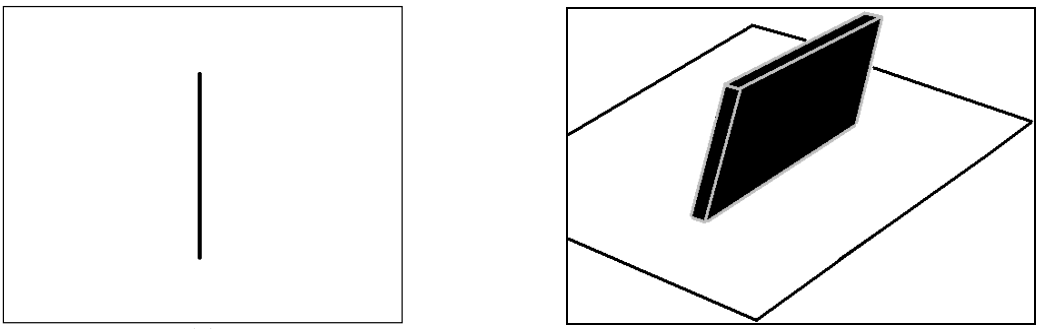
Este trabalho se propõe a desenvolver um estudo sobre técnicas de segmentação de linhas de documentos e analisar o desempenho de duas estratégias recentemente propostas nos documentos do nosso acervo. Uma das técnicas foi descrita brevemente na Seção 2.3.4 e pode ser vista com mais detalhes em Mello *et al.* [10] e em Severin [29]. A outra técnica, que é o objeto principal de estudo e implementação neste trabalho, será detalhada a seguir.

A técnica descrita a seguir foi desenvolvida por Basu *et al.* [11]. As etapas do algoritmo dessa técnica são mostradas na Figura 22; notar que não está sendo considerada a extração do texto dos documentos, assim como no trabalho de Basu *et al.*. A parte principal e inovadora desse método é a heurística de fluxo de água, que é usada para encontrar as regiões de linhas de texto. As outras etapas, como por exemplo a correção da inclinação das linhas, podem ser substituídas por métodos similares, dado que Basu *et al.* não faz um estudo sobre a variação dos resultados de acordo com as abordagens usadas. Cada etapa é detalhada nas Seções subseqüentes.

Figura 22. Diagrama do algoritmo de segmentação por fluxo de água.

3.1 Identificação das Linhas

A principal inovação do trabalho de Basu *et al.* foi a maneira de identificar as linhas de um documento digitalizado. A proposta foi utilizar uma heurística de fluxo hipotético de água passando sobre a imagem. Para ajudar a entender a heurística, tomemos como base a Figura 23.



(a) (b)

Figura 23. (a) Uma imagem binária de um objeto e (b) uma imagem do mesmo objeto observado sob um ângulo diferente.

A Figura 23 (a) mostra a imagem binária de um objeto e (b) uma imagem do mesmo objeto observado sob um ângulo diferente, hipoteticamente em três dimensões. Imaginando a reta da Figura 23 (a) como uma barreira, como mostra (b), se aplicarmos um fluxo de água da esquerda para a direita na imagem, teremos algo como mostrado na Figura 24.

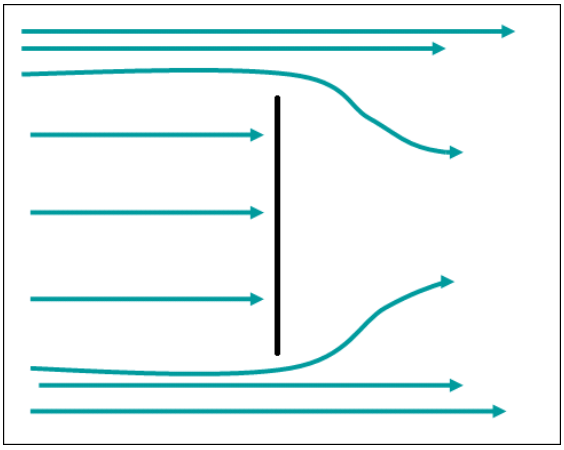


Figura 24. Fluxo de água passando da esquerda para a direita na imagem.

Na prática isso é feito percorrendo-se a imagem de cima para baixo e da esquerda para a direita, verificando se o *pixel* atual é um obstáculo (*pixel* preto). Se não for, muda a cor do *pixel* para uma cor definida como água. Se for um obstáculo a inundação daquela linha da imagem é interrompida. Um detalhe relevante é que toda vez que um *pixel* for identificado como obstáculo é necessário verificar se esse *pixel* faz parte do topo ou da base desse obstáculo. Se fizer parte do topo ou base, é preciso fazer a água escoar inclinadamente pelo obstáculo, como acontece na realidade que o algoritmo tenta simular. Como veremos mais adiante, esse ângulo pode ser controlado. A Figura 25 (a) ilustra a inundação realizada da esquerda para a direita. Após a

inundação da imagem da esquerda para a direita, fazemos a mesma inundação na direção oposta, isto é, da direita para a esquerda (Figura 25 (b)). Sobrepondo as duas imagens, temos como resultado a Figura 25 (c).

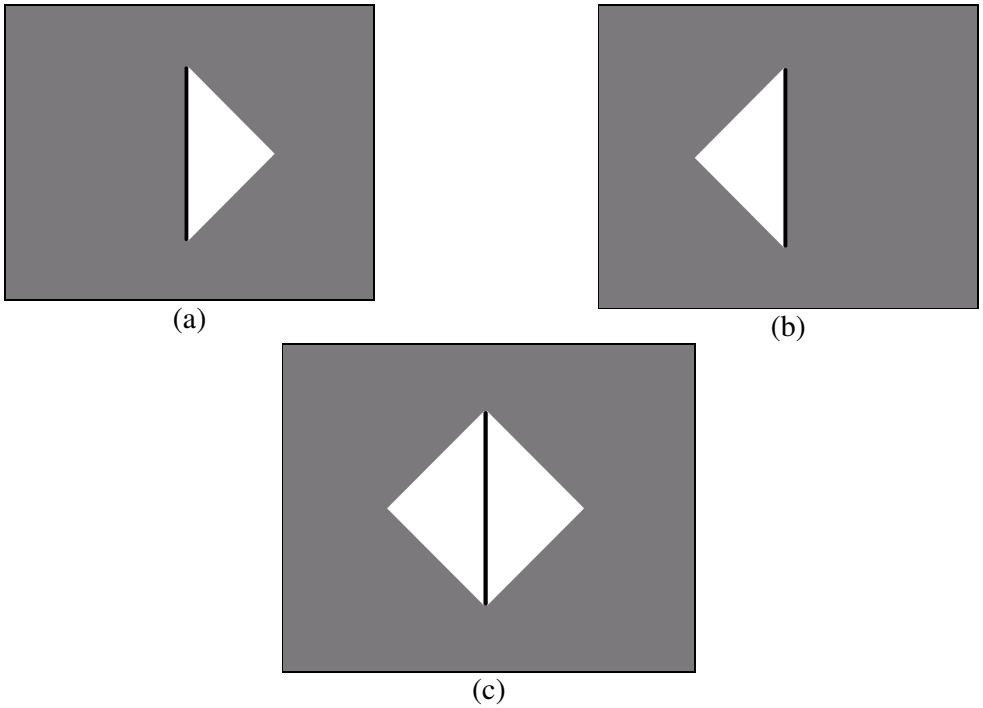


Figura 25. Imagens do processo de inundação por fluxo de água. As áreas brancas são áreas não-inundadas. (a) Imagem inundada da esquerda para a direita. (b) Imagem inundada da direita para a esquerda. (c) Imagem resultante da sobreposição das imagens (a) e (b).

Após as inundações pelos dois lados da imagem, fazemos a sobreposição delas, resultando numa imagem com os objetos identificados. A Figura 26 mostra a aplicação numa imagem com mais barreiras.

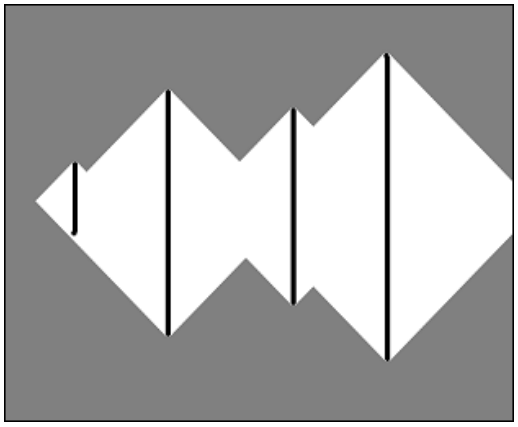


Figura 26. Resultado do algoritmo numa imagem com um número maior de barreiras.

O pseudo-código deste algoritmo percorrendo a imagem da esquerda para a direita é mostrado abaixo:

```
1.  Para x = 0 até x = altura(imagem)
2.      Para y = 0 até y = largura(imagem)
3.          Se imagem(x,y) == obstáculo
4.              Se ehBase(imagem(x,y)) ou ehTopo(imagem(x,y))
5.                  pintarInclinado(imagem(x,y), ângulo)
6.              Senão
7.                  y = y + 1
8.              Fim Se
9.          Senão
10.             imagem(x,y) = água
11.         Fim Se
12.     Fim Para
13. Fim Para
```

Antes da aplicação do algoritmo deve-se escolher o ângulo de fluxo da água pela imagem (Figura 27). Segundo Basu *et al.*, algumas representações de ângulos foram obtidas experimentalmente e são ilustradas na Figura 28. De acordo com a figura, vemos a distribuição de *pixels* para cada ângulo.

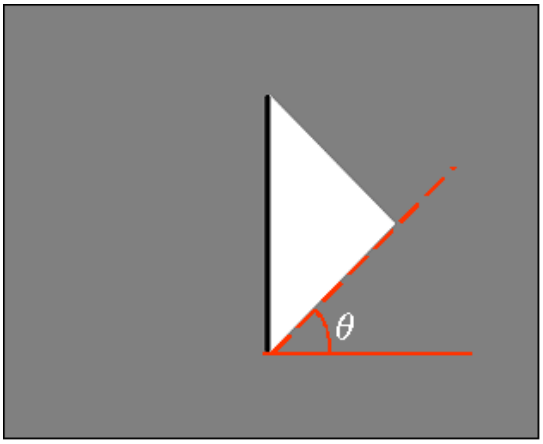
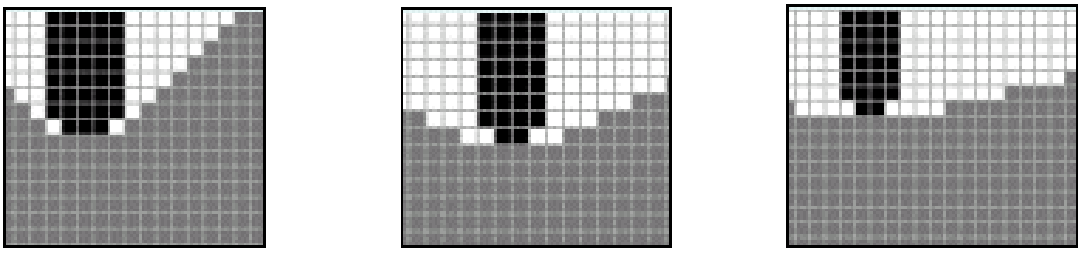


Figura 27. Ângulo θ do fluxo de água.



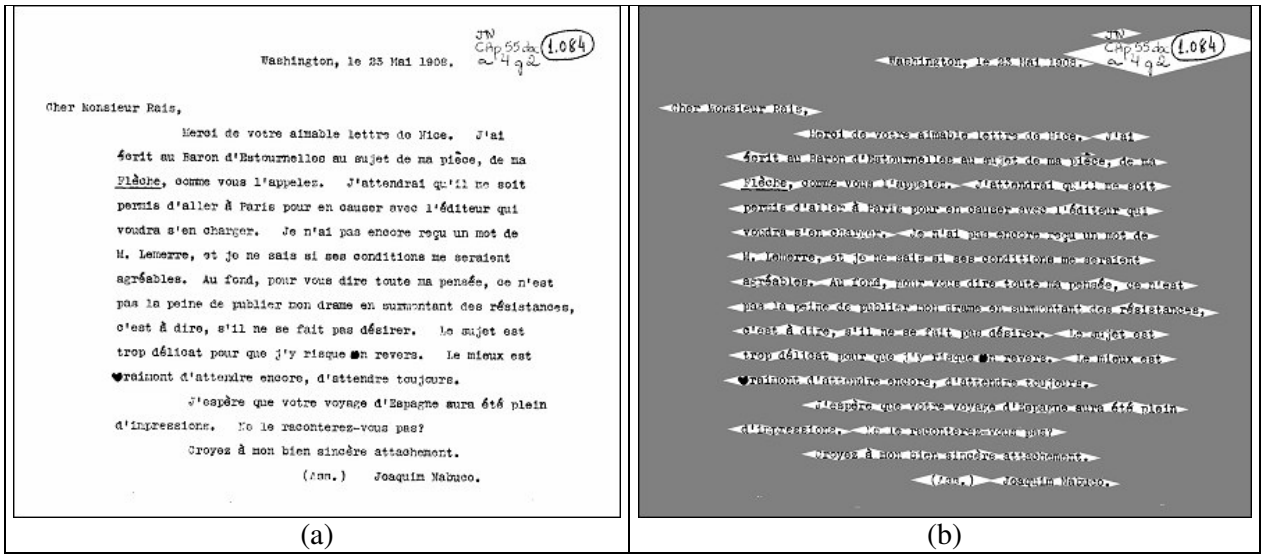
(a) (b) (c)

Figura 28. Representação matricial de alguns ângulos de fluxo. (a) 45°. (b) 26,6°. (c) 14°.

Foram feitos experimentos sobre as distribuições dos *pixels* e suas relações angulares. De acordo com a Figura 28, vemos que o fluxo de água de 45º é obtido inundando os *pixels* horizontais de maneira decremental, um por um, a partir de um limite superior ou inferior da barreira. Da mesma forma, o ângulo de 26,6º é obtido inundando os *pixels* de maneira decremental de dois em dois; e de quatro em quatro obtém-se o ângulo de 14º.

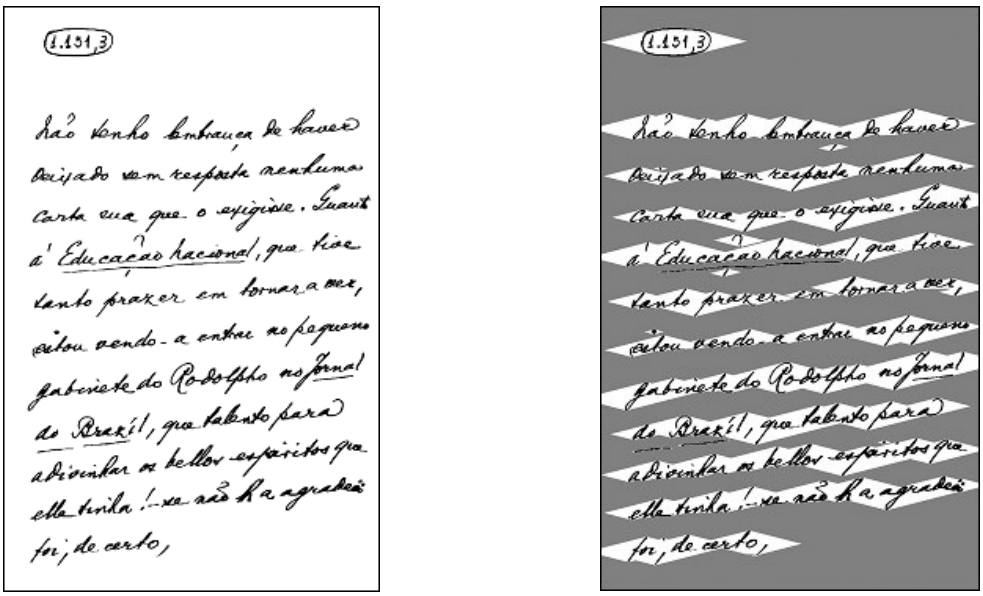
A aplicação desse algoritmo em um documento histórico datilografado e manuscrito pode ser vista, respectivamente, nas Figura 29 e Figura 30, ambas utilizando o ângulo de fluxo de 14°. Para aplicação em documentos, normalmente é necessário aplicar primeiro um filtro de erosão na imagem, para evitar que o algoritmo considere um acento ou pontuação gramatical como sendo um objeto independente [11].

Nos casos em que há palavras muito afastadas das outras da mesma linha, o algoritmo pode considerá-las como um elemento fora da linha. A operação morfológica de dilatação [1] também pode ser usada para prevenir esse caso, como mostra a Figura 31, mas dessa forma pode aproximar mais ainda linhas que já são próximas, aumentando a complexidade da extração. No trabalho de Basu *et al.* [11] foi utilizada uma elipsóide como elemento estruturante de raio variável, de acordo com o estilo da caligrafia do documento. No exemplo ilustrado na Figura 31 foi usada uma elipsóide de raio 8 como elemento estruturante da dilatação.



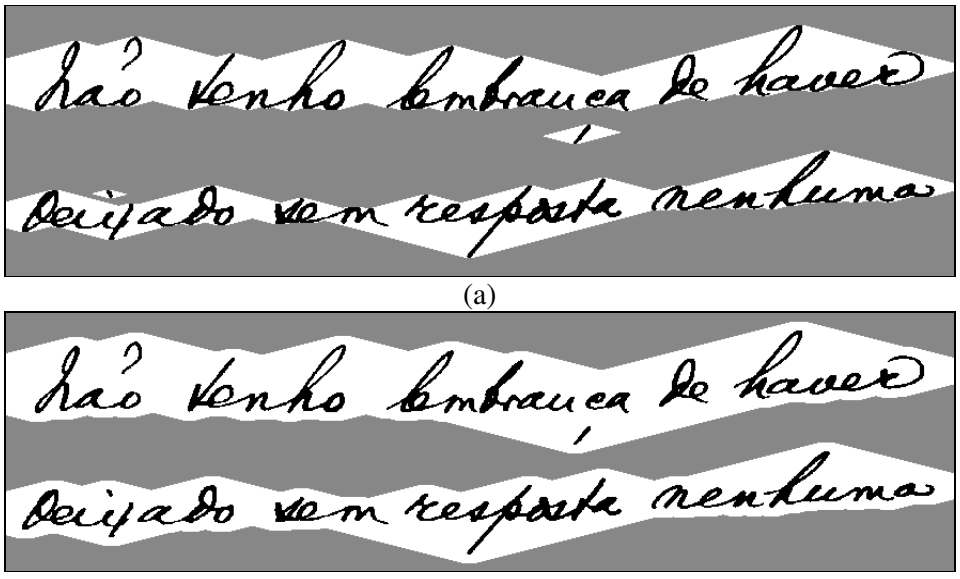
(a) (b)

Figura 29. Aplicação do algoritmo num documento datilografado: (a) imagem original e (b) imagem resultante do algoritmo aplicado com ângulo de 14°.



(a) (b)

Figura 30. Aplicação do algoritmo num documento manuscrito: (a) imagem original e (b) imagem resultante do algoritmo aplicado com ângulo de 14°.



(a) (b)

Figura 31. Exemplo do uso de dilatação para aproximar os objetos da imagem: (a) imagem segmentada sem dilatação e (b) imagem segmentada após dilatação.

3.2 Extração das Linhas

A extração das linhas identificadas é feita simplesmente removendo ordenadamente as áreas não-inundadas pelo algoritmo descrito na sub-Seção anterior. Essas linhas removidas são guardadas como imagens diferentes para serem usadas posteriormente. Um método indicado no trabalho de

Basu *et al.* para se fazer isso é rotulando cada área não-inundada com um símbolo diferente, através do algoritmo explicado a seguir. Esse algoritmo é uma versão modificada do apresentado em Gonzalez [5]. Primeiro encontra-se as linhas do documento, cuja imagem conterá três valores para seus *pixels*: 0,1 e 120, representando respectivamente áreas de papel, texto e inundação. A partir dessa imagem, para cada *pixel* P, verificam-se seus *pixels* vizinhos mais relevantes q_0, q_1, q_2 e q_3, ilustrados na Figura 32.

q_0	q_1	q_2
q_3	P	X
X	X	X

Figura 32. *Pixel* P e seus oito vizinhos.

Se esse *pixel* P for do tipo inundado, ele não é rotulado. Se for do tipo não-inundado ele é rotulado com um símbolo. Para decidir por qual símbolo o *pixel* P vai ser rotulado, observam-se os vizinhos relevantes. Se os quatro vizinhos não tiverem rótulo (forem *pixels* inundados) então usa-se um novo símbolo para rotular P. Caso contrário, rotula-se P com o mesmo símbolo de qualquer um dos quatro vizinhos. Após essa etapa de rotulação, percorre-se a imagem novamente, agrupando os pares de rótulos equivalentes num único símbolo, para evitar que cada objeto seja rotulado com mais de um símbolo [11]. Após o rotulamento das linhas, teremos algo parecido com a Figura 33 (d). Para extrair as linhas, basta comparar a imagem original com a máscara rotulada, escolher a linha a ser removida pelo rótulo na máscara e copiar os *pixels* equivalentes da imagem original para uma nova imagem.

De maneira equivalente, pode-se usar a função do Matlab *bwlabel*, que rotula cada objeto de uma imagem binária (nessa função um objeto é considerado um conjunto de *pixels* pretos), obtendo o mesmo efeito da Figura 33 (d).

Após a extração das linhas do documento, podem aparecer linhas de dimensões maiores que a média das outras linhas do mesmo documento. Isso pode indicar a presença de linhas conectadas e/ou sobrepostas, que o algoritmo não trata por si só. Para tratar esses problemas é possível a utilização de quaisquer técnicas, como as descritas na sub-seção 2.3.5, mas esse é um problema ainda sem solução efetiva e não será abordado neste trabalho. A Figura 34 mostra um exemplo de um documento no qual o algoritmo considerou uma única linha onde existem mais de uma.

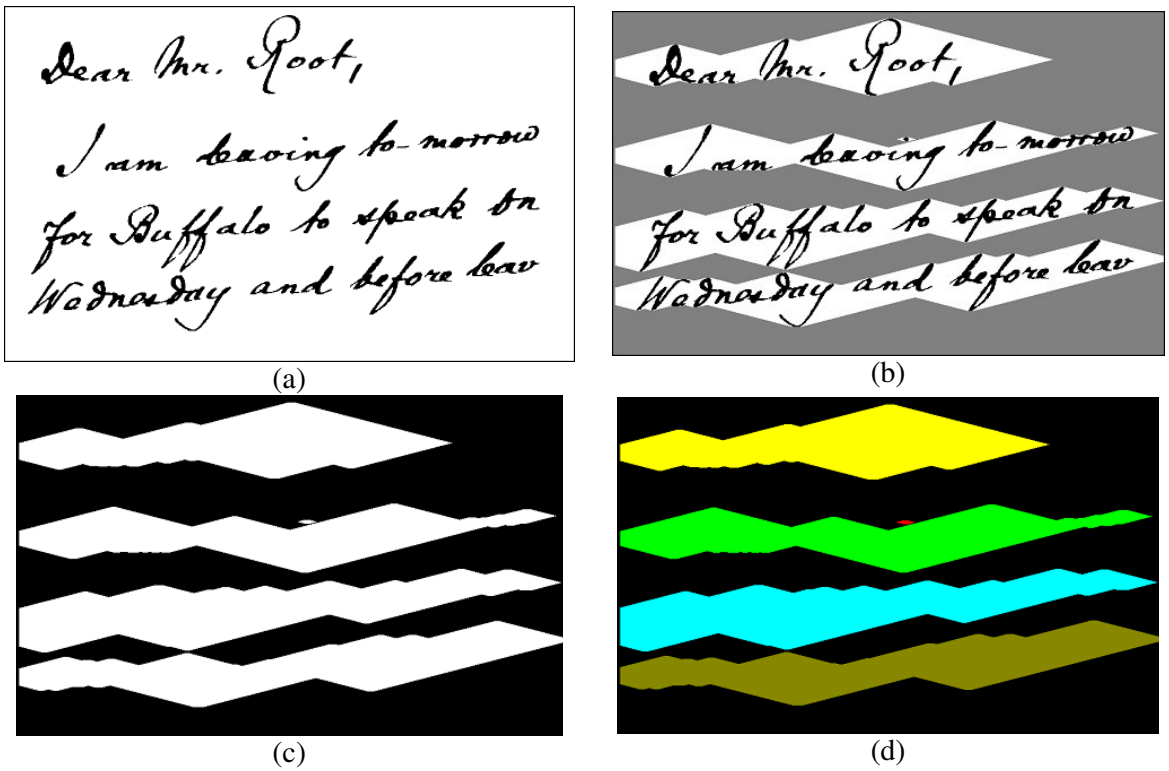


Figura 33. Exemplo de rotulamento das linhas de um documento. (a) Imagem original. (b) Imagem resultante do algoritmo analisado. (c) Máscara obtida através das áreas não-inundadas. (d) Máscara com os objetos (linhas) rotulados.

3.3 Correção da Inclinação das Linhas

Após a extração das linhas como imagens separadas, é necessário verificar se essas linhas estão com a orientação correta e, se necessário, corrigi-la. Neste trabalho, essa análise e correção da orientação das linhas será feita através da transformada de Hough, já descrita no Capítulo 2, Seção 2.2.3. A transformada de Hough é aplicada no documento inteiro antes e após da segmentação das linhas, em cada linha extraída. A Figura 35 mostra um exemplo de aplicação da correção da orientação em uma linha extraída de um documento manuscrito.

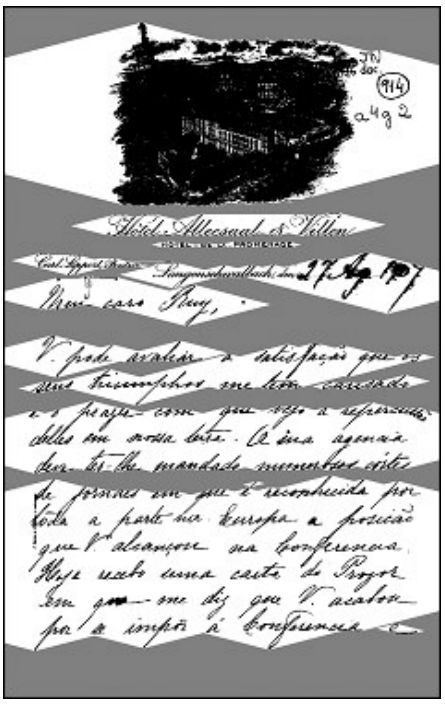
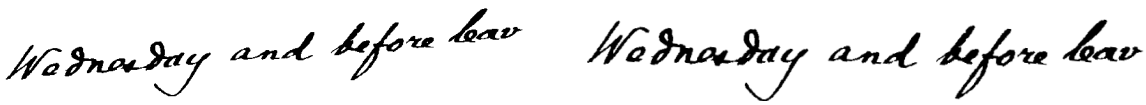


Figura 34. Documento com várias linhas sobrepostas e conectadas.



(a) (b)

Figura 35. Aplicação da correção de inclinação em uma linha. (a) Linha extraída de um documento. (b) Linha com a orientação corrigida.

Capítulo 4

Estudo de Casos

Este capítulo trata do estudo realizado em documentos históricos do acervo do projeto PROHIST, aplicando o método explicado no Capítulo 3. Além disso, faz parte do estudo uma análise comparativa dos resultados desse método com o método descrito na Seção 2.3.4, a fim de comparar os métodos, dado que essas técnicas foram divulgadas no âmbito científico recentemente [10][11].

A comparação entre os desempenhos dessas duas técnicas é a grande contribuição deste trabalho, sobretudo devido ao ineditismo que representa. Para essa comparação ser justa, utilizamos as mesmas imagens de documentos históricos usadas em Severin *et al.* [29], nas mesmas condições de pré-processamento e analisamos os mesmos aspectos dos resultados.

4.1 Material de Estudo

As imagens utilizadas neste trabalho são de documentos históricos brasileiros datados do período entre os séculos XVIII e XIX. No acervo possuíamos imagens de documentos tanto manuscritos quanto datilografados, mas só utilizamos imagens manuscritas para não fugir do escopo do trabalho. A Figura 36 mostra exemplos de documentos do acervo.

Figura 36. Exemplos de imagens de documentos do acervo.

Esses documentos foram digitalizados e arquivados, por questões de preservação, no formato de arquivo JPEG com 1% de perda. Para este estudo, utilizamos 14 imagens de documentos manuscritos, exatamente as mesmas usadas no trabalho de Severin *et al.* [29]. Essas imagens foram binarizadas utilizando o algoritmo proposto por Mello *et al.* em [16]. A Figura 37 mostra exemplos das imagens usadas para este estudo, já binarizadas.

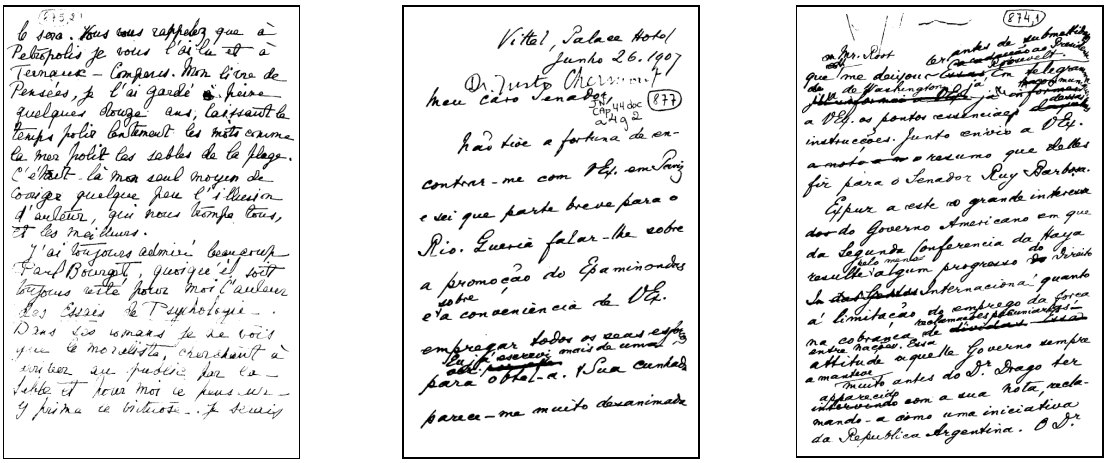


Figura 37. Exemplos de imagens binarizadas utilizadas nesse estudo.

Para poder analisar o desempenho do algoritmo descrito no Capítulo 3, ele foi implementado utilizando-se a ferramenta Matlab[3]. Seus módulos, como ilustrados na Figura 38, foram feitos da seguinte forma: a função *segmentation* recebe a imagem a ser segmentada e o valor do ângulo de fluxo; ela se utiliza de funções auxiliares para sua tarefa e então produz como saída a imagem segmentada; a imagem segmentada serve como parâmetro da função *extractor*, que chama a função nativa do Matlab de rotulamento de objetos e, de acordo com esses rótulos, copia cada linha segmentada para uma nova imagem; essa mesma função chama a função *imageRotate*, que detecta e corrige a orientação de cada linha, salvando-as no disco.

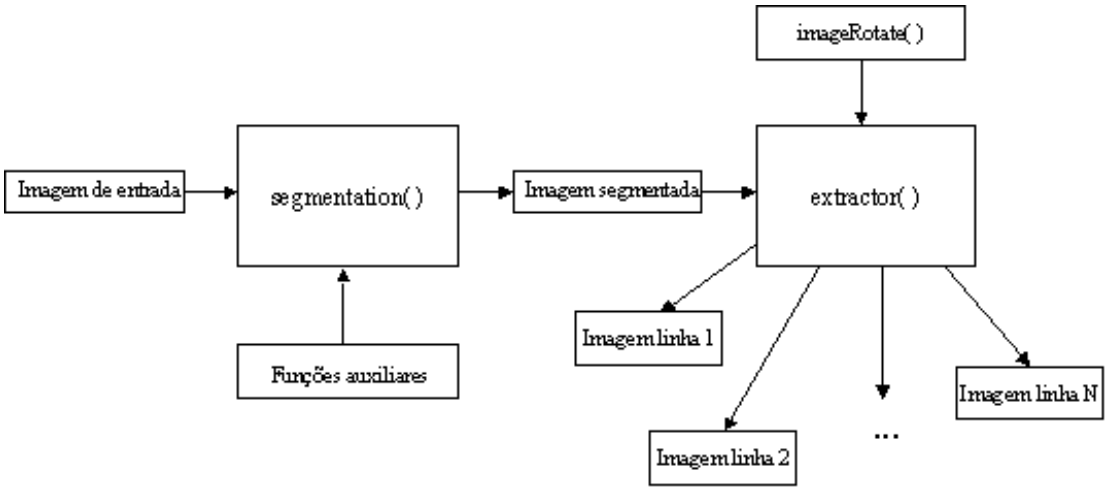


Figura 38. Diagrama mostrando os principais módulos da implementação, bem como suas interligações e produções.

4.2 Resultados

Para a análise dos resultados da aplicação do algoritmo implementado do fluxo de água, utilizamos os seguintes critérios, também usados em [29]:

- Linhas corretas: é o total de linhas que o algoritmo segmentou corretamente. Uma linha segmentada corretamente é aquela que foi totalmente encontrada, não deixando nenhuma palavra originalmente pertencente a essa linha fora dela.
- Linhas partidas: é o total de linhas que não foram detectadas inteiramente, deixando de fora alguma, ou algumas palavras.
- Linhas unidas: é o total de linhas que foram consideradas pelo algoritmo como sendo uma só.
- Linhas só de ruídos: é o total de objetos que foram considerados linhas pelo algoritmo, mas que são formados apenas por ruído.

[3] Matlab (Matrix Laboratory ou Laboratório de Matrizes) é uma ferramenta criada pela MathWorks. É uma plataforma que provê um ambiente para programação, muito usado para simulações, cálculo numérico e processamento de imagens, dado sua facilidade para tratamento de dados matriciais.

- Linhas eliminadas: é o total de linhas que o algoritmo não detectou, nem mesmo incorretamente.
- Linhas com palavras que não lhe pertencem: é o total de linhas detectadas pelo algoritmo que possuem palavras que não lhe pertencem originalmente.

Como esses aspectos também foram considerados em [29], podemos fazer uma comparação de desempenho entre as duas técnicas, apontando os pontos fortes e fracos do modelo de segmentação por fluxo de água.

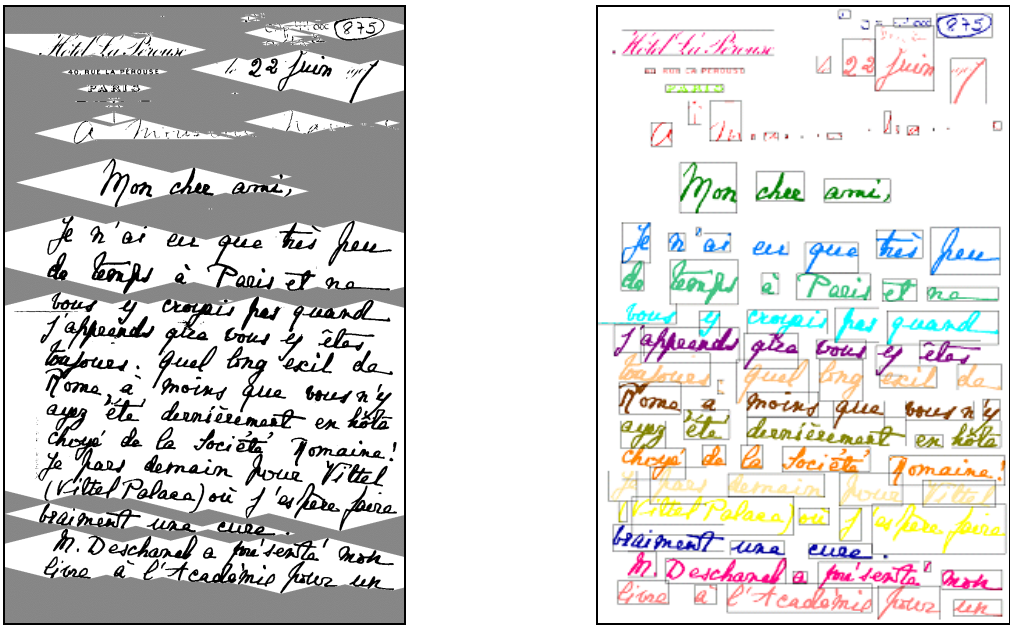
Para os experimentos, consideramos 2 ângulos para o fluxo de água do algoritmo: 14° e 26,6°, sendo esses valores aproximados. Esses ângulos foram escolhidos por serem valores intermediários entre os valores mínimo e máximo (0º e 45º, respectivamente) propostos por Basu *et al*, já se sabendo que os ângulos de fluxo próximos de 14º são os que proporcionam normalmente os melhores resultados, como mostrado em [11] e confirmado nos testes desse trabalho. As 14 imagens utilizadas nos experimentos são mostradas no Apêndice A.

A Tabela 1 mostra os resultados obtidos com a imagem mostrada na Figura A-44 (a) (Apêndice A), utilizando esses valores de ângulo, bem como os resultados da aplicação algoritmo de Severin *et al.* (Seção 2.3.4) nessa mesma imagem.

Tabela 1. Resultados obtidos com a imagem da Figura A-44 (a).

	Algoritmo de Fluxo Hipotético de Água				**Algoritmo de Severin *et al.***	
	14°		**26,6°**			
	#	**%**	**#**	**%**	**#**	**%**
Total de linhas	14	---	14	---	14	---
Linhas corretas	2	14,3	1	7,14	12	85,7
Linhas partidas	0	0	5	35,71	0	0
Linhas unidas	12	85,7	6	42,86	0	0
Linhas só de ruídos	3	---	15	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	0	0	2	14,29	2	14,3

De acordo com a tabela, vemos uma grande diferença no desempenho entre as duas técnicas. O algoritmo de Basu *et al.* apresentou 14,3% de identificação correta das linhas (no melhor caso), enquanto o algoritmo de Severin *et al.* apresentou 85,7%. A Figura 39 mostra os resultados obtidos com os dois algoritmos, para uma melhor observação. É uma diferença bastante significativa e se mantém semelhante nos resultados das outras imagens, como mostram os resultados das Tabelas 3 a 14 (Apêndice B).



(a) (b)

Figura 39. Resultados obtidos na imagem 1. (a) Segmentação por Fluxo de Água, com ângulo de 14°. (b) Novo Algoritmo de Segmentação de Documentos Históricos. A imagem (b) foi retirada da referência [29].

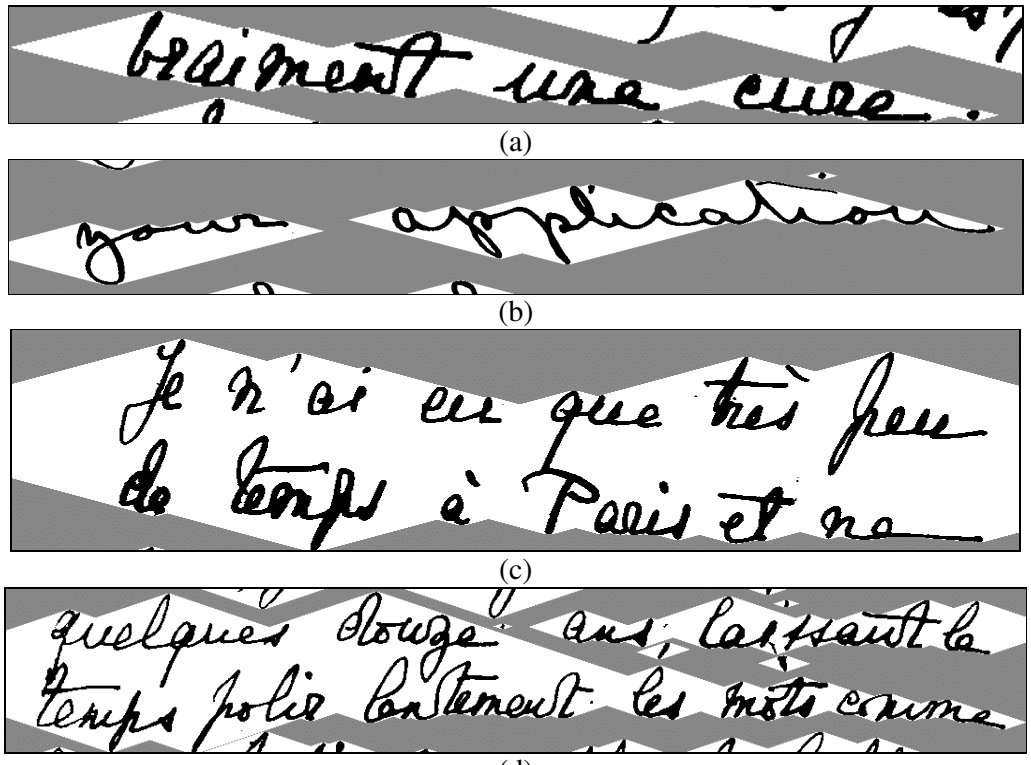
Tabela 2. Resultados obtidos com a imagem da Figura A-44 (o).

	Algoritmo de Fluxo Hipotético de Água				**Algoritmo de Severin *et al.***	
	14°		**26,6°**			
	#	**%**	**#**	**%**	**#**	**%**
Total de linhas	11	---	11	---	11	---
Linhas corretas	9	81,8	7	63,6	9	81,8
Linhas partidas	0	0	3	27,3	0	0
Linhas unidas	2	18,2	0	0	0	0
Linhas só de ruídos	8	---	8	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	0	0	1	9,1	2	18,2

De acordo com os resultados obtidos, podemos notar a superioridade do desempenho do algoritmo de Severin *et al.*. Na imagem da Figura A-44 (o) houve uma aparente equivalência entre os desempenhos, como mostrado na Tabela 2 (com o melhor ângulo, 14°). Porém o algoritmo de segmentação por fluxo de água apresenta muitas linhas formadas apenas de ruído da imagem. Essa quantidade pode ser reduzida no processo pós-segmentação, aplicando uma filtragem. No momento de rotulação e extração dos objetos-linhas do documento, pode-se desprezar os objetos que tiverem suas dimensões menores que 5x5 ou 7x7 *pixels*, por exemplo. Lembrando que esses ruídos são considerados linhas adicionais ao documento, portanto não interferem nas linhas corretamente identificadas.

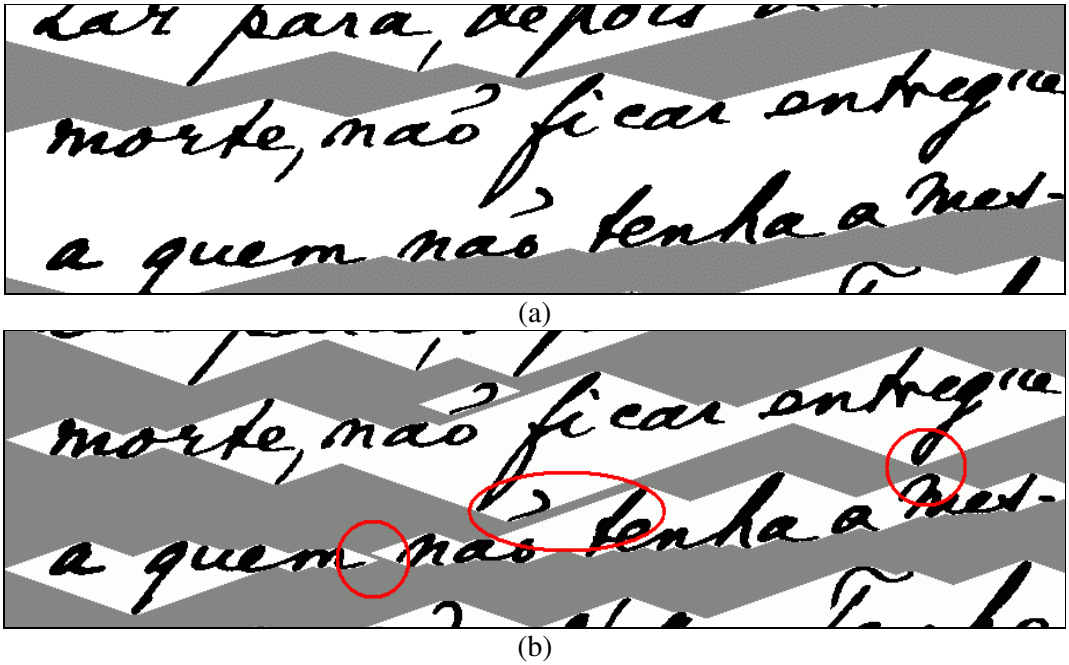
A Figura 40 mostra exemplos dos resultados obtidos, focando nos critérios e ilustrando os problemas que ocorrem na segmentação, utilizando o algoritmo de segmentação por Fluxo Hipotético de Água.

Sobre os ângulos estudados neste trabalho, 14° e 26,6° foram os intermediários entre o menor ângulo (0°) e o maior ângulo (45°). Dentre todos os resultados, os melhores foram conseguidos com o ângulo de fluxo de 14°. Observando os valores das tabelas de resultados, nota-se que aumentando o valor do ângulo, normalmente diminui-se a quantidade de linhas unidas e aumenta-se a quantidade de linhas partidas. Isso se deve ao fato de que aumentando o ângulo de fluxo, a água hipotética pode inundar áreas mais estreitas da imagem, conseguindo atingir as áreas de entrelinhas mais facilmente. Da mesma forma, palavras que estão em uma distância um pouco maior entre si não ficam numa mesma área não-inundada. A Figura 41 ilustra essa característica.



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

Figura 40. Exemplos das características encontradas nos resultados. (a) Uma linha segmentada corretamente. (b) Uma linha partida. (c) Uma linha dupla, formada por duas linhas conectadas. (d) Uma linha com palavras que pertencem à outra linha.



(a)

(b)

Figura 41. (a) Exemplo de segmentação com ângulo de fluxo 14°. (b) Mesmo trecho da imagem com segmentação com ângulo 18,4°, destacando (em vermelho) os locais onde se pode notar as diferenças, diminuindo as linhas duplas e aumentando as linhas partidas.

Quanto mais aumentamos o ângulo, mais essa característica vai se acentuando nos resultados. De forma que, para 45°, já não se consegue mais segmentar as linhas de texto, sendo mais útil para segmentação diretamente das palavras, como mostra a Figura 42.

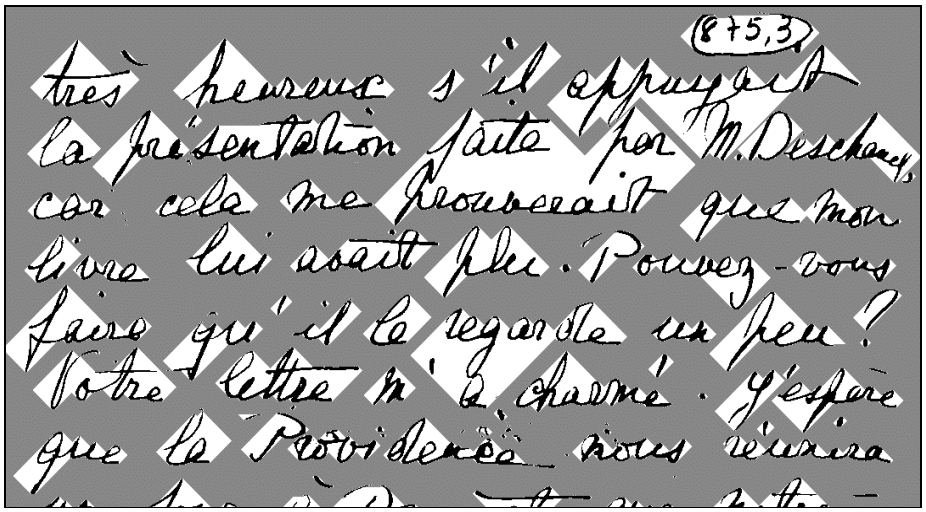
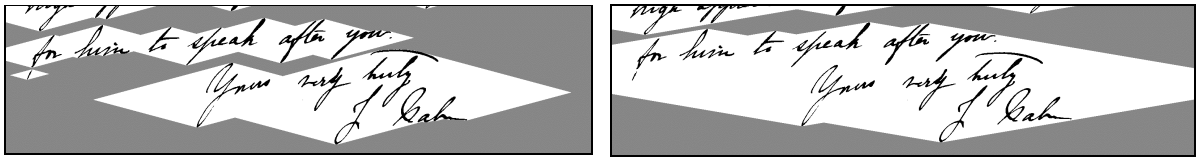


Figura 42. Exemplo de documento segmentado com ângulo de fluxo de 45°.

À medida que diminuímos o ângulo, acontece o efeito inverso. Aumenta-se o número de linhas conectadas e diminui-se o número de linhas partidas. Quanto maior o ângulo, menos áreas estreitas a água hipotética consegue alcançar, fazendo as "ilhas", áreas não inundadas, serem maiores. A Figura 43 mostra um trecho de documento segmentado usando 14° e 9,5°,

aproximadamente, ilustrando essas características. O ângulo de 14° apresentou, em média, os melhores resultados, balanceando melhor esses fatores.



(a) (b)

Figura 43. (a) Documento segmentado com fluxo de água com ângulo de 14°. (b) Documento segmentado com ângulo de 9,5°.

A análise dos resultados para o algoritmo de Severin *et al.* já foi feita em [29]. Neste trabalho, em todos os casos avaliados, ele se mostrou mais eficiente que o algoritmo de Basu *et al.* [11]. O principal problema das duas técnicas é a junção de linhas. Não existe um modelo ideal para se separar linhas conectadas, mas [29] propõe um modelo completo para identificação de linhas e palavras do texto, bem como uma heurística para separação de componentes conectados. Embora essa heurística não funcione em todos os casos, os resultados mostram uma taxa de segmentação correta aceitável. Por outro lado, a heurística proposta em [11]. não se preocupa com a separação de elementos conectados nem sobrepostos, durante a identificação das linhas. Por isso, como se pode observar pelos resultados, seus principais problemas são a falta de recorte de componentes conectados e a falta de tratamento para linhas muito próximas, resultando em muitas linhas duplas.

Capítulo 5

Conclusões

A transposição de documentos históricos para o meio digital é um importante modo de preservação e distribuição de informações da cultura de uma sociedade. O armazenamento das imagens digitalizadas desses documentos resolve o problema da preservação histórica, já que a fragilidade dos meios de comunicação impressos provoca uma baixa durabilidade desses documentos. Mas, tão importante quanto à preservação histórica, é a divulgação dessas informações. A transposição do texto desses documentos facilita o armazenamento e distribuição das informações, além de possibilitar a busca por algum dado específico.

Este trabalho se propôs a fazer um estudo sobre segmentação de linhas de documentos. Foi mostrada a importância da segmentação de linhas num sistema de OCR, o estado da arte dessa área de visão computacional, e foi feito um estudo mais aprofundado do método proposto por Basu *et al.* [11], implementando-o em Matlab. No estudo de casos, foi aplicado o algoritmo implementado em 14 imagens para avaliar o desempenho, comparando também com a técnica proposta em Severin *et al.* [29], a principal contribuição deste trabalho. Desta forma, neste capítulo, serão feitas algumas considerações sobre esse trabalho, suas contribuições e possíveis trabalhos futuros.

5.1 Considerações Finais

A segmentação de linhas de texto de documentos históricos é um dos processos mais complexos de um sistema de OCR. Neste trabalho foi estudada uma abordagem recente para este fim. Desse estudo e aplicação nas 14 imagens, foi possível verificar que a média de linhas segmentadas corretamente foi de 28,5%. Apesar de possuir uma heurística interessante e inovadora, essa técnica apresenta, como principal falha, deficiência no tratamento de linhas muito próximas e conectadas. Dessa forma, essa técnica não se adequou muito bem aos casos práticos, onde a presença desse tipo de problema é constante. Porém para casos de documentos menos problemáticos apresenta uma boa taxa de segmentação, independendo da angulação da caligrafia (*fluctuation*) do autor do texto. É uma técnica que, se complementada, pode gerar bons resultados para documentos complexos.

Do ponto de vista comparativo entre os resultados desta técnica e da técnica de Severin *et al.* [29], a última apresentou resultados de segmentação bem melhores (taxa média de 81,2% de

acerto contra 28,5% da abordagem de Basu *et al.* [11]). Um dos fatores é que ela é uma técnica mais completa e complexa. Mas mesmo assim ainda não possui tanta eficiência para tratar de componentes conectados. Dessa forma, das abordagens mais atuais pesquisadas, o uso do algoritmo de Severin *et al.* parece ser a melhor opção para textos manuscritos.

5.2 Contribuições

Este trabalhou procurou contribuir com os estudos que vêm sendo realizados nos últimos anos em processamento de documentos históricos, com a análise dos resultados da aplicação de uma técnica desenvolvida recentemente em documentos, visando facilitar o acesso à informação histórica do Brasil. Esse estudo possibilitará o início de uma análise aprofundada de segmentação de linhas nos documentos do acervo considerado, com a utilização de novas técnicas, ou até mesmo junção ou extensão de técnicas já existentes.

Além disso, a comparação entre os resultados das duas técnicas serviu para validar a técnica de Severin *et al.*, mostrando que os resultados da segmentação utilizando esse modelo foram consideravelmente melhores que os da abordagem de Basu *et al.* Essa comparação serve como contribuição científica, pois não havia sido feita até este momento.

A biblioteca desenvolvida neste trabalho também é uma contribuição importante. Ela foi feita de forma modular, podendo ser facilmente utilizada e expandida. Além disso, seus componentes podem ser facilmente substituídos, possibilitando, por exemplo, manter a forma de segmentação, mas substituir a forma de rotulação ou extração das linhas.

5.3 Trabalhos Futuros

Segmentação de linhas de texto é uma área de pesquisa em constante desenvolvimento. Como trabalho futuro, pode-se expandir a biblioteca desenvolvida neste trabalho, incorporando novas ou antigas técnicas, tornando-a uma ferramenta mais completa. Também se pode tentar expandir a abordagem de Basu *et al.* introduzindo algum método eficiente para detecção e separação de componentes duplos que foram considerados um só pelo algoritmo. Tentar desenvolver uma forma automática de escolha do ângulo de fluxo de água, de acordo com a caligrafia do autor do texto e de forma inteligente.

Um avanço interessante que pode ser dado a partir deste trabalho, além da detecção automática do ângulo de fluxo hipotético de água, seria uma variação desse fluxo no decorrer da segmentação, para melhor adequar a inundação às mudanças da caligrafia dentro de um mesmo texto. Estudar o modelo físico de fluxo de água e inundações e incorporar a essa técnica poderá apresentar melhorias para a heurística. Estudos futuros sobre a influência de operadores morfológicos no processo de segmentação e sobre métodos de escolha automática de elementos estruturantes para cada situação também podem trazer melhorias para a técnica de Basu *et al.*.

Um possível trabalho futuro também poderia ser a junção dos métodos estudados neste trabalho com algoritmos de segmentação de palavras e caracteres, servindo de entrada para algum algoritmo de reconhecimento de texto.

Bibliografia

[1] PARKER, J. R., "*Algorithms for Image Processing and Computer Vision*", Ed. John Wiley and Sons, Inc. 1995.

[2] O'GORMAN, L., KASTURI, R., "*Document Image Analysis*", Califórnia, IEEE Computer Society Press, 1995.

[3] WITTEN, I. H., MOFFAT, A., "*Managing Gigabytes Compressing and Indexing Documents and Images*", São Francisco, Ed. Morgan Kaufmann, 1999, 3ª ed.

[4] ANTONACOPOULOS, A., KARATZAS, D., "*Semantics-Based Content Extraction in Typewritten Histórical Documents*", Proc. of International Conference on Document Analysis and Recognition, ICDAR, Seul, Coréia, Agosto de 2005.

[5] GONZALEZ, R., WOODS, R., "*Digital Image Processing*", Ed. Addison-Wesley, 2002.

[6] OLIVEIRA, A. L. I., MELLO, C. A. B., SILVA JUNIOR, E. R. da, ALVES, V. M. O., "*Optical Digit Recognition for Images of Handwritten Historical Documents*", Simpósio Brasileiro de Redes Neurais, SBRN, p. 29, 2006.

[7] TRIER, O. D., JAIN, A. K., TAXT, T., "*Feature Extraction Methods for Character Recognition – A Survey*", Pattern Recognition 29, p. 641-661, 1996.

[8] MARINAI, S., GORI, M., SODA, G., "*Artificial Neural Networks for Document Analysis and Recognition*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, num. 1, Janeiro de 2005.

[9] LINKFORMAN-SULEM, L., ZAHOUR, A., TACONET, B., "*Text Line Segmentation of Historical Documents: A Survey*", International Journal on Document Analysis and Recognition, vol. 9, p. 123-138, Abril de 2007.

[10] MELLO, C. A. B., OLIVEIRA, A. L. I., SANCHEZ, A., "*PROHIST: An Environment for Image Processing of Historical Documents*", Workshop de Bibliotecas Digitais, p. 143-147, Gramado, 2007.

[11] BASU, S., CHAUDHURI, C., KUNDU, M., NASIPURI, M., BASU, D. K., "*Text Line Extraction from Multi-Skewed Handwritten Documents*", Pattern Recognition 40, p. 1825-1839, 2007.

[12] Projeto PROHIST. Acessível em: http://www.dsc.upe.br/~recpad/prohist. Acesso em: 8 de Novembro de 2007.

[13] PRATT, W. K., "*Digital Image Processing*", Ed. Addison-Wesley, Nova Iorque, 2001, 3ª ed.

[14] Tutorial disponível na internet em http://hanyfarid.org/tutorials/fip.pdf. Último acesso em 28/10/2007.

[15] CASEY, R., LECOLINET, E., "*A Survey of Methods in Strategies in Character Segmentation*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, p. 690-706, 1996.

[16] MELLO, C. A. B., "*New Tsallis Entropy Based Thresholding Algorithm for Images of Historical Documents*", ACM Document Engineering, p. 32-34, Winnipeg, 2007.

[17] MELLO, C. A. B., OLIVEIRA, A. L. I., SANCHEZ, A., "*Image Thresholding of Historical Documents: Application to the Joaquim Nabuco's File*", Digital Cultural Heritage Conference – EVA Vienna, Proc. of the 1st EVA 2006 Vienna Conference, p. 115-122, Vienna, 2006.

[18] VENTADERT, de G., ANDRE, J., RICHY, H., LINKFORMAN-SULEM, L., DESJARDIN, E., "*Les Documents Anciens*", Document Numérique, Hermès, vol. 3, p. 57-73, Junho de 1999.

[19] TAN, C. L., CAO, R., SHEN, P., "*Restoration of Archival Documents Using a Wavelet Technique*", IEEE PAMI, vol. 24, p. 1399-1404, 2002.

[20] LAMOUCHE, I., BELLISSANT, C., "*Séparation Recto/Verso d'Images de Manuscrits Anciens*", Proc. do Colloque National sur l'Ecrit et le Document, CNED, p. 199-206, Nantes, 1996.

[21] BRITO JUNIOR, A. S., FREITAS, C. O. A., JUSTINO, E., BORGES, D. L., FACON, J., BORTOLOZZI, F., SABOURIN, R., "*Técnicas em Processamento e Análise de Documentos Manuscritos*", Revista de Informática Teorica e Aplicada, vol. 8, num. 2, p. 47-68, Porto Alegre – UFRGS, 2001.

[22] MAO, S., ROSENFELD, A., KANUNGO, T., "*Document Structure Analysis Algorithms: A Literature Survey*", Proc. of SPIE/IS&T, vol. 5010, p. 197-207, 2003.

[23] MANMATHA, R., ROTHFEDER, J. L., "*A Scale Space Approach for Automatically Segmenting Words from Historical Handwritten Documents*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, num. 8, p. 1212-1225, Agosto de 2005.

[24] MARTI, U., BUNKE, H., "*On the Influence of Vocabulary Size and Language Models in Unconstrained Handwritten Text Recognition*", Proc. of International Conference on Document Analysis and Recognition, ICDAR, p. 260-265, Seattle, 2001.

[25] LIMA, J. C. B. de O., "*Análise de Algoritmos para Segmentação de Textos de Imagens de Documentos Históricos*", Trabalho de Conclusão de Curso, Engenharia da Computação, Departamento de Sistemas Computacionais, Universidade de Pernambuco, Brasil, 2006.

[26] SHI, Z., SETLUR, S., GOVINDARAJU, V., "*Text Extraction from Gray Scale Historical Document Images Using Adaptive Local Connectivity Map*", Proc. of International Conference on Document Analysis and Recognition, ICDAR, p. 794-798, Seul, Coréia, Agosto de 2005.

[27] WONG, K., CASEY, R., WAHL, F., "*Document Analysis Systems*", IBM Journal of Research and Development, vol. 26, num. 6, 1982.

[28] OLIVEIRA, L. E. S. de, "*Estudo Sobre a Extração das Estruturas Lógica e Física a Partir de Imagens de Cheques Bancários Brasileiros*", Dissertação de Mestrado, Centro Federal de Educação Tecnológica do Paraná, Curitiba, 1998.

[29] SEVERIN, P. D. S., SANCHEZ, A., "*Separación de Líneas y de Palabras en Imagenes de Documentos Históricos Manuscritos*", Trabalho de Conclusão de Curso, Ingeniería Informática, Universidad Rey Juan Carlos, Espanha, 2007.

[30] STOCKMAN, G., SHAPIRO, L., "*Computer Vision*", Ed. Prentice-Hall, 2000.

[31] ZAHOUR, A., TACONET, B., MERCY, P., RAMDANE, S., "*Arabic Handwritten Text-Line Extraction*", Proc. of International Conference on Document Analysis and Recognition, ICDAR, p. 281-285, Seattle, 2001.

[32] ZAHOUR, A., TACONET, B., RAMDANE, S., "*Contribution à la Segmentation de Textes Manuscrits Anciens*", Proc. do Colloque International Francophone sur l'Ecrit et le Document, La Rochelle, 2004.

[33] PIQUIN, P., VIARD-GAUDIN, C., BARBA, D., "*Coopération des Outils de Segmentation et de Binarisation de Documents*", Proc. do Colloque National sur l'Ecrit et le Document, p. 293-292, Rouen, 1994.

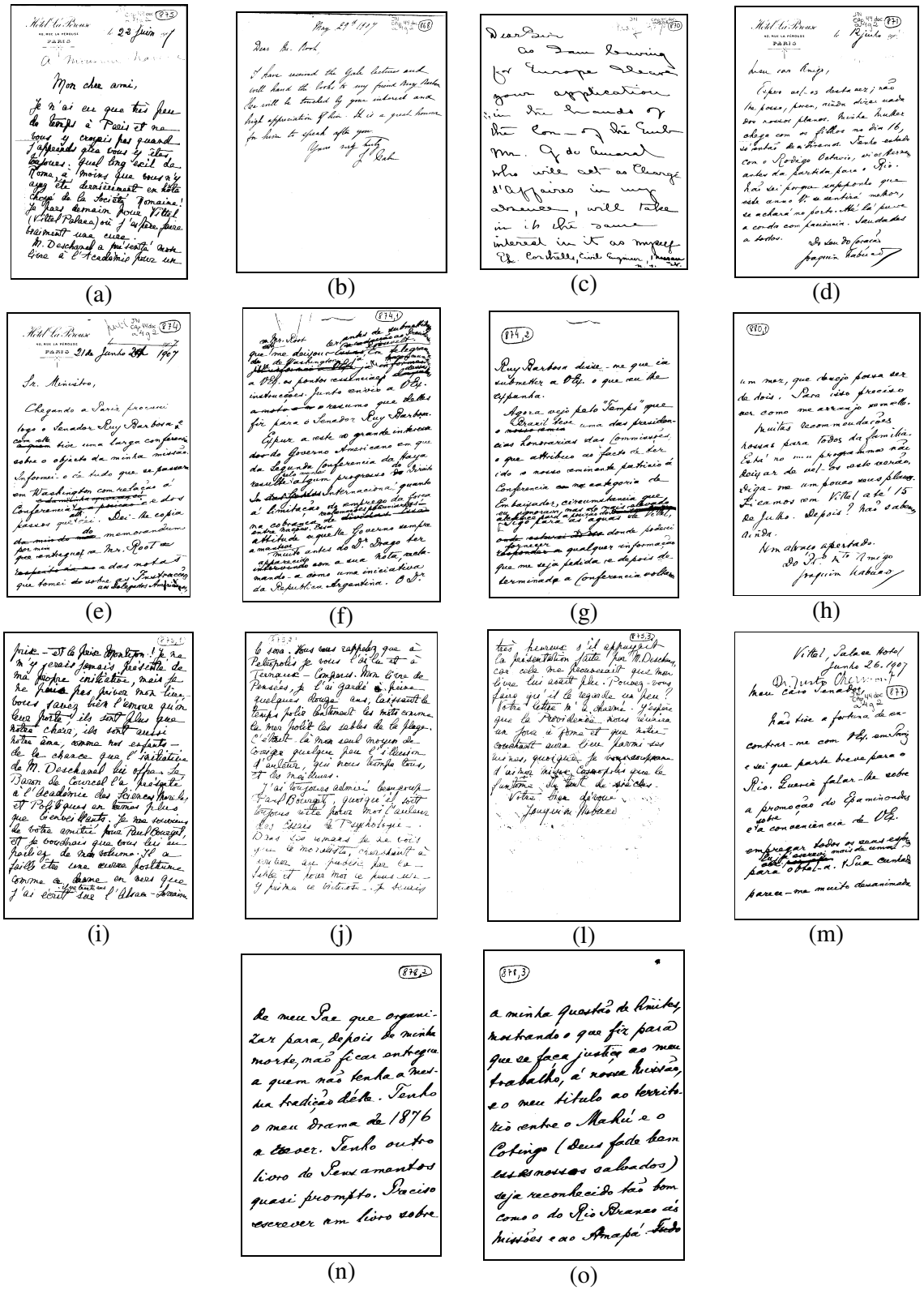
[34] BRUNOZZE, E., COFFETTI, M. C., "*An Algorithm for Extracting Cursive Text Lines*", Proc. of International Conference on Document Analysis and Recognition, ICDAR, p. 749-752, Bangalore, Índia, 1999.

Apêndice A

Imagens Utilizadas

Apresentamos aqui todas as imagens utilizadas nos experimentos descritos no Capítulo 4. As imagens apresentadas foram pré-processadas pelo algoritmo de binarização proposto por Mello *et al.* [16] e tiveram seus elementos não-processados de borda removidos manualmente. Esses elementos aparecem durante o processo de digitalização nas bordas do documento e não desaparecem no processo de binarização. Os únicos tratamentos dados as essas imagens antes da aplicação dos algoritmos foram esses.

As imagens utilizadas nos experimentos são mostradas na Figura A-44.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (l) (m) (n) (o)

Figura A-44. Todas as imagens utilizadas nos experimentos.

Apêndice B

Tabelas dos Resultados

Apresentaremos aqui as tabelas restantes de resultados dos experimentos descritos na Seção 4. A diferença de desempenho entre as técnicas avaliadas se mantém constante, exceto na imagem mostrada na Figura A-44 (o), que tem seus resultados mostrados na Tabela 2 (Capítulo 4). De fato, esta imagem é de um documento menos complexo.

As Tabelas 3 a 14 ilustram os resultados complementares descritos no Capítulo 4.

Tabela 3. Resultados obtidos com a imagem da Figura A-44 (b).

	Algoritmo de Fluxo Hipotético de Água				**Algoritmo de Severin *et al.***	
	14°		**26,6°**			
	#	**%**	**#**	**%**	**#**	**%**
Total de linhas	8	---	8	---	8	---
Linhas corretas	2	25	0	0	6	75
Linhas partidas	3	37,5	7	87,5	0	0
Linhas unidas	2	25	0	0	2	25
Linhas só de ruídos	2	---	5	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	1	12,5	1	12,5	0	0

Tabela 4. Resultados obtidos com a imagem da Figura A-44 (c).

	Algoritmo de Fluxo Hipotético de Água				**Algoritmo de Severin *et al.***	
	14°		**26,6°**			
	#	**%**	**#**	**%**	**#**	**%**
Total de linhas	13	---	13	---	13	---
Linhas corretas	2	15,4	1	7,7	11	84,6
Linhas partidas	9	69,2	10	76,9	0	0
Linhas unidas	0	0	0	0	0	0
Linhas só de ruídos	16	---	16	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	2	15,4	2	15,4	2	15,4

Tabela 5. Resultados obtidos com a imagem da Figura A-44 (d).

	Algoritmo de Fluxo Hipotético de Água				Algoritmo de Severin *et al.*	
	14°		26,6°			
	#	%	#	%	#	%
Total de linhas	14	---	14	---	14	---
Linhas corretas	3	21,4	1	7,1	12	85,7
Linhas partidas	7	50	10	71,4	0	0
Linhas unidas	2	14,3	0	0	2	14,3
Linhas só de ruídos	18	---	18	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	2	14,3	3	21,5	0	0

Tabela 6. Resultados obtidos com a imagem da Figura A-44 (e).

	Algoritmo de Fluxo Hipotético de Água				Algoritmo de Severin *et al.*	
	14°		26,6°			
	#	%	#	%	#	%
Total de linhas	14	---	14	---	14	---
Linhas corretas	9	64,3	3	21,4	12	85,8
Linhas partidas	2	14,3	10	71,4	0	0
Linhas unidas	2	14,3	0	0	1	7,1
Linhas só de ruídos	18	---	18	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	1	7,1	1	7,2	1	7,1

Tabela 7. Resultados obtidos com a imagem da Figura A-44 (f).

	Algoritmo de Fluxo Hipotético de Água				Algoritmo de Severin *et al.*	
	14°		26,6°			
	#	%	#	%	#	%
Total de linhas	19	---	19	---	19	---
Linhas corretas	5	26,4	2	10,5	8	42,1
Linhas partidas	3	15,8	8	42,1	2	10,5
Linhas unidas	10	52,6	6	31,6	8	42,1
Linhas só de ruídos	4	---	4	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	1	5,2	3	15,8	1	5,3

Tabela 8. Resultados obtidos com a imagem da Figura A-44 (g).

	Algoritmo de Fluxo Hipotético de Água				Algoritmo de Severin *et al.*	
	14°		26,6°			
	#	%	#	%	#	%
Total de linhas	17	---	17	---	17	---
Linhas corretas	7	41,2	5	29,4	11	64,7
Linhas partidas	4	23,5	7	41,2	0	0
Linhas unidas	4	23,5	4	23,5	2	11,8
Linhas só de ruídos	25	---	26	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	2	11,8	1	5,9	4	23,5

Tabela 9. Resultados obtidos com a imagem da Figura A-44 (h).

	Algoritmo de Fluxo Hipotético de Água				Algoritmo de Severin *et al.*	
	14°		26,6°			
	#	%	#	%	#	%
Total de linhas	14	---	14	---	14	---
Linhas corretas	4	28,6	2	14,3	14	100
Linhas partidas	3	21,4	9	64,3	0	0
Linhas unidas	7	50	0	0	0	0
Linhas só de ruídos	9	---	9	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	0	0	3	21,4	0	0

Tabela 10. Resultados obtidos com a imagem da Figura A-44 (i).

	Algoritmo de Fluxo Hipotético de Água				Algoritmo de Severin *et al.*	
	14°		26,6°			
	#	%	#	%	#	%
Total de linhas	21	---	21	---	21	---
Linhas corretas	0	0	0	0	17	80,9
Linhas partidas	0	0	11	52,4	0	0
Linhas unidas	21	100	10	47,6	1	4,8
Linhas só de ruídos	2	---	2	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	0	0	0	0	3	14,3

Tabela 11. Resultados obtidos com a imagem da Figura A-44 (j).

	Algoritmo de Fluxo Hipotético de Água				Algoritmo de Severin *et al.*	
	14°		26,6°			
	#	%	#	%	#	%
Total de linhas	20	---	20	---	20	---
Linhas corretas	2	10	1	5	20	100
Linhas partidas	1	5	15	75	0	0
Linhas unidas	17	85	1	5	0	0
Linhas só de ruídos	20	---	20	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	0	0	3	15	0	0

Tabela 12. Resultados obtidos com a imagem da Figura A-44 (l).

	Algoritmo de Fluxo Hipotético de Água				Algoritmo de Severin *et al.*	
	14°		26,6°			
	#	%	#	%	#	%
Total de linhas	14	---	14	---	14	---
Linhas corretas	1	7,1	0	0	14	100
Linhas partidas	1	7,1	6	42,8	0	0
Linhas unidas	11	78,7	3	21,5	0	0
Linhas só de ruídos	55	---	55	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	1	7,1	5	35,7	0	0

Tabela 13. Resultados obtidos com a imagem da Figura A-44 (m).

	Algoritmo de Fluxo Hipotético de Água				Algoritmo de Severin *et al.*	
	14°		26,6°			
	#	%	#	%	#	%
Total de linhas	14	---	14	---	14	---
Linhas corretas	6	42,8	3	21,4	11	78,6
Linhas partidas	3	21,4	9	64,3	0	0
Linhas unidas	4	28,6	0	0	3	21,4
Linhas só de ruídos	0	---	6	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	1	7,2	2	14,3	0	0

Tabela 14. Resultados obtidos com a imagem da Figura A-44 (n).

	Algoritmo de Fluxo Hipotético de Água				Algoritmo de Severin *et al.*	
	14°		26,6°			
	#	%	#	%	#	%
Total de linhas	10	---	10	---	10	---
Linhas corretas	6	60	1	10	8	80
Linhas partidas	0	0	8	80	1	10
Linhas unidas	4	40	0	0	0	0
Linhas só de ruídos	3	---	3	---	0	---
Linhas eliminadas	0	0	0	0	0	0
Linhas com palavras de outra linha	0	0	1	10	1	10